موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

پایگاه داده‌ها

(ویژه کنکور ارشد ۱۴۰۲)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

براساس کتب مرجع

راما کریشنان، آبراهام سیلبرشاتز و رامز المصری

ارسطو خلیلی‌فر

تقدیم به:

تمامی آنانی که برای پیشرفت و سعادت خود و بشریت

تلاش می کنند.

ارسطو خلیلی فر

فهرست مطالب

۱ مفاهیم اولیه

مقدمه

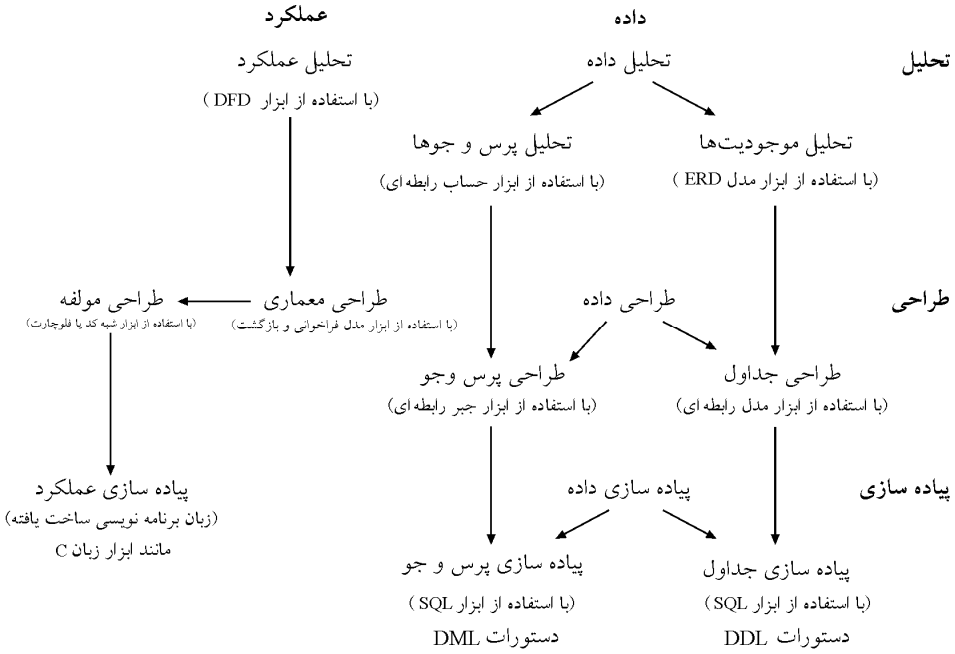
یک محصول نرم‌افزاری به واسطه‌ی فرآیند تولید نرم‌افزار که شامل فعالیت‌های مدل تحلیل، مدل طراحی، پیاده‌سازی و تست می‌باشد، ایجاد می‌گردد.

به طور کلی هر محصول نرم‌افزاری از دو وجه اساسی زیر تشکیل شده است:

۱- **ساختار داده:** محل نگهداری داده‌های محیط عملیاتی به شکل جداول.

۲- **عملکرد:** شامل برنامه‌ها و دستورالعمل‌ها یا همان برنامه کاربردی

مراحل ایجاد یک برنامه کاربردی به همراه بانک اطلاعات آن مطابق روال زیر می‌باشد:



توجه: از آنجا که ما قصد داریم در این کتاب مفاهیم مربوط بـه پایگـاه داده‌هـا را تشـریح نمـاییم، بنابراین از بیان مطلب مربوط به بخش عملکرد نرم‌افزار صرف‌نظر می‌نمائیم. بخـش عملکـرد را در کتاب مهندسی نرم‌افزار مورد بحث و بررسی قرار داده‌ایم. پس در ادامه به طور مفصـل بـه مفـاهیم مربوط به پایگاه داده‌ها می‌پردازیم. در ادامه با ذکر یک مثال، به طـور اجمـالی مراحـل ایجـاد یـک سیستم بانک اطلاعاتی را بیان می‌کنیم.

مثال: بانک اطلاعات و پرس‌وجوهای مربوط به نگهداری اطلاعات سیستم تولیدکنندگان و قطعات را تحلیل، طراحی و در نهایت پیاده‌سازی نمایید.

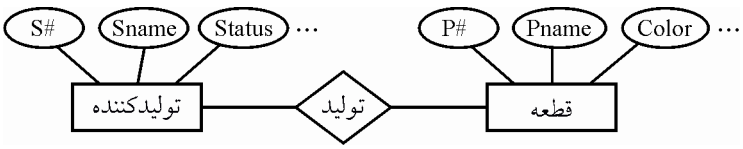
پاسخ:

مدل تحلیل

مدل تحلیل داده شامل تحلیل موجودیت‌ها و تحلیل پرس و جوها می‌باشد. تحلیـل موجودیـت‌هـا توسط ابزار مدل ER و تحلیل پرس و جوها توسط ابزار حساب رابطه‌ای مدل می‌شوند.

الف) تحلیل موجودیت ها (توسط مدل ER)

در یک محیط عملیاتی اسامی عام، موجودیت‌ها می‌باشند. بنابراین موجودیت‌های تهیه‌کننـده (S) و قطعه (P) موجود می‌باشند.



توجه: مبحث مدل ER در فصل مدل ER تشریح می‌گردد.

ب) تحلیل پرس و جوها (توسط حساب رابطه‌ای)

پرس و جو: نام تولیدکنندگان ساکن شهر C2 :

$$\{t \mid \exists\, sx \in s(t[sname] = sx[sname] \ \text{ and } sx[city] = 'c2')\} \quad \overset{\text{خروجی}}{\Rightarrow} \quad \begin{array}{c} \underline{\text{Sname}} \\ \text{Sn2} \\ \text{Sn3} \end{array}$$

توجه: حساب رابطه‌ای به دو طبقه کلی حساب رابطه‌ای دامنه‌ای و حساب رابطـه‌ای تـاپلی تقسـیم می‌شود. پرس و جوی فوق بر اساس حساب رابطه‌ای تاپلی نوشته شده است.

توجه: مبحث حساب رابطه‌ای (دامنه‌ای و تاپلی) در فصل حساب رابطه‌ای تشریح می‌گردد.

مدل طراحی

پس از مدل تحلیل داده، نوبت به مدل طراحی داده می‌رسد. طراحی داده بر دو بخش طراحی جداول و طراحی پرس و جو می‌باشد. طراحی جداول از بخش طراحی داده، تحلیل موجودیت (ERD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط مدل رابطه‌ای، طراحی جداول را انجام می‌دهد. طراحی پرس و جو از بخش طراحی داده، تحلیل پرس و جو (حساب رابطه‌ای) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط جبر رابطه‌ای، طراحی پرس و جو را انجام می‌دهد.

الف) طراحی جدول (توسط مدل رابطه‌ای)

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S
(تولیدکنندگان)

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP
(تولید)

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P
(قطعات)

توجه: مبحث مدل رابطه‌ای در فصل مدل رابطه‌ای تشریح می‌گردد.

ب) طراحی پرس و جو (توسط جبر رابطه‌ای)

پرس و جو: نام تولیدکنندگان ساکن شهر C2 :

$$\Pi_{sname}(\sigma_{city=c2}(s)) \quad \overset{\text{خروجی}}{\Rightarrow}$$

Sname
Sn2
Sn3

توجه: مبحث جبررابطه‌ای در فصل جبررابطه‌ای تشریح می‌گردد.

توجه: مدل تحلیل، مدل‌سازی عالم خارج به زبانی شبیه انسان است، اما مدل طراحی مدل‌سازی عالم داخل ماشین به زبانی شبیه زبان ماشین است. بنابراین برای اینکه ساخت برنامه کامپیوتری که به زبان ماشین است، امکان‌پذیر باشد، باید مدل تحلیل به مدل طراحی نگاشت شود تا مدل طراحی بتواند به عنوان نقشه راهی شبیه به زبان ماشین، راهگشا باشد.

مدل‌های تحلیل، کلی‌تر و انتزاعی‌تر هستند، و برای مدل‌سازی عالم خارج مورد استفاده قرار می‌گیرند، بدون آنکه به جزئیات نحوه پیاده‌سازی بپردازند، در واقع، مدل تحلیل به کلی‌گویی به زبان انسان و حذف جزئیات نحوه پیاده‌سازی می‌پردازد. اما مدل‌های طراحی، جزئی‌تر و غیرانتزاعی‌تر هستند، و برای مدل‌سازی عالم داخل ماشین مورد استفاده قرار می‌گیرند، و به بیان جزئیات نحوه پیاده‌سازی می‌پردازند، در واقع مدل طراحی به جزئی‌گویی به زبان شبیه ماشین و درج جزئیات نحوه پیاده‌سازی می‌پردازد.

با نگاهی سطح به سطح، به حل مساله، سطوح مختلفی از انتزاع را خواهیم داشت. در بالاترین سطح انتزاع، راه حل به صورت کلی به زبان محیط مساله بیان می‌گردد. در سطوح پایین‌تر، راه حل به جزئیات پیاده‌سازی نزدیک‌تر می‌شود و در پایین‌ترین سطح انتزاع راه حل به صورتی بیان می‌شود تا مستقیما قابل پیاده‌سازی باشد.

پیاده‌سازی

پس از مدل طراحی نوبت به پیاده‌سازی می‌رسد. پیاده‌سازی جداول از بخش پیاده‌سازی داده، طراحی جداول از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DDL در SQL، پیاده‌سازی جداول را انجام می‌دهد. پیاده‌سازی پرس و جو از بخش پیاده‌سازی داده، طراحی پرس و جو از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DML در SQL پیاده‌سازی پرس و جو را انجام می‌دهد.

الف) پیاده‌سازی جدول (توسط SQL:DDL)

پیاده‌سازی جدول تولیدکنندگان (S)

```
Create Table S
        (
        S# char (5),
        Sname char (20),
        City char (15),
        Primary key (S#)
        )
```

پیاده‌سازی جدول قطعات (P)

```
Create Table P
        (
        P# char (5),
        Pname char (20),
        Color char (10),
        Primary key (P#)
        )
```

پیاده‌سازی جدول تولید (SP)

```
Create Table SP
        (
        S# char (5),
        P# char (5),
        QTY numeric (10),
```

```
Primary key (S#, P#),
Foreign key (S#) References S(S#)
  on delete cascade
  on update cascade,
Foreign key (P#) References P(P#)
  on delete cascade
  on update cascade,
Check (QTY>1 AND QTY<1000)
)
```

توجه: مبحث SQL:DDL در فصل SQL:DDL تشریح می‌گردد.

ب) پیاده‌سازی پرس و جو (توسط SQL : DML)

پرس و جو: نام تولید کنندگان ساکن شهر C2 :

```
Select  Sname          خروجی      Sname
From S                 ⇒           Sn2
Where  City = 'C2'                 Sn3
```

توجه: مبحث SQL:DML در فصل SQL:DML تشریح می‌گردد.

حال که به طور اجمالی با مراحل ایجاد یک سیستم بانک اطلاعاتی آشنا شدید، در ادامه به تشریح دقیق‌تر مفاهیم پایگاه داده می‌پردازیم.

تعریف افزونگی

افزونگی در معنای عام به معنی تکرار ذخیره‌سازی داده‌ها می‌باشد. به عبارت بهتر تکرار مقادیر یک یا چند صفت خاصه در نمونه‌های مختلف یک نوع رکورد از یک فایل، به بیانی دیگر ذخیره‌سازی آن مقادیر در بیش از یک نقطه از فایل. نوع دیگری از افزونگی، تکرار اطلاعات در فایل‌های مختلف می‌باشد.

توجه: افزونگی سبب هدر رفتن حافظه می‌گردد.

انواع افزونگی

افزونگی بر دو نوع است:

۱- افزونگی طبیعی (محتوایی)

در افزونگی طبیعی، یک مقدار مشخص از صفت خاصه در تعدادی از نمونه رکوردها وجود دارد که بر دو نوع زیر است:

الف) تک فایلی(در یک فایل داده‌ای رخ می‌دهد.)

ب) چند فایلی (در چند فایل داده‌ای رخ می‌دهد.)

مثال: جدول studclg با مقادیر زیر را در نظر بگیرید:

S#	Sname	Clg#	Clgname
8621	Ali	12	Computer
8442	Reza	12	Computer
8731	Abbas	NULL	NULL

افزونگی طبیعی

در جدول studclg این اطلاعات که «کد دانشکده‌ی کامپیوتر برابر 12 است.» در سطرهای اول و دوم تکرار شده است، پس افزونگی داده (طبیعی) وجود دارد. اگر شخصی بخواهد کد دانشکده‌ی کامپیوتر را به 22 تغییر دهد و به اشتباه، فقط مقدار #Clg در سطر اول جدول studclg را عوض کند، از این به بعد دانشکده کامپیوتر دو کد خواهد داشت: 12 و 22. این موضوع یعنی **بی‌نظمی یا آنومالی یا ناهنجاری**.

در سطر آخر از این جدول، مقادیر NULL وجود دارد. به طور کلی، به ازای هر سطر از جدول studclg که #Clg آن برابر NULL باشد یک سطر در جدول studclg خواهیم داشت که مقادیر دو ستون آخر آن NULL خواهد بود. پس مشکل مقادیر NULL نیز وجود دارد.

بنابراین اگر تمامی اطلاعات جداول را در یک جدول بریزیم، مشکلات زیر را خواهیم داشت:
مانند جدول studclg.

۱- افزونگی داده‌ها (Data Redundancy)

۲- آنومالی یا بی‌نظمی یا ناهنجاری (Anomaly)

۳- مقادیر تهی (NULL Values)

حال اگر جدول studclg را به دو جدول clg و stud به شکل زیر تجزیه و نرمال کنیم:

S#	Sname	Clg#
8621	Ali	12
8442	Reza	12
8731	Abbas	NULL

جدول Stud

Clg#	Clgname
12	Computer

جدول clg

مشکلات فوق مرتفع و نتایج زیر حاصل خواهد شد.

- کاهش افزونگی طبیعی (محتوایی) مانند تکرار بی‌دلیل سطر (12,Computer)
- افزایش افزونگی تکنیکی به دلیل تعریف کلید خارجی (در قالب ستون های مشترک)

توجه: مبحث نرمال‌سازی در فصل نرمال‌سازی تشریح می‌گردد.

افزونگی تکنیکی

تکرار بعضی از مقادیر یک یا چند صفت خاصه در محیط ذخیره‌سازی جهت ایجاد یک شیوه دستیابی کاراتر برای جداول را افزونگی تکنیکی می‌گویند. مثل کلید خارجی برای ارتباط بین جداول یا وقتی که روی صفت خاصه‌ای از یک جدول، شاخص ایجاد می‌کنیم، مقادیر آن صفت در جدول شاخص تکرار خواهد شد.

توجه: مبحث کلید خارجی در فصل مدل رابطه‌ای تشریح می‌گردد.

توجه: مبحث شاخص در فصل SQL:DDL تشریح می‌گردد.

به منظور ایجاد یک سیستم نگهداری اطلاعات در یک محیط عملیاتی دو شیوه رایج می‌باشد:

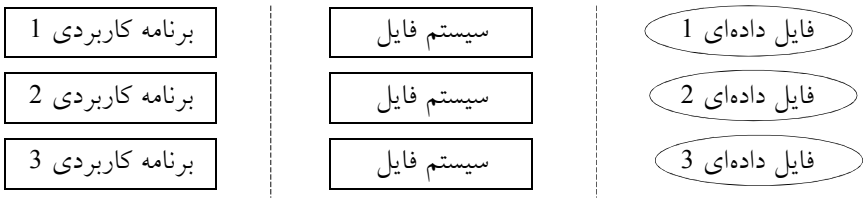
۱- سیستم فایلینگ (پرونده‌ای)

۲- سیستم بانکی (پایگاه داده)

۱- سیستم فایلینگ (پرونده‌ای)

در این روش با بهره‌گیری از امکانات سیستم فایل که در اختیار سیستم عامل می‌باشد و یک زبان برنامه‌نویسی سطح بالا و با نگرش به ورودی‌ها و خروجی‌های موردنظر، برنامه‌های کاربردی به منظور کنترل و پردازش فایل‌های داده‌ای، تحلیل، طراحی، و پیاده‌سازی می‌شوند.

همچنین برای هر فایل داده‌ای رکورد موردنظر با فیلدهای موردنیاز تعریف می‌شود:



سیستم حساب

آدرس	موجودی	نام و نام‌خانوادگی	شماره حساب	شماره مشتری
تهران	400	علوی	0313	6037
تهران	500	علوی	1314	6037

سیستم اعتباری

آدرس	بدهی	مقدار وام	نام و نام‌خانوادگی	شمار حساب	شماره مشتری
تهران	400	500	علوی	0313	6037

به طور کلی بسیاری از داده‌های مورد نیاز یک برنامه کاربردی، همان داده‌های مورد نیاز برنامه‌های دیگر است، در روش فایلینگ هر برنامه کاربردی فایل داده‌ای ویژه خود را دارد، بنابراین داده‌های مشترک بین برنامه‌های مختلف باید مجدداً ذخیره شوند که سبب تکرار اطلاعات (افزونگی چند

فایلی) و موجب هدر رفتن رسانه ذخیره‌سازی می‌گردد. در سیستم‌های فایلینگ به علت عدم **تجمع** داده‌های ذخیره شده (عدم اشتراک داده‌ها برای همه برنامه‌ها)، عدم وحدت ذخیره‌سازی و عدم نرمال‌سازی، پدیده افزونگی تک فایلی و چند فایلی را شاهد خواهیم بود.

مثال تک فایلی: مانند تکرار نام و نام‌خانوادگی

مثال چند فایلی: مانند آدرس یک شخص که در فایل داده‌ای هر برنامه کاربردی باید تکرار گردد.

آنومالی داده‌ای یا ناهنجاری داده‌ای

مقدمه ناهنجاری داده‌ای، افزونگی است. آنومالی داده‌ای یا بی‌نظمی داده‌ای یا ناهنجاری داده‌ای زمانی بروز می‌کند که بنابر دلایلی یک فقره اطلاع در بیش از یک نقطه ذخیره گردد و لازم باشد بروزرسانی شود. اگر عمل بروزرسانی در تمام نقاطی که آن فقره اطلاع وجود دارد، انجام نشود، ناهمگونی در اطلاعات و به عبارت دیگر پدیده ناهنجاری داده‌ای رخ داده‌است. به عنوان مثال برای حالت تک فایلی، اگر در فایل داده‌ای سیستم حساب، نام و نام‌خانوادگی «علوی» در سطر اول تغییر کند و فراموش گردد تا در سطر دوم نیز برورسانی گردد، آنگاه شماره مشتری «۶۰۳۷» دارای دو نام و نام‌خانوادگی متفاوت می‌باشد، که ناهنجاری داده‌ای رخ داده‌است. به عنوان مثال برای حالت چند فایلی، زمانی که آدرس «علوی» در فایل داده‌ای سیستم حساب تغییر کند و آدرس «علوی» در فایل داده‌ای سیستم اعتباری بروزرسانی نگردد، ناهنجاری داده‌ای رخ داده‌است.

توجه: در سیستم‌های فایلینگ کنترل پدیده ناهنجاری داده‌ای به دلیل عدم مجتمع بودن داده‌ها و عدم نرمال‌سازی بسیار مشکل و به صورت دستی است که خطای انسانی را به همراه خواهد داشت.

توجه: در سیستم‌های فایلینگ برقراری امنیت منطقی داده‌ها در برابر خطراتی از قبیل آتش‌سوزی و دستیابی غیرمجاز به دلیل عدم وجود یک مکانیزم پشتیبانی و حفاظتی و در نتیجه دسترسی ساده به داده‌ها بسیار مشکل است.

توجه: در سیستم‌های فایلینگ به علت جدا بودن فایل داده‌ای برنامه‌ها از یکدیگر امنیت فیزیکی داده‌ها در سطح مناسبی برقرار است. زیرا داده‌های همه برنامه‌های کاربردی به دلیل عدم مجتمع بودن فایل‌های داده‌ای در یک دیسک به یکباره در معرض تهدید قرار نمی‌گیرند.

توجه: در سیستم‌های فایلینگ هر برنامه کاربردی برای ایجاد و پردازش فایل داده‌ای مختص به خود نوشته شده‌است، بنابراین هرگونه تغییری در ساختار فایل داده‌ای منجر به تغییر در برنامه کاربردی می‌گردد. به بیان دیگر برنامه‌های کاربردی به جنبه و خصوصیات محیط فیزیکی ذخیره‌سازی داده‌ها وابسته است. زیرا کلیه تعاریف ایجاد فایل‌های داده‌ای و کار با آن‌ها، درون برنامه‌های کاربردی قرار دارد. که این موضوع به معنی عدم استقلال داده‌ای نیز می‌باشد.

توجه: با توجه به ماهیت سیستم‌های فایلینگ و محلی بودن برنامه‌ها و استفاده از الگوی صف جهت پردازش درخواست‌ها، دسترسی همزمان و اشتراکی به داده‌ها معنا و مفهومی ندارد.

توجه: در سیستم‌های فایلینگ، به دلیل عدم وجود سطوح دسترسی و مسائل امنیتی زمان اجرای برنامه‌های کاربردی کاهش می‌یابد، به دلیل عدم عبور از گیت امنیتی. اما زمان پاسخ به درخواست‌ها به دلیل وجود صف و عدم استفاده اشتراکی از داده‌ها افزایش می‌یابد.

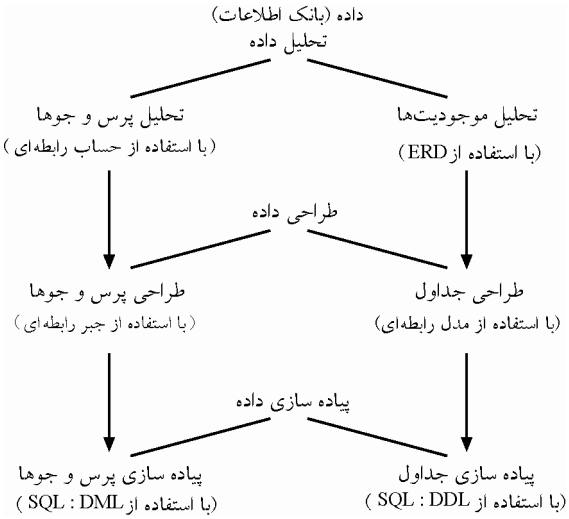
مثال: یک سیستم فایلینگ به صورت زیر است:

```
    type
        phone=record
        name:string [20];
        no:integer;
var
    fp:text;   یا   file of phone;
    i,n: integer;
begin
    assign (fp, 'sample.dat');
    rewrite (fp);
    readln (n);
for i:=1 to n do begin
    readln (no,name)
    writeln (fp,no,name)
end;
reset (fp);
    while TRUE do begin
    readln (fp,no,name);
    write (no,name)
if EOF (fp) then break;
    end;
    close (fp);
    end.
```

توجه: همانطور که در برنامه فوق ملاحظه می‌کنید، تعاریف مربوط به **عملکرد و داده** با هم در یک فایل قرار دارد.

۲- سیستم بانکی (پایگاه داده)

در این روش به منظور ایجاد بانک اطلاعات روال زیر را خواهیم داشت:



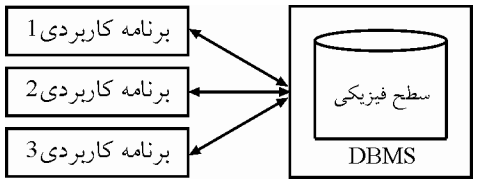
۱- تحلیل
(طراحی ادراکی (مفهومی) یا ادراکی عام)
توسط DA

۲- طراحی
(طراحی منطقی یا ادراکی خاص)
توسط DBA

۳- پیاده‌سازی
توسط DBP

در مرحله بعد یک برنامه کاربردی (عملکرد) مطابق فرآیند تولید نرم‌افزار باید ایجاد گردد. تا کاربران نهایی بتوانند به کمک DBMS در نهایت به داده‌های ویژه خود با توجه به سطوح دسترسی تعریف شده در DBMS برای آنها دسترسی یابند.

توجه: در واقع DBMS رابط بین برنامه‌های کاربردی و داده‌هاست و هرگونه دستیابی به داده‌ها از طریق DBMS صورت می‌گیرد. DBMS همچون قلعه‌ای می‌ماند که محل زندگی پادشاه، ملکه و شاهزادگان را درون خود محصور کرده‌است، که هرگونه آمد و شد به محل زندگی پادشاه باید با عبور از قلعه و اجازه قلعه‌بانان صورت گیرد. پادشاه، ملکه و شاهزادگان همان داده‌ها هستند و قلعه و قلعه‌بانان همان DBMS.



توجه: در سیستم‌های بانکی به دلیل مجتمع بودن داده‌های ذخیره شده (اشتراکی بودن داده‌ها برای برنامه‌ها)، وحدت ذخیره‌سازی و نرمال‌سازی، پدیده افزونگی به معنی آنچه در سیستم‌های فایلینگ دیدید رخ نمی‌دهد. و به تبع ناهنجاری داده‌ای نیز ایجاد نمی‌گردد. مجتمع بودن بدین معنی است که بانک اطلاعات مجموعه‌ای از جداول است که بخشی از ستون‌های اضافی بین آنها حذف شده است. غیر از کلیدهای خارجی که نقش ارتباط بین جداول را بازی می‌کنند که به آن افزونگی تکنیکی گفته می‌شود.

مثال: جداول زیر را در سیستم تولیدکنندگان و قطعات در نظر بگیرید.

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S
(تولیدکنندگان)

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP
(تولید)

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P
(قطعات)

برای‌مثال در بانک فوق که از سه جدول تشکیل شده است، لازم نیست که در جدول SP، فیلد City ذخیره شود، چرا که در صورت نیاز با رجوع به جدول S می‌توان آن را به دست آورد.

توجه: در سیستم‌های بانکی برقراری امنیت داده‌ها در برابر خطراتی از قبیل آتش‌سوزی و دستیابی غیرمجاز به دلیل وجود یک سیستم مدیریت بانک اطلاعات (DBMS) در سطح بسیار مناسبی قرار دارد. مانند پشتیبان‌گیری خودکار در یک رسانه ذخیره‌سازی دیگر، تعریف سطوح امنیتی و دسترسی ویژه کاربران و مدیران. برای مثال در SQL Server تا زمانی که در حال اجرا می‌باشد، امکان کپی و سرقت فایل‌های داده‌ای نمی‌باشد، مگر اجرای SQL Server یا همان نگهبان داده‌ها متوقف شود، که این عمل با مجوز مدیران انجام می‌گیرد.

توجه: در سیستم‌های بانکی به علت مجتمع بودن داده‌های ذخیره شده (اشتراکی بودن داده‌ها برای برنامه‌ها) و وحدت ذخیره‌سازی در یک دیسک امنیت فیزیکی داده‌ها در سطح مناسبی برقرار نیست. زیرا داده‌ها به دلیل مجتمع بودن و عدم توزیع‌شدگی در برخی موارد ممکن است به یکباره در معرض تهدید قرار گیرند. البته در اغلب موارد داده‌ها در دیسک‌های مختلف توزیع می‌شوند، که در این حالت مساله امنیت فیزیکی هم وجود نخواهد داشت. و به تبع امنیت فیزیکی در سیستم‌های بانکی نیز در سطح مناسبی برقرار خواهد بود.

توجه: مهمترین خصیصه استفاده از DBMS در سیستم‌های بانکی، مستقل شدن برنامه‌های کاربردی از جنبه و خصوصیات فیزیکی ذخیره‌سازی داده‌ها است. زیرا کلیه تعاریف و ایجاد داده‌ها و جداول در DBMS انجام می‌گیرد و مانند سیستم‌های فایلینگ تعاریف فایل‌های داده‌ای درون برنامه‌های کاربردی قرار ندارد که این موضوع به معنی استقلال داده‌ای نیز می‌باشد.

توجه: با توجه به ماهیت سیستم های بانکی و مبتنی بر شبکه بودن، کاربران مختلف می توانند به صورت همزمان با بانک کار کنند، یعنی هر کاربر بدون ایجاد محدودیت برای کاربر دیگر در هر لحظه می‌تواند با بانک کار کند.

توجه: با آنکه در روش بانکی وحدت ذخیره‌سازی داریم ولی هر کاربری دید خاص خود را نسبت به داده‌ها دارد، به دلیل تعریف سطوح دسترسی برای کاربران توسط DBMS.

توجه: در سیستم‌های بانکی، به دلیل وجود سطوح دسترسی و مسائل امنیتی زمان اجرای برنامه‌های کاربردی افزایش می‌یابد، به دلیل عبور از گیت امنیتی. اما زمان پاسخ به درخواست‌ها به دلیل عدم وجود صف و استفاده اشتراکی از داده‌ها کاهش می‌یابد.

بخش‌های مختلف سیستم‌های بانکی

محیط بانک اطلاعاتی از چهار بخش زیر تشکیل شده است:

(۱) سخت‌افزار

(۲) نرم‌افزار

(۳) داده

(۴) کاربر

۱- سخت‌افزار

در محیط بانک اطلاعات سه دسته سخت‌افزار وجود دارد:

الف) سخت‌افزار ذخیره‌سازی اطلاعات: منظور همان رسانه ذخیره‌سازی خارجی است. همانند دیسک که رسانه اصلی ذخیره‌سازی می‌باشد.

ب) سخت‌افزار پردازنده مرکزی: منظور همان کامپیوتر است.

ج) سخت‌افزار ارتباطی: منظور سخت‌افزارهای مورد نیاز جهت ارتباط بین کامپیوتر و دستگاه‌های جانبی و نیز سایر کامپیوترها می‌باشد.

۲- نرم‌افزار

در محیط بانک اطلاعاتی دو دسته نرم‌افزار وجود دارد:

نرم‌افزار کاربردی: واسط بین کاربر نهایی و سیستم مدیریت پایگاه داده‌ها (DBMS) می‌باشد. این نرم‌افزار به کمک یک زبان سطح بالا و یک زبان داده‌ای (مانند SQL) ایجاد می‌گردد.

نرم‌افزار سیستمی: واسط بین نرم‌افزار کاربردی و بانک اطلاعاتی می‌باشد. نرم‌افزار سیستمی از دو جزء DBMS و سیستم عامل تشکیل شده است. DBMS که نرم‌افزاری پیچیده است میهمان یک سیستم عامل است و از امکانات سیستم عامل در انجام وظایفش استفاده می‌کند. DBMS به برنامه‌ساز امکان می‌دهد تا :

- پایگاه داده خود را تعریف کند (توسط دستورات DDL در SQL)
- پایگاه داده خود را تغییر دهد (توسط دستورات DML در SQL)

- پایگاه داده خود را کنترل کند (توسط دستورات DCL در SQL)

۳- داده

منظور داده‌های مربوط به موجودیت‌های مختلف در یک محیط عملیاتی می‌باشد.
مثال: مانند موجودیت‌های تولیدکننده و قطعه در محیط عملیاتی تولیدکنندگان و قطعات.

۴- کاربر

در محیط بانک اطلاعاتی چهار نوع کاربر وجود دارد:

الف) مدیر داده‌ها (DA : Data Administrator)

انجام مدل تحلیل یا همان طراحی ادراکی(مفهومی) بر عهده DA است، مدیر داده‌ها یک شخصی مدیریتی است و نه یک شخص فنی. DA شخصی است که مسئولیت کنترل اصلی بر روی داده ها را بر عهده دارد. وظیفه مدیر داده ها آن است که در مرحله اول تصمیم بگیرد که چه داده‌هایی باید در بانک اطلاعاتی ذخیره شود و سپس هنگامی که داده ها ذخیره شدند، سیاست‌گذاری های لازم را جهت نگهداری و کار با آن‌ها تعیین کند. برای مثال اینکه چه کسی می‌تواند چه اعمالی را بر روی چه داده‌هایی انجام دهد، به معنی تعیین سطوح دسترسی به داده‌ها.

وظایف مدیر داده‌ها (DA)

- انجام تحلیل موجودیت‌ها توسط مدل ER.
- انجام تحلیل پرس و جوها توسط حساب رابطه‌ای.

ب) مدیر بانک اطلاعات (DBA : Data Base Administrator)

انجام مدل طراحی یا همان طراحی منطقی بر عهده DBA است، مدیر بانک اطلاعات فردی است فنی که پشتیبانی‌های تکنیکی لازم را برای پیاده‌سازی تصمیمات مدیر داده‌ها (DA) فراهم می‌سازد. وظیفه DBA ایجاد بانک اطلاعات و پیاده‌سازی کنترل‌های جامعیتی و امنیتی است که سیاست گذاری‌های مدیر داده‌ها (DA) را اعمال کند. همچنین DBA مسئول نظارت بر کارایی و پاسخ به تغییر نیازهاست، بدین معنی که DBA مسئول سازماندهی سیستم برای بدست آوردن بهترین کارایی برای سازمان می‌باشد و همزمان با تغییر نیازهای سازمان، تنظیمات متناسب با نظارت DA اعمال می‌گردد. همچنین DBA مجموعه‌ای از برنامه‌نویسان و سایر افراد فنی را همچون DBPها جهت پیاده‌سازی کارها در اختیار دارد.

وظایف مدیر بانک اطلاعات (DBA)

- انجام طراحی جداول توسط مدل رابطه‌ای.
- انجام طراحی پرس و جوها توسط جبر رابطه‌ای.

ج) برنامه‌نویسان بانک اطلاعات (DBP : Data Base Programming)

انجام فعالیت پیاده‌سازی بر عهده DBP است، برنامه‌نویسانی هستند که از طریق دستورات DML در SQL با بانک اطلاعات برای انجام عملیات درج، حذف و بروزرسانی در ارتباط هستند. این افراد دستورات DML را درون یک زبان سطح بالا (زبان میزبان) قرار می‌دهند و برنامه‌های کاربردی را می‌سازند تا نیازهای کاربران نهایی را برای استفاده از داده‌های درون بانک اطلاعاتی بر طرف سازند.

وظایف برنامه‌نویسان بانک اطلاعات (DBP)

- انجام پیاده‌سازی جداول توسط دستورات DDL در SQL.
- انجام پیاده‌سازی پرس و جوها توسط دستورات DML در SQL.
- انجام پیاده‌سازی کنترل‌های امنیتی توسط دستورات DCL در SQL.

د) کاربر نهایی

فردی است که از برنامه‌های نوشته شده استفاده می‌کند. این افراد با سیستم مدیریت پایگاه داده‌ها از طریق برنامه‌های کاربردی که توسط DBP ایجاد گردیده در تعامل هستند.

توجه: تنها مدیران (DBA, DA) و برنامه‌سازان (DBP) می‌توانند بدون دخالت DBMS به داده‌ها دسترسی داشته باشند و تمام کاربران نهایی فقط از طریق DBMS به داده های داخل بانک اطلاعات دسترسی دارند.

معماری بانک اطلاعات

معماری ANSI برای پایگاه داده‌ها شامل سه لایه زیر است:

۱- لایه خارجی.

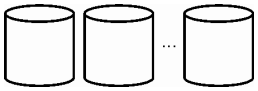
۲- لایه ادراکی شامل زیر لایه‌های مدل تحلیل (طراحی ادراکی یا ادراکی عام) و مدل طراحی (طراحی منطقی یا ادراکی خاص).

۳- لایه داخلی (فیزیکی).

توجه: در معماری بانک اطلاعات، کلمه تصویر، مترادف لایه می‌باشد. برای مثال لایه خارجی همان مفهوم تصویر خارجی را دارد.

یک محصول نرم‌افزاری به واسطه فرآیند تولید نرم‌افزار که شامل فعالیت‌های مدل تحلیل، مدل طراحی، پیاده‌سازی و تست می‌باشد، ایجاد می‌گردد. کاربران نهایی در لایه خارجی، مدل تحلیل و مدل طراحی در لایه ادراکی و فعالیت پیاده‌سازی در لایه داخلی قرار دارند.

شکل زیر گویای مطلب است:

	تصویر خارجی	کاربر n ... کاربر 2 کاربر 1	دیدهای کاربران مختلف (view)
تحلیل	تصویر ادراکی عام (طراحی ادراکی)	مدل ER موجودیت 1 — ارتباط — موجودیت 2	کل بانک بدون توجه به مدل خاصی
طراحی	تصویر ادراکی خاص (طراحی منطقی)	مدل رابطه‌ای A B C	کل بانک در قالب مدل انتخابی
پیاده‌سازی	تصویر فیزیکی	SQL 	کل بانک روی رسانه

لایه خارجی

در این لایه، میزان دسترسی کاربران به پایگاه داده، مشخص و مدیریت می‌گردد. لایه خارجی، تنها لایه‌ای است که به کاربران مربوط می‌شود. هر کاربر با بخشی از بانک سر و کار دارد و فقط اجازه دستیابی به بخشی از بانک به او داده می‌شود. برای مثال در بانک اطلاعات مربوط به بانک مرکزی، کاربر نهایی (کارمندان) بخش ارز مسافری، حق دستیابی به بخش‌های دیگر بانک مثل حساب افراد و وام ها را ندارند. قانون «**پنهان‌سازی اطلاعات**» که می‌گوید «**به هر کس به همان اندازه اطلاعات بده که نیاز دارد و نه بیشتر**» در اینجا صادق است. بنابراین در لایه خارجی دیدهای مختلف کاربران مطرح است و هر کدام بخشی از بانک را می‌بینند.

توجه: لایه خارجی لایه‌ای است که کاربران با آن سر و کار دارند، لایه‌های دیگر به مدیر و برنامه‌سازان بانک اطلاعات مربوط می‌شود.

لایه ادراکی

لایه ادراکی شامل دو زیر لایه مدل تحلیل(طراحی ادراکی یا ادراکی عام) و مدل طراحی(طراحی منطقی یا ادراکی خاص) می‌باشد.

مدل تحلیل (طراحی ادراکی یا ادراکی عام)

مدل تحلیل داده شامل تحلیل موجودیت‌ها و تحلیل پرس و جوها می‌باشد. تحلیل موجودیت‌ها توسط ابزار مدل ER و تحلیل پرس و جوها توسط ابزار حساب رابطه‌ای مدل می‌شوند.

مدل طراحی (طراحی منطقی یا ادراکی خاص)

پس از مدل تحلیل داده، نوبت به مدل طراحی داده می‌رسد. طراحی داده بر دو بخش طراحی جداول و طراحی پرس و جو می‌باشد. طراحی جداول از بخش طراحی داده، تحلیل موجودیت (ERD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط مدل رابطه‌ای، طراحی جداول را انجام می‌دهد. طراحی پرس و جو از بخش طراحی داده، تحلیل پرس و جو(حساب رابطه‌ای) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط جبر رابطه‌ای، طراحی پرس و جو را انجام می‌دهد.

توجه: مدل تحلیل (طراحی ادراکی یا ادراکی عام) به مدل خاصی وابستگی ندارد چون در شروع فعالیت‌ها قرار دارد، اما مدل طراحی (طراحی منطقی یا ادراکی خاص) به مدل خاصی بستگی دارد، بسته به اینکه در مدل تحلیل چه مدلی انتخاب شود، مدل طراحی باید تبعیت کند. اگر مدل تحلیل ساخت یافته بود، مدل طراحی نیز باید ساخت یافتگی را دنبال کند و اگر مدل تحلیل شیءگرا بود، مدل طراحی نیز باید شیءگرایی را دنبال کند.

لایه داخلی(فیزیکی)

پس از مدل طراحی نوبت به پیاده‌سازی می‌رسد. پیاده‌سازی جداول از بخش پیاده‌سازی داده، طراحی جداول از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DDL در SQL، پیاده‌سازی جداول را انجام می‌دهد. پیاده‌سازی پرس و جو از بخش پیاده‌سازی داده، طراحی پرس و جو از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DML در SQL پیاده‌سازی پرس و جو را انجام می‌دهد.

زبان‌های پیاده‌سازی

همانطور که گفتیم یک محصول نرم‌افزاری از دو وجه عملکرد (برنامه کاربردی) و داده (بانک اطلاعات) تشکیل می‌شود. در ادامه به معرفی انواع زبان‌های برنامه‌سازی می‌پردازیم.

زبان پیاده‌سازی برنامه کاربردی (وجه عملکرد)

برنامه کاربردی نیز مانند بخش داده، حاصل مراحل تحلیل، طراحی و پیاده سازی می باشد که شرح این مراحل مربوط به درس مهندسی‌نرم‌افزار می‌باشد. مرحله پیاده‌سازی برنامه کاربردی توسط یکی از زبان‌های برنامه‌نویسی سطح بالا انجام می‌شود.

توجه: به زبان‌های سطح بالا، زبان میزبان یا زبان روالی (Procedural) نیز گفته می‌شود.

زبان پیاده‌سازی بانک اطلاعات (وجه داده)

در بانک اطلاعات از زبان‌های بیانی (Declarative) که به آنها زبان پرس‌وجو (Query Language) نیز گفته می‌شود، استفاده می‌شود. در زبان‌های بیانی کاربر برنامه‌ساز کافیست

بگوید چه چیزی لازم دارد تا سیستم برای او ایجاد (مثل جداول) یا استخراج (مثل پرس و جوها) کند. در واقع چگونگی ایجاد جداول یا استخراج پرس‌وجوها از دید کاربر برنامه‌ساز وکاربر نهایی مخفی است.

برای مثال در یک سیستم کامپیوتری بخش آموزش یک دانشگاه برای استخراج «نام و شماره دانشجویانی که معدل آنها بالای 18 است» کافی است در ابتدا نام جدول یا جداول به عنوان مکان یا محل پرس و جو، سپس شرط انتخاب سطر به عنوان ملاک انتخاب سطرها و در نهایت ستون‌های نام و شماره دانشجو به عنوان ستون‌های مورد نیاز بیان شود. و به همین دلیل است که کلمه «بیانی» به این نوع زبان‌ها اطلاق می‌شود، زیرا کاربر برنامه‌ساز فقط مشخصات کامل وکلی آنچه را که لازم دارد بیان می‌کند و به جزئیات چگونگی و نحوه استخراج کاری ندارد، این مشخصات کامل و کلی به ترتیب شامل سه بخش نام جدول یا جداول به عنوان مکان یا محل پرس و جو، شرط انتخاب سطر به عنوان ملاک انتخاب سطرها و مدل انتخاب ستون به عنوان ستون های مورد نیاز است. در مثال فوق لازم نیست از تعداد سطرهای جدول، حلقه تکرار و غیره حرفی به میان آید. اصولاً حلقه تکرار با زبان‌های بیانی بیگانه است.

توجه: به زبان‌های بیانی، زبان‌های میهمان نیز گفته می‌شود.

توجه: یک برنامه کاربردی نوشته شده به یک زبان روالی سطح بالا پس از ارتباط با یک سرویس‌دهنده زبان بیانی مانند SQL Server از طریق مکانیزم‌های ویژه (Connection String) اقدام به استفاده از بانک اطلاعات می‌نماید. همچنین برای اتصال به پایگاه داده، در رشته اتصال (Connection String) نحوه احراز هویت (Authentication) کاربر مشخص می‌شود.

توجه: هر مدل داده‌ای (مدل رابطه‌ای)، زبان بیانی خاص خود را دارد. به طور مثال SQL برای مدل رابطه‌ای ایجاد گردیده است.

در مرحله پیاده سازی بانک اطلاعات سه مقوله ایجاد جداول، ایجاد پرس وجوها و ایجاد سطوح امنیتی کاربران نهایی انجام می‌گردد. بنابراین دستورات SQL به سه دسته زیر تقسیم می‌گردد:

۱ـ دستورات تعریف داده‌ها (DDL: Data Definition Language)

کارهای مربوط به ظرف‌سازی و ساختارسازی، کارهایی از قبیل ایجاد و حذف جداول

مثال: دستور Create Table برای ایجاد جداول و دستور Drop Table برای حذف جداول

۲ـ دستورات تغییر داده‌ها (DML: Data Manipulation Language)

کارهای مربوط به تغییرات محتوای جداول، کارهایی از قبیل ذخیره داده‌ها در جداول، بازیابی داده‌ها از جداول، بروزرسانی داده‌ها در جداول و حذف داده‌ها از جداول

مثال : دستورات Insert، Select، Update و Delete

۳ـ دستورات کنترل داده‌ها (Data Control Language : DCL)

کارهای مربوط به امنیت جداول، کارهایی از قبیل ایجاد سطوح دسترسی برای کاربران نهایی

مثال: دستور Grant برای ایجاد حق دسترسی و دستور Revoke برای حذف حق دسترسی

توجه: اجتماع مجموعه دستورات DDL، DML و DCL را DSL می‌گویند.

توجه: DSL سرواژه عبارت Data Sub Language است.

توجه: به اجتماع مجموعه دستورات DDL، DML و DCL، زبان پرس و جو (QL: Query Language) نیز گفته می‌شود.

به طور کلی دو دسته زبان داده‌ای وجود دارد :

الف) زبان‌های داده‌ای نامستقل یا ادغام شده

در این نوع زبان‌ها، زبان داده‌ای حتماً باید میهمان یک زبان سطح بالا باشد مثل SQL که در ویژوال بیسیک استفاده می‌شود.

ب) زبان‌های داده‌ای مستقل

در این نوع زبان‌ها، زبان داده‌ای نیازی به زبان سطح بالا ندارد. مانند Access و Foxpro.

شِمای بانک اطلاعات

به مجموعه ساختارهای طراحی شده در یک بانک بدون توجه به داده‌هایی که در آن‌ها قرار می‌گیرند، شِمای (Schema) بانک اطلاعات گفته می‌شود، برای مثال در مدل رابطه‌ای، شِمای یک بانک را جداول تشکیل می‌دهند یا برای نمونه نوع داده هر ستون به شِمای بانک مربوط می‌شود ولی تعداد سطرهای موجود در جدول ربطی به شِمای بانک ندارد.

توجه: هر پایگاه داده دارای قالب یا شِما (Schema) است و مجموعه‌ای از اطلاعات ذخیره شده در پایگاه داده در یک زمان خاص را نمونه پایگاه داده می‌نامند. شِمای پایگاه داده مانند تعریف نوع متغیر و نمونه شِمای پایگاه داده، مشابه مقدار متغیر در یک لحظه خاص است.

دیکشنری داده‌ها (کاتالوگ سیستم)

در یک سیستم بانک اطلاعات، اسامی زیادی مورد استفاده قرار می‌گیرند، از آنجایی که افراد زیاد و متفاوتی در یک مجموعه بانک اطلاعات درگیر هستند، مرجعی برای ایجاد یکنواختی و هماهنگی در نام داده‌ها و معنای آنها ضروری است. این مرجع، دیکشنری داده‌ها نام دارد.

در اختصار کاتالوگ سیستم یا دیکشنری داده ها شامل تمامی اطلاعات سیستمی مربوط به پایگاه داده‌ها همچون مشخصات سیستمی جداول و سطوح دسترسی کاربران می‌باشد که به طور خودکار توسط DBMS بروزرسانی می گردد. به بیان کامل تر هرگونه تغییرات حاصل از دستورات DDL در پایگاه داده همچون ایجاد جداول، حذف جداول، ایجاد شاخص، حذف شاخص، ایجاد View، حذف View و یا هرگونه تغییرات حاصل از دستورات DML در پایگاه داده همچون درج، حذف

و بروزرسانی رکوردها که منجر به تغییر تعداد رکوردها و به تبع آن تغییر اندازه جداول پایگاه داده‌ها می‌شود و یا هر گونه تغییرات حاصل از ایجاد و تغییر سطوح دسترسی توسط دستورات DCL در پایگاه داده همچون تخصیص سطوح دسترسی به کاربران و هر آنچه که مربوط به مشخصات سیستمی پایگاه داده باشد در کاتالوگ سیستم نگهداری می‌شود. در واقع شناسنامه بانک اطلاعات، دیکشنری داده‌ها یا کاتالوگ سیستم است.

توجه: دیکشنری داده در جایگاه خود، پایگاه داده‌ای سیستمی شامل اطلاعاتی سیستمی در مورد پایگاه داده یک محیط عملیاتی می‌باشد که حاوی «داده‌هایی درباره داده‌ها» است که گاهی اوقات به نام «فراداده» یا «دادگان» به معنی داده در مورد داده معرفی می‌گردد.

توجه: از آنجا که فضای ذخیره‌سازی دیکشنری داده ارتباطی به پایگاه داده محیط عملیاتی ندارد، بنابراین منجر به استقلال داده (پایگاه داده) از دیکشنری داده‌ها می‌گردد.

توجه: کاتالوگ سیستم به واسطه اجرای تمامی دستورات DDL و برخی دستورات DML مانند Insert و Delete و نه Select و Update دستخوش تغییر می‌گردد.

محتویات کاتالوگ سیستم

- مشخصات سیستمی جداول (مانند نام جداول و نام و نوع ستون‌های جداول)
- مشخصات سیستمی ارتباطات جداول
- مشخصات سیستمی دیدهای بانک اطلاعات (viewها)
- مشخصات سیستمی شاخص‌های بانک اطلاعات (Indexها)
- مشخصات سیستمی روال‌های ذخیره شده (Stored Procedure)
- مشخصات سیستمی تراکنش‌های بانک اطلاعات
- مشخصات سیستمی تاریخ ایجاد و بروزرسانی جداول داده‌ای
- مشخصات سیستمی کاربران و چگونگی حق دستیابی آنها به داده‌ها و محدوده مجاز عملیات آنها
- مشخصات سیستمی پایانه‌های متصل به بانک

توجه: در برخی کتب غیرمرجع، دیکشنری داده‌ها زیرمجموعه کاتالوگ سیستم در نظر گرفته شده است. اما در کتب مرجع اصلی همچون آبراهام سیلبرشاتز صفحه ۴۶۲ تا ۴۶۸ راما کریشنان و رامزالمصری، دیکشنری داده‌ها معادل کاتالوگ سیستم در نظر گرفته شده است.

استقلال داده‌ای

یکی از مهم‌ترین مزایای تکنولوژی پایگاه داده‌ها (مدل مفهومی پایگاه داده)، بلکه مهمترین هدف آن تأمین و افزایش استقلال داده‌ای است، به معنی وابسته نبودن برنامه‌های کاربردی به داده‌های

ذخیره شده.
استقلال داده‌ای بر دو نوع می‌باشد:

۱- استقلال فیزیکی داده‌ها

به معنی مصونیت برنامه های کاربردی در قبال تغییراتی که در سطح فیزیکی (رسانه ذخیره سازی) پایگاه داده‌ها بروز می‌کند. یعنی اگر تغییری در ذخیره‌سازی داده‌ها انجام گیرد (برای مثال نوع دیسک عوض شود) برنامه‌های کاربردی هیچ تغییری نکند.

۲- استقلال منطقی داده‌ها

به معنی مصونیت برنامه‌های کاربردی در قبال تعاریف و تغییراتی که در سطح مدل طراحی (مدل رابطه‌ای) پایگاه داده بروز می‌کند. یعنی تعریف و تغییر مدل طراحی بانک (ادراکی خاص یا طراحی منطقی) از دید برنامه‌های کاربردی آنها مخفی بماند.
برای مثال مدل رابطه‌ای از تجریدی به نام جدول استفاده می‌کند و داده‌ها هر چه باشند در قالب چند جدول ریخته می‌شوند و نحوه ذخیره‌سازی داده‌ها روی رسانه‌ها از دید برنامه کاربردی مخفی است. در حالی که در روش فایلینگ تعاریف مربوط به فایل های داده‌ای، در فایل برنامه کاربردی می‌آمد. از آنجاکه برنامه‌های کاربردی براساس مدل طراحی بانک (ادراکی خاص یا طراحی منطقی) تعریف می‌شوند، بنابراین به طور بالقوه در معرض تأثیرپذیری از **تغییرات** در مدل طراحی بانک (ادراکی خاص یا طراحی منطقی) قرار دارند.
توجه: در سیستم‌های امروزی، این نوع استقلال هم تا حدی (و نه صددرصد) تأمین شده است.

انواع تغییر در مدل طراحی (طراحی منطقی یا ادراکی خاص)

۱- رشد پایگاه داده ها به دلیل مطرح شدن نیازهای جدید مشتری: مانند درج جدول جدید، ترکیب جداول، تجزیه جداول.
۲- سازماندهی مجدد: مانند تغییر در نوع صفات خاصه، تغییر در اندازه صفات.
مثال: اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیز به آن اضافه گردد، در صورتی که برنامه کاربردی سابق نیاز به دستکاری و تغییر نداشته باشد، استقلال منطقی داده‌ها براساس تغییرات نیز لحاظ شده است.
توجه: از آن جا که با حذف جداول، داده ها هم از بین می رود، بنابراین برنامه های کاربردی نسبت به حذف جداول هیچگاه استقلال منطقی نخواهند داشت.

تراکنش

هر برنامه ای که در محیط بانک اطلاعاتی توسط کاربر اجرا گردد یک تراکنش نام دارد. (مانند عملیات کارت به کارت در یک دستگاه خودپرداز بانک) تراکنش واحد کار DBMS است. به طور

کلی هر عملیاتی در پایگاه داده در قالب یک تراکنش تعریف و اجرا می‌شود. هر تراکنش شامل دو یا چند دستور SQL است. تفاوت اصلی یک تراکنش با یک برنامه معمولی در محیط غیربانکی این است که تراکنش همواره به DBMS تسلیم می شود و DBMS در اعمال هر گونه کنترل و حتی به تعویق انداختن و ساقط کردن آن آزادی عمل دارد. هدف اصلی از اینگونه کنترل‌ها، حذف‌ها و تعویق‌ها، حفظ جامعیت داخلی و خارجی بانک اطلاعات است.

تضمین جامعیت داخلی و خارجی بانک اطلاعات

چهار کنترل زیر لازم است روی تمامی تراکنش ها در بانک اطلاعات اعمال گردد تا جامعیت داخلی و خارجی یعنی رعایت اصل سازگاری آن تضمین شود.

این کنترل‌ها به خواص ACID معروفند:

۱- یکپارچگی یا تجزیه‌ناپذیری (Atomicity)

این خاصیت به همه یا هیچ موسوم است. منظور این است که یا تمام دستورات یک تراکنش باید اجرا شود یا هیچکدام از آنها نباید اجرا شود. این به معنی تجزیه‌ناپذیر بودن بخش‌های مختلف یک تراکنش است.

برای مثال تراکنش انتقال پول از حساب A به حساب B از دو بخش جداگانه تشکیل یافته است:

الف) بخش اول (برداشت پول)

پول را از حساب A برداشت می‌کند.

ب) بخش دوم (واریز پول)

همان پول را به حساب B واریز می‌کند.

بخش اول حساب A را بدهکار و بخش دوم حساب B را بستانکار می‌کند. در شروع و پایان یک تراکنش سیستم باید سازگار باشد ولی در اثنای اجرای تراکنش ممکن است موقتاً نیاز به ناسازگاری باشد. برای مثال هنگام واریز پول از حساب A به B، پس از برداشت پول از حساب A سیستم به طور موقت ناسازگار است و پس از واریز آن به حساب B دوباره سیستم سازگار می‌شود. لذا برنامه برداشت از حساب A یا واریز به حساب B به تنهایی تراکنش نیستند.

این دو بخش ممکن است روی دو کامپیوتر جداگانه اجرا شوند. فرض کنید بخش اول تراکنش اجرا شود اما ناگهان ارتباط با ماشین دوم قطع گردد. و بخش دوم قابل انجام نباشد. بدیهی است که در این حالت باید پول برداشت شده دوباره به همان حساب اول بازگردانده شود تا جامعیت بانک اطلاعات حفظ شود. این عمل معادل این است که بگوییم هیچ دستورالعملی از تراکنش انجام نشده است.

به عنوان مثالی دیگر، هنگام خرید اینترنتی با کارت عضو شتاب ممکن است پول از حساب شما

کسر گردد، اما به حساب فروشگاه مورد نظر واریز نگردد، بنابراین خرید شما ناموفق اعلام می‌گردد، که در این حالت پول حداکثر تا ۴۸ ساعت دیگر به حساب شما بازمی‌گردد.
در بیانی دیگر تراکنش را می توان اینگونه تعریف کرد، تراکنش مجموعه ای از دستورات تعریف و دستکاری داده هاست که DBMS تضمین می کند یا همه آن دستورات اجرا شوند و یا هیچکدام از آن دستورات اجرا نشوند. برای محقق‌کردن خاصیت یکپارچگی (Atomicity) هر تراکنش می‌بایست بین دو دستور Begin Transaction و End Transaction قرار گیرد. به طور کلی هر عملیاتی در پایگاه داده در قالب یک تراکنش تعریف و اجرا می‌شود. پس تراکنش واحد کار DBMS است. هر تراکنش شامل دو یا چند دستور SQL است. بر این اساس می‌توان گفت که هیچ پرس و جویی برای پایگاه داده هویت مستقل ندارد، بلکه DBMS فقط تراکنش‌ها را می‌شناسد و اجرا می‌کند. به این ترتیب باید گفت که هر پرس و جویی در پایگاه داده ابتدا به یک تراکنش تبدیل می‌شود و سپس اجرا می‌گردد. در SQL برای نمایش ابتدای یک تراکنش از دستور Begin Transaction و برای نمایش خاتمه یک تراکنش از دستور End Transaction استفاده می‌شود. تراکنش با اجرای Begin Transaction شروع می‌گردد و در صورت اجرای موفق commit و در صورت عدم موفقیت یعنی عدم اجرای همه بخش‌های مختلف تراکنش با اجرای دستور Rollback خاتمه می‌یابد. با اجرای این دستور کلیه تغییراتی که تراکنش روی پایگاه داده اعمال نموده است، ابطال می‌شود و وضعیت پایگاه داده به آخرین وضعیت قبل از اجرای تراکنش برگردانده می‌شود.

۲- سازگاری (Consistency)

به طور کلی **جامعیت** در سیستم‌های بانکی به دو طبقه‌ی جامعیت داخلی و خارجی تقسیم می‌گردد. به حفظ قوانین مطرح شده از سوی مدل رابطه‌ای و DBMS در سطح پیاده‌سازی، جامعیت داخلی و به حفظ قوانین مطرح شده از سوی طراحان و برنامه‌نویسان بانک در سطح پیاده‌سازی، جامعیت خارجی گفته می‌شود. در صورتی که در یک بانک جامعیت داخلی و خارجی هر دو توام باهم برقرار باشد، در آن بانک، **اصل سازگاری** برقرار شده است. به عبارت دیگر سازگاری، به معنی رعایت قوانین داخلی و خارجی در بانک است. DBMS مسئول کنترل قوانین داخلی و خارجی در بانک است و هرگونه عاملی که باعث نقض قوانین داخلی و خارجی و به تبع سازگاری بانک گردد را رد می‌کند. قوانین داخلی بانک شامل قانون جامعیت درون رابطه‌ای، قانون جامعیت موجودیت، قانون جامعیت ارجاعی و قانون جامعیت دامنه‌ای است، این موارد در فصل مدل رابطه‌ای بررسی خواهد شد. قوانین خارجی بانک هم شامل هر قانونی است که طراحان و برنامه‌نویسان بانک آنرا وضع می‌کنند، مانند تعریف زیردامنه برای ورود اطلاعات، تعریف بازه 0 تا

20 برای نمرات، تعریف بازه 0 تا بینهایت برای حساب‌های بانکی به معنی عدم وجود موجودی منفی در حساب‌های بانکی...

در سیستم‌های بانکی کنترل جامعیت داخلی و خارجی به صورت خودکار توسط مکانیزم‌های موجود در DBMS انجام می‌گردد.

حال یکبار دیگر کد تعریف جدول SP را در نظر بگیرید:

```
Create Table SP
        (
        S# char (5),
        P# char (5),
        QTY numeric (10),
        Primary key (S#, P#),
        Foreign key (S#) References S(S#)
          on delete cascade
          on update cascade,
        Foreign key (P#) References P(P#)
          on delete cascade
          on update cascade,
        Check (QTY>1 AND QTY<1000)
        )
```

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (S#) References S(S#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #S در جدول S به ستون #S در جدول SP نیز اعمال گردد. بنابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

یا به طور مشابه، کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (P#) References P(P#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #P در جدول P به ستون #P در جدول SP نیز اعمال گردد. بنابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

Check (QTY>1 AND QTY<1000)

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه مقداردهی در ستون QTY از جدول SP در بازه 1 تا 1000 باشد، بنابراین **جامعیت خارجی** نقض نمی‌گردد.

توجه: همانطور که واضح است DBMS در حفظ جامعیت داخلی و خارجی به دقت نظارت دارد.

خاصیت سازگاری (Consistency) بیانگر این است که اگر یک تراکنش در محیط بانک اطلاعات انجام شود باید بانک اطلاعات را از حالتی سازگار به حالت سازگار دیگری منتقل کند.

به بیان دیگر هر تراکنش باید تمامی قوانین جامعیت داخلی و خارجی بانک اطلاعات را رعایت کند. خاصیت سازگاری می‌گوید انجام تراکنشی از سوی DBMS باید پذیرفته شود که جامعیت داخلی و خارجی به معنی حفظ قوانین داخلی و خارجی پایگاه داده را رعایت کند، یعنی پس از انجام تراکنش‌ها اصل سازگاری برقرار باشد، در غیراینصورت انجام تراکنش رد شود.

بنابراین تا به اینجا مشاهده شد که تراکنش ممکن است دو نوع پایان داشته باشد:

الف) پایان موفق که آن را انجام (commit) می‌نامند.

ب) پایان ناموفق که آن را سقوط (abort) می‌نامند.

۳- انزوا یا جداسازی (Isolation)

همزمانی در دسترسی به داده‌ها موجب بهبود کارایی و کاهش زمان پاسخ‌گویی سیستم می‌گردد. و این امر تسریع عملکرد برنامه‌ها را در پی دارد. بسیاری از سیستم‌ها اجازه می‌دهند که چندین کاربر به صورت همروند به داده ها دسترسی یابند و تغییرات مورد نظر خود را بر روی آنها اعمال نمایند. در چنین محیط هایی تغییرات همروند ایجاد شده بر روی داده ممکن است منجر به ایجاد ناسازگاری در داده‌ها گردد.

در سیستم‌های بانکی کاربران مختلف می‌توانند به صورت همزمان با بانک کار کنند. بنابراین اگر یک داده **خاص** بین کاربران مختلف به صورت اشتراکی مورد بازیابی و دستکاری قرار گرفت سیستم پایگاه داده باید محیطی را ایجاد نماید که مانع از بروز مشکلات و یا ایجاد نتایج نامطلوب گردد، مانند مطالب مربوط به ناحیه بحرانی در فرآیندهای همروند در درس سیستم عامل.

مثال : فرض کنید کاربر A و B به ترتیب در تهران و شیراز همزمان قصد برداشت وجه از حساب آقای 6037 از طریق برگ چک را دارند. بنابراین روال‌های زیر را خواهیم داشت، از آنجا که برداشت وجه از یک رکورد مشترک (عامل مشترک) صورت می‌گیرد، به تبع وقوع پدیده همزمانی برای هر دو تراکنش رخ می دهد. روال کار بدین صورت است که هر دو تراکنش مبلغ موجودی حساب که برابر مقدار 500 هزارتومان می‌باشد را خوانده و با توجه به مبالغ برگ چک A و B به ترتیب مبالغ 100 و 50 هزارتومان را از حساب کسر می کنند و بر حسب اینکه کدام تراکنش آخرین بروزرسانی را انجام دهد مبلغ مانده حساب 400 تا 450 هزارتومان ذخیره می‌گردد. که در

هر دو صورت اطلاعات نادرستی ذخیره شده است. در حالی که مبلغ 350 هزار تومان باید جهت مبلغ مانده حساب ذخیره می‌شد!

شماره مشتری	شماره حساب	نام	نام‌خانوادگی	موجودی
6037	0313	Ali	Alavi	500

Read (A)	Read (B)
A ← 500	B ← 500
R=500-100	R=500-50
R ← 400	R ← 450
Write (R) ← 400	Write (R) ← 450

بدین‌ترتیب مقادیر نادرستی توسط برنامه‌ها ذخیره شده است که این نادرستی به دلیل تداخل عملیات دو برنامه همروند است. برای جلوگیری از ایجاد چنین نتایج نامطلوبی سیستم باید نوعی نظارت بر عملکرد برنامه‌های همروند داشته باشد. مانند روش قفل‌گذاری.

در بانک اطلاعات ممکن است تراکنش‌های همروند وجود داشته باشد (مثل سیستم‌های چند برنامه‌ای) بر طبق خاصیت انزوا همروندی تراکنش‌ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند به بیان دیگر اثر تراکنش های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا انجام می‌شود.

به تعریفی دیگر تراکنش‌ها جدا از یکدیگر هستند. اگر چند تراکنش به طور همزمان اجرا شوند، به هنگام‌سازی‌های هر کدام از یکدیگر مخفی می‌مانند تا به اتمام برسند. به عبارتی دیگر، برای دو تراکنش مجزای A و B، تراکنش A می‌تواند بهنگام سازی‌های B را پس از پذیرفته شدن آن (commit) یا B می‌تواند بهنگام سازی‌های A را پس از پذیرفته شدن A ببیند. اما این دو تراکنش به طور همزمان نمی‌توانند، بهنگام‌سازی‌های یکدیگر ببینند.

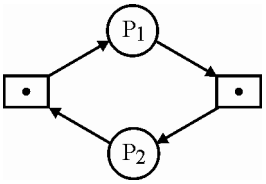
توجه: کنترل هم‌روندی توسط بخشی از DBMS به نام واحد کنترل همروندی (concurrency control) انجام می‌شود.

روش قفل‌گذاری

یکی از روش‌های اعمال خاصیت جداسازی در تراکنش‌ها و جلوگیری از اثر مخرب تراکنش‌های همروند بر روی یکدیگر، روش قفل‌گذاری است. در این روش هنگامی که تراکنش به داده‌ای نیاز داشته باشد. تقاضای قفل کردن آن را می‌دهد و در این حالت بقیه تراکنش‌ها تا اتمام کار آن نمی‌توانند از آن استفاده کنند. در واقع استفاده از انواع قفل‌ها در مکانیسم قفل‌گذاری، روش‌های دسترسی به فیلدها و یا داده‌های اشتراکی را در کاربردهای دیگر، امکان‌پذیر می‌سازد. وجود قفل‌ها سبب می‌شود که در اجرای همروند چند تراکنش، یک رکورد مشترک (عامل مشترک) به صورت

همزمان توسط دو تراکنش مورد استفاده قرار نگیرد. این کار سرعت عملیات را افزایش می‌دهد زیرا داده اشتراکی به صورت انحصاری فقط در اختیار یک تراکنش است. ولی احتمال بروز بن‌بست را به دلیل برقراری شرط انحصار متقابل زیاد می‌کند.

بر اساس قاعده کافمن شرایط وقوع بن‌بست به صورت زیر است:



۱ـ انحصار متقابل

۲ـ انحصاری بودن

۳ـ نگهداری و انتظار

۴ـ انتظار چرخشی

۴- پایائی یا ماندگاری (Durabiltiy)

براساس این خاصیت تراکنش هایی که به مرحله انجام (commit) برسند اثرشان ماندنی است و هرگز به طور تصادفی از بین نمی‌روند. برای مثال اگر مبلغی به حسابی واریز شود و تراکنش مربوطه انجام یافته(commit) اعلام شود حتی در صورت وقوع حادثه در آن شعبه بانک، مشتری نباید متضرر شود، برای مثال عمل واریز قبل از اعلام انجام موفق (commit) باید در جای دیگر نیز ثبت شده باشد، مثل دیسک اصلی یا دیسک پشتیبان. یعنی تأثیرات تراکنش در پایگاه داده، ماندگار باشد. هنگامی‌که یک تراکنش دستور commit را اجرا می‌کند نتایج اجرای آن به نسخه‌ی اصلی پایگاه داده در دیسک منتقل می‌شود. بنابراین می‌توان گفت تا زمانی‌که یک تراکنش دستور commit را اجرا نکرده است یا به اصطلاح تثبیت نشده است، ویژگی ماندگاری در مورد آن تراکنش تضمین شده نیست. اما پس از اجرای دستور ماندگاری نتایج تراکنش تضمین می‌شود.

توجه: دو خاصیت یکپارچگی و پایایی توسط واحدی از DBMS به نام واحد مدیریت بازگرد (Recovery management) کنترل می‌گردد.

معماری‌های بانک اطلاعات

منظور از معماری بانک اطلاعات، پیکربندی اجزای سیستمی است که در آن حداقل یک بانک اطلاعات، یک سیستم مدیریت بانک اطلاعات (DBMS)، یک سیستم عامل، یک کامپیوتر با دستگاه های جانبی و تعدادی کاربر (کاربران نهایی، مدیران و برنامه سازان) وجود دارد و خدماتی به کاربران نهایی ارائه می‌کند. نوع معماری به عواملی همچون سخت‌افزار، نرم‌افزار مدیریت بانک اطلاعات، موقعیت جغرافیایی کاربران، نیازهای کاربران ماهیت تراکنش‌ها، حجم داده‌ها و موقعیت مکانی داده‌ها و ارتباطات بین آنها بستگی دارد.

به طور کلی دو نوع معماری برای سیستم‌های بانکی وجود دارد :

۱- معماری متمرکز

۲- معماری نامتمرکز
- معماری مشتری ـ سرویس دهنده
- معماری توزیع

معماری متمرکز

در این معماری یک پایگاه داده روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر ایجاد می‌شود. و برای کاربردهای کوچک و با امکانات محدود است. سخت‌افزار این سیستم می‌تواند در حد یک کامپیوتر شخصی و یا بالاتر باشد. (مانند برنامه دفترچه تلفن شخصی)

معماری نامتمرکز

معماری مشتری - سرویس‌دهنده

هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگری از پردازش را از برنامه، سیستم یا ماشین دیگری بخواهد، معماری مشتری - سرویس‌دهنده نامیده می‌شود.

در واقع وظایفی که باید سیستم انجام دهد به دو رده تقسیم می شوند: رده‌ای که انجام آنها بر عهده سرویس دهنده است و رده‌ای که توسط مشتری انجام می شود. بدین ترتیب یک معماری دو سطحی داریم که مشکل توزیع را تسهیل می کند. با این تعریف، در این معماری یک ماشین (یا سیستم یا برنامه) سرویسی را به ماشین (یا سیستم یا برنامه) دیگر ارائه می کند. بنابراین به این ماشین (یا سیستم یا برنامه) سرویس دهنده می گویند. خدماتی که سرویس‌دهنده ارائه می کند متنوع است، برای مثال خدمات چاپ، خدمات فایل، خدمات پست الکترونیک، خدمات اینترنت، خدمات بانک اطلاعات و بسته به نوع خدمت، سرویس‌دهنده را با نامی مناسب می‌نامند. برای نمونه سرویس‌دهنده چاپ و سرویس‌دهنده فایل، بنابراین منظور از معماری مشتری - سرویس‌دهنده آن نوع از معماری است که در آن مسئولیت‌ها به طور منطقی تقسیم شده است.

انواع معماری مشتری ـ سرویس‌دهنده از نظر تعداد مشتری و سرویس‌دهنده:

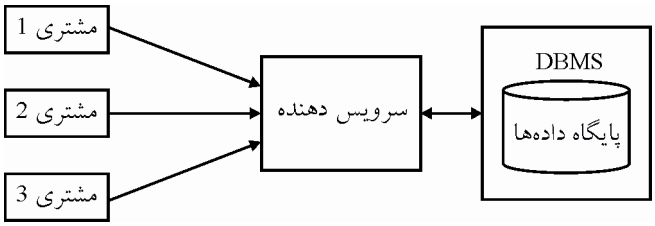
۱ـ چند مشتری ـ یک سرویس‌دهنده
۲ـ یک مشتری ـ چند سرویس‌دهنده
۳ـ چند مشتری ـچند سرویس‌دهنده

انواع معماری مشتری ـ سرویس‌دهنده از نظر پیکربندی سخت‌افزاری

معماری حول سرویس‌دهنده

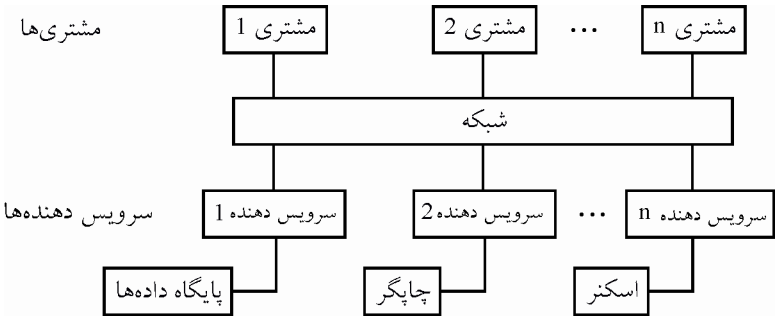
در این طرح، ماشین سرویس‌دهنده یک ابر کامپیوتر است و پایگاه داده ها روی همین کامپیوتر

ایجاد و مدیریت می‌شود و تعدادی مشتری، از طریق شبکه از خدمات آن استفاده می‌کنند.



معماری حول شبکه

در این طرح، تعدادی کامپیوتر شخصی به عنوان سرویس‌دهنده و تعدادی دیگر به عنوان مشتری از طریق شبکه بهم مرتبط هستند.



معماری توزیع شده

اصطلاح پردازش توزیعی بدین مفهوم است که ماشین های مجزا می توانند در یک شبکه ارتباطی به یکدیگر متصل شوند. به گونه ای که یک عملکرد پردازش داده منفرد می تواند بر روی چندین ماشین شبکه پراکنده شود. در سیستم پایگاه های توزیع شده، پایگاه داده ها بر روی چند کامپیوتر ذخیره می شود. کامپیوترها در یک سیستم توزیع شده از طریق رسانه های مختلف ارتباطی نظیر شبکه‌های با سرعت بالا یا خطوط تلفن به یکدیگر مرتبط هستند.

می توان گفت که در این معماری تعدادی پایگاه داده های ذخیره شده روی کامپیوترهای مختلف داریم که از نظر کاربران، پایگاه داده واحدی هستند.

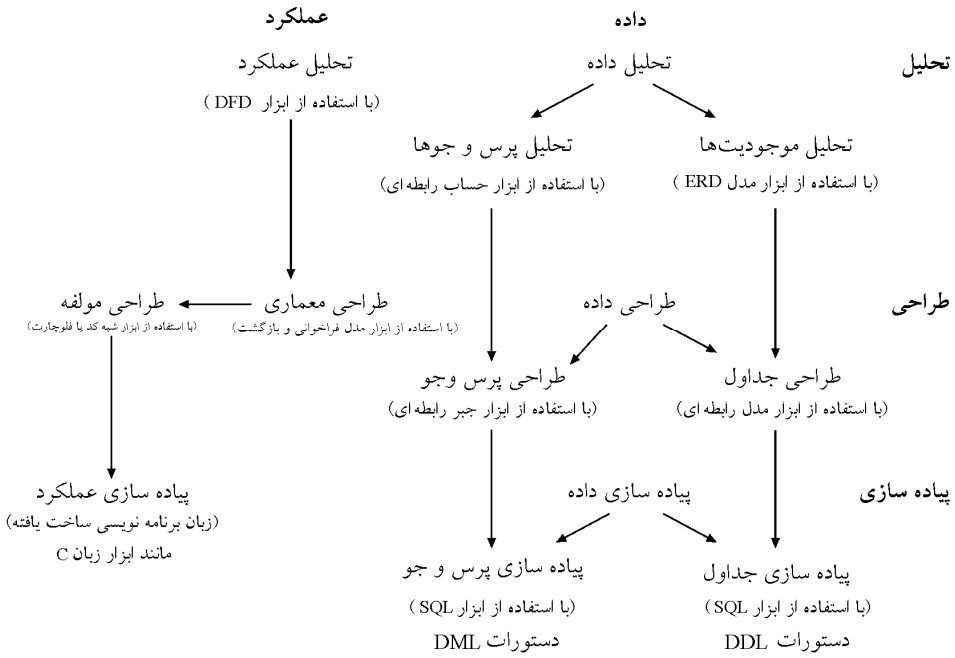
۲ مدل نهاد و رابطه

مقدمه

یک محصول نرم‌افزاری به واسطه‌ی فرآیند تولید نرم‌افزار که شامل فعالیت‌های مدل تحلیل، مدل طراحی، پیاده‌سازی و تست می‌باشد، ایجاد می‌گردد.

مدل‌سازی

یک مدل، ساده شده یک واقعیت است. ایجاد یک مدل برای سیستم‌های نرم‌افزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری وحیاتی است. بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که باید ساخته شوند را ترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدل‌ها و توصیف‌ها ساخته شوند. مدل‌های خوب و دقیق در برقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می‌توانند داشته باشند. علت اصلی مدل کردن سیستم‌های پیچیده این است که نمی‌توان به یکباره کل سیستم را تجسم کرد و ممکن است سیستم دارای ابهامات بسیاری باشد. لذا برای رفع این ابهامات و نیز برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمت‌های مختلف آن، از مدل‌سازی استفاده می‌شود. مدل‌سازی خود شامل دو مرحله‌ی مدل تحلیل و مدل طراحی می‌باشد. مدل تحلیل قبل از مدل طراحی انجام می‌شود، در واقع خروجی مدل تحلیل، ورودی مدل طراحی می‌باشد. شکل زیر گویای این مطلب می‌باشد:



توجه: از آنجا که ما قصد داریم در این کتاب مفاهیم مربوط به پایگاه داده را تشریح نماییم، بنابراین از بیان مطلب مربوط به بخش عملکرد نرم‌افزار صرف‌نظر می‌نمائیم. بخش عملکرد را در کتاب مهندسی نرم‌افزار مورد بحث و بررسی قرار داده‌ایم. پس در ادامه به طور مفصل به مفاهیم مربوط به پایگاه داده می‌پردازیم.

مدل تحلیل

تحلیل داده شامل تحلیل موجودیت‌ها و تحلیل پرس و جوها می‌باشد. تحلیل موجودیت‌ها توسط ابزار مدل ER و تحلیل پرس و جوها توسط ابزار حساب رابطه‌ای مدل می‌شوند.

مدل ER[1]

برای شناسایی هر مدل یا ساختاری ابتدا باید به بررسی بخش‌های مختلف آن مدل یا ساختار بپردازیم.

در مدل ER، سه مفهوم معنایی وجود دارد و معنای داده‌های هر محیط عملیاتی (محیطی که می‌خواهیم در مورد آن اطلاع کسب کنیم) به کمک همین سه مفهوم نمایش داده می‌شود:

۱- موجودیت یا نهاد (Entity)

۲- صفت (Attribute)

[1] Entity Relationship

۳- ارتباط یا رابطه (Relationship)

توجه: در مدل تحلیل منظور از رابطه، ارتباط بین موجودیت‌هاست، اما در مدل طراحی منظور از رابطه، جداول مدل رابطه‌ای می‌باشد.

۱- نهاد (موجودیت)

موجودیت عبارتند از مفهوم کلی «شیء»، «چیز»، «پدیده» و به طور کلی هر آنچه که می‌خواهیم در موردش «اطلاع» داشته باشیم.

مثال: در محیط عملیاتی دانشگاه، انواع موجودیت‌های دانشجو، درس، استاد، گروه آموزشی و غیره وجود دارد.

توجه: در کتب مختلف گاهاً به نهاد، مجموعه نهادی (Entity Set) نیز گفته می‌شود.

توجه: خروجی مدل‌سازی یک سیستم به کمک مدل ER، نموداری است که آن را ERD می‌نامند.

توجه: برای نمایش یک موجودیت (مجموعه نهادی) در نمودار ER از یک مستطیل نام‌دار استفاده می‌گردد.

مثال: موجودیت‌های درس و دانشجو در محیط عملیاتی دانشگاه.

درس	دانشجو

توجه: توصیه می‌شود در تشخیص مجموعه‌های نهادی و تحلیل سیستم به دنبال اسامی عام بگردیم. معمولاً اسامی عام یک موجودیت یا مجموعه نهادی را نشان می‌دهند (مانند دانشجو) و اسامی خاص اعضای یک مجموعه نهادی را نشان می‌دهند (مانند دانشجو احمدی)

توجه: از آنجا که نمادهای گرافیکی نمی‌توانند تمامی جزئیات مربوط به موجودیت‌ها یا اشیاء داده‌ای را بیان کنند، جهت تشریح شرح حال موجودیت‌ها یا اشیاء داده‌ای از مستنداتی به نام شرح حال موجودیت‌ها یا اشیاء داده‌ای یا فرهنگ داده‌ها (دیکشنری داده) استفاده می‌شود. فرهنگ داده‌ها، به بیان جزئیات مربوط به موجودیت‌ها یا اشیاء داده‌ای مورد نظر می‌پردازد.

برای‌مثال بر روی نماد مستطیل شکل موجودیت ها اسم برده می‌شود، اما این‌که دقیقاً این موجودیت‌ها یا اشیاء داده‌ای چه هستند مشخص نمی‌شود. برای این منظور از فرهنگ داده‌ها استفاده می‌شود که موجودیت‌ها یا اشیاء داده‌ای را به صورت دقیق تشریح می‌کند.

۲- صفت

صفت در واقع خصیصه یا ویژگی یک نوع موجودیت است و هر نوع موجودیت مجموعه‌ای از صفات (موسوم به مجموعه صفات موجودیت) را دارد. هر صفت از نظر کاربران یک نام، یک نوع و یک معنای مشخص دارد. به عنوان مثال، موجودیت درس را در نظر بگیرید. صفات درس

عبارتند از: شماره درس، عنوان درس، تعداد واحد درس، نوع درس (پایه، تخصصی، اختیاری و غیره)، ماهیت درس (نظری، عملی و غیره)، سطح درس (کاردانی، کارشناسی و غیره).
توجه: هر صفتی از یک مجموعه مقادیر معتبر و مجاز مقدار می‌گیرد که به این مجموعه مقادیر، اصطلاحاً دامنه یا میدان (Domain) مقادیر آن صفت می‌گویند.

انواع صفات در مدل ER

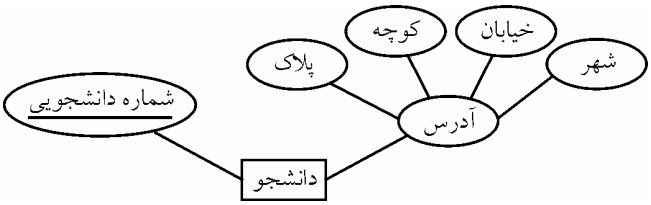
هر موجودیت دارای مجموعه‌ای از صفات است که در نمودار ER با بیضی نشان داده می‌شود.

صفت کلید

کلید عبارتند از یک یا چند صفت خاصه که در یک موجودیت منحصر به فرد باشد. به عبارت دقیق‌تر هر ترکیبی از صفت یا صفات که اگر بر اساس آن جستجو انجام گردد فقط و فقط یک مقدار را بازگرداند. بر اساس کدملی جستجو انجام دهید، واضح است که فقط و فقط یک مقدار را باز می‌گرداند. به عنوان مثالی دیگر در موجودیت دانشجو، شماره دانشجویی کلید است، چون هر دانشجو شماره‌ای یکتا دارد. ولی نام نمی‌تواند کلید باشد.
توجه: گاهی یک صفت به تنهایی نمی‌تواند کلید باشد بلکه مجموعه‌ای از دو یا چند صفت، تشکیل کلید را می‌دهند.
توجه: صفت کلید نمی‌تواند مقدار NULL بگیرد.
توجه: در نمودار ER زیر صفت یا صفات کلیدی یک خط می‌کشند.

صفت ساده و مرکب

صفت ساده صفتی است که مقدار آن از لحاظ معنایی ساده یا اتومیک یا تجزیه نشدنی باشد، به این معنا که اگر مقدار آن را به اجزایی تجزیه کنیم، مقادیر جزیی حاصله فاقد معنا باشد. برای‌مثال صفت درس و شماره دانشجویی یک صفت ساده است. صفت مرکب صفتی است که از چند صفت ساده تشکیل شده باشد به گونه‌ای که تجزیه شدنی باشند و اجزاء حاصله خود صفات ساده باشند. مانند صفت آدرس که از اجزاء نام استان، نام شهر، نام خیابان، نام کوچه، شماره پلاک و کدپستی تشکیل شده است.
توجه: برای نشان دادن صفت مرکب در نمودار ER اجزای صفت مرکب خود به عنوان صفت برای صفت مذکور نشان داده می‌شود.



توجه: در بانک اطلاعاتی مبتنی بر مدل رابطه‌ای (جدولی) صفت مرکب نداریم.

صفت تک مقداری و چند مقداری

بعضی از صفات چه ساده و چه مرکب فقط می‌توانند یک مقدار را بگیرند که به این صفات، صفت تک‌مقداری می‌گویند. مانند شماره دانشجویی که نمی‌تواند بیش از یک مقدار داشته باشد. این صفات در نمودار ER بصورت معمول نمایش داده می‌شوند.

صفاتی وجود دارند که می‌توانند چندین مقدار را بگیرند مانند صفت مدرک در موجودیت استاد که می‌تواند مقادیر لیسانس، فوق لیسانس و یا دکتری را در خود بگیرد.

مثال:

توجه: به مثال‌های زیر توجه کنید.

صفت ساده تک‌مقداری: مانند کدملی

صفت ساده چندمقداری: مانند مدرک تحصیلی

صفت مرکب تک‌مقداری: مانند تاریخ تولد

صفت مرکب چندمقداری: مانند آدرس

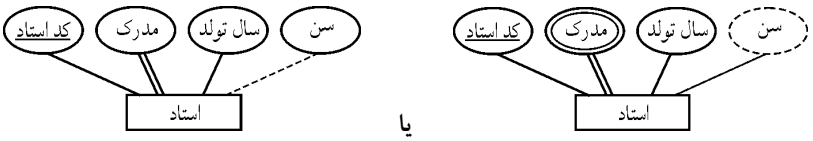
توجه: در بانک اطلاعاتی مبتنی بر مدل رابطه‌ای (جدولی) صفت چندمقداری نداریم.

صفت مشتق (پویا)

صفتی است که در موجودیت وجود خارجی ندارد ولی در صورت لزوم می‌توان آنرا بدست آورد. صفتی که مقادیر آن مدام در حال تغییر و تحول باشد، صفت پویا یا مشتق محسوب می‌گردد. بنابراین به دلیل تغییرات مداوم، توصیه می‌گردد صفت پویا در جداول بانک اطلاعات مورد استفاده قرار نگیرد و مقدار آن از طریق صفت مرتبط با آن محاسبه گردد. برای‌مثال برای محاسبه صفت سن، می‌توان صفت تاریخ تولد را در نظر گرفت و از روی این صفت، سن را محاسبه نمود.

توجه: صفت مشتق را در نمودار ER با نقطه چین به موجودیت مورد نظر متصل می‌کنند.

مثال:



۳- رابطه (ارتباط)

ارتباط، وابستگی بین دو مجموعه نهادی (موجودیت) را نشان می‌دهد.

توجه: در هنگام استخراج رابطه‌ها در مدل تحلیل، به دنبال افعال می‌گردیم.

توجه: برای نمایش یک رابطه در ER از یک لوزی استفاده می‌کنیم.

مثال:

در نمودار ER فوق، رابطه تهیه، مابین موجودیت تولیدکننده و قطعه برقرار است.

خواص رابطه

بطور کلی خواص رابطه به سه شکل زیر بررسی می‌گردد:

الف) درجه ارتباط

ب) کاردینالیتی ارتباط

ج) اجباری و اختیاری بودن ارتباط

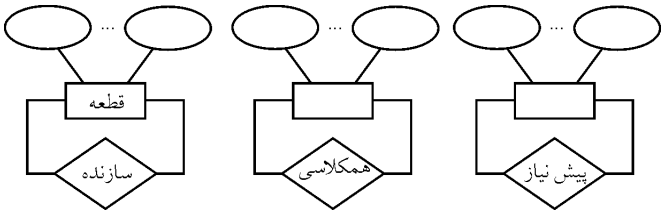
توجه: همه روابط سه خصیصه فوق را به طور همزمان دارند، اما مقادیر این خصیصه‌ها در روابط مختلف، متفاوت است.

الف) درجه ارتباط

به تعداد موجودیت‌هایی که در یک رابطه مشارکت دارند، درجه ارتباط گفته می‌شود. درجه در مدل ER عددی صحیح و کوچکتر از 5 است. ارتباط‌های درجه 1، 2 و 3 معمول، ارتباط درجه 4 کمیاب و غیر معمول است و ارتباط بالاتر از درجه 4 قابل رسم کردن نیست.

ارتباط درجه 1

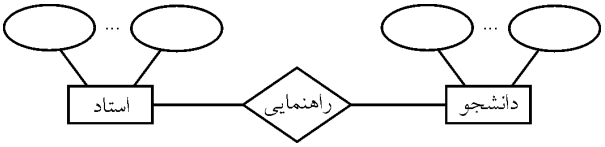
مثال:



فقط یک موجودیت در هر یک از شکل‌های فوق وجود دارد. دقت‌کنید که موجودیت یک نوع است و عضوهای آن، یک مجموعه را تشکیل می‌دهند. بنابراین در ارتباط یکتایی، عضوهایی از یک مجموعه در ارتباط با عضوهایی دیگر از همان مجموعه قرار می‌گیرند. برای‌مثال درسی پیش

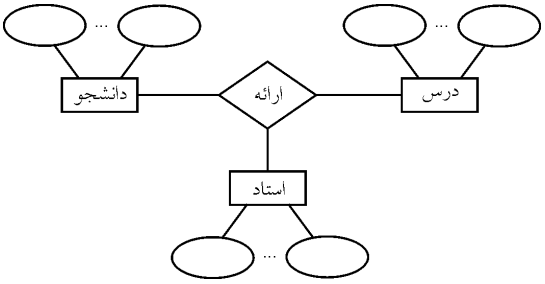
نیاز درس دیگر است، یا دانشجویی همکلاسی دانشجوی دیگری است، در حالیکه همه دورس به یک موجودیت و همه دانشجویان نیز به یک موجودیت تعلق دارند.
توجه: وقتی یک ارتباط بین یک نوع موجودیت و خودش برقرار باشد، آنرا ارتباط با خود (Self-Relationship) یا بازگشتی (Recursive) می‌گویند.

ارتباط درجه 2
مثال:



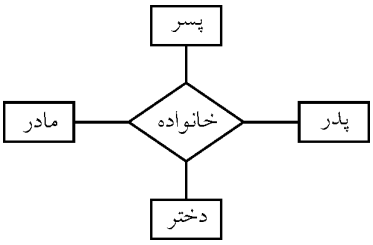
در این ارتباط دو موجودیت استاد و دانشجو مشارکت دارند.

ارتباط درجه 3
مثال:



در این ارتباط سه موجودیت استاد، درس و دانشجو مشارکت دارند.

ارتباط درجه 4
مثال:

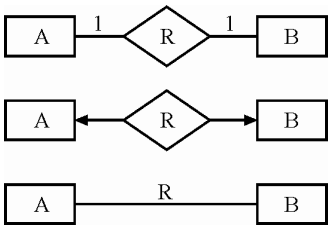


در این ارتباط چهار موجودیت پدر، مادر، پسر و دختر مشارکت دارند.

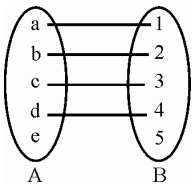
ب) کاردینالیتی ارتباط

کاردینالیتی ارتباط بر سه نوع است: یک به یک، یک به چند، چند به چند.

۱- یک به یک (1-1)



یعنی هر نمونه موجودیت از A با حداکثر با یک نمونه موجودیت از B ارتباط دارد و هر نمونه موجودیت از B با حداکثر یک نمونه موجودیت از A ارتباط دارد.



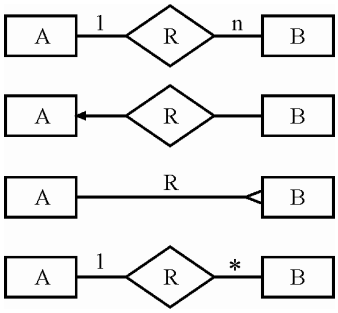
ارتباط یک به یک

مثال:

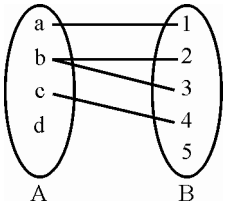
بدین معنی که هر استاد حداکثر یک درس ارائه می‌کند و هر درس حداکثر یک استاد برای ارائه دارد.

توجه: یعنی ممکن است استادی اصلاً درس نداشته باشد و یا درسی توسط هیچ استادی در این ترم ارائه نگردد.

۲- یک به چند (1:n)



یعنی هر نمونه موجودیت از A با صفر یا بیشتر نمونـه موجودیـت از B ارتبـاط دارد و هـر نمونـه موجودیت از B با حداکثر یک نمونه موجودیت از A ارتباط دارد.



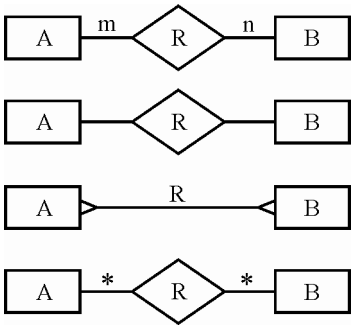
ارتباط یک به چند

مثال:

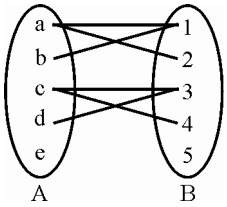
بدین معنی که هر استاد صفر یا بیشتر درس ارائه می‌کند و هر درس حداکثر یک استاد بـرای ارائـه دارد.

توجه: یعنی ممکن است استادی اصلاً درس نداشته باشد و یا درسی توسط هـیچ اسـتادی در ایـن ترم ارائه نگردد.

۳- چند به چند (m:n)



یعنی هر نمونه موجودیت از A با صفر یا بیشتر نمونـه موجودیـت از B ارتبـاط دارد و هـر نمونـه موجودیت از B با صفر یا بیشتر نمونه موجودیت از A ارتباط دارد.



مثال:

بدین معنی که هر استاد صفر یا بیشتر درس ارائه می‌کند و هر درس صفر یا بیشتر استاد برای ارائه دارد.

توجه: یعنی ممکن است استادی اصلاً درس نداشته باشد و یا درسی توسط هیچ استادی در این ترم ارائه نگردد.

توجه: در ارتباط، شرکت موجودیت‌ها در ارتباط به طور پیش‌فرض اختیاری است. برای‌مثال ممکن است استادی درسی ارائه نکند و یا درسی این ترم ارائه نشود. در ادامه ارتباط اجباری را مورد بررسی قرار خواهیم داد.

حد

مشخصه دیگر ارتباط، حد آن است که بیانگر حداقل و حداکثر تعداد نمونه موجودیت‌های شرکت‌کننده در ارتباط است. این مقادیر در پایین خط ارتباط، توسط پرانتز نشان داده می‌شود.

مثال:

در شکل بالا حد (0,10) نشان می‌دهد که یک استاد ممکن است راهنمای هیچ دانشجویی نباشد و یا حداکثر 10 دانشجو را راهنمایی کند. همچنین حد (0,1) نشان می‌دهد که یک دانشجو ممکن است استاد راهنما نداشته باشد و یا حداکثر توسط یک استاد راهنمایی شود.

ج) اجباری و اختیاری بودن ارتباط

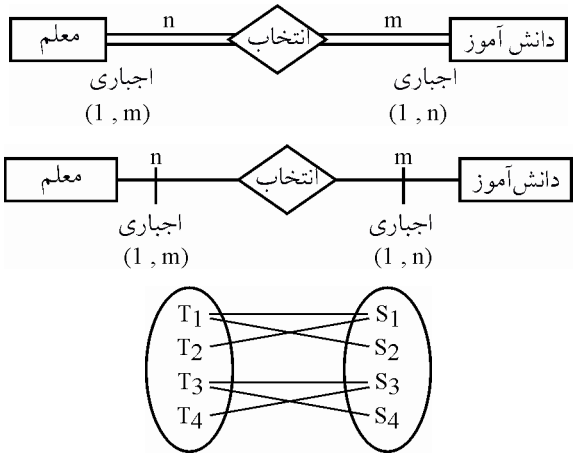
این نوع رابطه به دو دسته کلی زیر تقسیم می‌شود:

۱- اجباری یا کلی (Total)

یک رابطه اجباری است، اگر و تنها اگر تمام نمونه‌های موجودیت در رابطه شرکت کرده باشند. اجباری بودن در نمودار ER با نماد خط مضاعف افقی یا نماد | به معنی یک و الزام شرکت در رابطه نشان داده می‌شود.

توجه: نماد خط مضاعف افقی نشانه اجباری بودن موجودیت چسبیده به آن است، اما نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

مثال: در نظام آموزشی وزارت آموزش و پرورش حضور معلم و دانش آموز در محیط عملیاتی مدرسه اجباری است. زیرا مفاهیمی همچون مرخصی برای دانش‌آموز و یا پژوهش به‌جای تدریس برای معلم معنا ندارد.



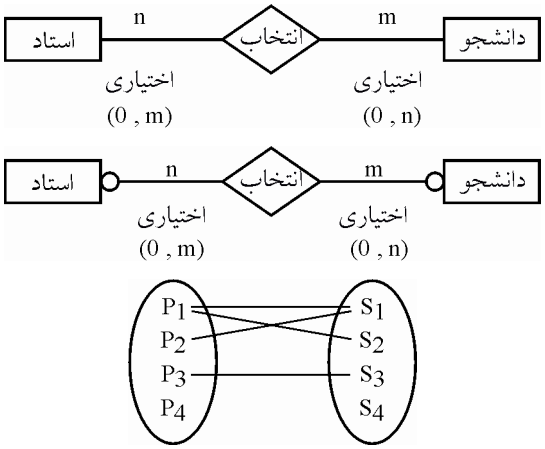
قید (1,m) نشان می‌دهد که یک معلم حداقل یک و حداکثر m دانش‌آموز دارد و قید (1 , n) نشان می‌دهد که یک دانش‌آموز حداقل یک و حداکثر n معلم دارد.

۲- اختیاری یا جزئی (Partial)

یک رابطه اختیاری است، اگر و تنها اگر حداقل یکی از نمونه‌های موجودیت در رابطه شرکت نکرده باشد. اختیاری بودن در نمودار ER با نماد خط افقی یا نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشان داده می‌شود.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است، اما نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

مثال: در نظام آموزشی وزارت علوم، تحقیقات و فناوری حضور استاد و دانشجو اختیاری است. زیرا مفاهیمی همچون مرخصی برای دانشجو و پژوهش به‌جای تدریس برای استاد معنا دارد.

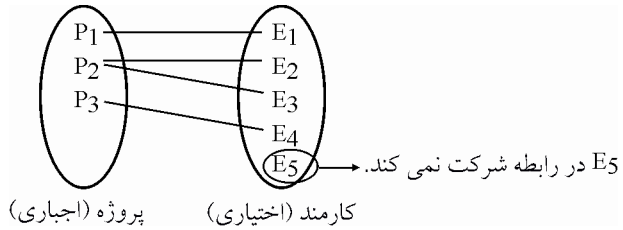


قید (m , 0) نشان می‌دهد که یک استاد حداقل هیچ و حداکثر m دانشجو دارد و قید (n , 0) نشان می‌دهد که یک دانشجو حداقل هیچ و حداکثر n استاد دارد.

توجه: از آنجا که حضور استاد اختیاری است، پس می‌تواند هیچ دانشجویی نداشته باشد. همین معنا برای دانشجو نیز صادق است.

مثال: نمودار ER زیر چه روابطی را نشان می‌دهد؟

شکل زیر گویای نمودار ER است.



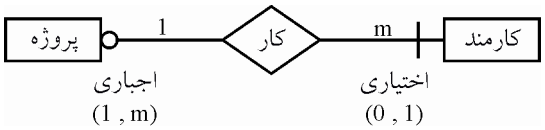
توجه: نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

توجه: نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

با توجه به ارتباط یک به چند پروژه با کارمند و اجباری بودن حضور نمونه موجودیت‌های پروژه، قید (1 , m)برای پروژه در نظر گرفته می‌شود، بدین معنی‌که، یک پروژه حداقل یک و حداکثر m کارمند دارد.

همچنین با توجه به ارتباط چند به یک کارمند با پروژه و اختیاری بودن حضور نمونه موجودیت‌های کارمند، قید (0,1) برای پروژه در نظر گرفته می‌شود، بدین معنی‌که، یک کارمند حداقل هیچ و حداکثر یک پروژه دارد.

شکل زیر گویای مطلب است:



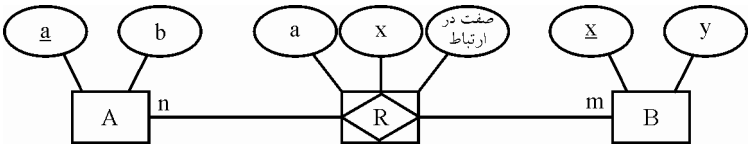
صفت در ارتباط

ارتباط‌ها نیز می‌توانند صفت داشته باشند (اغلب در ارتباطات n:m)

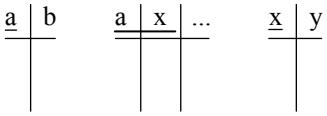
برای‌مثال صفت تعداد قطعات (QTY) در نمودار بانک اطلاعات تولیدکنندگان قطعات می‌تواند صفت در ارتباط، رابطه تهیه باشد.
شاید تصور شود که تعداد قطعات مربوط به موجودیت قطعه است. ولی این تصور غلط است زیرا یک تولیدکننده چند قطعه و یک قطعه نیز توسط چند تولیدکننده، تهیه می‌گردد. بنابراین صفت تعداد قطعات (QTY) را باید به ارتباط **تولید** که دو موجودیت قطعه و تولیدکننده را به هم مرتبط می‌کند نسبت داد.
توجه: چنین ارتباطاتی با یک لوزی درون یک مستطیل نشان داده می‌شود و کلید آنها کلیدهای همه موجودیت‌های مربوطه را شامل می‌شود.

نگاشت رابطه چند به چند بین دو موجودیت(مدل تحلیل) به مدل رابطه‌ای(مدل طراحی)
مستقل از اختیاری یا اجباری بودن موجودیت ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط‌دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد. روال کلی به‌صورت زیر است:

مدل تحلیل:



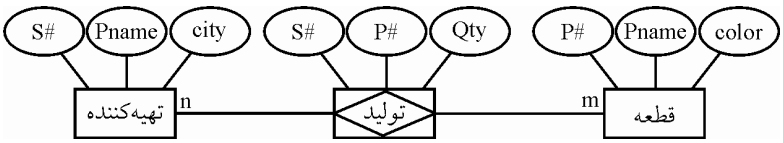
مدل طراحی:



جدول A جدول AB جدول B

مثال: ارتباط تولیدکنندگان و قطعات.

مدل تحلیل:



توجه: صفت QTY، به‌عنوان یک صفت در ارتباط، تعداد قطعات را مشخص می‌کند.

مدل طراحی:

S#	Sname	City
S_1	Sn1	C_1
S_2	Sn2	C_2
S_3	Sn3	C_2

جدول تولید کننده

S#	P#	QTY
S_1	P_1	10
S_1	P_2	20
S_2	P_1	30

جدول تولید

P#	Pname	Color
P_1	Pn1	Red
P_2	Pn2	Blue

جدول قطعه

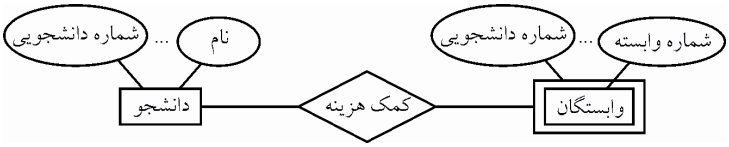
توجه: مابقی نگاشت‌ها از مدل تحلیل (مدل ER) به مدل طراحی (مدل رابطه‌ای) در فصل مدل رابطه‌ای بررسی می‌گردد.

وابستگی وجودی

اگر در یک بانک اطلاعاتی وجود یک موجودیت وابسته به موجودیت دیگری باشد که در صورت حذف و تغییر موجودیت اصلی این موجودیت نیز تغییر کند، این نوع وابستگی را وابستگی وجودی گفته و به پدیده وابسته، موجودیت ضعیف گویند. که موجودیت ضعیف کلید موجودیت اصلی را در بر دارد تا هرگونه تغییر یا حذف در موجودیت اصلی به موجودیت وابسته اعمال شود.

توجه: موجودیت وابسته با دو مستطیل تو در تو نمایش داده می‌شود.

مثال:



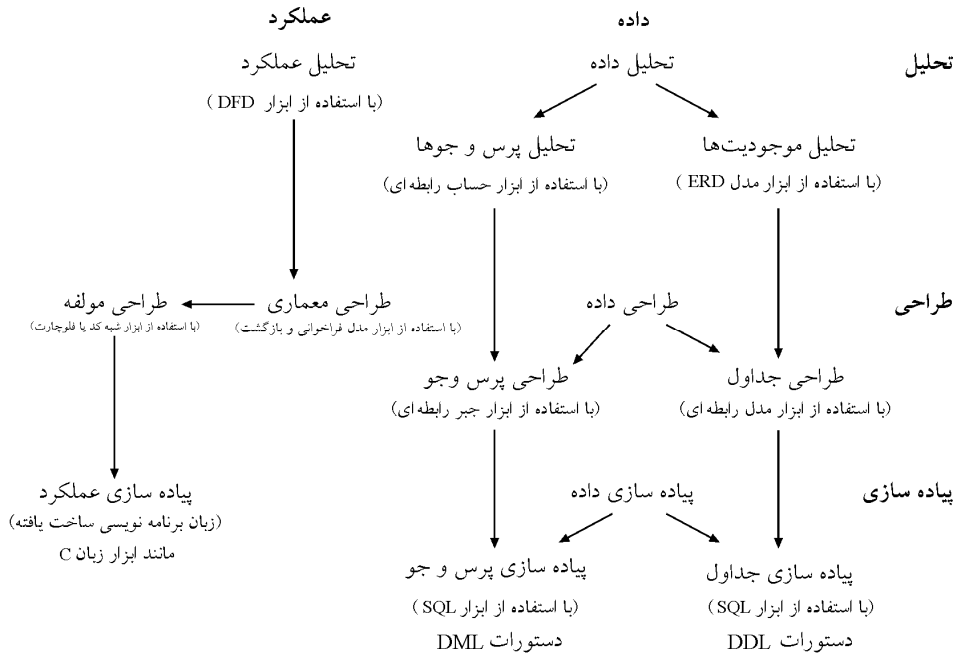
۳ مدل رابطه‌ای

مقدمه

یک محصول نرم‌افزاری به واسطه‌ی فرآیند تولید نرم‌افزار که شامل فعالیت‌های مدل تحلیل، مدل طراحی، پیاده‌سازی و تست می‌باشد، ایجاد می‌گردد.

مدل‌سازی

یک مدل، ساده شده یک واقعیت است. ایجاد یک مدل برای سیستم‌های نرم‌افزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری وحیاتی است. بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که باید ساخته شوند را ترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدل‌ها و توصیف‌ها ساخته شوند. مدل‌های خوب و دقیق در برقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می‌توانند داشته باشند. علت اصلی مدل کردن سیستم‌های پیچیده این است که نمی‌توان به یکباره کل سیستم را تجسم کرد و ممکن است سیستم دارای ابهامات بسیاری باشد. لذا برای رفع این ابهامات و نیز برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمت‌های مختلف آن، از مدل‌سازی استفاده می‌شود. مدل‌سازی خود شامل دو مرحله‌ی مدل تحلیل و مدل طراحی می‌باشد. مدل تحلیل قبل از مدل طراحی انجام می‌شود، در واقع خروجی مدل تحلیل، ورودی مدل طراحی می‌باشد. شکل زیر گویای این مطلب می‌باشد:



توجه: از آنجا که ما قصد داریم در این کتاب مفاهیم مربوط به پایگاه داده را تشریح نماییم، بنابراین از بیان مطلب مربوط به بخش عملکرد نرم‌افزار صرف‌نظر می‌نمائیم. بخش عملکرد را در کتاب مهندسی نرم‌افزار مورد بحث و بررسی قرار داده‌ایم. پس در ادامه به طور مفصل به مفاهیم مربوط به پایگاه داده می‌پردازیم.

مدل طراحی

پس از مدل تحلیل، نوبت به مدل طراحی می‌رسد، پس از مدل تحلیل، باید برای پیاده‌سازی نرم‌افزار آماده شد. واضح است که گذر مستقیم به مرحله تولید کد و پیاده‌سازی عاقلانه نیست.

مدل تحلیل، مدل‌سازی عالم خارج به زبانی شبیه انسان است، اما مدل طراحی مدل‌سازی عالم داخل ماشین به زبانی شبیه زبان ماشین است. بنابراین برای اینکه ساخت برنامه کامپیوتری که به زبان ماشین است، امکان‌پذیر باشد، باید مدل تحلیل به مدل طراحی نگاشت شود تا مدل طراحی بتواند به عنوان نقشه راهی شبیه به زبان ماشین، راهگشا باشد.

مدل‌های تحلیل، کلی‌تر و انتزاعی‌تر هستند، و برای مدل‌سازی عالم خارج مورد استفاده قرار می‌گیرند، بدون آنکه به جزئیات نحوه پیاده‌سازی بپردازند، در واقع، مدل تحلیل به کلی گویی به زبان انسان و حذف جزئیات نحوه پیاده‌سازی می‌پردازد. اما مدل‌های طراحی، جزئی‌تر و غیرانتزاعی‌تر هستند، و برای مدل‌سازی عالم داخل ماشین مورد استفاده قرار می‌گیرند، و به بیان

جزئیات نحوه پیاده‌سازی می‌پردازند، در واقع مدل طراحی به جزئی‌گویی به زبان شبیه ماشین و درج جزئیات نحوه پیاده‌سازی می‌پردازد.
با نگاهی سطح به سطح، به حل مساله، سطوح مختلفی از انتزاع را خواهیم داشت. در بالاترین سطح انتزاع، راه حل به صورت کلی به زبان محیط مساله بیان می‌گردد. در سطوح پایین‌تر، راه حل به جزئیات پیاده‌سازی نزدیک‌تر می‌شود و در پایین‌ترین سطح انتزاع راه حل به صورتی بیان می‌شود تا مستقیما قابل پیاده‌سازی باشد.
مدل طراحی مانند دوربین عکاسی می‌باشد که شرایطی را فراهم می‌آورد که تصویر عالم خارج را در عالم دوربین و نگاتیو ذخیره‌سازی نماید. هر چه در عالم خارج باشد، عینا، اما در ابعادی کوچکتر یا بزرگتر در دوربین ذخیره می‌شود، در این نگاشت چیزی جا نمی‌ماند، مدل تحلیل خوب به مدل طرحی خوب و مدل تحلیل بد به مدل طراحی بد نگاشت می‌شود، یک مدل طراحی خوب از یک مدل تحلیل خوب و یک مدل طراحی بد از یک مدل تحلیل بد نشات گرفته است.
اما این خیلی مهم است که بتوانید یک مدل تحلیل خوب را به یک مدل طراحی خوب نگاشت کنید. که این دیگر هنر طراح است!
کاپر، تعبیر جالبی را از مدل طراحی بیان نموده است، «مدل طراحی دقیقاً جایی است که همزمان بر روی دو عالم ایستاده‌اید، عالم انسان و عالم ماشین، و باید تلاش نمود این دو عالم را به هم رساند.»
طراحی داده بر دو بخش طراحی جدول و طراحی پرس و جو می‌باشد. طراحی جدول از بخش طراحی داده، تحلیل موجودیت (ERD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط مدل رابطه‌ای، طراحی جدول را انجام می‌دهد. طراحی پرس و جو از بخش طراحی داده، تحلیل پرس و جو از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط جبر رابطه‌ای، طراحی پرس و جو را انجام می‌دهد.
توجه: در طراحی داده، فرهنگ داده‌های موجود در مدل تحلیل، مورد استفاده قرار می‌گیرد.

مدل رابطه‌ای[1]

برای شناسایی هر مدل یا ساختاری ابتدا باید به بررسی بخش‌های مختلف آن مدل یا ساختار بپردازیم.
مدل رابطه‌ای از دو بخش زیر تشکیل شده‌است:
۱- رابطه[2] یا جدول
۲- قوانین مدل رابطه‌ای

[1] Relational Model
[2] Relation

توجه: در مدل تحلیل منظور از رابطه، ارتباط بین موجودیت‌هاست، اما در مدل طراحی منظور از رابطه، جداول مدل رابطه‌ای می‌باشد.

رابطه

به هر زیرمجموعه دلخواه از حاصلضرب دکارتی دنباله‌ای از دامنه‌ها یک رابطه گفته می‌شود. به مجموعه مقادیر ممکن یک صفت، دامنه یا میدان[1] گفته می‌شود.

مثال: دامنه‌های D_1 ، D_2و D_3را به صورت زیر در نظر بگیرید:

$D_1 = \{a, b\}$

$D_2 = \{x, y\}$

$D_3 = \{1, 2, 3\}$

حاصلضرب دکارتی دو دامنه D_1 و D_2 به صورت زیر است:

$$D_1 \times D_2 = \{(a,x),(a,y),(b,x),(b,y)\}$$

حاصلضرب دکارتی دو دامنه D_2 و D_3 به صورت زیر است:

$$D_2 \times D_3 = \{(x,1),(x,2),(x,3),(y,1),(y,2),(y,3)\}$$

حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 به صورت زیر است:

$$D_1 \times D_2 \times D_3 = \{(a,x,1),(a,x,2),(a,x,3),(a,y,1),(a,y,2),(a,y,3),$$
$$(b,x,1),(b,x,2),(b,x,3),(b,y,1),(b,y,2),(b,y,3)\}$$

حال هر زیر مجموعه دلخواه از حاصلضرب‌های فوق، یک رابطه است:

رابطه R_1 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی دو دامنه D_1 و D_2 به صورت زیر است:

$$R_1 = \{(a,x),(a,y),(b,y)\}$$

رابطه R_2 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی دو دامنه D_2 و D_3 به صورت زیر است:

$$R_2 = \{(x,1),(x,2),(y,1),(y,2)\}$$

رابطه R_3 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 به صورت زیر است:

$$R_3 = \{(a,x,1),(a,x,2),(b,x,1),(b,x,2)\}$$

رابطه R_4 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی دو دامنه D_1 و D_2 به صورت زیر است:

[1] Domain

$R_4 = \{\}$

رابطه R_5 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی دو دامنه D_2 و D_3 به صورت زیر است:

$R_5 = \{\}$

رابطه R_6 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 به صورت زیر است:

$R_6 = \{\}$

توجه: به هر یک از اعضای رابطه، یک چندتایی مرتب یا یک تاپل گفته می‌شود. برای مثال (a,x) یک تاپل از رابطه R_1، $(x,1)$ یک تاپل از رابطه R_2 و $(a,x,1)$ یک تاپل از رابطه R_3 است.
توجه: تمامی اعضای یک رابطه می‌بایست فقط زیرمجموعه‌ای از یک حاصلضرب دکارتی باشند، برای مثال $R_1 = \{(a,x),(x,1),(a,x,1)\}$ یک رابطه نیست، زیرا تاپل (a,x) از حاصلضرب دکارتی دو دامنه D_1 و D_2، تاپل $(x,1)$ از حاصلضرب دکارتی دو دامنه D_2 و D_3 و تاپل $(a,x,1)$ از حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 انتخاب شده است.

جدول[1]

بهترین راه نمایش و پیاده‌سازی رابطه استفاده از جدول است، پس جدول یک شیوه نمایش رابطه است. به بیان دیگر رابطه مفهومی ریاضی دارد، اما از نظر کاربردی نمایشی جدولی دارد، بنابراین از دیدگاه کاربردی رابطه یک جدول است. بانک اطلاعات از تعدادی جدول تشکیل یافته است. در مدل رابطه‌ای، هر رابطه معادل یک جدول است. جدول ساختاری است نامدار که از تعدادی سطر (رکورد یا تاپل[2]) و ستون تشکیل شده‌است. هر ستون نمایشگر یک صفت خاصه از یک نوع موجودیت است و هر سطر نمایشگر یک نمونه از یک نوع موجودیت است. در مدل رابطه‌ای مقادیر هر ستون حتما متعلق به یک دامنه نامدار است.
حال هر زیر مجموعه دلخواه از حاصلضرب‌های فوق، یک جدول است:
جدول T_1 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی دو دامنه D_1 و D_2 به صورت زیر است:

D_1	D_2
a	x
a	y
b	y

جدول T_1

[1] Table
[2] Tuple

جدول T_2 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی دو دامنه D_2 و D_3 به صورت زیر است:

D_2	D_3
x	1
x	2
y	1
y	2

جدول T_2

جدول T_3 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 به صورت زیر است:

D_1	D_2	D_3
a	x	1
a	x	2
b	x	1
b	x	2

جدول T_3

جدول T_4 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی دو دامنه D_1 و D_2 به صورت زیر است:

D_1	D_2

جدول T_4

جدول T_5 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی دو دامنه D_2 و D_3 به صورت زیر است:

D_2	D_3

جدول T_5

جدول T_6 به عنوان یک زیر مجموعه تهی از حاصلضرب دکارتی سه دامنه D_1 ، D_2و D_3 به صورت زیر است:

D_1	D_2	D_3

جدول T_6

ویژگی‌های رابطه در مدل رابطه‌ای

۱- در رابطه، تاپل تکراری وجود ندارد. زیرا رابطه یک مجموعه است و مجموعه در ریاضیات طبق تعریف تاپل تکراری ندارد. بنابراین در جدول هم رکوردهای تکراری نداریم.

۲- تاپل‌ها در رابطه نظم مکانی ندارند. زیرا مجموعه نظم ندارد. بنابراین ترتیب رکوردها در جدول اهمیت ندارد.

۳- ترتیب صفات خاصه در یک تاپل مهم نیست. البته به شرطی که ترتیب صفات خاصه در تمام تاپل‌های موجود در یک رابطه با یکدیگر سازگار باشند. بنابراین ترتیب فیلدها در یک جدول نیز مهم نیست، نظم ندارد و می‌توان آنرا جابه‌جا کرد. البته به شرطی که ترتیب فیلدها در تمام رکوردهای موجود در یک جدول با یکدیگر سازگار باشند.

برای مثال رابطه R_3 به عنوان یک زیر مجموعه دلخواه از حاصلضرب دکارتی سه دامنه D_1، D_2 و D_3 را در نظر بگیرید:

$$R_3 = \{(a,x,1),(a,x,2),(b,x,1),(b,x,2)\}$$

اگر ترتیب تاپل‌های رابطه R_7 به صورت زیر باشد:

$$R_7 = \{(x,1,a),(x,2,a),(x,1,b),(x,2,b)\}$$

رابطه R_7 معادل رابطه R_3 است، زیرا ترتیب صفات خاصه در تمام تاپل‌های موجود در رابطه R_7 با یکدیگر سازگار است. $(R_3 = R_7)$

اگر ترتیب تاپل‌های رابطه R_8 به صورت زیر باشد:

$$R_8 = \{(x,1,a),(2,x,a),(1,b,x),(x,2,b)\}$$

رابطه R_8 معادل رابطه R_3 نیست، زیرا ترتیب صفات خاصه در تمام تاپل‌های موجود در رابطه R_8 با یکدیگر سازگار نیست. $(R_3 \neq R_8)$

۴- همه‌ی مقادیر صفات خاصه در تاپل‌ها تجزیه‌ناپذیرند. به عبارت دیگر در مدل رابطه‌ای نمی‌توان فیلد مرکبی که از فیلدهای ساده تشکیل شده‌است، تعریف نمود. یعنی فیلدهای موجود در جداول مدل رابطه‌ای نباید مرکب مثل تاریخ تولد و آدرس باشد و نباید چندمقداری مثل مدرک تحصیلی باشد. به عبارت دیگر هیچ رابطه‌ای نمی‌تواند شامل تاپل‌های تو در تو، تاپل تو تاپل یا جدول تو جدول باشد، به بیان دیگر مولفه‌های هریک از تاپل‌های رابطه نباید مرکب و چندمقداری باشد، به عبارت ساده‌تر هر مولفه یک تاپل نمی‌تواند خودش یک تاپل باشد.

توجه: همانطور که گفتیم بهترین راه نمایش و پیاده‌سازی رابطه استفاده از جدول است، بنابراین جدول معادل رابطه در نظر گرفته می‌شود.

خواص رابطه

بطور کلی خواص رابطه به دو شکل زیر بررسی می‌گردد:

الف) درجه رابطه

ب) کاردینالیتی رابطه

توجه: همه روابط دو خصیصه فوق را به طور همزمان دارند، اما مقادیر این خصیصه‌ها در روابط مختلف، متفاوت است.

الف) درجه ارتباط

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود.

توجه: تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم.

ب) کاردینالیتی ارتباط

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود.

توجه: تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است.

انواع کلید در پایگاه داده‌ها

ابرکلید (Supper key)

هر ترکیبی از صفات (ستون‌ها) که خاصیت کلیدی داشته باشد، یک ابرکلید است. دقت کنید که خاصیت کلیدی بر اساس محیط خارج مشخص می‌گردد و نه محتوی جدول.

تعریف خاصیت کلیدی: اگر بر اساس آن مورد، جستجو انجام شود، فقط و فقط یک مورد را به صورت یکتا برگرداند. در واقع قدرت تفکیک داشته باشد.

مثال: شماره دانشجویی یا شماره ملی

توجه: هر ترکیبی از ابرکلید با سایر ستون‌های جدول مورد نظر نیز یک ابرکلید است. بنابراین ابرکلید می‌تواند عضو زائد داشته باشد و کمینه (minimal) نباشد.

مثال:

S#	Sname	City
S_1	Sn1	C_1
S_2	Sn_2	C_2
S_3	Sn_3	C_2
S_4	Sn_4	C_2

#S: ابرکلید است.
Sname: ابرکلید نیست.
(Sname,City): ابرکلید نیست.
(S#,Sname): ابرکلید است.
(S#,City): ابرکلید است.

مثال: تعداد ابرکلیدهای مثال فوق چند عدد است؟
پاسخ: برابر 4 عدد می‌باشد.

S# +	Sname	City	ابر کلید
خاصیت	0	0 →	(S#)
کلیدی دارد	0	1 →	(S#,City)
بنابراین ابر	1	0 →	(S#,Sname)
کلید است.	1	1 →	(S#,Sname,City)

کلید کاندید (Candidate key)

ابرکلیدی که عضو زائد نداشته باشد، کلید کاندید است، به عبارت دیگر ابرکلید **کمینه** را کلید کاندید می‌گویند. منظور از ابرکلید کمینه، ابرکلیدی نیست که کمترین تعداد صفت را داشته باشد، بلکه منظور ابرکلیدی است که صفت زائد نداشته باشد.

مثال:

S#	Sname	City
S_1	Sn_1	C_1
S_2	Sn_2	C_2
S_3	Sn_3	C_2

جدول S

S#	P#	QTY
S_1	P_1	10
S_1	P_2	20
S_2	P_1	30

جدول SP

P#	Pname	Color
P_1	Pn_1	Red
P_2	Pn_2	Blue
P_3	Pn_3	Blue

جدول P

#S: ابرکلید است. کلید کاندید نیز هست. (در جدول S)
(S#,Sname): ابرکلید است، زیرا خاصیت کلیدی دارد، اما کلید کاندید نیست، زیرا عضو زائد Sname را دارد. در واقع صفت #S، به تنهایی خاصیت کلیدی دارد، بنابراین صفت Sname، عضو زائد است. (در جدول S)
(S#,P#): ابرکلید است. کلید کاندید نیز هست. (در جدول SP).

مثال:

شماره ملی: ابرکلید است. کلید کاندید نیز هست.

(شماره ملی و نام خانوادگی): ابرکلید است. زیرا خاصیت کلیدی دارد، اما کلید کاندید نیست، زیرا عضو زائد نام خانوادگی را دارد. در واقع صفت شماره ملی، به تنهایی خاصیت کلیدی دارد، بنابراین صفت نام‌خانوادگی، عضو زائد است.

توجه: یک جدول می‌تواند چندین کلید کاندید داشته باشد.

مثال:

نام	نام خانوادگی	شماره ملی	شماره دانشجویی
		کلید کاندید (کلید فرعی)	کلید کاندید (کلید اصلی)

توجه: در مدل رابطه‌ای، هر رابطه حتماً حداقل یک کلید کاندید دارد، زیرا در بدترین شرایط، همه صفات با هم کلید کاندید می‌شوند، که به این رابطه تمام کلید (All key) گفته می‌شود.

توجه: یک رابطه، تحت هیچ شرایطی نمی‌تواند به دلیل استفاده از خاصیت مجموعه‌ای بودن، سطر تکراری داشته باشد. بنابراین یک رابطه، حداقل یک کلید کاندید دارد.

مثال: یک جدول تمام کلید.

a	b	c
1	2	3
1	6	3
1	2	7
8	2	3

کلید اصلی (Primary key)

کلید اصلی، یکی از کلیدهای کاندید است که بر اساس سلیقه طراح پایگاه داده، انتخاب می‌شود.

توجه: مطابق قوانین مدل رابطه‌ای، هر جدول حتماً باید یک کلید اصلی داشته باشد.

توجه: از آنجا که وظیفه کلید اصلی ایجاد تمایز میان سطرهای مختلف یک جدول است، هیچگاه کلید اصلی نمی‌تواند مقدار NULL داشته باشد.

توجه: اگر چندین کلید کاندید در یک جدول وجود داشته باشد، در انتخاب کلید اصلی همواره اولویت با کلید کاندیدی است که کمترین تعداد صفت را دارد.

توجه: هر جدول فقط می‌تواند یک کلید اصلی داشته باشد.

کلید فرعی (Alternative key)

هر کلید کاندیدی که به عنوان کلید اصلی انتخاب نشده است، یک کلید فرعی است.

مثال:

نام	نام خانوادگی	شماره ملی	شماره دانشجویی
		کلید کاندید (کلید فرعی)	کلید کاندید (کلید اصلی)

کلید خارجی (Foreign key)

کلید خارجی برای ارتباط میان جداول مورد استفاده قرار می‌گیرد.
کلید خارجی در دو دیگاه ساختاری و محتوایی مورد بررسی قرار می‌گیرد:

دیدگاه ساختاری کلید خارجی

تعریف: اگر صفت(هایی) در یک جدول به عنوان کلید خارجی تعریف شود،
اول اینکه: این صفت(ها) در جدول خودش شرط خاصی ندارد یعنی الزاما خاصیت کلیدی ندارد.
و دوم اینکه: این صفت(ها) در همان جدول یا جدول دیگری، **باید کلید کاندید** (اصلی یا فرعی) باشد.

مثال:

کلید کاندید ← کلید خارجی (جدول S ← جدول SP)
کلید خارجی ← کلید کاندید (جدول SP ← جدول P)

S#	Sname	City
S_1	Sn_1	C_1
S_2	Sn_2	C_2
S_3	Sn_3	C_2

جدول S

S#	P#	QTY
S_1	P_1	10
S_1	P_2	20
S_2	P_1	30

جدول SP

P#	Pname	Color
P_1	Pn_1	Red
P_2	Pn_2	Blue
P_3	Pn_3	Blue

جدول P

مثال:

نام	نام خانوادگی	شماره ملی
		کلید خارجی

جدول R_2

شماره ملی	شماره دانشجویی
کلید فرعی	کلید اصلی

جدول R_1

توجه: در مثال فوق، از آنجا که شماره‌ملی در جدول R_2، به طور ذاتی خاصیت کلیدی دارد، بنابراین کلید خارجی، خاصیت کلیدی دارد، اما الزامی بر این شرط نبوده است.
توجه: S# در جدول S، کلید کاندید است، پس عناصرش یکتاست.
توجه: S# در جدول SP، به عنوان کلید خارجی، خاصیت کلیدی ندارد، پس ممکن است، مقادیر تکراری داشته باشد.

دیدگاه محتوایی کلید خارجی

به ازای هر مقدار موجود در یک کلید خارجی، باید دقیقاً یک مقدار متناظر در کلید کاندید متناظر آن وجود داشته باشد، در غیر این صورت می‌گوییم، کلید خارجی دارای ارجاع NULL است. به بیان دیگر، مقادیر کلید خارجی همواره باید زیرمجموعه مقادیر کلید کاندید باشد.

توجه: یک کلید خارجی در یک رابطه هیچگاه نباید ارجاع NULL، داشته باشد، این مسأله را به عنوان یک قانون جامعیتی در مدل رابطه‌ای، می‌شناسیم و آن را **قانون جامعیت ارجاعی** می‌نامیم.

مثال:

S#	Sname	City
S_1	Sn_1	C_1
S_2	Sn_2	C_2
S_3	Sn_3	C_3

جدول S

S#	P#	QTY
S_1	P_1	10
S_1	P_2	20
S_2	P_1	30
S_4	P_3	40

ارجاع NULL ×

جدول SP

P#	Pname	Color
P_1	Pn_1	Red
P_2	Pn_2	Blue
P_3	Pn_3	Blue

جدول P

درج رکورد $(S_4, P_3, 40)$ در جدول SP، غیر مجاز و غیر ممکن است، زیرا سبب ارجاع NULL می‌گردد.

توجه: هر مقداری که در کلید خارجی وجود دارد، باید دارای مقدار متناظر در کلید کاندید مقصد باشد ولی عکس آن صادق نیست.

مثال: مانند S_3 که در جدول S قرار دارد ولی لزومی ندارد در جدول SP هم باشد. اما باید همه مقادیر کلید خارجی در کلید کاندید باشد، از بالا به پایین می‌بارد، از پایین به بالا که نمی‌بارد!

توجه: اگرچه کلید خارجی هیچگاه نباید، ارجاع NULL داشته باشد، اما می‌تواند مقدار NULL داشته باشد، البته به شرطی که کلید خارجی در شرایط قوانین بالا دستی قرار نگیرد.

مثال:

کلید کاندید — کلید خارجی

a	b	c
1	4	7
2	6	5
3	4	5

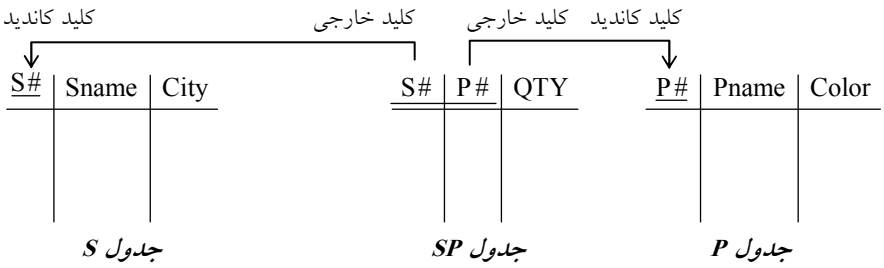
R_1

a	d
2	1
3	2
2	3
1	4

R_2

توجه: می‌توان مقادیر ستون a در جدول R_2 را NULL قرار داد. کلید کاندید (کلید اصلی) جدول R_2 ستون d است.

مثال:



توجه: می‌توان مقادیر کلید خارجی را برابر NULL قرار داد، البته به شرطی که قوانین بالا دستی بر کلید خارجی حاکم نباشد. در مثال فوق S# و P# در جدول SP به عنوان کلید خارجی نمی‌توانند NULL باشند، زیرا (S#,P#) با هم قبل از اینکه به تنهایی کلید خارجی باشند، در جدول SP کلید اصلی جدول SP بوده‌اند. بنابراین قانون بالا دستی **جامعیت موجودیت**، بدین معنی که هیچگاه نباید تمام یا بخشی از کلید اصلی NULL باشد، جلوی NULL شدن S# و P# به عنوان کلید خارجی را در جدول SP می‌گیرد.

توجه: مطابق تعریف کلید خارجی، کلید خارجی باید به کلید کاندید همان رابطه یا رابطه دیگر ارجاع کند، در واقع لزومی ندارد که روابط فرضی R_1 و R_2 از هم مجزا باشند تا کلید خارجی در یک رابطه باشد و کلید کاندید در رابطه‌ای دیگر. زیرا هم کلید کاندید و هم کلید خارجی می‌توانند در یک رابطه باشند، یعنی R_1 و R_2 می‌توانند یک رابطه باشند، بنابراین کلید خارجی یک رابطه می‌تواند متناظر با کلید کاندید همان رابطه باشد. در واقع ممکن است، دیاگرام ارجاعی دارای طوقه باشد، به این معنی که مبداء و مقصد کلید خارجی با هم یکی باشد. این مورد زمانی رخ می‌دهد که درجه ارتباط موجودیت‌ها در مدل تحلیل(مدل ER) برابر مقدار 1 یعنی درجه 1 باشد.

مثال: نام رئیس D چیست؟

شماره کارمندی	نام	رئیس
1	A	1
2	B	1
3	C	1
4	D	3
5	E	2
6	F	5

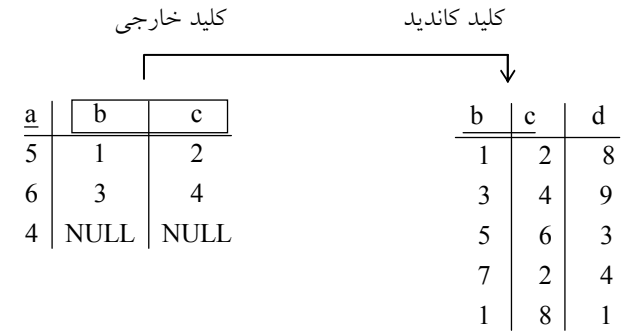
کلید کاندید — کلید خارجی

طوقه

پاسخ: شماره رئیس D برابر 3 است و نام شماره 3 برابر C است.

توجه: مجموعه‌ای از صفات رابطه R_2 را کلید خارجی این رابطه می‌گویند، هرگاه در همان رابطه یا رابطه دیگر مثل R_1 کلید کاندید (اصلی یا فرعی) باشد، بنابراین هیچ الزامی برای اینکه کلید خارجی ترکیبی نباشد و ساده باشد، وجود ندارد، بنابراین کلید خارجی می‌تواند یک کلید ترکیبی باشد، یعنی بیش از یک صفت داشته باشد، لذا اگر کلید خارجی ترکیبی باشد حتماً کلید کاندید متناظر نیز ترکیبی است.

مثال:



a	b	c
5	1	2
6	3	4
4	NULL	NULL

b	c	d
1	2	8
3	4	9
5	6	3
7	2	4
1	8	1

توجه: اگر کلید خارجی ترکیبی باشد (یعنی بیش از یک صفت داشته باشد) یا تماماً NULL است یا تماماً غیر NULL ، به عبارت دیگر هیچگاه بخشی از یک کلید خارجی ترکیبی نمی‌تواند مقدار NULL بگیرد.

توجه: ابرکلید و کلیدکاندید در مباحث تئوری و کلید اصلی و کلید خارجی در کاربرد مورد استفاده قرار می‌گیرند.

توجه: برای اینکه یک رابطه، در مدل رابطه‌ای صدق کند، باید دارای دو شرط زیر باشد:

۱- اتومیک باشد.

یعنی صفت چندمقداری و مرکب نباشد. به عبارت دیگر یعنی اینکه هیچ یک از سلول‌های این رابطه دارای بیش از یک مقدار نباشد.

۲- رابطه باید حتماً دارای حداقل یک کلید کاندید باشد.

توجه: از آنجاییکه رابطه در مدل رابطه‌ای یک مجموعه است، در بدترین حالت تمام صفات یک سطر با هم، باید تشکیل یک کلید را بدهد، در غیر این صورت تعریف مجموعه نقض شده است. (مجموعه تاپل تکراری ندارد.)

توجه: یک رابطه در مدل رابطه‌ای دارای حداقل یک کلید کاندید است. اما می‌تواند، بیش از یک کلید کاندید نیز داشته باشد.

نگاشت مدل تحلیل (نمودار نهاد و رابطه) به مدل طراحی (مدل رابطه‌ای)

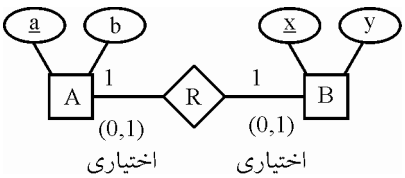
به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در مدل تحلیل (نمودار نهاد – رابطه) هنگام نگاشت به مدل طراحی (مدل رابطه‌ای) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

نگاشت رابطه یک به یک بین دو موجودیت به مدل رابطه‌ای

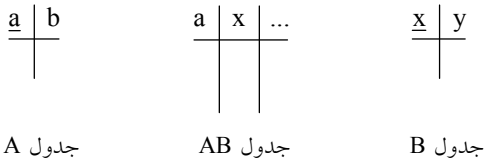
حالت اول: اگر هر دو طرف رابطه اختیاری باشد. هر موجودیت به یک جدول تبدیل می‌گردد. و یک جدول ارتباط نیز به عنوان ارتباط دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول ارتباط یکی از کلیدهای دو جدول دیگر می‌باشد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

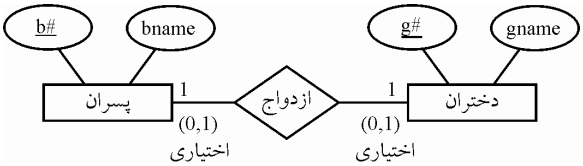


توجه: کلید کاندید جدول AB، ستون a یا ستون x می‌باشد. زیرا هر یک به تنهایی قدرت تفکیک سطرها را دارند. ترکیب ستون‌های a و x با هم به عنوان ax، ابرکلید ایجاد می‌کند.

توجه: ستون a در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون x در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند.

مثال:

مدل تحلیل:



مدل طراحی:

g#	gname
g_1	gn1
g_2	gn2
g_3	gn3

جدول دختران

b#	g#	سال ازدواج
b_1	g1	60
b_2	g2	70

جدول ازدواج

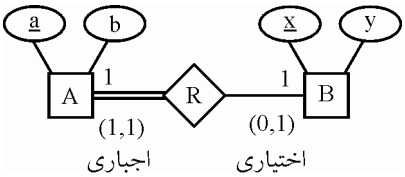
b#	bname
b_1	bn1
b_2	bn2
b_3	bn3

جدول پسران

حالت دوم: اگر یک طرف رابطه اختیاری و طرف دیگر رابطه اجباری باشد، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جدول موجودیت اختیاری در جدول موجودیت اجباری به عنوان کلید خارجی تعریف می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

a	b	x

جدول A

x	y

جدول B

توجه: این نگاشت منطقی هم هست، هرگاه سطری در جدول موجودیت اختیاری وجود داشت، توسط کلید خارجی به جدول موجودیت اجباری می‌تواند رجوع کند.

توجه: اگر بالعکس نگاشت فوق عمل کنیم و کلید کاندید جدول موجودیت اجباری را در جدول موجودیت اختیاری به عنوان کلید خارجی تعریف کنیم، آنگاه جدول موجودیت اختیاری با مقادیر زیاد NULL مواجه می‌گردد.

مدل طراحی نادرست:

a	b

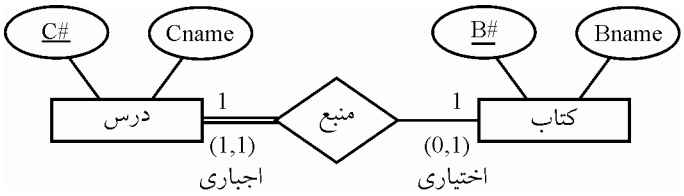
جدول A

x	y	a

جدول B

مثال:

مدل تحلیل:



مدل طراحی درست:

C#	Cname	B#
C_1	Cn1	B_1
C_2	Cn2	B_2

جدول درس

B#	Bname
B_1	Bn1
B_2	Bn2
B_3	Bn3
B_4	Bn4

جدول کتاب

مدل طراحی نادرست:

C#	Cname
C_1	Cn1
C_2	Cn_2

جدول درس

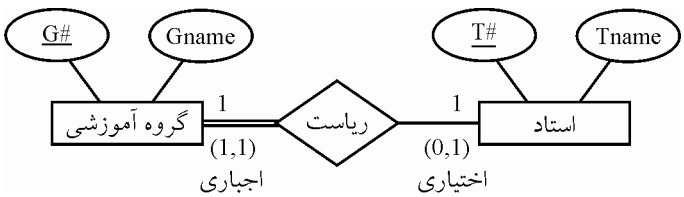
B#	Bname	C#
B_1	Bn1	C_1
B_2	Bn2	C_2
B_3	Bn3	NULL
B_4	Bn4	NULL

جدول کتاب

توجه: مقادیر NULL در ستون C# قابل مشاهده است، زیرا بسیاری از کتاب‌ها، مرجع هیچ درسی نیستند!

مثال:

مدل تحلیل:



توجه: رابطه ریاست بین استاد و گروه آموزشی به صورت فوق است، چون هر گروه آموزشی حتماً می‌بایست رئیس داشته باشد، اما یک استاد ممکن است رئیس یک گروه آموزشی باشد یا نباشد.

مدل طراحی:

G#	Gname	T#
G_1	Gn1	T_1
G_2	Gn2	T_2

جدول گروه آموزشی

T#	Tname
T_1	Tn1
T_2	Tn2
T_3	Tn3
T_4	Tn4

جدول استاد

مدل طراحی نادرست:

G#	Gname
G_1	Gn1
G_2	Gn2

جدول گروه آموزشی

T#	Tname	G#
T_1	Tn1	G1
T_2	Tn2	G2
T_3	Tn3	NULL
T_4	Tn4	NULL

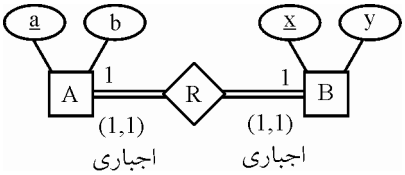
جدول استاد

توجه: مقادیر NULL در ستون #G قابل مشاهده است، زیرا بسیاری از اساتید، رئیس هیچ گروه آموزشی نیستند!

حالت سوم: اگر هر دو طرف رابطه اجباری باشد، هر دو موجودیت در قالب یک جدول قرار می‌گیرند. و کلید کاندید جدول حاصل، کلید کاندید هر یک از موجودیت‌ها می‌تواند باشد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

a	b	x	y

جدول AB

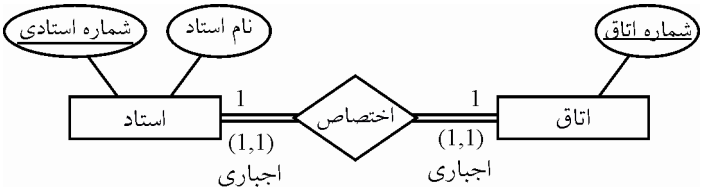
یا

a	b	x	y

جدول AB

توجه: در مواقع اختیاری، با ایجاد جداول دیگر، از تکرار بی‌رویه مقادیر NULL، جلوگیری می‌کنیم، ولی در موارد اجباری مانند حالت سوم، ایجاد جداول دیگر، معنا ندارد.
مثال: فرض کنید در یک دانشگاه هر استاد، یک اتاق اختصاصی دارد.

مدل تحلیل:



مدل طراحی:

شماره استادی	نام استاد	شماره اتاق

یا

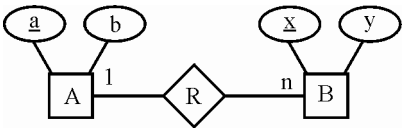
شماره اتاق	نام استاد	شماره استادی

نگاشت رابطه یک به چند بین دو موجودیت به مدل رابطه‌ای

مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جدول یک در جدول چند به عنوان کلید خارجی تعریف می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

a	b

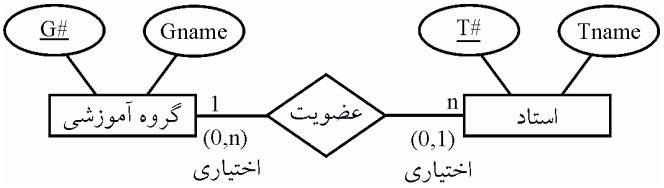
جدول A

x	y	a

جدول B

مثال: حالت اختیاری

مدل تحلیل:



مدل طراحی:

T#	Tname	G#
T_1	Tn1	G1
T_2	Tn2	G1
T_3	Tn3	G2
T_4	Tn4	NULL

جدول استاد

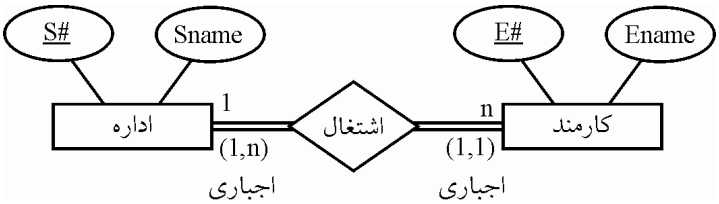
G#	Tname
G_1	Gn1
G_2	Gn2
G_3	Gn3

جدول گروه آموزشی

توجه: اگر استادی عضو گروه آموزشی نباشد، ستون #G برای آن مقدار NULL می‌گیرد، مانند T_4 در جدول استاد.

مثال: حالت اجباری

مدل تحلیل:



مدل طراحی:

E#	Ename	S#
E_1	En1	S_1
E_2	En2	S_1
E_3	En3	S_2
E_4	En4	S_3

جدول کارمند

S#	Sname
S_1	Sn1
S_2	Sn2
S_3	Sn3

جدول اداره

توجه: با توجه به اجباری بودن ارتباط، یک اداره حداقل یک کارمند و حداکثر چند کارمند دارد و هر کارمند حداقل و حداکثر در یک اداره اشتغال دارد.

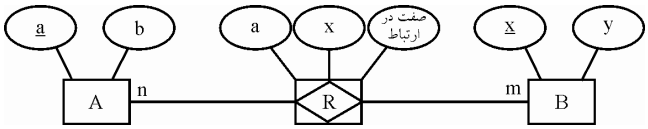
توجه: با توجه به اجباری بودن موجودیت کارمند، هنگام پیاده‌سازی، ستون #S در جدول کارمند می‌بایست NOT NULL تعریف گردد.

نگاشت رابطه چند به چند بین دو موجودیت به مدل رابطه‌ای

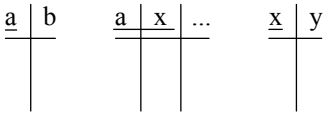
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



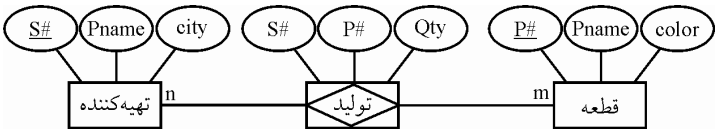
مدل طراحی:



توجه: ستون a در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون x در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند.

مثال: سیستم تولیدکنندگان و قطعات.

مدل تحلیل:



توجه: صفت QTY، به عنوان یک صفت در ارتباط، تعداد قطعات را مشخص می‌کند.

مدل طراحی:

S#	Sname	city
S_1	Sn1	C_1
S_2	Sn2	C_1
S_3	Sn3	C_3

جدول تهیه کننده

S#	P#	QTY
S_1	P_1	10
S_1	P_2	20
S_2	P_1	30

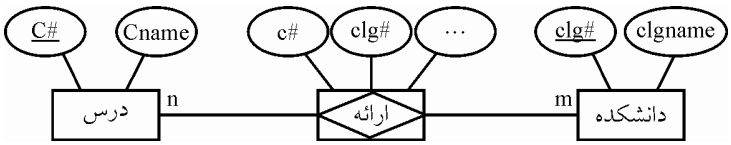
جدول تولید

P#	Pname	color
P_1	Pn1	Red
P_2	Pn2	Blue

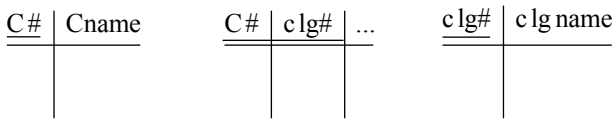
جدول قطعه

مثال:

مدل تحلیل:



مدل طراحی:



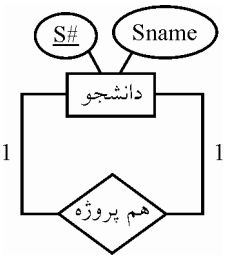
جدول درس جدول ارائه جدول دانشکده

نگاشت رابطه یک به یک در یک موجودیت به مدل رابطه‌ای

حالت اول: اگر موجودیت اختیاری باشد. موجودیت به یک جدول تبدیل می‌گردد و یک جدول دیگر برای نمایش روابط موجود مورد استفاده قرار می‌گیرد.

مثال: فرض کنید در یک دانشگاه، هر دانشجو به صورت اختیاری، حداکثر باید با یک دانشجوی دیگر هم پروژه باشد.

مدل تحلیل:



مدل طراحی:

S#	Sname
S_1	Sn1
S_2	Sn2
S_3	Sn3
S_4	Sn4

جدول دانشجو

S#	SP#
S_1	S_2
S_2	S_1

جدول هم پروژه‌ای

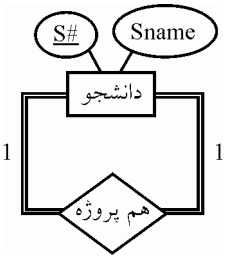
توجه: ستون #SPدر جدول هم پروژه‌ای به عنوان کلید خارجی بـه سـتون #S از جـدول دانشـجو ارجاع می‌کند.

توجه: انجام فعالیت طراحی در حالت اختیاری، توسط یک جـدول، سـبب ایجـاد مقـادیر NULL زیاد می‌گردد. بنابراین همان طراحی دو جدولی توصیه می‌گردد.

حالت دوم: اگر موجودیت اجباری باشد. موجودیت به یک جدول تبدیل می‌گردد.

مثال: فرض کنید در یک دانشگاه، هر دانشجو به صورت اجباری، باید با یک دانشجوی دیگـر هـم پروژه باشد.

مدل تحلیل:



مدل طراحی:

S#	Sname	SP#
S_1	Sn1	S_2
S_2	Sn2	S_1
S_3	Sn3	S_4
S_4	Sn4	S_3

کلید کاندید کلید خارجی

طوقه

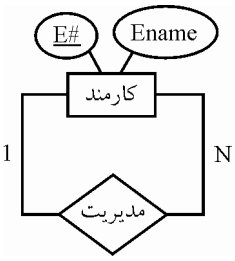
توجه: مطابق تعریف کلید خارجی، کلید خارجی باید به کلید کاندید همان رابطه یـا رابطـه دیگـر، ارجاع کند، در واقع لزومی ندارد که روابط فرضی R_1 و R_2 از هم مجزا باشند تا کلید خـارجی در یک رابطه باشد و کلید کاندید در رابطه‌ای دیگر. زیرا هم کلید کاندید و هم کلید خارجی می‌توانند در یک رابطه باشند، یعنی R_1 و R_2 می‌توانند یک رابطه باشند، بنابراین کلید خـارجی یـک رابطـه می‌تواند متناظر با کلید کاندید همان رابطه باشد. در واقع دیاگرام ارجاعی دارای طوقه است به ایـن معنی که مبدأ و مقصد کلید خارجی با هم یکی است.

نگاشت رابطه یک به چند در یک موجودیت به مدل رابطه‌ای

مستقل از اختیاری یا اجباری بودن موجودیت، موجودیت به یک جدول تبدیل می‌گردد.

مثال: فرض کنید در یک سازمان هر کارمند، تحت مدیریت یک کارمند دیگر باشد.
توجه: یک رئیس چندین کارمند دارد ولی هر کارمند حداکثر یک رئیس دارد.

مدل تحلیل:



مدل طراحی:

E#	Sname	M#
S_1	Sn1	NULL
S_2	Sn2	S_1
S_3	Sn3	S_1
S_4	Sn4	S_3
S_5	Sn5	S_2
S_6	Sn6	S_5

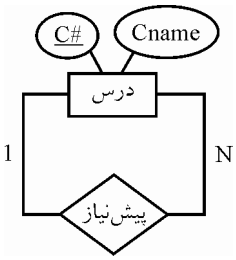
کلید خارجی (شماره مدیر) کلید کاندید (شماره کارمندی)

طوقه

توجه: سطر مربوط به مدیر کل، که مدیر بالاتری ندارد، مقدار Null در فیلد #M خود می‌گیرد.
توجه: شماره مدیر، برای شخصی با شماره کارمندی S_4 برابر S_3 است و نام S_3 برابر Sn3 است.
توجه: مطابق تعریف کلید خارجی، کلید خارجی باید به کلید کاندید همان رابطه یا رابطه دیگر، ارجاع کند، در واقع لزومی ندارد که روابط فرضی R_1 و R_2 از هم مجزا باشند، تا کلید خارجی در یک رابطه باشد و کلید کاندید در رابطه‌ای دیگر. زیرا هم کلید کاندید و هم کلید خارجی می‌توانند در یک رابطه باشند، یعنی R_1 و R_2 می‌توانند یک رابطه باشند، بنابراین کلید خارجی یک رابطه می‌تواند متناظر با کلید کاندید همان رابطه باشد. در واقع دیاگرام ارجاعی دارای طوقه است به این معنی که مبداء و مقصد کلید خارجی با هم یکی است.

مثال: فرض کنید در یک دانشگاه، هر درس، پیش نیازهای دیگری دارد.
توجه: در رابطه یک به چند میان درس و پیش‌نیازهای آن، یک درس چندین پیش نیاز دارد ولی هر پیش‌نیاز، حداکثر پیش نیاز یک درس است.

مدل تحلیل:



مدل طراحی:

C#	Cname	CC#
C_1	Cn1	NULL
C_2	Cn2	C_1
C_3	Cn3	C_1
C_4	Cn4	C_3
C_5	Cn5	C_2
C_6	Cn6	C_5

کلید خارجی (شماره درس) کلید کاندید (پیش‌نیازها)

طوقه

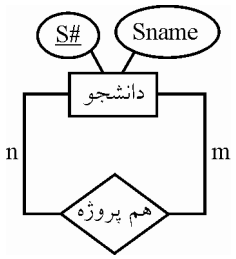
توجه: C_2 و C_3 پیش‌نیازهای درس C_1 هستند و C_6 پیش‌نیاز درس C_5 و C_1 پیش نیاز هیچ درسی نیست.

نگاشت رابطه چند به چند در یک موجودیت به مدل رابطه‌ای

مستقل از اختیاری یا اجباری بودن موجودیت. موجودیت به یک جدول اصلی تبدیل می‌گردد. و یک جدول ارتباط برای نمایش روابط موجود مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول اصلی، در جدول ارتباط به عنوان کلید خارجی تعریف می‌گردد.

مثال: فرض کنید در یک دانشگاه، هر دانشجو، با چندین دانشجوی دیگر هم پروژه باشد.

مدل تحلیل:



مدل طراحی:

S#	Sname
S_1	Sn1
S_2	Sn2
S_3	Sn3
S_4	Sn4

جدول دانشجو

S#	SP#
S_1	S_2
S_1	S_3
S_2	S_1
S_3	S_1

جدول هم‌پروژه‌ای

توجه: ستون #SP در جدول هم پروژه‌ای به عنوان کلید خارجی بـه سـتون #S از جـدول دانشـجو ارجاع می‌کند.

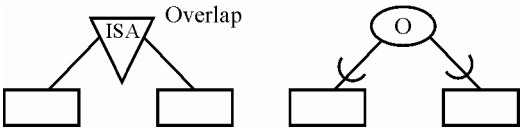
توجه: کلید کاندید جدول دانشجو #S است و کلید کاندید جدول هم پروژه‌ای (#SP و #S) است.

توجه: انجام فعالیت طراحی در حالت اجباری، توسط یک جدول، سبب افزایش افزونگی می‌گردد. بنابراین همان طراحی دو جدولی توصیه می‌گردد.

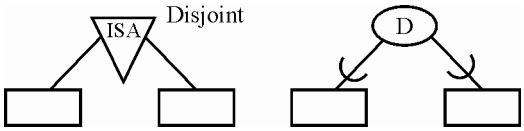
نگاشت رابطه ISA یا وراثت به مدل رابطه‌ای

در رابطه ISA رابطه پدر با فرزندان به دو صورت رابطه اختیاری یا جزئی (Partial) بـا نمـاد خـط عمودی و رابطه اجباری یا کلی (Total) با نماد خط مضاعف عمودی است و رابطه فرزندان با پدر به دو صورت رابطه پوشا یا تخصیص غیرمجزا (Overlap) و رابطه غیرپوشـا یـا تخصـیص مجـزا (Disjoint) می‌باشد.

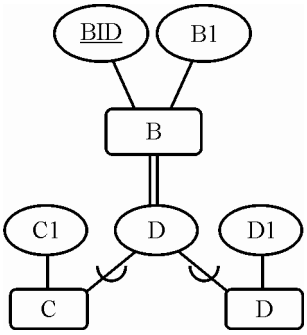
رابطه پوشا یا تخصیص غیرمجزا (Overlap) مـابین فرزنـدان و پـدر بـه دو شـیوه زیـر نشـان داده می‌شود:



رابطه غیرپوشا یا تخصیص مجزا (Disjoint) مـابین فرزنـدان و پـدر بـه دو شـیوه زیـر نشـان داده می‌شود:



مثال: نمودار نهاد و رابطه زیر را در نظر بگیرید:



مدل EER رسم شده در شکل فوق، یک رابطه اجباری یا کلی (Total) مابین موجودیت B و موجودیت‌های C و D و یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint) را مابین موجودیت‌های C و D و موجودیت B نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را بیان می‌کنیم.

در یک رابطه اجباری یا کلی (Total)، هر نمونه از موجودیت پدر حتماً می‌بایست با یکی از نمونه موجودیت‌های فرزند در ارتباط باشد. برای نمونه در این مثال، هر نمونه از موجودیت B حتماً می‌بایست با یکی از نمونه موجودیت‌های C یا D در ارتباط باشد. به عبارت دیگر نمی‌توان نمونه‌ای از موجودیت B داشت که با هیچ یک از نمونه موجودیت‌های C یا D در ارتباط نیست.

هم‌چنین در یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint)، نمونه موجودیت‌های فرزند نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند. برای نمونه در این مثال، نمونه موجودیت‌های C و D نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت B در ارتباط باشند. به عبارت دیگر نمی‌توان نمونه‌هایی از موجودیت‌های C و D داشت که به طور همزمان با نمونه‌ای از موجودیت B در ارتباط هستند.

به عنوان مثالی دیگر هر نمونه از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد حتماً می‌بایست با یکی از نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است در ارتباط باشد. به عبارت دیگر نمی‌توان نمونه‌ای از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد داشت که با هیچ یک از نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است در ارتباط نباشد. این یعنی رابطه اجباری یا کلی (Total).

همچنین نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد در ارتباط باشند، به عبارت دیگر نمی‌توان نمونه‌هایی از موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است داشت که به طور همزمان با نمونه‌ای از موجودیت ماشین که

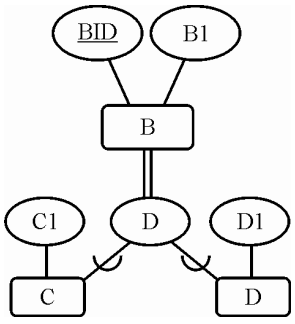
شماره شاسی یکتا و منحصر به فرد خود را دارد در ارتباط باشند. این یعنی رابطه غیرپوشا یا تخصیص مجزا (Disjoint).

از آنجاکه رابطه مابین موجودیت B و موجودیت‌های C و D یک رابطه اجباری یا کلی (Total) است، پس رکوردهای حاوی محتوای مقدار NULL درستون‌های مربوط به موجودیت B در طراحی به شکل مدل دو جدولی در جداول C و D به ازای یک نمونه موجودیت از B به دلیل عدم ارتباط با برخی از نمونه موجودیت‌های C و D ایجاد نمی‌گردد، که باعث شود این محتوای NULL در جداول C و D حاصل از عدم ارتباط برخی از نمونه موجودیت‌های موجودیت B با نمونه موجودیت‌های C و D در جدول B، به شکل مدل سه جدولی نگهداری شود.

در حالت رابطه اجباری مابین موجودیت B و موجودیت‌های C و D به ازای هر نمونه از موجودیت B، حتما نمونه موجودیتی از C یا D وجود دارد که با B رابطه برقرار کند، پس در این حالت طراحی بهینه این است که کل صفات موجودیت B در دو جدول موجودیت‌های C و D قرار داده شود و یک طراحی به شکل مدل دو جدولی ایجاد گردد، همچنین از آنجاکه رابطه مابین موجودیت‌های C و D و موجودیت B یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint) است، پس رکوردهای تکراری در جداول C و D به ازای یک نمونه موجودیت از موجودیت B ایجاد نمی‌گردد، که افزونگی حاصل از تکرار رکوردها در جداول C و D سبب شود رکورد نمونه موجودیت‌های B در جدول B نگهداری شود و یک مدل سه جدولی ایجاد گردد. پس در این حالت طراحی بهینه این است که کل صفات موجودیت B در دو جدول موجودیت های C و D قرار داده شود و یک طراحی به شکل مدل دو جدولی ایجاد گردد.

در ادامه فرآیند نگاشت نمودار (ISA) به مدل رابطه‌ای را شرح می‌دهیم:

مدل تحلیل (نمودار ISA)



مدل طراحی (مدل رابطه‌ای)

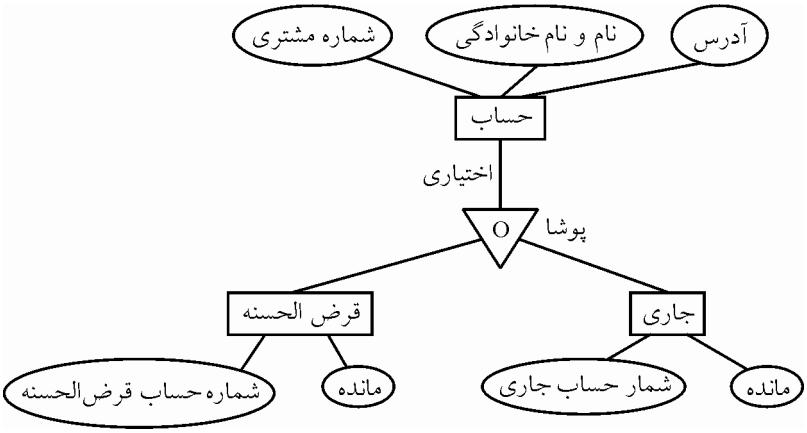
BID	B1	C1

جدول C

BID	B1	D1

جدول D

مثال: نمودار نهاد و رابطه زیر را در نظر بگیرید:



مدل EER رسم شده در شکل فوق، یک رابطه اختیاری یا جزئی (Partial) مابین موجودیت حساب و موجودیت‌های حساب جاری و حساب قرض الحسنه و یک رابطه پوشا یا تخصیص غیرمجزا (Overlap) را مابین موجودیت‌های حساب جاری وحساب قرض الحسنه و موجودیت حساب نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را بیان می‌کنیم.

در یک رابطه اختیاری یا جزئی (Partial)، هر نمونه از موجودیت پدر می‌تواند با یکی از نمونه موجودیت‌های فرزند در ارتباط باشد یا نباشد. برای نمونه در این مثال، هر نمونه از موجودیت حساب می‌تواند با یکی از نمونه موجودیت‌های حساب جاری و حساب قرض الحسنه در ارتباط باشد یا نباشد. به عبارت دیگر می‌توان نمونه‌ای از موجودیت حساب داشت که با هیچ یک از نمونه موجودیت‌های حساب جاری و حساب قرض الحسنه در ارتباط نیست.

هم‌چنین در یک رابطه پوشا یا تخصیص غیرمجزا (Overlap)، نمونه موجودیت‌های فرزند می‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند. برای نمونه در این مثال، نمونه موجودیت‌های حساب جاری و حساب قرض الحسنه می‌توانند به طور همزمان با نمونه‌ای از موجودیت حساب در ارتباط باشند. به عبارت دیگر می‌توان نمونه‌هایی از موجودیت‌های حساب جاری و حساب قرض الحسنه داشت که به طور همزمان با نمونه‌ای از موجودیت حساب در ارتباط هستند. از آنجاکه رابطه مابین موجودیت حساب و موجودیت‌های حساب جاری و حساب

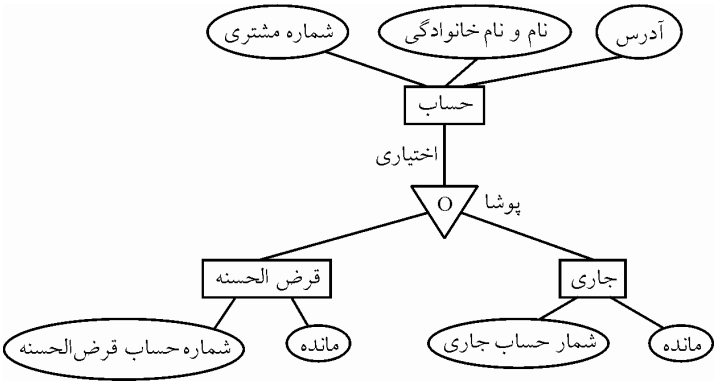
قرض الحسنه یک رابطه اختیاری یا جزئی (Partial) است، پس رکوردهای حاوی محتوای مقدار NULL درستون‌های مربوط به موجودیت حساب در طراحی به شکل مدل دو جدولی در جداول حساب جاری و حساب قرض الحسنه به ازای یک نمونه موجودیت از حساب به دلیل عدم ارتباط با برخی از نمونه موجودیت‌های حساب جاری و حساب قرض الحسنه ایجاد می‌گردد، که این محتوای NULL در جداول حساب جاری و حساب قرض الحسنه حاصل از عدم ارتباط برخی از نمونه موجودیت‌های موجودیت حساب با نمونه موجودیت‌های حساب جاری و حساب قرض الحسنه در مدل دو جدولی شامل جدول حساب جاری و جدول حساب قرض‌الحسنه باعث می‌شود اطلاعات موجودیت حساب در جدول حساب، به شکل مدل سه جدولی شامل جدول حساب، جدول حساب جاری و جدول حساب قرض‌الحسنه نگهداری شود.

در حالت رابطه اختیاری مابین موجودیت حساب و موجودیت‌های حساب جاری و حساب قرض الحسنه به ازای برخی از نمونه‌ها از موجودیت حساب، ممکن است نمونه موجودیتی از حساب جاری یا حساب قرض الحسنه وجود نداشته باشد که با موجودیت حساب رابطه برقرار کند، پس در این حالت طراحی بهینه این است که صفات موجودیت حساب در جدول حساب، صفات موجودیت حساب جاری در جدول حساب جاری و صفات موجودیت حساب قرض الحسنه در جدول حساب قرض الحسنه قرار داده شود و یک طراحی به شکل مدل سه جدولی اما با تحمل سربار حاصل از عمل الحاق ایجاد گردد و جهت ارتباط جداول حساب جاری و حساب قرض الحسنه با جدول حساب کلید کاندید جدول حساب در جداول حساب جاری و حساب قرض الحسنه به عنوان کلید خارجی تعریف گردد، همچنین از آنجاکه رابطه مابین موجودیت‌های حساب جاری و حساب قرض الحسنه و موجودیت حساب یک رابطه پوشا یا تخصیص غیرمجزا (Overlap) است، پس رکوردهای تکراری در جداول حساب جاری و حساب قرض‌الحسنه به ازای یک نمونه موجودیت از موجودیت حساب ایجاد می‌گردد، که افزونگی حاصل از تکرار رکوردها در جداول حساب جاری و حساب قرض الحسنه سبب می‌شود رکورد نمونه موجودیت‌های حساب در جدول حساب نگهداری شود و یک مدل سه جدولی ایجاد گردد. پس در این حالت طراحی بهینه این است که صفات موجودیت حساب در جدول حساب، صفات موجودیت حساب جاری در جدول حساب جاری و صفات موجودیت حساب قرض الحسنه در جدول حساب قرض الحسنه قرار داده شود و یک طراحی به شکل مدل سه جدولی اما با تحمل سربار حاصل از عمل الحاق ایجاد گردد و جهت ارتباط جداول حساب جاری و حساب قرض الحسنه با جدول حساب کلید کاندید جدول حساب در جداول حساب جاری و حساب قرض الحسنه به عنوان کلید خارجی تعریف گردد.

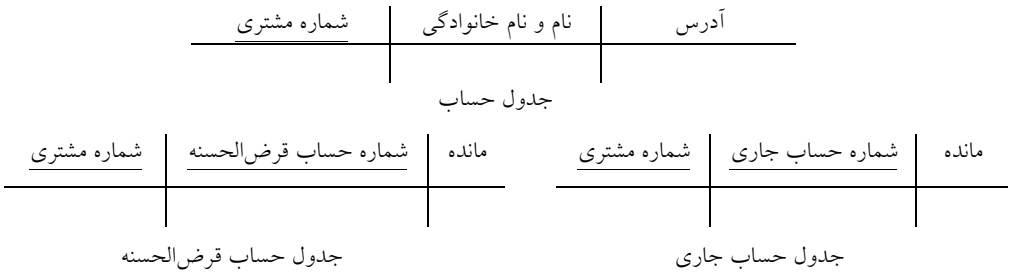
توجه: اگر شماره مشتری در یک بانک باشد، ممکن است برای آن حساب جاری یا حساب قرض‌الحسنه موجود نباشد، البته اگر نظر طراح بانک اطلاعات بر این باشد که شماره مشتری‌های فاقد حساب در بانک اطلاعات باقی بمانند. در بانکداری ایران هیچ‌گاه شماره مشتری‌ها (حتی فاقد حساب) پاک نمی‌شوند، هم‌چنین حتی پس از بستن حساب از آن‌جا که شماره مشتری‌ها باقی

می‌مانند، تمامی تراکنش‌های انجام شده، همچنان موجود و قابل دسترسی برای بانکداران می‌باشد. در این شرایط شماره مشتری از موجودیت حساب با موجودیت حساب جاری یا موجودیت حساب قرض‌الحسنه رابطه اختیاری یا جزئی (Partial) دارد. اما اگر یک شماره مشتری از موجودیت حساب بخواهد صاحب حساب جاری یا حساب قرض‌الحسنه هم باشد، می‌تواند چندین نوع حساب جاری یا حساب قرض‌الحسنه داشته باشد. این یعنی پوشا یا تخصیص غیرمجزا (Overlap) بین حساب جاری و حساب قرض الحسنه.

مدل تحلیل (نمودار ISA)



مدل طراحی (مدل رابطه‌ای)



توجه: جداول فوق را می‌توان به این صورت تفسیر معنایی نمود که، هنگام ایجاد یک حساب جاری یا حساب قرض‌الحسنه در پایگاه داده، اطلاعات بسته به نوع حساب به دو بخش اطلاعات عمومی و اطلاعات اختصاصی تقسیم می‌شود. البته اطلاعات عمومی حساب در جدول حساب درج می‌شود، سپس اطلاعات اختصاصی در یکی از جداول حساب جاری یا حساب قرض‌الحسنه متناسب با نوع حساب درج می‌شود و در نهایت دو سطر درج شده در دو جدول پدر و فرزند از طریق شماره مشتری، که کلید اصلی جدول حساب و کلید خارجی جداول حساب جاری یا حساب قرض‌الحسنه است به یک‌دیگر مرتبط می‌شوند. به این ترتیب می‌توان گفت که اطلاعات

عمومی حساب در جدول حساب و اطلاعات اختصاصی حساب جاری یا حساب قرض‌الحسنه در جداول حساب جاری و حساب قرض‌الحسنه قرار دارند.

توجه: نگاشت فوق، یک راه‌حل غیربهینه و ایده‌آل نیز دارد، بدین شکل که برای موجودیت پدر، جدول در نظر گرفته نشود و فقط برای موجودیت‌های فرزند جدول در نظر گرفته شود و تمام صفات پدر در جدول‌های فرزند درج گردند. این رویکرد، ساده‌ترین راه‌حل محسوب می‌گردد، اما چنان‌چه گفته شد، راه‌حل غیربهینه و دارای تکرار اطلاعات و به تبع افزونگی است.

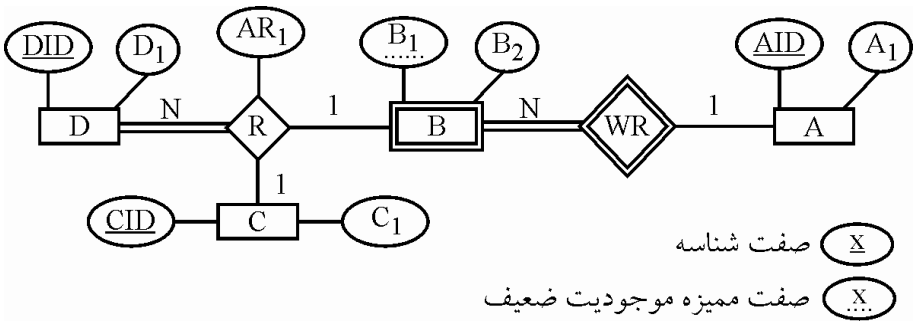
مانده	شماره حساب جاری	آدرس	نام و نام‌خانوادگی	شماره مشتری

جدول حساب جاری

مانده	شماره حساب قرض‌الحسنه	آدرس	نام و نام‌خانوادگی	شماره مشتری

جدول حساب قرض‌الحسنه

مثال: کدام مورد، شامل طراحی منطقی درست ارتباط R در نمودار ER زیر است؟



۱) $R(\underline{DID, AID, B_1, CID}, AR_1)$ به صورت یک جدول مستقل
P.K.

۲) $B(\underline{B_1}, B_2, \underline{DID}, \underline{CID}, AR_1)$ مستتر در جدول موجودیت B
P.K. F.K. F.K.

۳) $C(\underline{CID}, C_1, \underline{DID}, \underline{AID, B_1}, AR_1)$ مستتر در جدول موجودیت C
P.K. F.K. F.K.

۴) $D(\underline{DID}, D_1, \underline{CID}, \underline{AID, B_1}, AR_1)$ مستتر در جدول موجودیت D
P.K. F.K. F.K.

پاسخ: گزینه (۴) صحیح است.

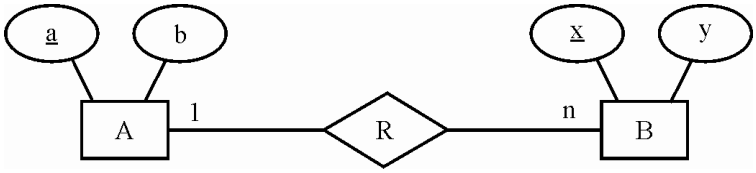
به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در نمودار ER (مدل تحلیل) هنگام نگاشت به مدل رابطه‌ای (مدل طراحی) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

نگاشت رابطه یک به چند بین دو موجودیت به مدل رابطه‌ای

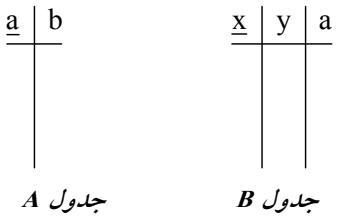
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جدول یک در جدول چند به عنوان کلید خارجی تعریف می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:



جدول A ***جدول B***

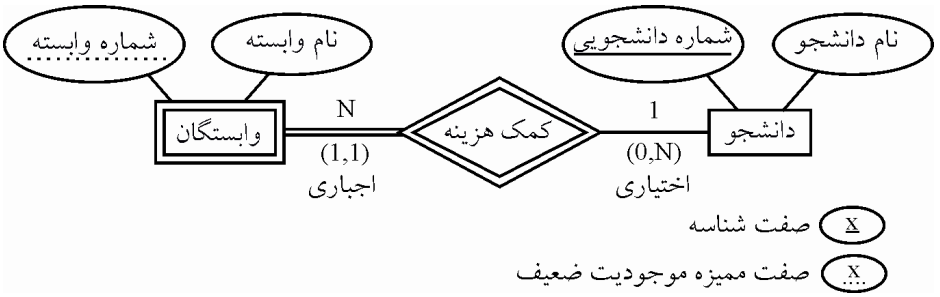
وابستگی وجودی

اگر در یک بانک اطلاعاتی، وجود یک موجودیت، وابسته به موجودیت دیگری باشد که در صورت حذف و تغییر موجودیت اصلی یعنی موجودیت قوی این موجودیت نیز تغییر کند، این نوع وابستگی را وابستگی وجودی گفته و به پدیده وابسته، **موجودیت ضعیف** گویند. همچنین **موجودیت ضعیف** کلید **موجودیت قوی** را در بر دارد تا هرگونه تغییر یا حذف در موجودیت قوی به موجودیت ضعیف اعمال شود.

توجه: موجودیت ضعیف با دو مستطیل تو در تو نمایش داده می‌شود.

مثال: حالت اجباری و اختیاری

مدل تحلیل:



توجه: نماد خط مضاعف افقی نشانه اجباری بودن موجودیت چسبیده به آن است، اما نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است، اما نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

توجه: قید (0,N) نشان می‌دهد که هر نمونه موجودیت از دانشجو حداقل با صفر و حداکثر با N نمونه موجودیت از وابستگان ارتباط دارد و قید (1,1) نشان می‌دهد که هر نمونه موجودیت از وابستگان حداقل با یک و حداکثر با یک نمونه موجودیت از دانشجو ارتباط دارد.

مدل طراحی:

S#	Sname

جدول دانشجو

S#	SD#	SDname

جدول وابستگان

توجه: کلید کاندید جدول یک یعنی **موجودیت قوی** در جدول چند یعنی **موجودیت ضعیف** به عنوان کلید خارجی تعریف می‌گردد.

توجه: کلید کاندید جدول چند یعنی **موجودیت ضعیف** برابر ترکیب **کلید خارجی** و **صفت ممیزه** در جدول موجودیت ضعیف است. یعنی کلید کاندید جدول وابستگان برابر (S#,SD#) است.

توجه: صفت ممیزه یا **کلید جزئی** به طور سراسری در یک موجودیت ضعیف یکتا نیست، بلکه فقط در بین نمونه‌ها یا دسته‌هایی که با موجودیت قوی ارتباط دارند، یکتا است.

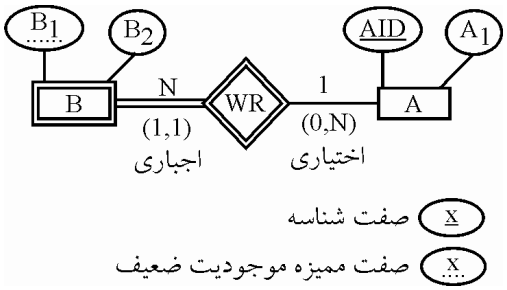
توجه: یک موجودیت ضعیف **همیشه** در ارتباطش با موجودیت قوی رابطه **اجباری** دارد.

بخشی از مدل EER رسم شده در صورت سوال، یک رابطه یک به چند بین دو موجودیت قوی و

ضعیف را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

حالت اجباری و اختیاری

مدل تحلیل:



توجه: در شکل فوق صفت AID کلید موجودیت قوی A و صفت B_1، صفت ممیزه موجودیت ضعیف B است.

توجه: نماد خط مضاعف افقی نشانه اجباری بودن موجودیت چسبیده به آن است، اما نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است، اما نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

توجه: قید (0,N) نشان می‌دهد که هر نمونه موجودیت از A حداقل با صفر و حداکثر با N نمونه موجودیت از B ارتباط دارد و قید (1,1) نشان می‌دهد که هر نمونه موجودیت از B حداقل با یک و حداکثر با یک نمونه موجودیت از A ارتباط دارد.

مدل طراحی:

AID	B_1	B_2

جدول B

AID	A_1

جدول A

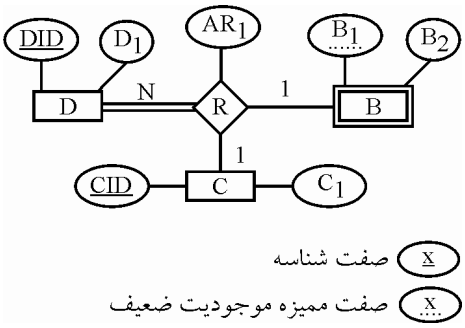
توجه: کلید کاندید جدول یک یعنی **موجودیت قوی** در جدول چند یعنی **موجودیت ضعیف** به عنوان کلید خارجی تعریف می‌گردد.

توجه: کلید کاندید جدول چند یعنی **موجودیت ضعیف** برابر ترکیب **کلید خارجی** و **صفت ممیزه** در جدول موجودیت ضعیف است. یعنی کلید کاندید جدول B برابر (AID,B_1) است.

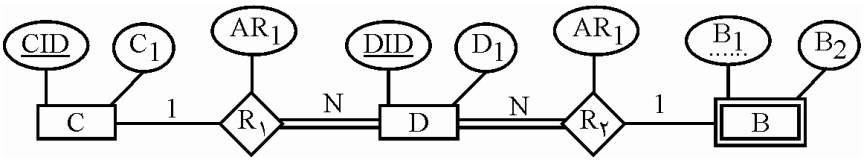
توجه: صفت ممیزه یا **کلید جزئی** به طور سراسری در یک موجودیت ضعیف یکتا نیست، بلکه فقط در بین نمونه‌ها یا دسته‌هایی که با موجودیت قوی ارتباط دارند، یکتا است.
توجه: یک موجودیت ضعیف **همیشه** در ارتباطش با موجودیت قوی رابطه **اجباری** دارد.
همچنین در ادامه نگاشت، بخشی از مدل EER رسم شده در صورت سوال، یک رابطه یک به چند و یک به یک بین سه موجودیت را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

نگاشت رابطه یک به چند و یک به یک بین سه موجودیت به مدل رابطه‌ای
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جداول یک در جدول چند به عنوان کلید خارجی تعریف می‌گردند. همچنین صفات متصل به **رابطه**، درون جدول چند **مستتر** می‌شود.

مدل تحلیل:



مدل طراحی:



CID	C_1

جدول C

DID	D_1	CID	AID	B_1	AR_1

جدول D

AID	B_1	B_2

جدول B

توجه: ستون CID در جدول D به عنوان کلید خارجی تعریف می‌گردد که به جدول C ارجاع می‌کند. همچنین ستون‌های (AID,B_1) در جدول D به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند.

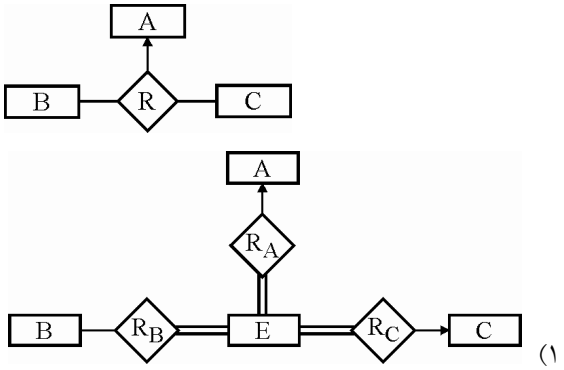
توجه: کلید کاندید جدول D فقط و فقط کلید کاندید موجودیت D یعنی صفت DID است.

توجه: جدول B در نگاشت مرحله قبل ایجاد شده است که در اینجا دقیقا به همان شکل و همان مشخصات، استفاده شده است.

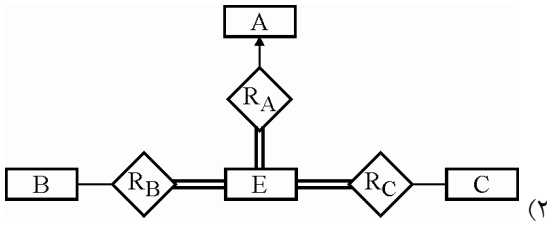
توجه: نتیجه اینکه رابطه درجه سوم یا ارتباط سه‌گانی R، به دو رابطه درجه دوم یا ارتباط دوگانی توسط تعریف کلید خارجی تبدیل شد.

توجه: همانطور که واضح است، طراحی جدول D در گزینه چهارم به صورت ($\underline{DID},D_1,\underline{CID},\underline{AID},\underline{B_1},AR_1$) در نظر گرفته شده است، که مطابق آنچه بیان کردیم، طراحی درستی است. بنابراین پرواضح است که گزینه چهارم پاسخ سوال است.

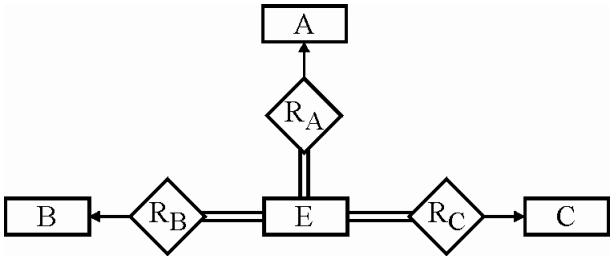
مثال: فرض کنید که رابطه سه‌تایی زیر بین موجودیت‌های A, B, C وجود دارد، حال اگر بخواهیم این رابطه سه‌تایی را با رابطه‌های دودویی نمایش دهیم، کدام‌یک از نمودارهای موجودیت و رابطه (ERD) زیر دقیقا معادل با این رابطه سه‌تایی می‌باشد؟



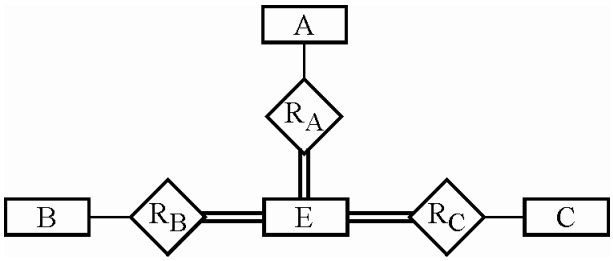
۱)



۲)



۳)



۴)

پاسخ: گزینه (۳) صحیح است.

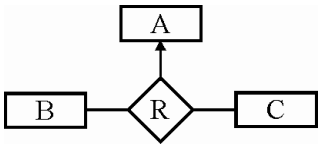
به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در نمودار ER (مدل تحلیل) هنگام نگاشت به مدل رابطه‌ای (مدل طراحی) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

مدل ER رسم شده در صورت سوال، یک رابطه چند به چند و یک به چند بین سه موجودیت را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

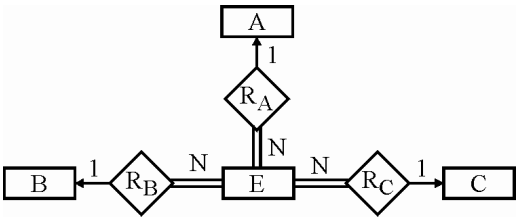
نگاشت رابطه چند به چند و یک به چند بین سه موجودیت به مدل رابطه‌ای

مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط دهنده سه جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید سه جدول دیگر ایجاد می‌گردد. همچنین صفات متصل به **رابطه** در صورت وجود، درون جدول پُل **مستتر** می‌شود.

مدل تحلیل:



مدل طراحی:



a	…

جدول A

b	…

جدول B

a	b	c	x

جدول ABC

c	…

جدول C

توجه: ستون a در جدول ABC به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون b در جدول ABC به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند. همچنین ستون c در جدول ABC به عنوان کلید خارجی تعریف می‌گردد که به جدول C ارجاع می‌کند.

توجه: کلید کاندید جدول ABC برابر abc است.

توجه: ستون x، می‌تواند به عنوان یک صفت در ارتباط در جدول ABC باشد.

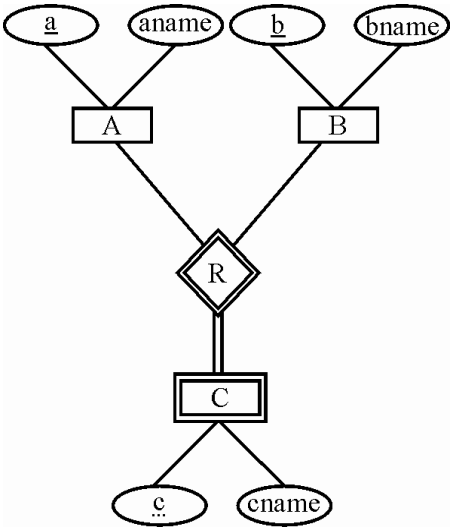
توجه: نتیجه اینکه رابطه درجه سوم یا ارتباط سه‌گانی R، به سه رابطه درجه دوم یا ارتباط دوگانی توسط تعریف کلید خارجی تبدیل شد.

توجه: همانطور که واضح است، مدل طراحی در گزینه سوم به صورت شکل فوق در نظر گرفته شده‌است، که مطابق آنچه بیان کردیم، طراحی درستی است. بنابراین پرواضح است که گزینه سوم پاسخ سوال است.

توجه: موجودیت ضعیف وابسته به یک موجودیت قوی (مانند E_1) می‌تواند خود با موجودیت قوی دیگری مانند (مانند E_2) رابطه داشته باشد.»

مثال: فرض کنید که موجودیت ضعیف C، با دو موجودیت قوی A و B رابطه داشته باشد، نمودار موجودیت و رابطه (ERD) یعنی مدل تحلیل و مدل طراحی آن به چه شکل خواهد بود؟

مدل تحلیل:



توجه: در شکل فوق صفت a کلید موجودیت قوی A و صفت b کلید موجودیت قوی B است، همچنین صفت c، صفت ممیزه موجودیت ضعیف C است.

توجه: نماد خط مضاعف افقی نشانه اجباری بودن موجودیت چسبیده به آن است، اما نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است، اما نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

مدل طراحی:

a	aname	…

جدول A

a	b	c	cname

جدول C

b	bname	…

جدول B

توجه: کلیدهای کاندید دو جدول **موجودیت قوی** در جدول **موجودیت ضعیف** به عنوان کلید خارجی تعریف می‌گردند.

توجه: ستون a در جدول C به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون b در جدول C به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند.

توجه: کلید کاندید جدول **موجودیت ضعیف** برابر ترکیب **کلیدهای خارجی** و **صفت ممیزه** در جدول موجودیت ضعیف است. یعنی کلید کاندید جدول C برابر (a,b,c) است.

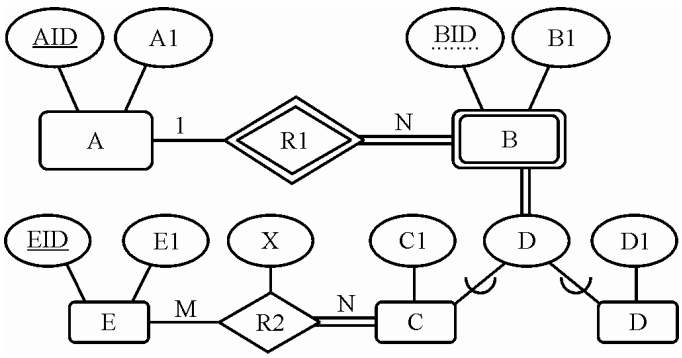
توجه: صفت ممیزه یا **کلید جزئی** به طور سراسری در یک موجودیت ضعیف یکتا نیست، بلکه فقط در بین نمونه‌ها یا دسته‌هایی که با موجودیت قوی ارتباط دارند، یکتا است.

توجه: یک موجودیت ضعیف **همیشه** در ارتباطش با موجودیت قوی رابطه **اجباری** دارد.

توجه: اگر یک نوع موجودیت (مانند E_1) تنها یک صفت داشته باشد و تنها با یک نوع موجودیت دیگر (مانند E_2) در ارتباط باشد، می‌توانیم آن را حذف و به عنوان صفت موجودیت دوم (یعنی E_2) در نظر بگیریم. زیرا بهترین راه شناسایی موجودیت‌های یک محیط عملیاتی، بررسی مقادیری است که قرار است ذخیره شود، ذخیره‌سازی مقادیر داخل صفات (ستون‌ها) انجام می‌شود و این صفات قطعا مرتبط با یک موجودیت هستند. بنابراین هر موجودیت باید دارای چندین صفت باشد. پس موجودیت فاقد صفت معنا ندارد. حتی در اغلب مواقع، موجودیت تک خصیصه‌ای نیز قابل قبول نمی‌باشد و به احتمال فراوان آن صفت مرتبط به یک موجودیت دیگر است که به خطا به عنوان یک موجودیت تک خصیصه‌ای شناخته شده است.

مثال: با فرض وجود نمودار EER زیر کدام‌یک از رابطه‌های زیر به درستی طراحی نشده است؟

A (AID, A1) B (BID, B1) E (EID, E1, AID, BID)
C (AID, BID, B1, C1) D (AID, BID, B1, D1) R2 (AID, BID, EID, X)



۱) A, B
۲) B, E
۳) C, D
۴) E, C

پاسخ: گزینه (۲) صحیح است.

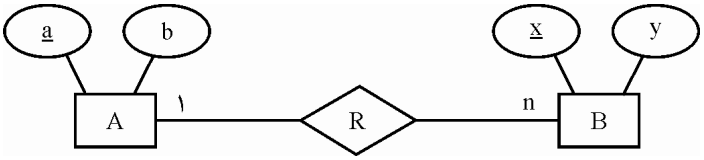
به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در نمودار ER (مدل تحلیل) هنگام نگاشت به مدل رابطه‌ای (مدل طراحی) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

نگاشت رابطه یک به چند بین دو موجودیت به مدل رابطه‌ای

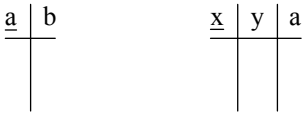
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جدول یک در جدول چند به عنوان کلید خارجی تعریف می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:



جدول A ***جدول B***

وابستگی وجودی

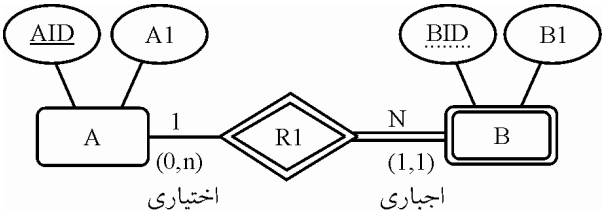
اگر در یک بانک اطلاعاتی، وجود یک موجودیت، وابسته به موجودیت دیگری باشد که در صورت حذف و تغییر موجودیت اصلی یعنی موجودیت قوی این موجودیت نیز تغییر کند، این نوع وابستگی را وابستگی وجودی گفته و به پدیده وابسته، **موجودیت ضعیف** گویند. همچنین **موجودیت ضعیف** کلید **موجودیت قوی** را در بر دارد تا هرگونه تغییر یا حذف در موجودیت قوی به موجودیت ضعیف اعمال شود.

توجه: موجودیت ضعیف با دو مستطیل تو در تو نمایش داده می‌شود.

بخشی از مدل EER رسم شده در صورت سوال، یک رابطه یک به چند بین دو موجودیت قوی و ضعیف را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

حالت اجباری و اختیاری

مدل تحلیل:



توجه: در شکل فوق صفت AID کلید موجودیت قوی A و صفت BID، صفت ممیزه موجودیت ضعیف B است.

توجه: نماد خط مضاعف افقی نشانه اجباری بودن موجودیت چسبیده به آن است، اما نماد | به معنی یک و الزام شرکت در رابطه نشانه اجباری بودن موجودیت طرف مقابل است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است، اما نماد دایره کوچک توخالی به معنی صفر و عدم الزام شرکت در رابطه نشانه اختیاری بودن موجودیت طرف مقابل است.

توجه: قید (0,N) نشان می‌دهد که هر نمونه موجودیت از A حداقل با صفر و حداکثر با N نمونه موجودیت از B ارتباط دارد و قید (1,1) نشان می‌دهد که هر نمونه موجودیت از B حداقل با یک و حداکثر با یک نمونه موجودیت از A ارتباط دارد.

مدل طراحی:

AID	A1

جدول A

AID	BID	B1

جدول B

توجه: کلید کاندید جدول یک یعنی **موجودیت قوی** در جدول چند یعنی **موجودیت ضعیف** به عنوان کلید خارجی تعریف می‌گردد.

توجه: کلید کاندید جدول چند یعنی **موجودیت ضعیف** برابر ترکیب **کلید خارجی** و **صفت ممیزه** در جدول موجودیت ضعیف است. یعنی کلید کاندید جدول B برابر (AID,BID) است.

توجه: **صفت ممیزه** یا **کلید جزئی** به طور سراسری در یک موجودیت ضعیف یکتا نیست، بلکه فقط در بین نمونه‌ها یا دسته‌هایی که با موجودیت قوی ارتباط دارند، یکتا است.

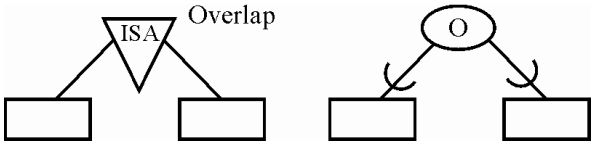
توجه: یک موجودیت ضعیف **همیشه** در ارتباطش با موجودیت قوی رابطه **اجباری** دارد.

توجه: همانطور که واضح است، طراحی جدول B در صورت سوال به صورت (BID,B1) در نظر گرفته شده‌است، که مطابق آنچه بیان کردیم، طراحی درست آن به صورت (AID,BID,B1) است. همچنین طراحی جدول A در صورت سوال به صورت (AID,A1) در نظر گرفته شده‌است، که مطابق آنچه بیان کردیم، طراحی درست آن هم به صورت (AID,A1) است. بنابراین گزینه‌های اول، سوم و چهارم را کنار می‌گذاریم، پس تا همینجا واضح است که گزینه دوم پاسخ سوال است.

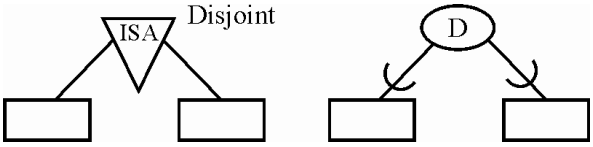
نگاشت رابطه ISA به مدل رابطه‌ای

در رابطه ISA رابطه پدر با فرزندان به دو صورت رابطه اختیاری یا جزئی (Partial) با نماد خط عمودی و رابطه اجباری یا کلی (Total) با نماد خط مضاعف عمودی است و رابطه فرزندان با پدر به دو صورت رابطه پوشا یا تخصیص غیرمجزا (Overlap) و رابطه غیرپوشا یا تخصیص مجزا (Disjoint) می‌باشد.

رابطه پوشا یا تخصیص غیرمجزا (Overlap) مابین فرزندان و پدر به دو شیوه زیر نشان داده می‌شود:



رابطه غیرپوشا یا تخصیص مجزا (Disjoint) مابین فرزندان و پدر به دو شیوه زیر نشان داده می‌شود:



بخشی از مدل EER رسم شده در صورت سوال، یک رابطه اجباری یا کلی (Total) مابین موجودیت B و موجودیت‌های C و D و یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint) را مابین موجودیت‌های C و D و موجودیت B نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را بیان می‌کنیم.

در یک رابطه اجباری یا کلی (Total)، هر نمونه از موجودیت پدر حتماً می‌بایست با یکی از نمونه موجودیت‌های فرزند در ارتباط باشد. برای‌مثال در این سؤال، هر نمونه از موجودیت B حتماً می‌بایست با یکی از نمونه موجودیت‌های C یا D در ارتباط باشد. به عبارت دیگر نمی‌توان نمونه‌ای از موجودیت B داشت که با هیچ یک از نمونه موجودیت‌های C یا D در ارتباط نیست.

هم‌چنین در یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint)، نمونه موجودیت‌های فرزند نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند. برای‌مثال در این سؤال، نمونه موجودیت‌های C و D نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت B در ارتباط باشند. به عبارت دیگر نمی‌توان نمونه‌هایی از موجودیت‌های C و D داشت که به طور همزمان با نمونه‌ای از موجودیت B در ارتباط هستند.

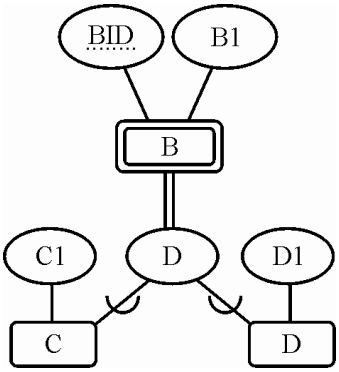
به عنوان مثالی دیگر هر نمونه از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خـود را دارد حتماً می‌بایست با یکی از نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است در ارتباط باشد. به عبارت دیگر نمی‌توان نمونه‌ای از موجودیت ماشـین کـه شـماره شاسی یکتا و منحصر به فرد خود را دارد داشت که با هیچ یک از نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است در ارتباط نباشد. این یعنی رابطه اجباری یا کلـی (Total).

همچنین نمونه موجودیت‌های نـوع ماشـین کـه دومحـور (2WD) یـا چهـارمحور (4WD) اسـت نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت ماشین که شماره شاسی یکتا و منحصر بـه فـرد خود را دارد در ارتباط باشند، به عبارت دیگر نمی‌توان نمونه‌هایی از موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است داشت که به طور همزمان با نمونـه‌ای از موجودیـت ماشین که شماره شاسی یکتا و منحصر به فـرد خـود را دارد در ارتبـاط باشـند. ایـن یعنـی رابطـه غیرپوشا یا تخصیص مجزا (Disjoint).

از آنجاکه رابطه مابین موجودیت B و موجودیت‌هـای C و D یـک رابطـه اجبـاری یـا کلـی (Total) است، پس رکوردهای حاوی محتـوای مقـدار NULL درسـتون‌هـای مربـوط بـه موجودیـت B در طراحی به شکل مدل دو جدولی در جداول C و D به ازای یک نمونـه موجودیـت از B بـه دلیـل عدم ارتباط با برخی از نمونه موجودیت‌های C و D ایجاد نمی‌گردد، که باعث شـود ایـن محتـوای NULL در جداول C و D حاصل از عدم ارتباط برخی از نمونه موجودیت‌هـای موجودیـت B بـا نمونه موجودیت‌های C و D در جدول B، به شکل مدل سه جـدولی نگهـداری شـود. در حالـت رابطه اجباری مابین موجودیت B و موجودیت‌هـای C و D بـه ازای هـر نمونـه از موجودیـت B، حتما نمونه موجودیتی از C یا D وجود دارد که با B رابطه برقرار کند، پس در این حالت طراحـی بهینه این است که کل صفات موجودیت B در دو جدول موجودیت‌های C و D قـرار داده شـود و یک طراحی به شکل مدل دو جدولی ایجاد گردد، همچنین از آنجاکه رابطه مابین موجودیـت‌هـای C و D و موجودیت B یک رابطه غیرپوشا یا تخصیص مجـزا (Disjoint) اسـت، پـس رکوردهـای تکراری در جداول C و D به ازای یک نمونه موجودیـت از موجودیـت B ایجـاد نمـی‌گـردد، کـه افزونگی حاصل از تکرار رکوردها در جداول C و D سبب شود رکورد نمونه موجودیـت‌هـای B در جدول B نگهداری شود و یک مدل سه جدولی ایجاد گردد. پس در این حالـت طراحـی بهینـه این است که کل صفات موجودیت B در دو جدول موجودیت های C و D قرار داده شـود و یـک طراحی به شکل مدل دو جدولی ایجاد گردد.

در ادامه فرآیند نگاشت نمودار (ISA) به مدل رابطه‌ای را شرح می‌دهیم:

مدل تحلیل (نمودار (ISA))



مدل طراحی (مدل رابطه‌ای)

همانطور که در مدل طراحی قبل‌تر گفتیم، مدل طراحی درست جدول B به صورت (AID,BID,B1) در نظر گرفته شد.

AID	BID	B1	C1

جدول C

AID	BID	B1	D1

جدول D

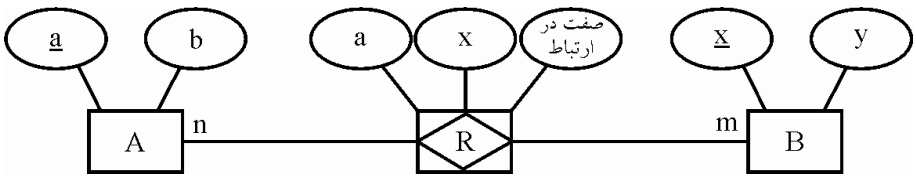
توجه: چون رابطه موجودیت B با موجودیت‌های C و D اجباری است، و رابطه موجودیت‌های C و D با موجودیت B از نوع Disjoint است. و به ازای هر نمونه موجودیت از B حتماً یک نمونه موجودیت از C و یا D وجود دارد و به تبع عدم مقادیر NULL جلوی نمونه موجودیت‌های B و عدم نیاز به عمل‌الحاق، نگاشت فوق به عنوان یک تبدیل ایده‌آل توصیه می‌گردد.

نگاشت رابطه چند به چند بین دو موجودیت به مدل رابطه‌ای

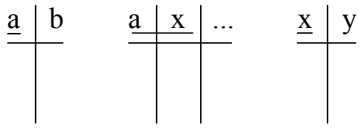
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

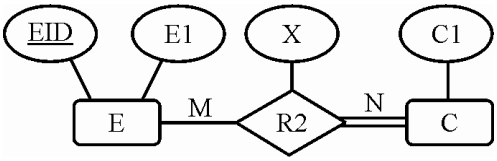


جدول A *جدول AB* *جدول B*

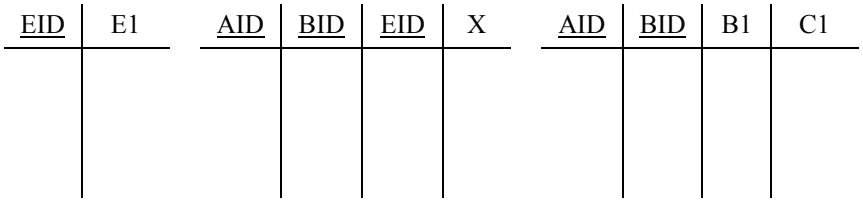
توجه: ستون a در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون x در جدول AB به عنوان کلید خارجی تعریف می‌گردد که به جدول B ارجاع می‌کند.

بخشی از مدل EER رسم شده در صورت سوال، یک رابطه چند به چند بین دو موجودیت را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

مدل تحلیل:



مدل طراحی:

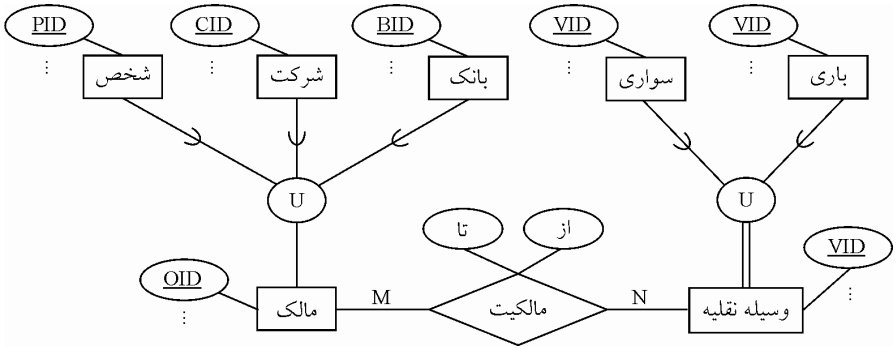


جدول E *جدول R2* *جدول C*

توجه: جدول C در نگاشت مرحله قبل ایجاد شده است که در اینجا دقیقاً به همان شکل و همان مشخصات، استفاده شده است.

توجه: همانطور که واضح است، طراحی جدول E در صورت سوال به صورت (EID,E1,AID,BID) در نظر گرفته شده‌است، که مطابق آنچه بیان کردیم، طراحی درست آن به صورت (EID,E1) است. بنابراین پرواضح است که گزینه دوم پاسخ سوال است.

مثال- مدل رابطه‌ای متناظر با نمودار ER زیر به چه صورتی است؟

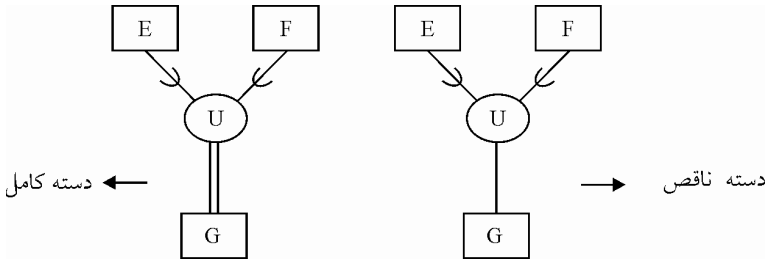


به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در نمودار ER (مدل تحلیل) هنگام نگاشت به مدل رابطه‌ای (مدل طراحی) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

زیرنوع اجتماع (U-Type) یا دسته (Category)

زیرنوع موجودیت G را زیرنوع U-Type زیرنوع‌های E,F,... گوییم، هرگاه در مجموعه نمونه‌های G نمونه‌هایی (اگر همه نمونه‌ها یعنی دسته کامل و اگر بعضی نمونه‌ها یعنی دسته ناقص) از E,F,... وجود داشته باشد. در واقع نمایانگر اجتماعی از نمونه‌هایی از انواع مختلف است.

توجه: یک نمونه از زیرنوع اجتماع (دسته)، بسته به اینکه از نوع کدام زیرنوع باشد، صفات همان زیرنوع را به ارث می‌برد.



توجه: شناسه‌های زیرنوع‌ها می‌تواند از دامنه‌های متفاوت باشد، به صورت زیر:

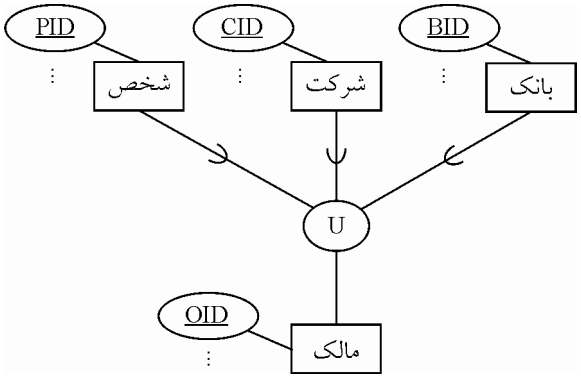
فرم متفاوت: شناسه زیرنوع شناسه‌ای است که خود باید در نظر بگیریم.

فرم یکسان: شناسه زیرنوع می‌تواند همان شناسه زبرنوع‌ها باشد.

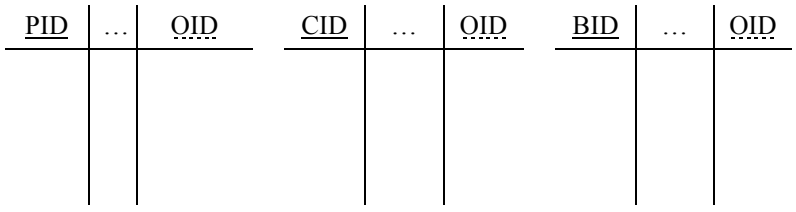
نگاشت رابطه زیرنوع U-Type ناقص و زبرنوع‌ها به مدل رابطه‌ای

هر موجودیت به یک جدول تبدیل می‌گردد، اگر شناسه زبرنوع‌ها از دامنه‌های متفاوت باشد، کلید کاندید موجودیت زیرنوع به عنوان کلیدخارجی در موجودیت زبرنوع‌ها تعریف می‌شود.

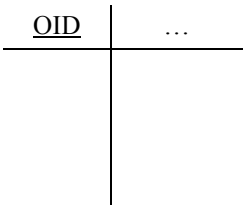
مدل تحلیل:



مدل طراحی:



جدول Pers ***جدول Comp*** ***جدول Bank***



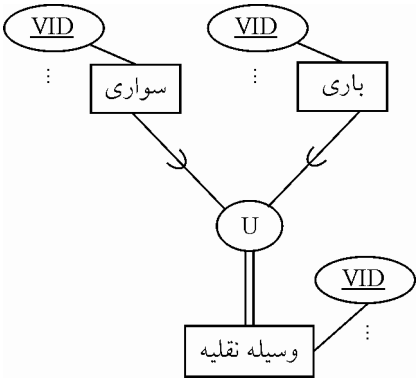
جدول Owner

توجه: ستون OID در جدول‌های Pers، Comp و Bank به عنوان کلید خارجی تعریف می‌گردد که به جدول Owner ارجاع می‌کند.

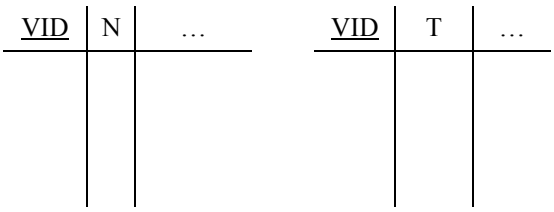
نگاشت رابطه زیرنوع U-Type کامل و زیرنوع‌ها به مدل رابطه‌ای

هر موجودیت به یک جدول تبدیل می‌گردد، اگر شناسه زیرنوع‌ها از یک دامنه باشد و مقادیر شناسه در همه نمونه‌های زیرنوع‌ها یکتا باشد، کلید رابطه نمایشگر زیرنوع، همان کلید رابطه‌های نمایشگر زیرنوع‌ها است.

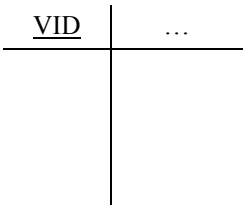
مدل تحلیل:



مدل طراحی:



جدول Savary ***جدول Bary***



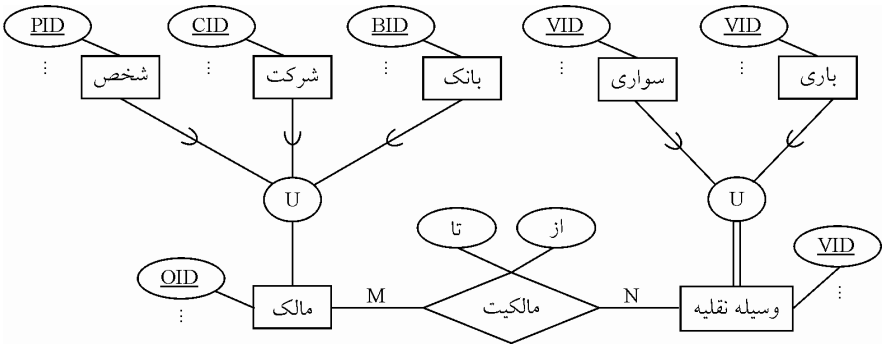
جدول Vehic

نگاشت رابطه چند به چند بین دو موجودیت به مدل رابطه‌ای

مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد. همچنین صفات متصل به **رابطه**، درون جدول پُل **مستتر** می‌شود.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



توجه: در شکل فوق صفت OID کلید موجودیت Owner و صفت VID کلید موجودیت Vehic است.

مدل طراحی:

OID	…

جدول Owner

OID	VID	F	T

جدول Owns

VID	…

جدول Vehic

توجه: ستون OID در جدول Owns به عنوان کلید خارجی تعریف می‌گردد که به جدول Owner ارجاع می‌کند. همچنین ستون VID در جدول Owns به عنوان کلید خارجی تعریف می‌گردد که به جدول Vehic ارجاع می‌کند.

توجه: همچنین صفات متصل (مالکیت) به **رابطه**، درون جدول پُل یعنی جدول Owns **مستتر** می‌شود.

توجه: همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد. یعنی کلید کاندید جدول Owns برابر (OID,VID) است.

توجه: کلید کاندید جدول چند چپ یعنی **موجودیت Owner** برابر همان **کلید کاندید سابق** در جدول موجودیت Owner است. یعنی کلید کاندید جدول Owner برابر (OID) است.

توجه: کلید کاندید جدول چند راست یعنی **موجودیت Vehic** برابر همان **کلید کاندید سابق** در جدول موجودیت Vehic است. یعنی کلید کاندید جدول Vehic برابر (VID) است.

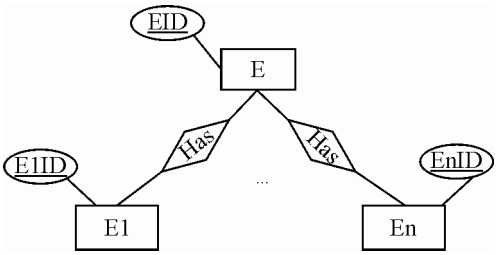
توجه: جدول Owner در نگاشت مرحله قبل ایجاد شده است که در اینجا دقیقاً به همان شکل و همان مشخصات، استفاده شده است.

توجه: جدول Vehic در نگاشت مرحله قبل ایجاد شده است که در اینجا دقیقاً بـه همـان شـکل و همان مشخصات، استفاده شده است.

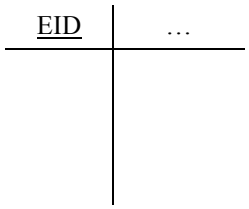
نگاشت رابطه IS-A-PART-OF به مدل رابطه‌ای

هر موجودیت به یک جدول تبدیل می‌گردد و کلید کاندید موجودیت زبرنـوع در موجودیـت‌هـای زیرنوع به عنوان کلید خارجی تعریف می‌گردد.

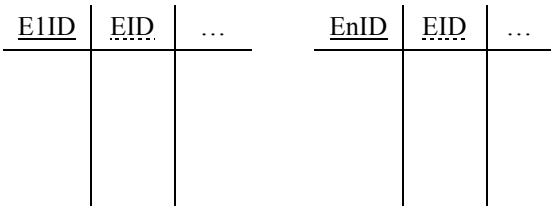
مدل تحلیل:



مدل طراحی:



جدول E



جدول E1 ***جدول En***

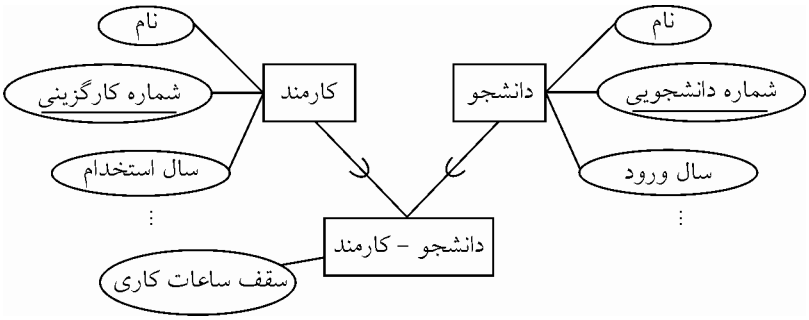
توجه: موجودیت کل (E) به طور مستقل برای خود کلید کاندید دارد و همچنـین موجودیـت‌هـای جز (E1 و En) به طور مستقل برای خود کلید کاندید دارند.

توجه: ستون EID در جدول‌های E1 و Enk به عنوان کلید خارجی تعریف می‌گردد که به جـدول E ارجاع می‌کند.

نگاشت رابطه ارث‌بری چندگانه به مدل رابطه‌ای

هر موجودیت به یک جدول تبدیل می‌گردد و کلید کاندید موجودیت‌های زیرنوع در موجودیت زیرنوع به عنوان کلید خارجی تعریف می‌گردد.

مدل تحلیل:



مدل طراحی:

EID	Ename	…

جدول Empl

STID	STname	..

جدول Stud

EID	STID	MAXW

جدول Stem

همانطور که گفتیم مدل رابطه‌ای از دو بخش زیر تشکیل شده‌است:

۱- رابطه یا جدول

۲- قوانین مدل رابطه‌ای

رابطه یا جدول بررسی شد، در ادامه به بررسی قوانین مدل رابطه‌ای می‌پردازیم:

قوانین مدل رابطه‌ای

همانطور که گفتیم به طور کلی جامعیت در سیستم‌های بانکی به دو طبقه‌ی جامعیت داخلی و خارجی تقسیم می‌گردد. به حفظ قوانین مطرح شده از سوی مدل رابطه‌ای و DBMS در سطح پیاده‌سازی، جامعیت داخلی و به حفظ قوانین مطرح شده از سوی طراحان و برنامه‌نویسان بانک در سطح پیاده‌سازی، جامعیت خارجی گفته می‌شود. در صورتی که در یک بانک جامعیت داخلی و خارجی هر دو توام باهم برقرار باشد، در آن بانک، اصل سازگاری برقرار شده است. به عبارت دیگر سازگاری، به معنی رعایت قوانین داخلی و خارجی در بانک است. DBMS مسئول کنترل قوانین داخلی و خارجی در بانک است و هرگونه عاملی که باعث نقض قوانین داخلی و خارجی و به تبع سازگاری بانک گردد را رد می‌کند. قوانین داخلی بانک شامل قانون جامعیت درون رابطه‌ای، قانون جامعیت موجودیت، قانون جامعیت ارجاعی و قانون جامعیت دامنه‌ای است، این موارد در ادامه بررسی خواهند شد. قوانین خارجی بانک هم شامل هر قانونی است که طراحان و برنامه‌نویسان بانک آنرا وضع می‌کنند، مانند تعریف زیردامنه برای ورود اطلاعات، تعریف بازه 0 تا 20 برای نمرات، تعریف بازه 0 تا بینهایت برای حساب‌های بانکی به معنی عدم وجود موجودی منفی در حساب‌های بانکی...

در سیستم‌های بانکی کنترل جامعیت داخلی و خارجی به صورت خودکار توسط مکانیزم‌های موجود در DBMS انجام می‌گردد.

حال یکبار دیگر کد تعریف جدول SP را در نظر بگیرید:

```
Create Table SP
        (
        S# char (5),
        P# char (5),
        QTY numeric (10),
        Primary key (S#, P#),
        Foreign key (S#) References S(S#)
          on delete cascade
          on update cascade,
        Foreign key (P#) References P(P#)
          on delete cascade
          on update cascade,
        Check (QTY>1 AND QTY<1000)
        )
```

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (S#) References S(S#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #S در جدول S به ستون #S در جدول SP نیز اعمال گردد. بنابراین جامعیت داخلی از نوع جامعیت ارجاعی نقض نمی‌گردد.

یا به طور مشابه، کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (P#) References P(P#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #P در جدول P به ستون #P در جدول SP نیز اعمال گردد. بنابراین جامعیت داخلی از نوع جامعیت ارجاعی نقض نمی‌گردد.

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Check (QTY>1 AND QTY<1000)
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه مقداردهی در ستون QTY از جدول SP در بازه 1 تا 1000 باشد، بنابراین جامعیت خارجی نقض نمی‌گردد.

در ادامه به بررسی قوانین جامعیت داخلی می‌پردازیم:

۱- قانون جامعیت درون رابطه‌ای[1]

مطابق قانون جامعیت درون رابطه‌ای، هر رابطه (جدول) باید به تنهایی درست باشد.

یعنی صفات رابطه اتومیک باشد (چند مقداری و مرکب نباشد) و همچنین هر رابطه باید حتماً حداقل دارای یک کلید کاندید باشد.

۲- قانون جامعیت موجودیت[2]

مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد.

۳- قانون جامعیت دامنه‌ای[3]

مطابق قانون جامعیت دامنه‌ای، صفات رابطه می‌بایست دارای نوع(دامنه) باشد و مقادیر صفات در محدوده همان دامنه باشد.

۴- قانون جامعیت ارجاعی[4]

مطابق قانون جامعیت ارجاعی، هیچگاه نباید کلید خارجی، دچار ارجاع NULL گردد. برای این منظور باید تعریف ساختاری و محتوایی کلید خارجی برقرار باشد. همانطور که گفتیم کلید

[1] Intra-Relational Integrity Rule
[2] Entity Integrity Rule
[3] Domain Integrity Rule
[4] Referential Integrity Rule

خارجی برای ارتباط میان جداول مورد استفاده قرار می‌گیرد. کلید خارجی در دو دیگاه ساختاری و محتوایی مورد بررسی قرار می‌گیرد:

دیدگاه ساختاری کلید خارجی

تعریف: اگر صفت(هایی) در یک جدول به عنوان کلید خارجی تعریف شود،

اول اینکه: این صفت(ها) در جدول خودش شرط خاصی ندارد یعنی الزاما خاصیت کلیدی ندارد.

و دوم اینکه: این صفت(ها) در همان جدول یا جدول دیگری، **باید کلید کاندید** (اصلی یا فرعی) باشد.

دیدگاه محتوایی کلید خارجی

به ازای هر مقدار موجود در یک کلید خارجی، باید دقیقاً یک مقدار متناظر در کلید کاندید متناظر آن وجود داشته باشد، در غیر این صورت می‌گوییم، کلید خارجی دارای ارجاع NULL است. به بیان دیگر، مقادیر کلید خارجی همواره باید زیرمجموعه مقادیر کلید کاندید باشد.

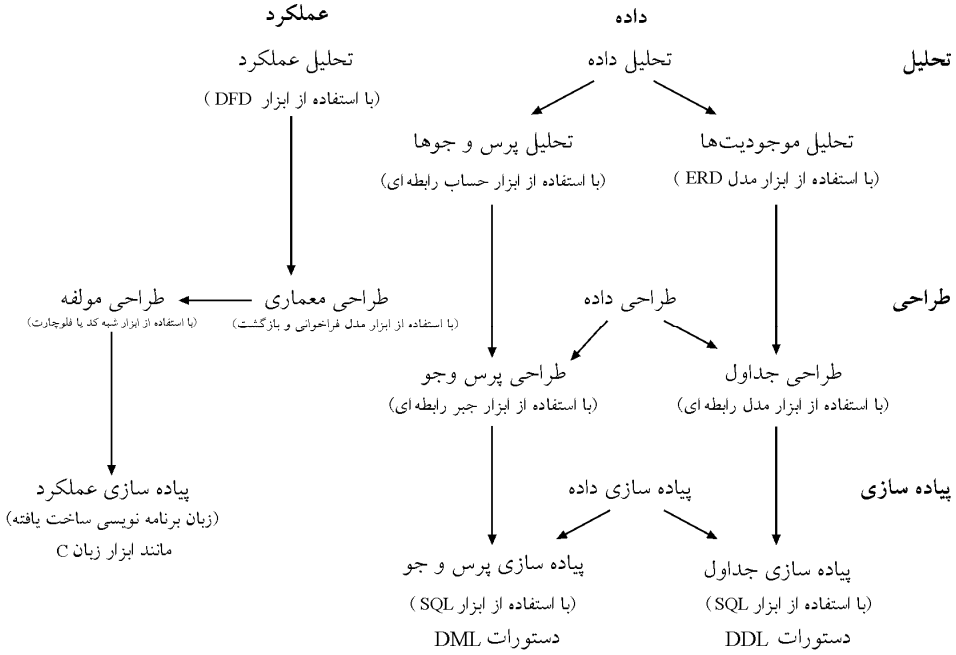
۴ جبر رابطه‌ای

مقدمه

یک محصول نرم‌افزاری به واسطه‌ی فرآیند تولید نرم‌افزار که شامل فعالیت‌های مدل تحلیل، مدل طراحی، پیاده‌سازی و تست می‌باشد، ایجاد می‌گردد.

مدل‌سازی

یک مدل، ساده شده یک واقعیت است. ایجاد یک مدل برای سیستم‌های نرم‌افزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری وحیاتی است. بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که باید ساخته شوند را ترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدل‌ها و توصیف‌ها ساخته شوند. مدل‌های خوب و دقیق در برقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می‌توانند داشته باشند. علت اصلی مدل کردن سیستم‌های پیچیده این است که نمی‌توان به یکباره کل سیستم را تجسم کرد و ممکن است سیستم دارای ابهامات بسیاری باشد. لذا برای رفع این ابهامات و نیز برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمت‌های مختلف آن، از مدل‌سازی استفاده می‌شود. مدل‌سازی خود شامل دو مرحله‌ی مدل تحلیل و مدل طراحی می‌باشد. مدل تحلیل قبل از مدل طراحی انجام می‌شود، در واقع خروجی مدل تحلیل، ورودی مدل طراحی می‌باشد. شکل زیر گویای این مطلب می‌باشد:



توجه: از آنجا که ما قصد داریم در این کتاب مفاهیم مربوط به پایگاه داده را تشریح نماییم، بنابراین از بیان مطلب مربوط به بخش عملکرد نرم‌افزار صرف‌نظر می‌نمائیم. بخش عملکرد را در کتاب مهندسی نرم‌افزار مورد بحث و بررسی قرار داده‌ایم. پس در ادامه به طور مفصل به مفاهیم مربوط به پایگاه داده می‌پردازیم.

مدل طراحی

پس از مدل تحلیل، نوبت به مدل طراحی می‌رسد، پس از مدل تحلیل، باید برای پیاده‌سازی نرم‌افزار آماده شد. واضح است که گذر مستقیم به مرحله تولید کد و پیاده‌سازی عاقلانه نیست.

مدل تحلیل، مدل‌سازی عالم خارج به زبانی شبیه انسان است، اما مدل طراحی مدل‌سازی عالم داخل ماشین به زبانی شبیه زبان ماشین است. بنابراین برای اینکه ساخت برنامه کامپیوتری که به زبان ماشین است، امکان‌پذیر باشد، باید مدل تحلیل به مدل طراحی نگاشت شود تا مدل طراحی بتواند به عنوان نقشه راهی شبیه به زبان ماشین، راهگشا باشد.

مدل‌های تحلیل، کلی‌تر و انتزاعی‌تر هستند، و برای مدل‌سازی عالم خارج مورد استفاده قرار می‌گیرند، بدون آنکه به جزئیات نحوه پیاده‌سازی بپردازند، در واقع، مدل تحلیل به کلی گویی به زبان انسان و حذف جزئیات نحوه پیاده‌سازی می‌پردازد. اما مدل‌های طراحی، جزئی‌تر و غیرانتزاعی‌تر هستند، و برای مدل‌سازی عالم داخل ماشین مورد استفاده قرار می‌گیرند، و به بیان

جزئیات نحوه پیاده‌سازی می‌پردازند، در واقع مدل طراحی به جزئی‌گویی به زبان شبیه ماشین و درج جزئیات نحوه پیاده‌سازی می‌پردازد.
با نگاهی سطح به سطح، به حل مساله، سطوح مختلفی از انتزاع را خواهیم داشت. در بالاترین سطح انتزاع، راه حل به صورت کلی به زبان محیط مساله بیان می‌گردد. در سطوح پایین‌تر، راه حل به جزئیات پیاده‌سازی نزدیک‌تر می‌شود و در پایین‌ترین سطح انتزاع راه حل به صورتی بیان می‌شود تا مستقیما قابل پیاده‌سازی باشد.
مدل طراحی مانند دوربین عکاسی می‌باشد که شرایطی را فراهم می‌آورد که تصویر عالم خارج را در عالم دوربین و نگاتیو ذخیره‌سازی نماید. هر چه در عالم خارج باشد، عینا، اما در ابعادی کوچکتر یا بزرگتر در دوربین ذخیره می‌شود، در این نگاشت چیزی جا نمی‌ماند، مدل تحلیل خوب به مدل طرحی خوب و مدل تحلیل بد به مدل طراحی بد نگاشت می‌شود، یک مدل طراحی خوب از یک مدل تحلیل خوب و یک مدل طراحی بد از یک مدل تحلیل بد نشات گرفته است.
اما این خیلی مهم است که بتوانید یک مدل تحلیل خوب را به یک مدل طراحی خوب نگاشت کنید. که این دیگر هنر طراح است!
کاپر، تعبیر جالبی را از مدل طراحی بیان نموده است، «مدل طراحی دقیقاً جایی است که همزمان بر روی دو عالم ایستاده‌اید، عالم انسان و عالم ماشین، و باید تلاش نمود این دو عالم را به هم رساند.»
طراحی داده بر دو بخش طراحی جدول و طراحی پرس و جو می‌باشد. طراحی جدول از بخش طراحی داده، تحلیل موجودیت (ERD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط مدل رابطه‌ای، طراحی جدول را انجام می‌دهد. طراحی پرس و جو از بخش طراحی داده، تحلیل پرس و جو از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط جبر رابطه‌ای، طراحی پرس و جو را انجام می‌دهد.
توجه: در طراحی داده، فرهنگ داده‌های موجود در مدل تحلیل، مورد استفاده قرار می‌گیرد.

جبر رابطه‌ای[1]

برای شناسایی هر مدل یا ساختاری ابتدا باید به بررسی بخش‌های مختلف آن مدل یا ساختار بپردازیم.
جبر رابطه‌ای از دو بخش زیر تشکیل شده است:
۱- عملوندها
۲- عملگرها

[1] Relational Algebra

عملوندها

در جبر رابطه‌ای، عملوندها همان روابط (جداول) هستند که پرس و جوها بـر روی همـین روابـط جهت استخراج اطلاعات نوشته می‌شود.

عملگرها

در جبر رابطه‌ای، پرس و جوها توسط عملگرهای جبر رابطه‌ای نوشته می‌شوند. ایـن عملگرهـا بـه دو دسته کلی عملگرهای اصلی و فرعی تقسیم می‌شوند. کلیه‌ی پـرس و جوهـای ممکـن در جبـر رابطه‌ای را می‌توان توسط عملگرهای اصلی نوشت. به عبارت دیگر هیچ پـرس و جـویی در جبـر رابطه‌ای وجود ندارد که با استفاده از عملگرهای اصلی قابل نوشتن نباشـد. عملگرهـای فرعـی آن دسته از عملگرهایی هستند که توسط عملگرهای اصلی قابـل تولیـد هسـتند. بنـابراین عملگرهـای فرعی جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند.

روال کلی پرس و جو نویسی در جبر رابطه‌ای

روال کلی پرس و جو نویسی در جبر رابطه‌ای به ترتیب شامل بیان نام جـدول (R) یـا جـداول بـه عنوان مکان یا محل پرس و جو، سپس شرط انتخاب سطر (σ) به عنوان ملاک انتخاب سـطرها و در نهایت مدل انتخاب ستون (Π) به عنوان ستون‌های مورد نیاز است.
در ادامه به بررسی عملگرهای جبر رابطه‌ای می‌پردازیم:

عملگر انتخاب (Select)

این عملگر توسط نماد σ نمایش داده می‌شود. عملگر σ یک عملگر اصلی است. عملگـر σ جهـت انتخاب سطر در یک جدول مورد استفاده قرار می‌گیرد.
فرم کلی عملگر σ به صورت زیر است:

$$R_2 = \sigma_{\theta}(R_1)$$

در بخش θ شرط انتخاب سطر به عنوان ملاک انتخاب سطر مشخص می‌گـردد. همچنـین در ایـن بخش علائم ریاضی $=, \neq, >, \geq, <, \leq$ و عملگرهای منطقی $\wedge, \vee, \neg$ می‌تواند مورد اسـتفاده قـرار گیرد.
در بخش R_1 نام جدول یا جداول به عنوان مکان یا محل پرس و جو مشخص می‌گردد، این مکان جستجو می‌تواند خروجی یک پرس و جوی دیگر یعنی یک عبارت جبر رابطه‌ای دیگر باشد.
اگر رابطه R_1 به عنوان رابطه مبدا و رابطه R_2 را به عنوان رابطه مقصد، حاصل از عملگـر σ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر σ همواره برابر درجه رابطه مبدا می‌باشد، به صورت زیر:

$$\deg(R_2) = \deg(R_1)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

درجه رابطه فوق به عنوان رابطه مبدا برابر مقدار 3 است، به صورت زیر:

$$\deg(R_1) = \deg(S) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \sigma_{City=C2}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

درجه رابطه فوق به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$\deg(R_2) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_2) = \deg(R_1) = 3$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر σ همواره کوچکتر یا مساوی کاردینالیتی رابطه مبدا می‌باشد، به صورت زیر:

$$\text{card}(R_2) \leq \text{card}(R_1)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

کاردینالیتی رابطه فوق به عنوان رابطه مبدا برابر مقدار 3 است، به صورت زیر:

$$card(R_1) = card(S) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \sigma_{City=C2}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

کاردینالیتی رابطه فوق به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$card(R_2) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_2) \le card(R_1)$$

تعداد روابط حاصل

تعداد روابط حاصل از عملگر σ به صورت زیر محاسبه می‌گردد:

$$2^{card(R_1)} = 2^n = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \ldots + \binom{n}{n}$$

همان‌طور که می‌دانید یک مجموعه با n عضو، دارای 2^n زیر مجموعه است که البته یکی از آن‌ها مجموعه تهی است. بنابراین اگر رابطه‌ای دارای n سطر باشد، تعداد روابط حاصل از عملگر σ برابر 2^n خواهد بود.

مثال: جدول S را به صورت زیر در نظر بگیرید:

نام سطر	S#	Sname	City
X	S1	Sn1	C1
Y	S2	Sn2	C2
Z	S3	Sn3	C2

کاردینالیتی رابطه فوق برابر مقدار 3 است، به صورت زیر:

$$card(R_1) = card(S) = 3$$

X	Y	Z	انتخاب سطر
0	0	0	تهی
0	0	1	Z
0	1	0	Y
0	1	1	YZ
1	0	0	X
1	0	1	XZ
1	1	0	XY
1	1	1	XYZ

بنابراین تعداد روابط حاصل از عملگر σ برابر $2^3 = 8$ خواهد بود.

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر σ همواره برابر کلید کاندید رابطه مبدا می‌باشد، به صورت زیر:

$$C.K.(R_2) = C.K.(R_1)$$

زیرا رابطه مقصد حاصل از عملگر σ همواره زیر مجموعه رابطه مبدا است.

شرط لازم برای زیرمجموعه بودن رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

کلید کاندید رابطه فوق به عنوان رابطه مبدا برابر #S است، به صورت زیر:

$$C.K.(R_1) = C.K.(S) = S\#$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \sigma_{City=C2}(S)$$

خروجی پرس و جوی فوق به عنوان رابطه R_2 به صورت زیر است:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

از آنجا که شروط سازگاری مابین رابطه مبدا یعنی R_1 و رابطه مقصد یعنی R_2 برقرار است، و همچنین همه سطرهای رابطه مقصد R_2 در رابطه R_1 قرار دارد، پس رابطه R_2 زیر مجموعه رابطه R_1 می‌باشد، یعنی:

$$R_2 \subseteq R_1$$

پس کلید کاندید رابطه مقصد یعنی R_2 برابر همان کلید کاندید رابطه مبدا یعنی R_1 است، که برابر #S است، به صورت زیر:

$$C.K.(R_2) = S\#$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_2) = C.K.(R_1) = S\#$$

خاصیت جابه‌جایی

عملگر σ نسبت به خودش داری خاصیت جابه‌جایی است، یعنی اگر چندین عملگر σ به صورت متوالی روی یک رابطه مورد استفاده قرار گیرند، می‌توان ترتیب آنها را به هر شکل دلخواهی تغییر داد. به طور کلی اگر R یک رابطه و θ_1 و θ_2 بیانگر دو شرط باشند، همواره تساوی زیر برقرار است:

$$\sigma_{\theta_1}\left(\sigma_{\theta_2}(R)\right) = \sigma_{\theta_2}\left(\sigma_{\theta_1}(R)\right) = \sigma_{\theta_1 \wedge \theta_2}(R)$$

همچنین در یک قاعده کلی تساوی زیر نیز همواره برقرار است:

$$\sigma_{\theta_1}\left(\sigma_{\theta_2} \cdots \sigma_{\theta_n}(R)\right) = \sigma_{\theta_1 \wedge \theta_2 \cdots \wedge \theta_n}(R)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{Sname='Sn2'}\left(\sigma_{City='C2'}(S)\right)=\sigma_{City='C2'}\left(\sigma_{Sname='Sn2'}(S)\right)=\sigma_{Sname='Sn2'\wedge City='C2'}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname	City
S2	Sn2	C2

عملگر پرتو یا تصویر (Project)

این عملگر توسط نماد **Π** نمایش داده می‌شود. عملگر **Π** جهت انتخاب ستون در یک جدول مورد استفاده قرار می‌گیرد.

فرم کلی عملگر **Π** به صورت زیر است:

$$R_2=\Pi_L(R_1)$$

در بخش L مدل انتخاب ستون به عنوان ستون‌های مورد نیاز مشخص می‌گردد.

در بخش R_1 نام جدول یا جداول به عنوان مکان یا محل پرس و جو مشخص می‌گردد، این مکان جستجو می‌تواند خروجی یک پرس و جوی دیگر یعنی یک عبارت جبر رابطه‌ای دیگر باشد.

اگر رابطه R_1 به عنوان رابطه مبدا و رابطه R_2 را به عنوان رابطه مقصد، حاصل از عملگر **Π** در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر **Π** کوچکتر یا مساوی درجه رابطه مبدا می‌باشد، به صورت زیر:

$$\deg(R_2)\le\deg(R_1)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

درجه رابطه فوق به عنوان رابطه مبدا برابر مقدار 3 است، به صورت زیر:

$$\deg(R_1)=\deg(S)=3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2=\Pi_{S\#,Sname,City}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

درجه رابطه فوق به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$\deg(R_2) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_2) \le \deg(R_1)$$

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#,Sname}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

درجه رابطه فوق به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$\deg(R_2) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_2) \le \deg(R_1)$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر Π کوچکتر یا مساوی کاردینالیتی رابطه مبدا می‌باشد، به صورت زیر:

$$card(R_2) \le card(R_1)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

کاردینالیتی رابطه فوق به عنوان رابطه مبدا برابر مقدار 3 است، به صورت زیر:

$\text{card}(R_1) = \text{card}(S) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_2 = \Pi_{S\#,Sname}(S)$

خروجی پرس و جوی فوق به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

کاردینالیتی رابطه فوق به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$\text{card}(R_2) = 3$

بنابراین رابطه زیر برقرار خواهد بود:

$\text{card}(R_2) \le \text{card}(R_1)$

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_2 = \Pi_{City}(S)$

خروجی پرس و جوی فوق به صورت زیر است:

City
C1
C2

کاردینالیتی رابطه فوق به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$\text{card}(R_2) = 2$

بنابراین رابطه زیر برقرار خواهد بود:

$\text{card}(R_2) \le \text{card}(R_1)$

توجه: در جبر رابطه‌ای هنگام استفاده از عملگر **Π**، تاپل‌های تکراری فقط یک بار در خروجی ظاهر می‌شوند.

تعداد روابط حاصل

تعداد روابط حاصل از عملگر **Π** به صورت زیر محاسبه می‌گردد:

$$2^{\deg(R_1)} - 1 = 2^n - 1 = \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \ldots + \binom{n}{n}$$

همان‌طور که می‌دانید یک مجموعه با n عضو، دارای 2^n زیر مجموعه است که البته یکی از آن‌ها

مجموعه تهی است. بنابراین اگر رابطه‌ای دارای n ستون باشد، تعداد روابط حاصل از عملگر **Π** برابر $2^n - 1$ خواهد بود، دقت کنید که جدول بدون ستون نداریم، پس مجموعه تهی کنار گذاشته شده است.

مثال: جدول S را به صورت زیر در نظر بگیرید:

نام ستون	X	Y	Z
	S#	Sname	City
	S1	Sn1	C1
	S2	Sn2	C2
	S3	Sn3	C2

درجه رابطه فوق برابر مقدار 3 است، به صورت زیر:

$$\deg(R_1) = \deg(S) = 3$$

X	Y	Z	انتخاب ستون
0	0	0	تهی
0	0	1	Z
0	1	0	Y
0	1	1	YZ
1	0	0	X
1	0	1	XZ
1	1	0	XY
1	1	1	XYZ

(جدول بدون ستون نداریم)

بنابراین تعداد روابط حاصل از عملگر **Π** برابر $2^3 - 1 = 7$ خواهد بود.

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر **Π** به دو فرم زیر وجود دارد:

فرم اول: اگر کلید کاندید رابطه مبدا یعنی R_1 در رابطه مقصد یعنی R_2 باشد، آنگاه کلید کاندید رابطه مقصد یعنی R_2 برابر همان کلید کاندید رابطه مبدا یعنی R_1 خواهد بود، به صورت زیر:

$$C.K.(R_2) = C.K.(R_1)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C2
S2	Sn2	C2
S3	Sn2	C3

کلید کاندید رابطه فوق به عنوان رابطه مبدا برابر S# است، به صورت زیر:

$$C.K.(R_1) = C.K.(S) = S\#$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#,Sname}(S)$$

خروجی پرس و جوی فوق به عنوان رابطه R_2 به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn2

از آنجا که کلید کاندید رابطه مبدا یعنی R_1 در رابطه مقصد یعنی R_2 وجود دارد، پس کلید کاندید رابطه مقصد یعنی R_2 برابر همان کلید کاندید رابطه مبدا یعنی R_1 خواهد بود، به صورت زیر:

$$C.K.(R_2) = S\#$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_2) = C.K.(R_1)$$

فرم دوم: اما اگر کلید کاندید رابطه مبدا یعنی R_1 در رابطه مقصد یعنی R_2 نباشد، آنگاه کلید کاندید رابطه مقصد تمام کلید خواهد بود، به صورت زیر:

$$C.K.(R_2) = \text{All Key}$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C2
S2	Sn2	C2
S3	Sn2	C3

کلید کاندید رابطه فوق به عنوان رابطه مبدا برابر S# است، به صورت زیر:

$$C.K.(R_1) = C.K.(S) = S\#$$

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{Sname,City}(S)$$

خروجی پرس و جوی فوق به صورت زیر است:

Sname	City
Sn1	C2
Sn2	C2
Sn2	C3

از آنجا که کلید کاندید رابطه مبدا یعنی R_1 در رابطه مقصد یعنی R_2 وجود ندارد، پس کلید کاندید رابطه مقصد یعنی R_2 برابر Sname,City خواهد بود، به صورت زیر:

$$C.K.(R_2) = Sname, City$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_2) = All\ key$$

قاعده کاربرد تو در تو

کاربرد تو در توی عملگر Π فقط به صورت مشروط امکان‌پذیر است. به طور کلی اگر R یک رابطه و L_1 و L_2 بیانگر دو زیر مجموعه متفاوت از ستون‌ها باشند، آنگاه عبارت زیر زمانی برقرار است که $L_1 \subseteq L_2$:

$$\Pi_{L_1}\left(\Pi_{L_2}(R)\right)$$

همچنین در یک قاعده کلی اگر R یک رابطه و L_1 تا L_n بیانگر زیر مجموعه‌های متفاوت از ستون‌ها باشند، آنگاه عبارت زیر زمانی برقرار است که $1 \le i < n,\ L_i \subseteq L_{i+1}$:

$$\Pi_{L_1}\left(\Pi_{L_2}\left(\ldots \Pi_{L_n}(R)\right)\right)$$

همچنین در یک قاعده کلی تساوی زیر نیز همواره برقرار است:

$$\Pi_{L_1}\left(\Pi_{L_2}\left(\ldots \Pi_{L_n}(R)\right)\right) = \Pi_{L_1}(R)$$

توجه: در جبر رابطه‌ای همواره پرس و جو از داخلی‌ترین پرانتز آغاز، سپس در ادامه به سمت خارجی‌ترین پرانتز حرکت و در نهایت پایان می‌یابد.

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#}\left(\Pi_{S\#,sname}(S)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2
S3

خاصیت جابه‌جایی

عملگر Π نسبت به خودش همواره دارای خاصیت جابه‌جایی نیست، یعنی اگر چندین عملگر Π به صورت متوالی روی یک رابطه مورد استفاده قرار گیرند، نمی‌توان ترتیب آنها را به هر شکل دلخواهی تغییر داد. به طور کلی اگر R یک رابطه و L_1 و L_2 بیانگر دو زیر مجموعه متفاوت از ستون‌ها باشند، یعنی $L_1 \neq L_2$، آنگاه رابطه زیر برقرار است:

$$\Pi_{L_1}\left(\Pi_{L_2}(R)\right) \neq \Pi_{L_2}\left(\Pi_{L_1}(R)\right)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#}\left(\Pi_{S\#,sname}(R)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2
S3

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#,Sname}\left(\Pi_{S\#}(R)\right)$$

مطابق قاعده کاربرد تو در تو S#,Sname زیر مجموعه S# نیست، یعنی $S\#,Sname \not\subset S\#$ بنابراین، پرس و جوی فوق قابل انجام نیست.

پس رابطه زیر برقرار است:

$$\Pi_{S\#}\left(\Pi_{S\#,Sname}(R)\right) \neq \Pi_{S\#,Sname}\left(\Pi_{S\#}(R)\right)$$

همچنین در یک قاعده کلی اگر R یک رابطه و L_1 و L_2 بیانگر دو زیر مجموعه یکسان از ستون‌ها باشند، یعنی $L_1 = L_2$، آنگاه رابطه زیر برقرار است:

$$\Pi_{L_1}\left(\Pi_{L_2}(R)\right) = \Pi_{L_2}\left(\Pi_{L_1}(R)\right)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#}\left(\Pi_{S\#}(R)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#
S1
S2
S3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2
S3

پس رابطه زیر برقرار است:

$$\Pi_{S\#}(\Pi_{S\#}(R)) = \Pi_{S\#}(\Pi_{S\#}(R))$$

خاصیت جابه‌جایی دو عملگر σ و Π

دو عملگر **σ** و **Π** به صورت مشروط دارای خاصیت جابه‌جایی هستند. به طور کلی اگر R یک رابطه، L زیر مجموعه‌ای از ستون‌ها و **θ** مجموعه‌ای از شروط بر روی سطرها باشد، آنگاه تساوی زیر زمانی برقرار است که ستون‌های عملگر **σ** زیر مجموعه ستون‌های عملگر **Π** باشد. یعنی **σ ⊆ Π**:

$$\Pi_{L}(\sigma_{\theta}(R)) = \sigma_{\theta}(\Pi_{L}(R))$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#,City}(\sigma_{City='C2'}(S))$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	City
S2	C2
S3	C2

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \sigma_{City='C2'}(\Pi_{S\#,City}(S))$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#	City
S1	C1
S2	C2
S3	C2

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	City
S2	C2
S3	C2

پس رابطه زیر برقرار است:

$$\Pi_{S\#,City}\left(\sigma_{City='C2'}(S)\right)=\sigma_{City='C2'}\left(\Pi_{S\#,City}(S)\right)$$

مثال: جدول S را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \Pi_{S\#}\left(\sigma_{Cty='C2'}(S)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S2
S3

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_2 = \sigma_{City='C2'}\left(\Pi_{S\#}(S)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

S#
S1
S2
S3

که در نهایت پس از حرکت به سمت خارج، انجام عملگر انتخاب امکان پذیر نخواهد بود، زیرا ستون‌های عملگر σ یعنی City زیر مجموعه ستون‌های عملگر Π یعنی S# نیست، یعنی $City \not\subset S\#$. در واقع ستون City در پرانتز داخلی توسط عملگر Π انتخاب نشده است، بنابراین اجرای عملگر σ بر روی ستون City امکان‌پذیر نیست.

پس رابطه زیر برقرار است:

$$\Pi_{S\#}\left(\sigma_{City='C2'}(S)\right) \neq \sigma_{City='C2'}\left(\Pi_{S\#}(S)\right)$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: مشخصات تولیدکنندگانی که ساکن شهر C2 هستند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$R_2 = \sigma_{City='C2'}(S)$$

در پرانتز داخلی یعنی داخل جدول S مشخصات تولیدکنندگان قرار دارد که توسط عملگر σ تولیدکنندگانی که ساکن شهر C2 هستند انتخاب می‌گردند، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

پرس و جو: شماره تولیدکنندگانی که ساکن شهر C2 هستند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$R_2 = \Pi_{S\#}\left(\sigma_{City='C2'}(S)\right)$$

در پرانتز داخلی یعنی داخل جدول S مشخصات تولیدکنندگان قرار دارد که توسط عملگر σ تولیدکنندگانی که ساکن شهر C2 هستند انتخاب می‌گردند، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S2
S3

پرس و جو: شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$R_2 = \Pi_{S\#}(SP)$$

در پرانتز داخلی یعنی داخل جدول SP تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2

عملگر اجتماع (Union)

این عملگر توسط نماد $\cup$ نمایش داده می‌شود. عملگر $\cup$ یک عملگر اصلی است. عملگر $\cup$ جهت تجمیع کلیه سطرهای دو جدول مورد استفاده قرار می‌گیرد. در جبر رابطه‌ای اجتماع هر دو رابطه دلخواه امکان‌پذیر نیست. مگر اینکه شروط سازگاری در مورد آنها برقرار باشد. در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

فرم کلی عملگر $\cup$ به صورت زیر است:

$$R_3 = R_1 \cup R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\cup$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\cup$ همواره برابر درجه روابط مبدا می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1) = \deg(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	7
5	4

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 است، به صورت زیر:

$$\deg(R_1) = \deg(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cup R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\cup$

a	b
1	7
5	4

$=$

a	b
1	2
3	4
1	7
5	4

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$\deg(R_3) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) = \deg(R_2) = 2$$

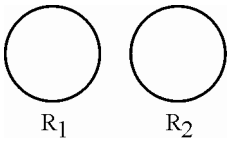
کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه

مقصد حاصل از عملگر $\cup$ در **کمترین** و **بیشترین** حالت به صورت زیر است:

برای دو رابطه $R_1(a,b)$ دارای $card(R_1)$ تاپل و رابطه $R_2(a,b)$ دارای $card(R_2)$ تاپل دو فرم زیر را داریم:

فرم اول: اگر $R_1 \cap R_2 = \phi$ باشد، یعنی:



آنگاه، تعداد تاپل‌های حاصل از عملگر اجتماع یعنی $card(R_3)$، برابر $card(R_1)+card(R_2)$ است که **بیشترین** مقدار ممکن خواهد بود. یعنی:

$$card(R_3) = card(R_1) + card(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	7
5	4
6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cup R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\cup$

a	b
1	7
5	4
6	9

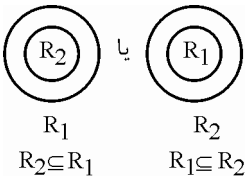
=

a	b
1	2
3	4
1	7
5	4
6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$card(R_3) = card(R_1) + card(R_2) = 2 + 3 = 5$$

فرم دوم: اگر $R_1 \subseteq R_2$ یا $R_2 \subseteq R_1$ باشد، یعنی:



آنگاه، تعداد تاپل‌های حاصل از عملگر اجتماع یعنی $card(R_3)$، برابر $\max(card(R_1), card(R_2))$ است که **کمترین** مقدار ممکن خواهد بود. یعنی:

$$card(R_3) = \max(card(R_1), card(R_2))$$

مثال: جداول R_1 و R_2 را با فرض $R_1 \subseteq R_2$ به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	2
3	4
5	6

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cup R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\cup$

a	b
1	2
3	4
5	6

$=$

a	b
1	2
3	4
5	6

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = \max(card(R_1), card(R_2)) = \max(2,3) = 3$$

توجه: در جبر رابطه‌ای تاپل‌های تکراری حاصل از عملگر $\cup$ حذف می‌شوند. به بیان دیگر، اگر یک تاپل در دو رابطه وجود داشته باشد در اجتماع آن دو رابطه فقط یکبار ظاهر می‌شود، به

عبارت دیگر در اجتماع دو رابطه تاپل‌های تکراری وجود ندارد. بر این اساس می‌توان گفت که در جبر رابطه‌ای اجتماع هر رابطه با خودش برابر با همان رابطه است.

مثال: جداول R_1 و R_2 را با فرض $R_2 \subseteq R_1$ به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6

R_1

a	b
1	2
3	4

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$$card(R_1) = 3$$
$$card(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cup R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6

$\cup$

a	b
1	2
3	4

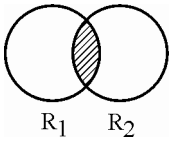
$=$

a	b
1	2
3	4
5	6

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = \max\big(card(R_1), card(R_2)\big) = \max(3,2) = 3$$

توجه: اگر دو رابطه R_1 و R_2 به ترتیب دارای $card(R_1)$ و $card(R_2)$ تاپل باشند و دو رابطه دارای k تاپل مشترک باشند، یعنی:



آنگاه حاصل $card(R_1 \cup R_2)$ به صورت زیر خواهد بود:

$$card(R_1 \cup R_2) = \big(card(R_1) + card(R_2)\big) - k$$

مثال: جداول R_1 و R_2 را با دو تاپل مشترک به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

a	b
1	2
3	4
9	10

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$$card(R_1) = 4$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cup R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	8

$\cup$

a	b
1	2
3	4
9	10

$=$

a	b
1	2
3	4
5	6
7	8
9	10

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$card(R_3) = card(R_1 \cup R_2) = \big(card(R_1) + card(R_2)\big) - k = (4+3) - 2 = 7 - 2 = 5$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره تمام کلید می‌باشد، به صورت زیر:

$$C.K.(R_3) = \text{All key}$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
2	1	3
5	2	6
3	7	8

R_1

a	b	c
2	1	4
5	4	6
9	7	8

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cup R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
2	1	3
5	2	6
3	7	8

$\cup$

a	b	c
2	1	4
5	4	6
9	7	8

=

a	b	c
2	1	3
5	2	6
3	7	8
2	1	4
5	4	6
9	7	8

از آنجا که کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره تمام کلید (All key) می‌باشد، پس کلید کاندید رابطه مقصد یعنی R_3 برابر abc است، به صورت زیر:

$C.K.(R_3) = abc$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = All\ key$

توجه: کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره برابر تمام ستون‌های یکی از دو جدول R_1 و R_2 است. در مثال فوق تمام ستون‌های جدول R_1 برابر abc و تمام ستون‌های جدول R_2 نیز برابر abc است، از آنجاکه شروط سازگاری مابین روابط عملگر $\cup$ یعنی R_1 و R_2 برقرار است، پس ستون‌های دو جدول R_1 و R_2 یکسان است، بنابراین کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره برابر تمام ستون‌های یکی از دو جدول R_1 و R_2 یعنی abc است. که این همان معنای تمام کلید است اما به بیان دیگر.

توجه: کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره برابر اجتماع تمام ستون‌های دو جدول R_1 و R_2 است. در مثال فوق تمام ستون‌های جدول R_1 برابر abc و تمام ستون‌های جدول R_2 نیز برابر abc است، از آنجاکه شروط سازگاری مابین روابط عملگر $\cup$ یعنی R_1 و R_2 برقرار است، پس ستون‌های دو جدول R_1 و R_2 یکسان است و اجتماع تمام ستون‌های دو جدول R_1 و R_2 برابر abc می‌شود. بنابراین کلید کاندید رابطه مقصد حاصل از عملگر $\cup$ مابین دو رابطه R_1 و R_2 همواره برابر اجتماع تمام ستون‌های دو جدول R_1 و R_2 یعنی abc است. که این همان معنای تمام کلید است اما به بیان دیگر.

توجه: کلید کاندید رابطه مقصد حاصل از عملگر ∪ مابین دو رابطه R_1 و R_2 همواره برابر اشتراک تمام ستون‌های دو جدول R_1 و R_2 است. در مثال فوق تمام ستون‌های جدول R_1 برابر abc و تمام ستون‌های جدول R_2 نیز برابر abc است، از آنجاکه شروط سازگاری مابین روابط عملگر ∪ یعنی R_1 و R_2 برقرار است، پس ستون‌های دو جدول R_1 و R_2 یکسان است و اشتراک تمام ستون‌های دو جدول R_1 و R_2 برابر abc می‌شود. بنابراین کلید کاندید رابطه مقصد حاصل از عملگر ∪ مابین دو رابطه R_1 و R_2 همواره برابر اشتراک تمام ستون‌های دو جدول R_1 و R_2 یعنی abc است. که این همان معنای تمام کلید است اما به بیان دیگر.

خاصیت جابه‌جایی

عملگر ∪ دارای خاصیت جابه‌جایی است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر ∪ بیانگر عمل اجتماع باشد، همواره تساوی زیر برقرار است:

$$R_1 \cup R_2 = R_2 \cup R_1$$

خاصیت شرکت‌پذیری

عملگر ∪ دارای خاصیت شرکت‌پذیری است. به طور کلی اگر R_1، R_2 و R_3 سه رابطه و عملگر ∪ بیانگر عمل اجتماع باشد، همواره تساوی زیر برقرار است:

$$R_1 \cup (R_2 \cup R_3) = (R_1 \cup R_2) \cup R_3$$

خاصیت توزیع‌پذیری

عملگر انتخاب داری خاصیت توزیع‌پذیری بر روی عملگر اجتماع است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر ∪ بیانگر عمل اجتماع باشد، همواره تساوی زیر برقرار است:

$$\sigma_{\theta}(R_1 \cup R_2) = \sigma_{\theta}(R_1) \cup \sigma_{\theta}(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
2	5	6
3	8	9
4	5	6

R_1

a	b	c
8	1	4
1	2	6
4	3	7
4	5	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{c=6}(R_1 \cup R_2)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
2	5	6
3	8	9
4	5	6

∪

a	b	c
8	1	4
1	2	6
4	3	7
4	5	6

=

a	b	c
1	2	3
2	5	6
3	8	9
4	5	6
8	1	4
1	2	6
4	3	7
4	5	6

که در نهایت پس از حرکت به سمت خارج و انجام عملگر انتخاب، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	b	c
2	5	6
4	5	6
1	2	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$\sigma_{c=6}(R_1) \cup \sigma_{c=6}(R_2)$

خروجی $\sigma_{c=6}(R_1)$ به صورت زیر است:

a	b	c
2	5	6
4	5	6

خروجی $\sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
1	2	6
4	5	6

که خروجی نهایی پرس و جوی $\sigma_{c=6}(R_1) \cup \sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
2	5	6
4	5	6

∪

a	b	c
1	2	6
4	5	6

=

a	b	c
2	5	6
4	5	6
1	2	6

پس رابطه زیر برقرار است:

$$\sigma_{c=6}(R_1 \cup R_2) = \sigma_{c=6}(R_1) \cup \sigma_{c=6}(R_2)$$

همچنین به طور کلی اگر R یک رابطه و عملگر ∪ بیانگر عمل اجتماع باشد، همواره تساوی زیر برقرار است:

$$\sigma_{\theta_1 \vee \theta_2}(R) = \sigma_{\theta_1}(R) \cup \sigma_{\theta_2}(R)$$

مثال: جدول R را به صورت زیر در نظر بگیرید:

a	b	c
2	3	4
3	4	5
4	4	6
1	4	3
5	6	5

پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{b=4 \vee c=5}(R)$$

خروجی پرس و جو به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3
5	6	5

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{b=4}(R) \cup \sigma_{c=5}(R)$$

خروجی $\sigma_{b=4}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

خروجی $\sigma_{c=5}(R)$ به صورت زیر است:

a	b	c
3	4	5
5	6	5

که خروجی نهایی پرس و جوی $\sigma_{b=4}(R) \cup \sigma_{c=5}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

∪

a	b	c
3	4	5
5	6	5

=

a	b	c
3	4	5
4	4	6
1	4	3
5	6	5

پس رابطه زیر برقرار است:

$$\sigma_{b=4 \vee c=5}(R) = \sigma_{b=4}(R) \cup \sigma_{c=5}(R)$$

توجه: در جبر رابطه‌ای تاپل‌های تکراری حاصل از عملگر ∪ حذف می‌شوند. به بیان دیگر، اگر یک تاپل در دو رابطه وجود داشته باشد در اجتماع آن دو رابطه فقط یکبار ظاهر می‌شود، به عبارت دیگر در اجتماع دو رابطه تاپل‌های تکراری وجود ندارد. بر این اساس می‌توان گفت که در جبر رابطه‌ای اجتماع هر رابطه با خودش برابر با همان رابطه است.

عملگر اشتراک (Intersect)

این عملگر توسط نماد ∩ نمایش داده می‌شود. عملگر ∩ یک عملگر فرعی است. عملگر ∩ جهت کشف سطرهای مشترک دو جدول مورد استفاده قرار می‌گیرد. در جبر رابطه‌ای اشتراک هر دو رابطه دلخواه امکان‌پذیر نیست. مگر اینکه شروط سازگاری در مورد آنها برقرار باشد. در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

فرم کلی عملگر ∩ به صورت زیر است:

$$R_3 = R_1 \cap R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر ∩ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\cap$ همواره برابر درجه روابط مبدا می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1) = \deg(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	2
3	4
5	6

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 است، به صورت زیر:

$$\deg(R_1) = \deg(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cap R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\cap$

a	b
1	2
3	4
5	6

=

a	b
1	2
3	4

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$\deg(R_3) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

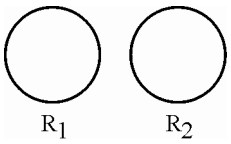
$$\deg(R_3) = \deg(R_1) = \deg(R_2) = 2$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $\cap$ در **کمترین** و **بیشترین** حالت به صورت زیر است:

برای دو رابطه $R_1(a,b)$ دارای $\text{card}(R_1)$ تاپل و رابطه $R_2(a,b)$ دارای $\text{card}(R_2)$ تاپل داریم:

فرم اول: اگر $R_1 \cap R_2 = \phi$ باشد، یعنی:



آنگاه، تعداد تاپل‌های حاصل از عملگر اشتراک یعنی $card(R_3)$، برابر صفر یعنی جدول تهی است که **کمترین** مقدار ممکن خواهد بود. یعنی:

$$card(R_3) = 0$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
5	6
7	8
9	10

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cap R_2$$

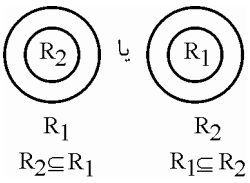
خروجی پرس و جوی فوق به صورت زیر است:

a	b	∩	a	b	=	a	b
1	2		5	6			
3	4		7	8			
			9	10			

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$card(R_3) = 0$$

اگر $R_1 \subseteq R_2$ یا $R_2 \subseteq R_1$ باشد، یعنی:



آنگـــاه، تعـــداد تاپـــل‌هـــای حاصـــل از عملگـــر اشـــتراک یعنـــی $\text{card}(R_3)$، برابـــر $\min(\text{card}(R_1), \text{card}(R_2))$ است که **بیشترین** مقدار ممکن خواهد بود. یعنی:

$$\text{card}(R_3) = \min(\text{card}(R_1), \text{card}(R_2))$$

مثال: جداول R_1 و R_2 را با فرض $R_1 \subseteq R_2$ به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	2
3	4
5	6

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$$\text{card}(R_1) = 2$$
$$\text{card}(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cap R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\cap$

a	b
1	2
3	4
5	6

$=$

a	b
1	2
3	4

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$\text{card}(R_3) = \min(\text{card}(R_1), \text{card}(R_2)) = \min(2,3) = 2$$

مثال: جداول R_1 و R_2 را با فرض $R_2 \subseteq R_1$ به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6

R_1

a	b
1	2
3	4

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$card(R_1) = 3$

$card(R_2) = 2$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cap R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6

$\cap$

a	b
1	2
3	4

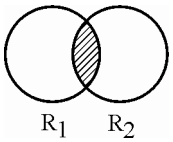
$=$

a	b
1	2
3	4

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$card(R_3) = \min(card(R_1), card(R_2)) = \min(3,2) = 2$

توجه: اگر دو رابطه R_1 و R_2 به ترتیب دارای $card(R_1)$ و $card(R_2)$ تاپل باشند و دو رابطه دارای k تاپل مشترک باشند، یعنی:



آنگاه حاصل $card(R_1 \cap R_2)$ به صورت زیر خواهد بود:

$card(R_1 \cap R_2) = k$

مثال: جداول R_1 و R_2 را با دو تاپل مشترک به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

a	b
1	2
3	4
9	10

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$card(R_1) = 4$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cap R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	8

$\cap$

a	b
1	2
3	4
9	10

=

a	b
1	2
3	4

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$\text{card}(R_3) = \text{card}(R_1 \cap R_2) = k = 2$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\cap$ مابین دو رابطه R_1 و R_2 همواره برابر یکی از کلیدهای کاندید روابط مبدا یعنی کلید کاندید R_1 یا کلید کاندید R_2 می‌باشد، به صورت زیر:

$C.K.(R_3) = C.K.(R_1) \text{ OR } C.K.(R_2)$

زیرا رابطه مقصد حاصل از عملگر $\cap$ همواره زیر مجموعه روابط مبدا است. یعنی $R_1 \cap R_2 \subseteq R_1$ و هم $R_1 \cap R_2 \subseteq R_2$ بنابراین هم کلید کاندید R_1 و هم کلید کاندید R_2 قدرت ایجاد تمایز بین سطرهای $R_1 \cap R_2$ را دارد.

شرط لازم برای زیرمجموعه بودن رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	9

R_1

a	b	c
1	2	3
4	5	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cap R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	9

$\cap$

a	b	c
1	2	3
4	5	6

$=$

a	b	c
1	2	3
4	5	6

از آنجا که کلید کاندید رابطه مقصد حاصل از عملگر $\cap$ مابین دو رابطه R_1 و R_2 همواره برابر یکی از کلیدهای کاندید روابط مبدا یعنی کلید کاندید R_1 یا کلید کاندید R_2 می‌باشد، پس کلید کاندید رابطه مقصد یعنی R_3 برابر کلید کاندید رابطه R_1 یعنی a است و یا برابر کلید کاندید رابطه R_2 یعنی b است، به صورت زیر:

$C.K.(R_3) = a \text{ OR } b$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1) \text{ OR } C.K.(R_2)$

از آنجا که شروط سازگاری مابین رابطه مبدا یعنی R_1 و رابطه مقصد R_3 یعنی $R_1 \cap R_2$ برقرار است، و همچنین همه سطرهای رابطه مقصد R_3 یعنی $R_1 \cap R_2$ در رابطه R_1 قرار دارد، پس رابطه R_3 یعنی $R_1 \cap R_2$ زیر مجموعه رابطه R_1 می‌باشد، یعنی:

$R_3 = R_1 \cap R_2 \subseteq R_1$

پس کلید کاندید رابطه مقصد یعنی R_3 حاصل از عملگر اشتراک می‌تواند برابر همان کلید کاندید رابطه مبدا یعنی R_1 باشد، که برابر a است، به صورت زیر:

$C.K.(R_3) = a$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1) = a$

همچنین از آنجا که شروط سازگاری مابین رابطه مبدا یعنی R_2 و رابطه مقصد R_3 یعنی $R_1 \cap R_2$ برقرار است، و همچنین همه سطرهای رابطه مقصد R_3 یعنی $R_1 \cap R_2$ در رابطه R_2 قرار دارد، پس رابطه R_3 یعنی $R_1 \cap R_2$ زیر مجموعه رابطه R_2 می‌باشد، یعنی:

$$R_3 = R_1 \cap R_2 \subseteq R_2$$

پس کلید کاندید رابطه مقصد یعنی R_3 حاصل از عملگر اشتراک می‌تواند برابر همان کلید کاندید رابطه مبدا یعنی R_2 باشد، که برابر b است، به صورت زیر:

$$C.K.(R_3) = b$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_3) = C.K.(R_2) = b$$

خاصیت جابه‌جایی

عملگر $\cap$ دارای خاصیت جابه‌جایی است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\cap$ بیانگر عمل اشتراک باشد، همواره تساوی زیر برقرار است:

$$R_1 \cap R_2 = R_2 \cap R_1$$

خاصیت شرکت‌پذیری

عملگر $\cap$ دارای خاصیت شرکت‌پذیری است. به طور کلی اگر R_1، R_2 و R_3 سه رابطه و عملگر $\cap$ بیانگر عمل اشتراک باشد، همواره تساوی زیر برقرار است:

$$R_1 \cap (R_2 \cap R_3) = (R_1 \cap R_2) \cap R_3$$

خاصیت توزیع‌پذیری

عملگر انتخاب دارای خاصیت توزیع‌پذیری بر روی عملگر اشتراک است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\cap$ بیانگر عمل اشتراک باشد، همواره تساوی زیر برقرار است:

$$\sigma_\theta(R_1 \cap R_2) = \sigma_\theta(R_1) \cap \sigma_\theta(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$\sigma_{c=6}(R_1 \cap R_2)$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\cap$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

که در نهایت پس از حرکت به سمت خارج و انجام عملگر انتخاب، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	b	c
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$\sigma_{c=6}(R_1) \cap \sigma_{c=6}(R_2)$

خروجی $\sigma_{c=6}(R_1)$ به صورت زیر است:

a	b	c
4	5	6
7	8	6

خروجی $\sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
4	5	6
7	9	6

که خروجی نهایی پرس و جوی $\sigma_{c=6}(R_1) \cap \sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
4	5	6
7	8	6

$\cap$

a	b	c
4	5	6
7	9	6

=

a	b	c
4	5	6

پس رابطه زیر برقرار است:

$\sigma_{c=6}(R_1 \cap R_2) = \sigma_{c=6}(R_1) \cap \sigma_{c=6}(R_2)$

همچنین به طور کلی اگر R یک رابطه و عملگر $\cap$ بیانگر عمل اشتراک باشد، همواره تساوی زیر برقرار است:

$\sigma_{\theta_1 \wedge \theta_2}(R) = \sigma_{\theta_1}(R) \cap \sigma_{\theta_2}(R)$

مثال: جدول R را به صورت زیر در نظر بگیرید:

a	b	c
2	3	4
3	4	5
4	4	6
1	4	3
5	6	5

پرس و جوی زیر را در نظر بگیرید:

$\sigma_{b=4 \wedge c=5}(R)$

خروجی پرس و جو به صورت زیر است:

a	b	c
3	4	5

همچنین پرس و جوی زیر را در نظر بگیرید:

$\sigma_{b=4}(R) \cap \sigma_{c=5}(R)$

خروجی $\sigma_{b=4}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

خروجی $\sigma_{c=5}(R)$ به صورت زیر است:

a	b	c
3	4	5
5	6	5

که خروجی نهایی پرس و جوی $\sigma_{b=4}(R) \cap \sigma_{c=5}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

$\cap$

a	b	c
3	4	5
5	6	5

=

a	b	c
3	4	5

پس رابطه زیر برقرار است:

$\sigma_{b=4 \wedge c=5}(R) = \sigma_{b=4}(R) \cap \sigma_{c=5}(R)$

عملگر تفاضل (Set Difference)

این عملگر توسط نماد – نمایش داده می‌شود. عملگر – یک عملگر اصلی است. عملگر – جهت تفاضل سطرهای دو جدول مورد استفاده قرار می‌گیرد. اگر R_1 و R_2 دو رابطه باشند، منظور از $R_1 - R_2$ مجموعه کلیه تاپل‌هایی است که عضو R_1 هستند اما در R_2 حضور ندارند. در جبر رابطه‌ای تفاضل هر دو رابطه دلخواه امکان‌پذیر نیست. مگر اینکه شروط سازگاری در مورد آنها برقرار باشد. در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

فرم کلی عملگر – به صورت زیر است:

$$R_3 = R_1 - R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر – در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر – همواره برابر درجه روابط مبدا می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1) = \deg(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

a	b
1	2
3	4

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 است، به صورت زیر:

$$\deg(R_1) = \deg(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 - R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	8

$-$

a	b
1	2
3	4

$=$

a	b
5	6
7	8

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$\deg(R_3) = 2$

بنابراین رابطه زیر برقرار خواهد بود:

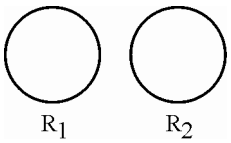
$\deg(R_3) = \deg(R_1) = \deg(R_2) = 2$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $-$ در **کمترین** و **بیشترین** حالت به صورت زیر است:

برای دو رابطه $R_1(a,b)$ دارای $\text{card}(R_1)$ تاپل و رابطه $R_2(a,b)$ دارای $\text{card}(R_2)$ تاپل داریم:

فرم اول: اگر $R_1 \cap R_2 = \phi$ باشد، یعنی:



آنگاه، تعداد تاپل‌های حاصل از عملگر تفاضل یعنی $\text{card}(R_3)$، برابر $\text{card}(R_1)$ است که **بیشترین** مقدار ممکن خواهد بود. یعنی:

$\text{card}(R_3) = \text{card}(R_1)$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
5	6
7	8
9	10

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر:

$\text{card}(R_1) = 2$

$\text{card}(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 - R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$-$

a	b
5	6
7	8
9	10

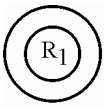
$=$

a	b
1	2
3	4

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$\text{card}(R_3) = \text{card}(R_1) = 2$

فرم دوم: اگر $R_1 \subseteq R_2$ باشد، یعنی:



R_2

$R_1 \subseteq R_2$

آنگاه، تعداد تاپل‌های حاصل از عملگر اشتراک یعنی $\text{card}(R_3)$، برابر صفر یعنی جدول تهی است که **کمترین** مقدار ممکن خواهد بود. یعنی:

$\text{card}(R_3) = 0$

مثال: جداول R_1 و R_2 را با فرض $R_1 \subseteq R_2$ به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

a	b
1	2
3	4
5	6
7	8

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$\text{card}(R_1) = 2$

$\text{card}(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 - R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$-$

a	b
1	2
3	4
5	6
7	8

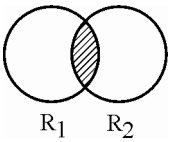
$=$

a	b

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$card(R_3) = 0$

توجه: اگر دو رابطه R_1 و R_2 به ترتیب دارای $card(R_1)$ و $card(R_2)$ تاپل باشند و دو رابطه دارای k تاپل مشترک باشند، یعنی:



آنگاه حاصل $card(R_1 - R_2)$ به صورت زیر خواهد بود:

$card(R_1 - R_2) = card(R_1) - k$

مثال: جداول R_1 و R_2 را با دو تاپل مشترک به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

a	b
1	2
3	4
9	10

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا، به صورت زیر است:

$card(R_1) = 4$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 - R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	8

$-$

a	b
1	2
3	4
9	10

$=$

a	b
5	6
7	8

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$card(R_3) = card(R_1 - R_2) = card(R_1) - k = 4 - 2 = 2$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $-$ مابین دو رابطه R_1 و R_2 همواره برابر کلید کاندید رابطه مبدا در سمت چپ عملگر تفاضل یعنی کلید کاندید R_1 می‌باشد، به صورت زیر:

$$C.K.(R_3) = C.K.(R_1)$$

زیرا رابطه مقصد حاصل از عملگر $-$ همواره زیر مجموعه رابطه مبدا در سمت چپ عملگر تفاضل است. یعنی $R_1 - R_2 \subseteq R_1$ بنابراین فقط کلید کاندید R_1 قدرت ایجاد تمایز بین سطرهای $R_1 - R_2$ را دارد.

شرط لازم برای زیر مجموعه بودن، رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	9
10	11	12

R_1

a	b	c
1	2	3
4	5	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 - R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	9
10	11	12

−

a	b	c
1	2	3
4	5	6

=

a	b	c
7	8	9
10	11	12

از آنجا که کلید کاندید رابطه مقصد حاصل از عملگر − مابین دو رابطه R_1 و R_2 همواره برابر کلید کاندید رابطه مبدا در سمت چپ عملگر تفاضل یعنی کلید کاندید R_1 می‌باشد، پس کلید کاندید رابطه مقصد یعنی R_3 برابر کلید کاندید رابطه R_1 یعنی a است، به صورت زیر:

$C.K.(R_3) = a$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1)$

از آنجا که شروط سازگاری مابین رابطه مبدا یعنی R_1 و رابطه مقصد R_3 یعنی $R_1 - R_2$ برقرار است، و همچنین همه سطرهای رابطه مقصد R_3 یعنی $R_1 - R_2$ در رابطه R_1 قرار دارد، پس رابطه R_3 یعنی $R_1 - R_2$ زیر مجموعه رابطه R_1 می‌باشد، یعنی:

$R_3 = R_1 - R_2 \subseteq R_1$

پس کلید کاندید رابطه مقصد یعنی R_3 حاصل از عملگر تفاضل برابر همان کلید کاندید رابطه مبدا یعنی R_1 است، که برابر a است، به صورت زیر:

$C.K.(R_3) = a$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1) = a$

خاصیت جابه‌جایی

عملگر − دارای خاصیت جابه‌جایی نیست. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر − بیانگر عمل اجتماع باشد، رابطه زیر برقرار است:

$R_1 - R_2 \neq R_2 - R_1$

خاصیت شرکت‌پذیری

عملگر – دارای خاصیت شرکت‌پذیری نیست. بـه طـور کلـی اگـر R_1، R_2 و R_3 سـه رابطـه و عملگر – بیانگر عمل تفاضل باشد، رابطه زیر برقرار است:

$$R_1 - (R_2 - R_3) \neq (R_1 - R_2) - R_3$$

خاصیت توزیع‌پذیری

عملگر انتخاب دارای خاصیت توزیع‌پذیری بر روی عملگر تفاضل است. به طور کلـی اگـر R_1 و R_2 دو رابطه و عملگر – بیانگر عمل تفاضل باشد، همواره تساوی زیر برقرار است:

$$\sigma_{\theta}(R_1 - R_2) = \sigma_{\theta}(R_1) - \sigma_{\theta}(R_2)$$

توجه: در عبارت فوق، حتی اعمال شرط θ بر روی رابطه R_2 در سمت راسـت تسـاوی ضـروری نیست. بنابراین همواره تساوی زیر نیز برقرار است:

$$\sigma_{\theta}(R_1 - R_2) = \sigma_{\theta}(R_1) - \sigma_{\theta}(R_2) = \sigma_{\theta}(R_1) - R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6
2	4	6
3	4	5

R_1

a	b	c
1	2	3
4	5	6
8	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{c=6}(R_1 - R_2)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6
2	4	6
3	4	5

–

a	b	c
1	2	3
4	5	6
8	9	6

=

a	b	c
7	8	6
2	4	6
3	4	5

که در نهایت پس از حرکت به سمت خارج و انجام عملگر انتخاب، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	b	c
7	8	6
2	4	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{c=6}(R_1) - \sigma_{c=6}(R_2)$$

خروجی $\sigma_{c=6}(R_1)$ به صورت زیر است:

a	b	c
4	5	6
7	8	6
2	4	6

خروجی $\sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
4	5	6
8	9	6

که خروجی نهایی پرس و جوی $\sigma_{c=6}(R_1) - \sigma_{c=6}(R_2)$ به صورت زیر است:

a	b	c
4	5	6
7	8	6
2	4	6

$-$

a	b	c
4	5	6
8	9	6

$=$

a	b	c
7	8	6
2	4	6

پس رابطه زیر برقرار است:

$$\sigma_{c=6}(R_1 - R_2) = \sigma_{c=6}(R_1) - \sigma_{c=6}(R_2) = \sigma_{c=6}(R_1) - R_2$$

همچنین به طور کلی اگر R یک رابطه و عملگر $-$ بیانگر عمل تفاضل باشد، همواره تساوی زیر برقرار است:

$$\sigma_{\theta_1 - \theta_2}(R) = \sigma_{\theta_1 \wedge \neg\theta_2}(R) = \sigma_{\theta_1}(R) \cap \sigma_{\neg\theta_2}(R)$$

مثال: جدول R را به صورت زیر در نظر بگیرید:

a	b	c
2	3	4
3	4	5
4	4	6
1	4	3
5	6	5

پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{b=4-c=5}(R)=\sigma_{b=4\wedge\neg(c=5)}(R)=\sigma_{b=4\wedge c\neq5}(R)$$

خروجی پرس و جو به صورت زیر است:

a	b	c
4	4	6
1	4	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\sigma_{b=4}(R)\cap\sigma_{\neg(c=5)}(R)=\sigma_{b=4}(R)\cap\sigma_{c\neq5}(R)$$

خروجی $\sigma_{b=4}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

خروجی $\sigma_{c\neq5}(R)$ به صورت زیر است:

a	b	c
2	3	4
4	4	6
1	4	3

که خروجی نهایی پرس و جوی $\sigma_{b=4}(R)\cap\sigma_{c\neq5}(R)$ به صورت زیر است:

a	b	c
3	4	5
4	4	6
1	4	3

$\cap$

a	b	c
2	3	4
4	4	6
1	4	3

=

a	b	c
4	4	6
1	4	3

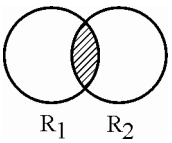
پس رابطه زیر برقرار است:

$$\sigma_{b=4-c=5}(R)=\sigma_{b=4\wedge\neg(c=5)}(R)=\sigma_{b=4\ \wedge\ c\neq5}(R)=\sigma_{b=4}(R)\cap\sigma_{c\neq5}(R)$$

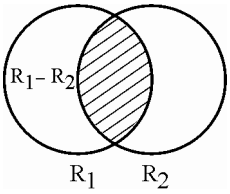
شبیه‌سازی

عملگر اشتراک یک عملگر فرعی است که می‌توان توسط عملگر اصلی تفاضل آنرا شبیه‌سازی نمود، به صورت زیر:

$$R_1\cap R_2=R_1-(R_1-R_2)=R_2-(R_2-R_1)$$



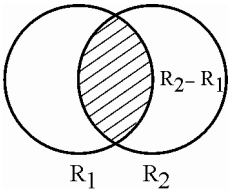
با توجه به شکل زیر، مشاهده می‌شود که با کنار گذاشتن ناحیه $R_1 - R_2$ از ناحیه R_1، قسمت باقی مانده برابر $R_1 \cap R_2$ است.



بنابراین رابطه زیر برقرار خواهد بود:

$$R_1 \cap R_2 = R_1 - (R_1 - R_2)$$

همچنین با توجه به شکل زیر، مشاهده می‌شود که با کنار گذاشتن ناحیه $R_2 - R_1$ از ناحیه R_2، قسمت باقی مانده برابر $R_1 \cap R_2$ است.



بنابراین رابطه زیر برقرار خواهد بود:

$$R_1 \cap R_2 = R_2 - (R_2 - R_1)$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: شماره تولیدکنندگانی که هیچ قطعه‌ای تولید نکرده‌اند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#}(S) - \Pi_{S\#}(SP)$$

در پرس و جوی سمت چپ عملگر تفاضل یعنی در جدول S مشخصات کلیه تولیدکنندگان قرار دارد، که پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگان استخراج می‌شود، به صورت زیر:

S#
S1
S2
S3

همچنین در پرس و جوی سمت راست عملگر تفاضل یعنی در جدول SP شماره کلیه تولیدکنندگانی قرار دارد که حداقل یک قطعه تولید کرده‌اند، که پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگانی که حداقل یک قطعه تولیدکرده‌اند استخراج می‌شود، به صورت زیر:

S#
S1
S2

که در نهایت پس از اجرای عملگر تفاضل، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2
S3

S#
S1
S2

S#
S3

به طور کلی، اگر عبارت نفی در پرس و جو استفاده شده باشد، این امر نشانه‌ای برای ضرورت استفاده از عملگر تفاضل است. دقت کنید که در انجام عمل تفاضل برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

پرس و جو: شماره تولیدکنندگانی که حداقل یک قطعه تولیدکرده‌اند اما قطعه P2 را تولید نکرده‌اند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#}(SP) - \Pi_{S\#}(\sigma_{P\#='P2'}(SP))$$

در پرس و جوی سمت چپ عملگر تفاضل یعنی در جدول SP مشخصات کلیه تولیدکنندگانی قرار دارد که حداقل یک قطعه تولیدکرده‌اند، که پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگانی که حداقل یک قطعه تولیدکرده‌اند استخراج می‌شود، به صورت زیر:

S#
S1
S2

همچنین در پرس و جوی سمت راست عملگر تفاضل یعنی در جدول SP شماره کلیه تولیدکنندگانی که حداقل یک قطعه تولیدکرده‌اند قرار دارد، که پس از اجرای عملگر انتخاب شماره کلیه تولیدکنندگانی که قطعه P2 را تولیدکرده‌اند استخراج می‌گردد، در نهایت پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگانی که قطعه P2 را تولید کرده‌اند استخراج می‌شود، به صورت زیر:

$$\frac{S\#}{S1}$$

که در نهایت پس از اجرای عملگر تفاضل، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

$$\frac{S\#}{S1} - \frac{S\#}{S1} = \frac{S\#}{S2}$$
$$S2$$

به طور کلی، اگر عبارت نفی در پرس و جو استفاده شده باشد، این امر نشانه‌ای برای ضرورت استفاده از عملگر تفاضل است. دقت کنید که در انجام عمل تفاضل برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

عملگر ضرب دکارتی (Cartesian Product)

این عملگر توسط نماد × نمایش داده می‌شود. عملگر × یک عملگر اصلی است. عملگر ضرب دکارتی گاها با نماد Times نیز نشان داده می‌شود. عملگر × جهت اتصال کلیه سطرهای دو جدول که در هم ضرب می‌شود، مورد استفاده قرار می‌گیرد. در جبر رابطه‌ای ضرب دکارتی هر دو رابطه دلخواه امکان‌پذیر است و شرط خاصی برای ضرب ندارد. عملگر ضرب دکارتی یکی از پرهزینه‌ترین عملگرهای جبر رابطه‌ای است و باید با دقت زیاد مورد استفاده قرار گیرد.

فرم کلی عملگر × به صورت زیر است: $R_3 = R_1 \times R_2$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر × در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر × همواره برابر حاصل جمع درجه روابط مبدا می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1) + \deg(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	15	16

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \times R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

×

c	d	e
5	6	7
8	9	10
11	12	13
14	15	16

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	15	16
3	4	5	6	7
3	4	8	9	10
3	4	11	12	13
3	4	14	15	16

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$\deg(R_3) = 5$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) + \deg(R_2) = 2 + 3 = 5$$

همچنین اگر دو جدول دارای ستون‌های مشترک باشند در خروجی از نقطه‌گذاری بـرای تشـخیص آنها استفاده می‌شود.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	D
5	6	7
8	9	10
11	12	13
14	15	16

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار ۲ و ۳ است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \times R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

×

b	c	d
5	6	7
8	9	10
11	12	13
14	15	16

=

a	R_1.b	R_2.b	c	d
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	15	16
3	4	5	6	7
3	4	8	9	10
3	4	11	12	13
3	4	14	15	16

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$\deg(R_3) = 5$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) + \deg(R_2) = 2 + 3 = 5$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر × همواره برابر حاصل ضرب کاردینالیتی روابط مبدا می‌باشد، به صورت زیر:

$$card(R_3) = card(R_1) \times card(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	15	16

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 4 است، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 4$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \times R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

×

c	d	e
5	6	7
8	9	10
11	12	13
14	15	16

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	15	16
3	4	5	6	7
3	4	8	9	10
3	4	11	12	13
3	4	14	15	16

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 8 است، به صورت زیر:

$$card(R_3) = 8$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) = card(R_1) \times card(R_2) = 2 \times 4 = 8$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر × مابین دو رابطه R_1 و R_2 همواره برابر هر ترکیبی از کلیدهای کاندید روابط مبدا یعنی هر ترکیبی از یک کلید کاندید R_1 و یک کلید کاندید R_2 می‌باشد، به صورت زیر:

$C.K.(R_3) = C.K.(R_1) + C.K.(R_2)$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	×	c	d	e	=	a	b	c	d	e
1	2		5	6	7		1	2	5	6	7
3	2		8	9	10		1	2	8	9	10
			11	12	13		1	2	11	12	13
			14	6	7		1	2	14	6	7
							3	2	5	6	7
							3	2	8	9	10
							3	2	11	12	13
							3	2	14	6	7

از آنجا که کلید کاندید رابطه مقصد حاصل از عملگر × مابین دو رابطه R_1 و R_2 همواره برابر هر ترکیبی از کلیدهای کاندید روابط مبدا یعنی هر ترکیبی از یک کلید کاندید R_1 و یک کلید کاندید R_2 می‌باشد، پس کلید کاندید رابطه مقصد یعنی R_3 برابر ترکیبی از یک کلید کاندید R_1 و یک کلید کاندید R_2 است، به صورت زیر:

$C.K.(R_3) = a + c = ac$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1) + C.K.(R_2)$

خاصیت جابه‌جایی

عملگر × دارای خاصیت جابه‌جایی است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر × بیانگر عملگر ضرب دکارتی باشد، همواره تساوی زیر برقرار است:

$R_1 \times R_2 = R_2 \times R_1$

خاصیت شرکت‌پذیری

عملگر × دارای خاصیت شرکت‌پذیری است. به طور کلی اگر R_1، R_2 و R_3 سه رابطه و عملگر × بیانگر عمل اجتماع باشد، همواره تساوی زیر برقرار است:

$R_1 \times (R_2 \times R_3) = (R_1 \times R_2) \times R_3$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$\Pi_{S\#}(SP)$

در پرانتز داخلی یعنی داخل جدول SP تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2

پرس و جو: مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$\Pi_{S.S\#,Sname,City}(\sigma_{S.S\#=SP.S\#}(S \times SP))$

در پرانتز داخلی یعنی حاصل ضرب دکارتی دو جدول S و SP به صورت زیر است:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

×

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

=

S.S#	Sname	City	SP.S#	P#	QTY
S1	Sn1	C1	S1	P1	10
S1	Sn1	C1	S1	P2	20
S1	Sn1	C1	S2	P1	30
S2	Sn2	C2	S1	P1	10
S2	Sn2	C2	S1	P2	20
S2	Sn2	C2	S2	P1	30
S3	Sn3	C2	S1	P1	10
S3	Sn3	C2	S1	P2	20
S3	Sn3	C2	S2	P1	30

در ادامه پس از اجرای عملگر انتخاب، خروجی پرس و جو به صورت زیر است:

S.S#	Sname	City	SP.S#	P#	QTY
S1	Sn1	C1	S1	P1	10
S1	Sn1	C1	S1	P2	20
S2	Sn2	C2	S2	P1	30

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S.S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

همچنین مطابق پرس و جوی مطرح شده در جبر رابطه‌ای به شکل بهینه‌تری داریم:

$$\Pi_{S.S\#,Sname,City}\left(\sigma_{S.S\#=SP.S\#}\left(S\times\left(\Pi_{s\#}(SP)\right)\right)\right)$$

در پرانتز داخلی توسط عملگر پرتو از جدول SP شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#
S1
S2

در ادامه پس از حرکت به سمت خارج و انجام عملگر ضرب دکارتی مابین خروجی پرانتز داخلی و جدول S خروجی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

×

S#
S1
S2

=

S#	Sname	City	SP.S#
S1	Sn1	C1	S1
S1	Sn1	C1	S2
S2	Sn2	C2	S1
S2	Sn2	C2	S2
S3	Sn3	C2	S1
S3	Sn3	C2	S2

که در نهایت پس از حرکت به سمت خارج و انجام عملگر انتخاب و بعد پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S.S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

پرس و جو: جفت شماره تولیدکننده و شماره قطعه‌ای که برای شماره تولیدکننده مورد نظر قطعه‌ای تولید نشده باشد:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#,P\#}(S\times P)-\Pi_{S\#,P\#}(SP)$$

در پرس و جوی سمت چپ عملگر تفاضل حاصل ضرب دکارتی دو جدول S و P قرار دارد، یعنی تمامی حالت‌های همه تولیدکنندگان با همه قطعات، به صورت زیر:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

×

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

=

S#	Sname	City	P#	Pname	Color
S1	Sn1	C1	P1	Pn1	Red
S1	Sn1	C1	P2	Pn2	Blue
S2	Sn2	C2	P1	Pn1	Red
S2	Sn2	C2	P2	Pn2	Blue
S3	Sn3	C2	P1	Pn1	Red
S3	Sn3	C2	P2	Pn2	Blue

که پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگان و کلیه قطعات در تمامی حالت‌ها استخراج می‌شود، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1
S2	P2
S3	P1
S3	P2

همچنین در پرس و جوی سمت راست عملگر تفاضل یعنی در جدول SP شماره کلیه تولیدکنندگانی که حداقل یک قطعه تولیدکرده‌اند قرار دارد، که در نهایت پس از اجرای عملگر پرتو شماره کلیه تولیدکنندگان به همراه قطعه‌ای که تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1

که در نهایت پس از اجرای عملگر تفاضل، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	P#
S1	P1
S1	P2
S2	P1
S2	P2
S3	P1
S3	P2

$-$

S#	P#
S1	P1
S1	P2
S2	P1

$=$

S#	P#
S2	P2
S3	P1
S3	P2

به طور کلی، اگر عبارت نفی در پرس و جو استفاده شده باشد، این امر نشانه‌ای برای ضرورت استفاده از عملگر تفاضل است. دقت کنید که در انجام عمل تفاضل برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

عملگر الحاق طبیعی (Natural Join)

این عملگر توسط نماد $\bowtie$ نمایش داده می‌شود. عملگر $\bowtie$ یک عملگر فرعی است. عملگر الحاق طبیعی گاها با نماد Join نیز نشان داده می‌شود. عملگر $\bowtie$ جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است. اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر

دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

فرم کلی عملگر ⋈ به صورت زیر است:

$$R_3 = R_1 \bowtie R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر ⋈ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر ⋈ همواره برابر حاصل جمع درجه روابط مبدا منهای ستون یا ستون‌های مشترک می‌باشد، به صورت زیر:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k$$

همچنین ستون‌های رابطه مقصد حاصل از عملگر ⋈ همواره برابر اجتماع تمام ستون‌های روابط مبدا است. به عبارت دیگر، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های R={a,b} و S={b,c,d} و ستون b به عنوان ستون مشترک باشد، آنگاه ستون‌های $r \bowtie s$ به صورت زیر خواهد بود:

$$r \bowtie s = \Pi_{R \cup S}\big(r \bowtie s\big) = \Pi_{a,b,c,d}\big(r \bowtie s\big)$$

$R \cup S$ به صورت زیر محاسبه می‌گردد:

$$R \cup S = \{a,b\} \cup \{b,c,d\} = \{a,b,c,d\}$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\deg(R_1) = 2$

$\deg(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\deg(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$\deg(R_3) = (\deg(R_1) + \deg(R_2)) - k = (2+3) - 1 = 5 - 1 = 4$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$\deg(R_1) = 3$

$\deg(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

⋈

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

a	b	c	d	E
1	2	3	5	1
1	2	3	9	1
3	6	7	8	4
5	2	3	5	1
5	2	3	9	1

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$\deg(R_3) = 5$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = (\deg(R_1) + \deg(R_2)) - k = (3+4) - 2 = 7 - 2 = 5$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر ⋈ همواره کوچکتر یا مساوی حاصل ضرب کاردینالیتی روابط مبدا می‌باشد، به صورت زیر:

$$card(R_3) \le card(R_1) \times card(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$card(R_3) = 5$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1) \times card(R_2)$$
$$5 \le 4 \times 5$$
$$5 \le 20$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	D
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 4$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		4	3	2		3	4	3	2
5	6		6	8	3		5	6	8	3
7	8		8	7	9		7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$$card(R_3) = 4$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1) \times card(R_2)$$
$$4 \le 4 \times 4$$
$$4 \le 16$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	D
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	1		1	5	7		1	1	5	7
2	1		1	3	8		1	1	3	8
			1	6	9		1	1	6	9
							2	1	5	7
							2	1	3	8
							2	1	6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 6 است، به صورت زیر:

$$card(R_3) = 6$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1) \times card(R_2)$$
$$6 \le 2 \times 3$$
$$6 \le 6$$

پس، **بیشترین** کاردینالیتی حاصل از عملگر الحاق طبیعی زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		5	6	7					
3	4		8	9	10					
			11	12	13					

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$card(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1) \times card(R_2)$$
$$0 \le 2 \times 3$$
$$0 \le 6$$

پس، **کمترین** کاردینالیتی حاصل از عملگر الحاق طبیعی زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، هیچ مقدار یکسانی نداشته باشند.

توجه: در صورتی که مابین دو رابطه‌ای که الحاق طبیعی می‌شوند، کلید خارجی تعریف گردد، آنگاه کاردینالیتی رابطه مقصد حاصل از عملگر ⋈ همواره کوچکتر یا مساوی کاردینالیتی رابطه مبدایی که در آن کلید خارجی قرار دارد می‌باشد، به صورت زیر:

$$card(R_3) \le card(R_2)$$

کلید خارجی برای ارتباط میان جداول مورد استفاده قرار می گیرد. کلید خارجی در دو دیگاه ساختاری و محتوایی مورد بررسی قرار می‌گیرد:

دیدگاه ساختاری کلید خارجی

تعریف: اگر صفت(هایی) در یک جدول به عنوان کلید خارجی تعریف شود،
اول اینکه: این صفت(ها) در جدول خودش شرط خاصی ندارد یعنی الزاما خاصیت کلیدی ندارد.
و دوم اینکه: این صفت(ها) در همان جدول یا جدول دیگری، **باید کلید کاندید** (اصلی یا فرعی) باشد.

دیدگاه محتوایی کلید خارجی

به ازای هر مقدار موجود در یک کلید خارجی، باید دقیقاً یک مقدار متناظر در کلید کاندید متناظر آن وجود داشته باشد، در غیر این صورت می‌گوییم، کلید خارجی دارای ارجاع NULL است. به بیان دیگر، مقادیر کلید خارجی همواره باید زیرمجموعه مقادیر کلید کاندید باشد.
مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
1	8

R_1

b	c	d
2	5	1
6	3	2
2	8	1

R_2

ستون b در جدول R_2 کلید خارجی و در جدول R_1 کلید کاندید است. بنابراین مقادیر ستون b در جدول R_2 به عنوان کلید خارجی همواره باید زیرمجموعه مقادیر ستون b در جدول R_1 به عنوان کلید کاندید باشد.
کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 3 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
1	8

$\bowtie$

b	c	d
2	5	1
6	3	2
2	8	1

=

a	b	c	d
1	2	5	1
1	2	8	1
5	6	3	2

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$card(R_3) = 3$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_3) \leq card(R_2)$

$3 \leq 3$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
1	8

R_1

b	c	d
2	5	1
NULL	3	2
2	8	1

R_2

ستون b در جدول R_2 کلید خارجی و در جدول R_1 کلید کاندید است. بنابراین مقادیر ستون b در جدول R_2 به عنوان کلید خارجی همواره باید زیرمجموعه مقادیر ستون b در جدول R_1 به عنوان کلید کاندید باشد.

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 3 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
1	8

$\bowtie$

b	c	d
2	5	1
NULL	3	2
2	8	1

=

a	b	c	d
1	2	5	1
1	2	8	1

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$card(R_3) = 2$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_3) \leq card(R_2)$

$2 \leq 3$

کلید کاندید رابطه مقصد

در صورتی‌که دو جدول R_1 و R_2 ستون یا ستون‌های مشترک داشته باشند، و تعریف کلید خارجی مابین دو جدول R_1 و R_2 برقرار باشد، و ستون مشترک در جدول R_2 کلید خارجی و

در جدول R_1 کلید کاندید باشد، آنگاه کلید کاندید رابطه مقصد حاصل از عملگر $\bowtie$ مابین دو رابطه R_1 و R_2، همواره برابر کلید کاندید رابطه R_2 می‌باشد، به عبارت دیگر، کلید کاندید رابطه مقصد حاصل از عملگر $\bowtie$ مابین دو رابطه R_1 و R_2، همواره برابر کلید کاندید رابطه مبدایی است که در آن کلید خارجی تعریف شده است، به صورت زیر:

$$C.K.(R_3) = C.K.(R_2)$$

زیرا رابطه مقصد حاصل از عملگر $\bowtie$ در شرایط تعریف کلید خارجی، شامل همان سطرهای جدول R_2 است که ستون‌های غیر مشترک جدول R_1 نیز به دلیل انجام الحاق طبیعی به آن اضافه شده است. پس کلید کاندید رابطه مقصد حاصل از عملگر $\bowtie$ مابین دو رابطه R_1 و R_2 همواره برابر کلید کاندید رابطه مبدایی است که در آن کلید خارجی تعریف شده است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	B
1	2
3	4
5	6
1	8

R_1

b	c	d
2	5	1
6	3	2
2	8	1

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
1	8

$\bowtie$

b	c	d
2	5	1
6	3	2
2	8	1

=

a	b	c	d
1	2	5	1
1	2	8	1
5	6	3	2

از آنجا که کلید کاندید رابطه مقصد حاصل از عملگر $\bowtie$ مابین دو رابطه R_1 و R_2 همواره برابر کلید کاندید رابطه مبدایی است که در آن کلید خارجی تعریف شده است، پس کلید کاندید رابطه مقصد یعنی R_3 برابر کلید کاندید R_2 است، به صورت زیر:

$$C.K.(R_3) = c$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_3) = C.K.(R_2)$$

خاصیت جابه‌جایی

عملگر $\bowtie$ دارای خاصیت جابه‌جایی است. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\bowtie$ بیانگر عمل الحاق طبیعی باشد، همواره تساوی زیر برقرار است:

$$R_1 \bowtie R_2 = R_2 \bowtie R_1$$

خاصیت شرکت‌پذیری

عملگر $\bowtie$ دارای خاصیت شرکت‌پذیری است. به طور کلی اگر R_1، R_2 و R_3 سه رابطه و عملگر $\bowtie$ بیانگر عمل الحاق طبیعی باشد، همواره تساوی زیر برقرار است:

$$R_1 \bowtie (R_2 \bowtie R_3) = (R_1 \bowtie R_2) \bowtie R_3$$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$$R_3 = R_1 \bowtie R_2 = R_1 \cap R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\bowtie$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \cap R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\cap$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$$R_3 = R_1 \bowtie R_2 = R_1 \cap R_2$$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. به صورت زیر:

$$R_3 = R_1 \bowtie R_2 = R_1 \times R_2$$

توجه: عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

$\bowtie$

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

×

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_3 = R_1 \bowtie R_2 = R_1 \times R_2$

شبیه‌سازی

عملگر الحاق طبیعی یک عملگر فرعی است که می‌توان توسط سه عملگر اصلی انتخاب، پرتو و ضرب دکارتی آنرا شبیه‌سازی نمود، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه $r \bowtie s$ به صورت زیر قابل شبیه‌سازی است:

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.b=s.b}(r \times s)) = \Pi_{a,s.b,c,d}(\sigma_{r.b=s.b}(r \times s))$$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \bowtie s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\Pi_{R \cup S}(\sigma_{r.b=s.b}(r \times s)) = \Pi_{a,s.b,c,d}(\sigma_{r.b=s.b}(r \times s))$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

×

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	r.b	s.b	c	d
1	2	2	5	1
1	2	2	8	3
5	6	6	3	2
7	2	2	5	1
7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	s.b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

بنابراین رابطه زیر برقرار خواهد بود:

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.b=s.b}(r \times s)) = \Pi_{a,s.b,c,d}(\sigma_{r.b=s.b}(r \times s))$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#}(SP)$$

در پرانتز داخلی یعنی داخل جدول SP تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2

پرس و جو: مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#,Sname,City}(S \bowtie SP)$$

در پرانتز داخلی یعنی حاصل الحاق طبیعی دو جدول S و SP مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، به صورت زیر:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\bowtie$

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

=

S#	Sname	City	P#	QTY
S1	Sn1	C1	P1	10
S1	Sn1	C1	P2	20
S2	Sn2	C2	P1	30

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

همچنین مطابق پرس و جوی مطرح شده در جبر رابطه‌ای به شکل بهینه‌تری داریم:

$$S \bowtie (\Pi_{S\#} SP)$$

در پرانتز داخلی توسط عملگر پرتو از جدول SP شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#
S1
S2

در نهایت پس از حرکت به سمت خارج و انجام عملگر الحاق طبیعی مابین خروجی پرانتز داخلی و جدول S مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\bowtie$

S#
S1
S2

=

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

عملگر نیم پیوند چپ (Left Semi Join)

این عملگر توسط نماد $\ltimes$ نمایش داده می‌شود. عملگر $\ltimes$ یک عملگر فرعی است. عملگر نیم پیوند چپ گاها با نماد **Left Semi Join** نیز نشان داده می‌شود. عملگر $\ltimes$ همانند عملگر الحاق طبیعی جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد اما با این تفاوت که عملگر نیم پیوند چپ فقط ستون‌های جدول سمت چپ را در خروجی قرار می‌دهد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است. اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

فرم کلی عملگر $\ltimes$ به صورت زیر است:

$$R_3 = R_1 \ltimes R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.
اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\ltimes$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\ltimes$ همواره برابر درجه رابطه مبدا سمت چپ می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b
1	2
5	6
7	2

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$\deg(R_3) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) = 2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$$\deg(R_1) = 3$$
$$\deg(R_2) = 4$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

$\ltimes$

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

a	b	c
1	2	3
3	6	7
5	2	3

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$\deg(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) = 3$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $\ltimes$ همواره کوچکتر یا مساوی کاردینالیتی رابطه مبدا سمت چپ می‌باشد، به صورت زیر:

$$card(R_3) \leq card(R_1)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

$=$

a	b
1	2
5	6
7	2

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1)$$
$$3 \le 4$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\ltimes$	b	c	d	=	a	b
1	2		2	5	1		1	2
3	4		4	3	2		3	4
5	6		6	8	3		5	6
7	8		8	7	9		7	8

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_3) \le card(R_1)$

$4 \le 4$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	d
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$card(R_1) = 2$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\ltimes$	b	c	d	=	a	b
1	1		1	5	7		1	1
2	1		1	3	8		2	1
			1	6	9			

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$$card(R_3) = 2$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1)$$
$$2 \le 2$$

پس، **بیشترین** کاردینالیتی حاصل از عملگر نیم پیوند چپ زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و یا تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	B
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\ltimes$	b	c	d	=	a	b
1	2		5	6	7			
3	4		8	9	10			
			11	12	13			

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$card(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_1)$$
$$0 \le 2$$

پس، **کمترین** کاردینالیتی حاصل از عملگر نیم پیوند چپ زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، هیچ مقدار یکسانی نداشته باشند.
همچنین کاردینالیتی رابطه مقصد حاصل از عملگر $\ltimes$ همواره کوچکتر یا مساوی کاردینالیتی الحاق طبیعی روابط مبدا می‌باشد، زیرا به دلیل حذف ستون‌های جدول مبدا سمت راست و انتخاب فقط ستون‌های جدول مبدا سمت چپ ممکن است سطرهای تکراری ایجاد گردد، که مطابق قاعده جبررابطه‌ای سطرهای تکراری حذف می‌شوند به صورت زیر:

$$card(R_1 \ltimes R_2) \leq card(R_1 \bowtie R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b
1	2
5	6
7	2

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$$card(R_3) = 5$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_1 \ltimes R_2) \le card(R_1 \bowtie R_2)$$
$$3 \le 5$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 4$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋉	b	c	d	=	a	b
1	2		2	5	1		1	2
3	4		4	3	2		3	4
5	6		6	8	3		5	6
7	8		8	7	9		7	8

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$$card(R_3) = 4$$

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		4	3	2		3	4	3	2
5	6		6	8	3		5	6	8	3
7	8		8	7	9		7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \ltimes R_2) \le card(R_1 \bowtie R_2)$

$4 \le 4$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\ltimes$ همواره برابر کلید کاندید رابطه مبدا سمت چپ می‌باشد، به صورت زیر:

$C.K.(R_3) = C.K.(R_1)$

زیرا رابطه مقصد حاصل از عملگر $\ltimes$ همواره زیر مجموعه رابطه مبدا سمت چپ است.

شرط لازم برای زیرمجموعه بودن رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b
1	2
5	6
7	2

از آنجا که شروط سازگاری مابین رابطه مبدا سمت چپ یعنی R_1 و رابطه مقصد یعنی R_3 برقرار است، و همچنین همه سطرهای رابطه مقصد R_3 یعنی $R_1 \ltimes R_2$ در رابطه R_1 قرار دارد، پس رابطه R_3 زیر مجموعه رابطه R_1 می‌باشد، یعنی:

$$R_3 \subseteq R_1$$

پس کلید کاندید رابطه مقصد یعنی R_3 برابر همان کلید کاندید رابطه مبدا سمت چپ یعنی R_1 است، که برابر a است، به صورت زیر:

$$C.K.(R_3) = a$$

بنابراین رابطه زیر برقرار خواهد بود:

$$C.K.(R_3) = C.K.(R_1) = a$$

خاصیت جابه‌جایی

عملگر $\ltimes$ داری خاصیت جابه‌جایی نیست. زیرا عملگر نیم پیوند چپ همواره ستون‌های جدول سمت چپ را در خروجی قرار می‌دهد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\ltimes$ بیانگر عمل نیم پیوند چپ باشد، رابطه زیر برقرار است:

$R_1 \ltimes R_2 \neq R_2 \ltimes R_1$

همچنین رابطه زیر نیز همواره برقرار است:

$R_1 \ltimes R_2 = R_2 \rtimes R_1$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$R_3 = R_1 \ltimes R_2 = R_1 \rtimes R_2 = R_1 \cap R_2$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\ltimes$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\rtimes$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cap R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\cap$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$$R_3 = R_1 \ltimes R_2 = R_1 \rtimes R_2 = R_1 \cap R_2$$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. دقت کنید که عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند. در عملگر نیم پیوند چپ اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر نیم پیوند چپ فقط جدول مبدا سمت چپ را به طور کامل با همه سطرها و ستون‌ها در خروجی قرار خواهد داد، و مانند عملگر ضرب دکارتی رفتار نخواهد کرد. زیرا عملگر نیم پیوند چپ فقط ستون‌های جدول سمت چپ را در خروجی قرار می‌دهد.

$$R_1 \ltimes R_2 \neq R_1 \times R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

$\ltimes$

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b
1	2
3	2

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

×

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_1 \ltimes R_2 \neq R_1 \times R_2$

شبیه‌سازی

عملگر نیم پیوند چپ یک عملگر فرعی است که می‌توان توسط سه عملگر اصلی انتخاب، پرتو و ضرب دکارتی آنرا شبیه‌سازی نمود. اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های R={a,b} و S={b,c,d} و ستون b به عنوان ستون مشترک باشد، آنگاه $r \ltimes s$ به صورت زیر قابل شبیه‌سازی است:

$$r \ltimes s = \Pi_R \left(\sigma_{r.b=s.b}(r \times s)\right) = \Pi_{a,r.b} \left(\sigma_{r.b=s.b}(r \times s)\right)$$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \ltimes s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋉	b	c	d	=	a	b
1	2		2	5	1		1	2
3	4		6	3	2		5	6
5	6		2	8	3		7	2
7	2		9	6	7			
			7	4	5			

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\Pi_{R}\left(\sigma_{r.b=s.b}(r\times s)\right)=\Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r\times s)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	×	b	c	d	=	a	r.b	s.b	c	d
1	2		2	5	1		1	2	2	5	1
3	4		6	3	2		1	2	2	8	3
5	6		2	8	3		5	6	6	3	2
7	2		9	6	7		7	2	2	5	1
			7	4	5		7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	r.b
1	2
5	6
7	2

بنابراین رابطه زیر برقرار خواهد بود:

$$r\ltimes s=\Pi_{R}\left(\sigma_{r.b=s.b}(r\times s)\right)=\Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r\times s)\right)$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$S \ltimes SP$$

داخل پرانتز یعنی حاصل عملگر نیم پیوند چپ دو جدول S و SP مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\ltimes$

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

=

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

همچنین مطابق پرس و جوی مطرح شده در جبر رابطه‌ای به شکل بهینه‌تری داریم:

$$S \ltimes \left(\Pi_{S\#} SP\right)$$

در پرانتز داخلی توسط عملگر پرتو از جدول SP شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#
S1
S2

در نهایت پس از حرکت به سمت خارج و انجام عملگر نیم پیوند چپ مابین خروجی پرانتز داخلی و جدول S مشخصات تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند استخراج می‌گردد، به صورت زیر:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\ltimes$

S#
S1
S2

=

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2

عملگر نیم پیوند راست (Right Semi Join)

این عملگر توسط نماد $\rtimes$ نمایش داده می‌شود. عملگر $\rtimes$ یک عملگر فرعی است. عملگر نیم پیوند راست گاها با نماد **Right Semi Join** نیز نشان داده می‌شود. عملگر $\rtimes$ همانند عملگر الحاق طبیعی جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد اما با این تفاوت که عملگر نیم پیوند راست فقط ستون‌های جدول سمت راست را در خروجی قرار می‌دهد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر

نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است. اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

فرم کلی عملگر $\bowtie$ به صورت زیر است:

$$R_3 = R_1 \bowtie R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\bowtie$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\bowtie$ همواره برابر درجه رابطه مبدا سمت راست می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\rtimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

b	c	d
2	5	1
2	8	3
6	3	2

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$\deg(R_3) = 3$

بنابراین رابطه زیر برقرار خواهد بود:

$\deg(R_3) = \deg(R_2) = 3$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$\deg(R_1) = 3$

$\deg(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

$\rtimes$

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\deg(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_2) = 4$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $\rtimes$ همواره کوچکتر یا مساوی کاردینالیتی رابطه مبدا سمت راست می‌باشد، به صورت زیر:

$$card(R_3) \le card(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \rtimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\rtimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

b	c	d
2	5	1
2	8	3
6	3	2

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_3) \le card(R_2)$

$3 \le 5$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\rtimes$	b	c	d	=	b	c	d
1	2		2	5	1		2	5	1
3	4		4	3	2		4	3	2
5	6		6	8	3		6	8	3
7	8		8	7	9		8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_3) \le card(R_2)$

$4 \le 4$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	d
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\text{card}(R_1) = 2$

$\text{card}(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋊	b	c	d	=	b	c	d
1	1		1	5	7		1	5	7
2	1		1	3	8		1	3	8
			1	6	9		1	6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$\text{card}(R_3) = 3$

بنابراین رابطه زیر برقرار خواهد بود:

$\text{card}(R_3) \le \text{card}(R_2)$

$3 \le 3$

پس، **بیشترین** کاردینالیتی حاصل از عملگر نیم پیوند راست زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و یا تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\text{card}(R_1) = 2$

$\text{card}(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋊	b	c	d	=	b	c	d
1	2		5	6	7				
3	4		8	9	10				
			11	12	13				

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$card(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_3) \le card(R_2)$$
$$0 \le 3$$

پس، **کمترین** کاردینالیتی حاصل از عملگر نیم پیوند راست زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، هیچ مقدار یکسانی نداشته باشند.

همچنین کاردینالیتی رابطه مقصد حاصل از عملگر ⋊ همواره کوچکتر یا مساوی کاردینالیتی الحاق طبیعی روابط مبدا می‌باشد، زیرا به دلیل حذف ستون‌های جدول مبدا سمت چپ و انتخاب فقط ستون‌های جدول مبدا سمت راست ممکن است سطرهای تکراری ایجاد گردد، که مطابق قاعده جبررابطه‌ای سطرهای تکراری حذف می‌شوند به صورت زیر:

$$card(R_1 \rtimes R_2) \le card(R_1 \bowtie R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$card(R_1) = 4$$
$$card(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \rtimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\rtimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

b	c	d
2	5	1
2	8	3
6	3	2

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$card(R_3) = 3$

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$card(R_3) = 5$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \rtimes R_2) \leq card(R_1 \bowtie R_2)$

$3 \leq 5$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋊	b	c	d	=	b	c	d
1	2		2	5	1		2	5	1
3	4		4	3	2		4	3	2
5	6		6	8	3		6	8	3
7	8		8	7	9		8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋊	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		4	3	2		3	4	3	2
5	6		6	8	3		5	6	8	3
7	8		8	7	9		7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \rtimes R_2) \le card(R_1 \bowtie R_2)$

$4 \le 4$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر ⋊ همواره برابر کلید کاندید رابطه مبدا سمت راست می‌باشد، به صورت زیر:

$C.K.(R_3) = C.K.(R_2)$

زیرا رابطه مقصد حاصل از عملگر ⋈ همواره زیر مجموعه رابطه مبدا سمت راست است.

شرط لازم برای زیرمجموعه بودن رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\rtimes$	b	c	d	=	b	c	d
1	2		2	5	1		2	5	1
3	4		6	3	2		2	8	3
5	6		2	8	3		6	3	2
7	2		9	6	7				
			7	4	5				

از آنجا که شروط سازگاری مابین رابطه مبدا سمت راست یعنی R_2 و رابطه مقصد یعنی R_3 برقرار است، و همچنین همه سطرهای رابطه مقصد R_3 یعنی $R_1 \rtimes R_2$ در رابطه R_2 قرار دارد، پس رابطه R_3 زیر مجموعه رابطه R_2 می‌باشد، یعنی:

$R_3 \subseteq R_2$

پس کلید کاندید رابطه مقصد یعنی R_3 برابر همان کلید کاندید رابطه مبدا سمت راست یعنی R_2 است، که برابر c است، به صورت زیر:

$C.K.(R_3) = c$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_2) = c$

خاصیت جابه‌جایی

عملگر $\rtimes$ داری خاصیت جابه‌جایی نیست. زیرا عملگر نیم پیوند راست همواره ستون‌های جدول سمت راست را در خروجی قرار می‌دهد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\rtimes$ بیانگر عمل نیم پیوند راست باشد، رابطه زیر برقرار است:

$R_1 \rtimes R_2 \neq R_2 \rtimes R_1$

همچنین رابطه زیر نیز همواره برقرار است:

$R_1 \rtimes R_2 = R_2 \rtimes R_1$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$R_3 = R_1 \rtimes R_2 = R_1 \ltimes R_2 = R_1 \cap R_2$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\rtimes$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\rtimes$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \cap R_2$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\cap$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_3 = R_1 \rtimes R_2 = R_1 \bowtie R_2 = R_1 \cap R_2$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. دقت کنید که عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند. در عملگر نیم پیوند راست اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر نیم پیوند راست فقط جدول مبدا سمت راست را به طور کامل با همه سطرها و ستون‌ها در خروجی قرار خواهد داد، و مانند عملگر ضرب دکارتی رفتار نخواهد کرد. زیرا عملگر نیم پیوند راست فقط ستون‌های جدول سمت راست را در خروجی قرار می‌دهد.

$R_1 \rtimes R_2 \neq R_1 \times R_2$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \rtimes R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋊	c	d	e	=	c	d	e
1	2		5	6	7		5	6	7
3	2		8	9	10		8	9	10
			11	12	13		11	12	13
			14	6	7		14	6	7

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	×	c	d	e	=	a	b	c	d	e
1	2		5	6	7		1	2	5	6	7
3	2		8	9	10		1	2	8	9	10
			11	12	13		1	2	11	12	13
			14	6	7		1	2	14	6	7
							3	2	5	6	7
							3	2	8	9	10
							3	2	11	12	13
							3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_1 \rtimes R_2 \neq R_1 \times R_2$

شبیه‌سازی

عملگر نیم پیوند راست یک عملگر فرعی است که می‌توان توسط سه عملگر اصلی انتخاب، پرتو و ضرب دکارتی آنرا شبیه‌سازی نمود. اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه $r \rtimes s$ به صورت زیر قابل شبیه‌سازی است:

$$r \rtimes s = \Pi_S (\sigma_{r.b=s.b}(r \times s)) = \Pi_{s.b,c,d} (\sigma_{r.b=s.b}(r \times s))$$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \rtimes s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\rtimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

b	c	d
2	5	1
2	8	3
6	3	2

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\Pi_{(S)}(\sigma_{r.b=s.b}(r \times s)) = \Pi_{(s.b,c,d)}(\sigma_{r.b=s.b}(r \times s))$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

×

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	r.b	s.b	c	d
1	2	2	5	1
1	2	2	8	3
5	6	6	3	2
7	2	2	5	1
7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

s.b	c	d
2	5	1
2	8	3
6	3	2

بنابراین رابطه زیر برقرار خواهد بود:

$$r \rtimes s = \Pi_S (\sigma_{r.b=s.b}(r \times s)) = \Pi_{s.b,c,d} (\sigma_{r.b=s.b}(r \times s))$$

الحاق طبیعی نیم پیوند چپ و نیم پیوند راست

در الحاق طبیعی نیم پیوند چپ و نیم پیوند راست، همواره تساوی زیر برقرار است:

$$(R_1 \ltimes R_2) \bowtie (R_1 \rtimes R_2) = R_1 \bowtie R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_1 \ltimes R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b
1	2
5	6
7	2

همچنین پرس و جوی زیر را در نظر بگیرید:

$$R_1 \rtimes R_2$$

a	b
1	2
3	4
5	6
7	2

$\rtimes$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

b	c	d
2	5	1
2	8	3
6	3	2

حال پرس و جوی زیر را در نظر بگیرید:

$(R_1 \ltimes R_2) \bowtie (R_1 \rtimes R_2)$

a	b
1	2
5	6
7	2

$\bowtie$

b	c	d
2	5	1
2	8	3
6	3	2

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_1 \bowtie R_2$

a	b
1	2
3	4
5	6
7	2

$\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

بنابراین رابطه زیر برقرار خواهد بود:

$(R_1 \ltimes R_2) \bowtie (R_1 \rtimes R_2) = R_1 \bowtie R_2$

همچنین رابطه زیر برقرار خواهد بود:

$\text{card}((R_1 \ltimes R_2) \bowtie (R_1 \rtimes R_2)) = \text{card}(R_1 \bowtie R_2) = 5$

عملگر الحاق خارجی چپ یا فراپیوند چپ (Left Outer Join)

این عملگر توسط نماد $\Rightarrow\!\bowtie$ نمایش داده می‌شود. عملگر $\Rightarrow\!\bowtie$ یک عملگر فرعی است. عملگر الحاق خارجی چپ گاها با نماد **Left Outer Join** نیز نشان داده می‌شود. عملگر $\Rightarrow\!\bowtie$ همانند عملگر الحاق طبیعی جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد اما با این تفاوت که عملگر الحاق خارجی چپ علاوه بر اینکه کلیه سطرهای پیوندپذیر را در خروجی قرار می‌دهد، کلیه سطرهای پیوندناپذیر جدول سمت چپ را نیز در خروجی قرار می‌دهد

و در این حالت برای ستون‌های غیرمشترک جدول سمت راست مقدار NULL در نظر می‌گیرد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است. اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

فرم کلی عملگر $\Rightarrow\!\bowtie$ به صورت زیر است:

$$R_3 = R_1 \Rightarrow\!\bowtie R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\Rightarrow\!\bowtie$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\Rightarrow\!\bowtie$ همواره برابر حاصل جمع درجه روابط مبدا منهای ستون یا ستون‌های مشترک می‌باشد، به صورت زیر:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k$$

همچنین ستون‌های رابطه مقصد حاصل از عملگر $\Rightarrow\!\bowtie$ همواره برابر اجتماع تمام ستون‌های روابط مبدا است. به عبارت دیگر، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه ستون‌های $r \Rightarrow\!\bowtie s$ به صورت زیر خواهد بود:

$$r \Rightarrow\!\bowtie s = \Pi_{R \cup S}\big(r \Rightarrow\!\bowtie s\big) = \Pi_{a,b,c,d}\big(r \Rightarrow\!\bowtie s\big)$$

$R \cup S$ به صورت زیر محاسبه می‌گردد:

$$R \cup S = \{a,b\} \cup \{b,c,d\} = \{a,b,c,d\}$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\deg(R_1) = 2$

$\deg(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\Rightarrow\!\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
3	4	NULL	NULL

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\deg(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = \big(2+3\big) - 1 = 5 - 1 = 4$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$\deg(R_1) = 3$

$\deg(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

$\Rightarrow\!\bowtie$

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

a	b	c	d	e
1	2	3	5	1
1	2	3	9	1
3	6	7	8	4
5	2	3	5	1
5	2	3	9	1
8	4	5	NULL	NULL

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$\deg(R_3) = 5$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = (3+4) - 2 = 7 - 2 = 5$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $\Rightarrow\!\bowtie$ همواره کوچکتر یا مساوی حاصل ضرب کاردینالیتی روابط مبدا و بزرگتر یا مساوی کاردینالیتی الحاق طبیعی روابط مبدا می‌باشد، زیرا در الحاق خارجی چپ علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت چپ نیز در خروجی قرار می‌گیرد، به صورت زیر:

$$\text{card}\big(R_1 \bowtie R_2\big) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 5$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\Rightarrow\!\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
3	4	NULL	NULL

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 6 است، به صورت زیر:

$card(R_3) = 6$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$5 \le 6 \le 4 \times 5$

$5 \le 6 \le 20$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		4	3	2		3	4	3	2
5	6		6	8	3		5	6	8	3
7	8		8	7	9		7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$card(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$4 \le 4 \le 4 \times 4$

$4 \le 4 \le 16$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	d
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$card(R_1) = 2$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	1		1	5	7		1	1	5	7
2	1		1	3	8		1	1	3	8
			1	6	9		1	1	6	9
							2	1	5	7
							2	1	3	8
							2	1	6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 6 است، به صورت زیر:

$card(R_3) = 6$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$6 \le 6 \le 2 \times 3$

$6 \le 6 \le 6$

پس، **بیشترین** کاردینالیتی حاصل از عملگر الحاق خارجی چپ زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$card(R_1) = 2$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	$\Rightarrow\!\bowtie$	b	c	d	=	a	b	c	d
1	2		5	6	7		1	2	NULL	NULL
3	4		8	9	10		3	4	NULL	NULL
			11	12	13					

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 2 است، به صورت زیر:

$card(R_3) = 2$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$0 \le 2 \le 2 \times 3$

$0 \le 2 \le 6$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 0 و 3 است، به صورت زیر:

$$card(R_1) = 0$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\bowtie R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b

$\Rightarrow\!\bowtie$

b	c	d
5	6	7
8	9	10
11	12	13

=

a	b	c	d

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$card(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$$
$$0 \le 0 \le 0$$
$$0 \le 0 \le 0$$

پس، **کمترین** کاردینالیتی حاصل از عملگر الحاق خارجی چپ زمانی رخ می‌دهد که جدول سمت چپ تهی باشد.

همچنین در یک قاعده کلی رابطه زیر برقرار است:

$$(R_1 \bowtie R_2) \subseteq (R_1 \Rightarrow\!\bowtie R_2)$$

یعنی الحاق طبیعی دو رابطه همواره زیر مجموعه الحاق خارجی چپ دو رابطه است.

شرط لازم برای زیر مجموعه بودن، رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$(R_1 \bowtie R_2)$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$(R_1 \Rightarrow\!\bowtie R_2)$

a	b	⇒⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							3	4	NULL	NULL

از آنجا که شروط سازگاری مابین رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ و رابطه سمت راست یعنی $(R_1 \Rightarrow\!\bowtie R_2)$ برقرار است، و همچنین همه سطرهای رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ در رابطه سمت راست یعنی $(R_1 \Rightarrow\!\bowtie R_2)$ قرار دارد، پس رابطه $(R_1 \bowtie R_2)$ زیر مجموعه رابطه $(R_1 \Rightarrow\!\bowtie R_2)$ می‌باشد، یعنی:

$$(R_1 \bowtie R_2) \subseteq (R_1 \Rightarrow\!\bowtie R_2)$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\Rightarrow\!\bowtie$ در شرایط کلی وجود ندارد، زیرا مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد.

خاصیت جابه‌جایی

عملگر $\Rightarrow\!\bowtie$ دارای خاصیت جابه‌جایی نیست. زیرا عملگر الحاق خارجی چپ علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت چپ را نیز در خروجی قرار می‌دهد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\Rightarrow\!\bowtie$ بیانگر عمل الحاق خارجی چپ باشد، رابطه زیر برقرار است:

$$R_1 \Rightarrow\!\bowtie R_2 \neq R_2 \Rightarrow\!\bowtie R_1$$

همچنین رابطه زیر نیز همواره برقرار است:

$$R_1 \Rightarrow\!\bowtie R_2 = R_2 \bowtie\!\Leftarrow R_1$$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$$R_3 = R_1 \Rightarrow\!\bowtie R_2 = R_1$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\bowtie R_2$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\Rightarrow\!\bowtie$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6
7	8	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_3 = R_1 \Rightarrow\!\bowtie R_2 = R_1$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. دقت کنید که عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند. در عملگر الحاق خارجی چپ اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق خارجی چپ دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

$R_1 \Rightarrow\!\bowtie R_2 = R_1 \times R_2$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\bowtie R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

$\Rightarrow\!\bowtie$

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

×

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_1 \Rightarrow\!\bowtie R_2 = R_1 \times R_2$

شبیه‌سازی

عملگر الحاق خارجی چپ یک عملگر فرعی است که می‌توان توسط پنج عملگر اصلی انتخاب، پرتو، تفاضل، ضرب دکارتی و اجتماع آنرا شبیه‌سازی نمود، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه $r \Rightarrow\!\bowtie s$ به صورت زیر قابل شبیه‌سازی است:

$$r \Rightarrow\!\bowtie = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \cup \left(\left(r - \Pi_{a,r.b}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \times \{NULL, NULL\}\right)$$

تعداد NULL به صورت زیر محاسبه می‌گردد:

S-R = {b, c, d}-{a, b} ={c, d}

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	D
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \Rightarrow\!\bowtie s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\Rightarrow\!\bowtie$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
3	4	NULL	NULL

همچنین پرس و جوی زیر را در نظر بگیرید:

$\left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

×

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	r.b	s.b	c	d
1	2	2	5	1
1	2	2	8	3
5	6	6	3	2
7	2	2	5	1
7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	s.b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$\left(\Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)$

a	b	×	b	c	d	=	a	r.b	s.b	c	d
1	2		2	5	1		1	2	2	5	1
3	4		6	3	2		1	2	2	8	3
5	6		2	8	3		5	6	6	3	2
7	2		9	6	7		7	2	2	5	1
			7	4	5		7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	r.b
1	2
5	6
7	2

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(r - \Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)$$

a	b	−	a	b	=	a	b
1	2		1	2		3	4
3	4		5	6			
5	6		7	2			
7	2						

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(\left(r - \Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r \times s)\right)\right) \times \{NULL, NULL\}\right)$$

a	b	×	c	d	=	a	b	c	d
3	4		NULL	NULL		3	4	NULL	NULL

حال پرس و جوی زیر را در نظر بگیرید:

$$\left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b\,=\,s.b}(r \times s)\right)\right) \cup \left(\left(r - \Pi_{a,r.b}\left(\sigma_{r.b\,=\,s.b}(r \times s)\right)\right) \times \{NULL, NULL\}\right)$$

a	s.b	c	d	∪	a	b	c	d	=	a	b	c	d
1	2	5	1		3	4	NULL	NULL		1	2	5	1
1	2	8	3							1	2	8	3
5	6	3	2							5	6	3	2
7	2	5	1							7	2	5	1
7	2	8	3							7	2	8	3
										3	4	NULL	NULL

بنابراین رابطه زیر برقرار خواهد بود:

$$r \Rightarrow\!\bowtie s = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \cup \left(\left(r - \Pi_{a,r.b}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \times \{NULL, NULL\}\right)$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: شماره، نام تولیدکننده، نام شهر تولیدکننده، شماره قطعه و تعداد قطعات تولیدی توسط تولیدکننده مربوط به تولیدکنندگان:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$(S \Rightarrow\!\bowtie SP)$$

داخل پرانتز یعنی حاصل عملگر الحاق خارجی چپ دو جدول S و SP مشخصات شماره، نام تولیدکننده، نام شهر تولیدکننده، شماره قطعه و تعداد قطعات تولیدی توسط تولیدکنندگان مربوط به تولیدکنندگان قرار دارد، که در نهایت، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City	P#	QTY
S1	Sn1	C1	P1	10
S1	Sn1	C1	P2	20
S2	Sn2	C2	P1	30
S3	Sn3	C2	NULL	NULL

عملگر الحاق خارجی راست یا فراپیوند راست (Right Outer Join)

این عملگر توسط نماد $\Rightarrow\!\bowtie$ نمایش داده می‌شود. عملگر الحاق خارجی راست گاها با نماد **Right Outer Join** نیز نشان داده می‌شود. عملگر $\Rightarrow\!\bowtie$ همانند عملگر الحاق طبیعی جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد اما با این تفاوت که عملگر الحاق خارجی راست علاوه بر اینکه کلیه سطرهای پیوندپذیر را در خروجی قرار می‌دهد، کلیه سطرهای پیوندناپذیر جدول سمت راست را نیز در خروجی قرار می‌دهد و در این حالت برای ستون‌های

غیرمشترک جدول سمت چپ مقدار NULL در نظر می‌گیرد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است.
اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.
فرم کلی عملگر $\bowtie$ به صورت زیر است:

$$R_3 = R_1 \bowtie R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.
اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\bowtie$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\bowtie$ همواره برابر حاصل جمع درجه روابط مبدا منهای ستون یا ستون‌های مشترک می‌باشد، به صورت زیر:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k$$

همچنین ستون‌های رابطه مقصد حاصل از عملگر $\bowtie$ همواره برابر اجتماع تمام ستون‌های روابط مبدا است. به عبارت دیگر، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های R={a,b} و S={b,c,d} و ستون b به عنوان ستون مشترک باشد، آنگاه ستون‌های s $\bowtie$ r به صورت زیر خواهد بود:

$$r \bowtie s = \Pi_{R \cup S}(r \bowtie s) = \Pi_{a,b,c,d}(r \bowtie s)$$

$R \cup S$ به صورت زیر محاسبه می‌گردد:

$$R \cup S = \{a,b\} \cup \{b,c,d\} = \{a,b,c,d\}$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\deg(R_1) = 2$

$\deg(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes\!\!= R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\ltimes\!\!=$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
NULL	9	6	7
NULL	7	4	5

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\deg(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = (2+3) - 1 = 5 - 1 = 4$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$\deg(R_1) = 3$

$\deg(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie\!\!\!\!= R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

$\bowtie\!\!\!\!=$

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

a	b	c	d	e
1	2	3	5	1
1	2	3	9	1
3	6	7	8	4
5	2	3	5	1
5	2	3	9	1
NULL	9	8	1	2

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$\deg(R_3) = 5$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = (3+4) - 2 = 7 - 2 = 5$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر $\bowtie\!\!\!\!=$ همواره کوچکتر یا مساوی حاصل ضرب کاردینالیتی روابط مبدا و بزرگتر یا مساوی کاردینالیتی الحاق طبیعی روابط مبدا می‌باشد، زیرا در الحاق خارجی راست علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت راست نیز در خروجی قرار می‌گیرند، به صورت زیر:

$$\text{card}\big(R_1 \bowtie R_2\big) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 5$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie\!\!\!\!= R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈=	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							NULL	9	6	7
							NULL	7	4	5

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 7 است، به صورت زیر:

$card(R_3) = 7$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$5 \le 7 \le 4 \times 5$

$5 \le 7 \le 20$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$card(R_1) = 4$

$card(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie\!\!\!\!= R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈⇐	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		4	3	2		3	4	3	2
5	6		6	8	3		5	6	8	3
7	8		8	7	9		7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\text{card}(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$

$4 \le 4 \le 4 \times 4$

$4 \le 4 \le 16$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	d
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$\text{card}(R_1) = 2$

$\text{card}(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie\!\!\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈⇐	b	c	d	=	a	b	c	d
1	1		1	5	7		1	1	5	7
2	1		1	3	8		1	1	3	8
			1	6	9		1	1	6	9
							2	1	5	7
							2	1	3	8
							2	1	6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 6 است، به صورت زیر:

$\text{card}(R_3) = 6$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$$
$$6 \le 6 \le 2 \times 3$$
$$6 \le 6 \le 6$$

پس، **بیشترین** کاردینالیتی حاصل از عملگر الحاق خارجی راست زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$card(R_1) = 2$$
$$card(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie\!\!= R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈=	b	c	d	=	a	b	c	d
1	2		5	6	7		NULL	5	6	7
3	4		8	9	10		NULL	8	9	10
			11	12	13		NULL	11	12	13

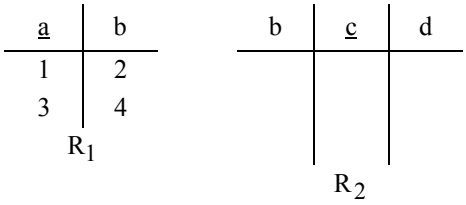
کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$$
$$0 \le 3 \le 2 \times 3$$
$$0 \le 3 \le 6$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:



کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 0 است، به صورت زیر:

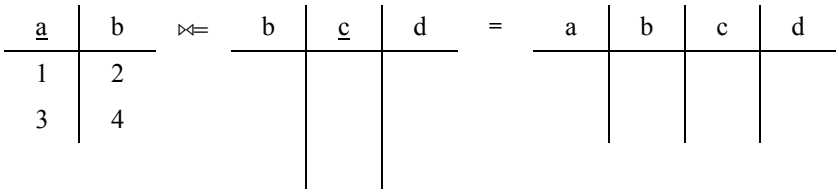
$card(R_1) = 2$

$card(R_2) = 0$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \ltimes\!\!= R_2$

خروجی پرس و جوی فوق به صورت زیر است:



کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$card(R_3) = 0$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \ltimes\!\!= R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$0 \le 0 \le 0$

$0 \le 0 \le 0$

پس، **کمترین** کاردینالیتی حاصل از عملگر الحاق خارجی راست زمانی رخ می‌دهد که جدول سمت راست تهی باشد.

همچنین در یک قاعده کلی رابطه زیر برقرار است:

$(R_1 \ltimes\!\!= R_2) \subseteq (R_1 \ltimes\!\!= R_2)$

یعنی الحاق طبیعی دو رابطه همواره زیر مجموعه الحاق خارجی راست دو رابطه است.

شرط لازم برای زیر مجموعه بودن، رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$(R_1 \bowtie R_2)$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$(R_1 \bowtie\!\!\!= R_2)$

a	b	⋈=	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							NULL	9	6	7
							NULL	7	4	4

از آنجا که شروط سازگاری مابین رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ و رابطه سمت راست یعنی $(R_1 \bowtie\!\!= R_2)$ برقرار است، و همچنین همه سطرهای رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ در رابطه سمت راست یعنی $(R_1 \bowtie\!\!= R_2)$ قرار دارد، پس رابطه $(R_1 \bowtie R_2)$ زیر مجموعه رابطه $(R_1 \bowtie\!\!= R_2)$ می‌باشد، یعنی:

$$(R_1 \bowtie R_2) \subseteq (R_1 \bowtie\!\!= R_2)$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\bowtie\!\!=$ در شرایط کلی وجود ندارد، زیرا مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد.

خاصیت جابه‌جایی

عملگر $\bowtie\!\!=$ دارای خاصیت جابه‌جایی نیست. زیرا عملگر الحاق خارجی راست علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت راست را نیز در خروجی قرار می‌دهد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\bowtie\!\!=$ بیانگر عمل الحاق خارجی راست باشد، رابطه زیر برقرار است:

$$R_1 \bowtie\!\!= R_2 \neq R_2 \bowtie\!\!= R_1$$

همچنین رابطه زیر نیز همواره برقرار است:

$$R_1 \bowtie\!\!= R_2 = R_2 =\!\!\bowtie R_1$$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$$R_3 = R_1 \bowtie\!\!= R_2 = R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \bowtie\!\!= R_2$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

⋈⇐

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6
7	9	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_3 = R_1 \Rightarrow\!\bowtie R_2 = R_2$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. دقت کنید که عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند. در عملگر الحاق خارجی راست اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق خارجی راست دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

$R_1 \bowtie\!\Leftarrow R_2 = R_1 \times R_2$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \bowtie\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

⋈⇐

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

×

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_1 \ltimes R_2 = R_1 \times R_2$

شبیه‌سازی

عملگر الحاق خارجی راست یک عملگر فرعی است که می‌توان توسط پنج عملگر اصلی انتخاب، پرتو، تفاضل، ضرب دکارتی و اجتماع آنرا شبیه‌سازی نمود، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه $r \ltimes s$ به صورت زیر قابل شبیه‌سازی است:

$$r \ltimes s = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \cup \left(\{NULL\} \times \left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right)\right)$$

تعداد NULL به صورت زیر محاسبه می‌گردد:

$R\text{-}S = \{a, b\}\text{-}\{b, c, d\} = \{a\}$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \rtimes\!\!\!\!\leftarrow s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈⇐	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							NULL	9	6	7
							NULL	7	4	5

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	×	b	c	d	=	a	r.b	s.b	c	d
1	2		2	5	1		1	2	2	5	1
3	4		6	3	2		1	2	2	8	3
5	6		2	8	3		5	6	6	3	2
7	2		9	6	7		7	2	2	5	1
			7	4	5		7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

a	s.b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(\Pi_{s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)$$

a	b	×	b	c	d	=	a	r.b	s.b	c	d
1	2		2	5	1		1	2	2	5	1
3	4		6	3	2		1	2	2	8	3
5	6		2	8	3		5	6	6	3	2
7	2		9	6	7		7	2	2	5	1
			7	4	5		7	2	2	8	3

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

s.b	c	d
2	5	1
2	8	3
6	3	2

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right)$$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

$-$

s.b	c	d
2	5	1
2	8	3
6	3	2

$=$

b	c	d
9	6	7
7	4	5

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\left(\{NULL\} \times \left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)\right)$$

a
NULL

$\times$

b	c	d
9	6	7
7	4	5

$=$

a	b	c	d
NULL	9	6	7
NULL	7	4	5

حال پرس و جوی زیر را در نظر بگیرید:

$$\left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right) \cup \left(\{NULL\} \times \left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)\right)$$

a	s.b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3

$\cup$

a	b	c	d
NULL	9	6	7
NULL	7	4	5

$=$

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
NULL	9	6	7
NULL	7	4	5

بنابراین رابطه زیر برقرار خواهد بود:

$$r \Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow s = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right) \cup \left(\{NULL\} \times \left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b = s.b}(r \times s)\right)\right)\right)$$

عملگر الحاق خارجی کامل یا فراپیوند کامل (Full Outer Join)

این عملگر توسط نماد $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ نمایش داده می‌شود. عملگر $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ یک عملگر فرعی است. عملگر الحاق خارجی کامل گاها با نماد Full Outer Join نیز نشان داده می‌شود. عملگر $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ همانند عملگر الحاق طبیعی جهت الحاق سطرهای پیوندپذیر مابین دو جدول مورد استفاده قرار می‌گیرد اما با این تفاوت که عملگر الحاق خارجی کامل علاوه بر اینکه کلیه سطرهای پیوندپذیر را در خروجی قرار می‌دهد، کلیه سطرهای پیوندناپذیر جدول سمت چپ و راست را نیز در خروجی قرار می‌دهد و در این حالت برای ستون‌های غیرمشترک جدول سمت راست و چپ مقدار NULL در نظر می‌گیرد. سطرهایی از دو جدول که مقدارشان در ستون یا ستون‌های مشترک مساوی است به عنوان سطرهای پیوندپذیر نامیده می‌شوند. در جبر رابطه‌ای الحاق طبیعی هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای الحاق طبیعی دارد. داشتن ستون یا ستون‌های مشترک مابین دو جدول شرط انجام عملگر الحاق طبیعی است. اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک باشند، آنگاه سطرهایی از دو جدول در خروجی قرار می‌گیرد که مقادیرشان در ستون یا ستون‌های مشترک برابر، یکسان و مساوی باشد. همچنین ستون یا ستون‌های مشترک فقط یکبار در خروجی ظاهر می‌شوند. اما اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

فرم کلی عملگر $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ به صورت زیر است:

$$R_3 = R_1 \Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر روابط R_1 و R_2 به عنوان روابط مبدا و رابطه R_3 را به عنوان رابطه مقصد، حاصل از عملگر $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر $\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$ همواره برابر حاصل جمع درجه روابط مبدا منهای ستون یا ستون‌های مشترک می‌باشد، به صورت زیر:

$$\deg(R_3) = (\deg(R_1) + \deg(R_2)) - k$$

همچنین ستون‌های رابطه مقصد حاصل از عملگر $\Rightarrow\!\!\bowtie\!\!\Leftarrow$ همواره برابر اجتماع تمام ستون‌های روابط مبدا است. به عبارت دیگر، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه ستون‌های $r \Rightarrow\!\!\bowtie\!\!\Leftarrow s$ به صورت زیر خواهد بود:

$$r \Rightarrow\!\!\bowtie\!\!\Leftarrow s = \Pi_{R \cup S}(r \Rightarrow\!\!\bowtie\!\!\Leftarrow s) = \Pi_{a,b,c,d}(r \Rightarrow\!\!\bowtie\!\!\Leftarrow s)$$

$R \cup S$ به صورت زیر محاسبه می‌گردد:

$$R \cup S = \{a,b\} \cup \{b,c,d\} = \{a,b,c,d\}$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 3$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\!\bowtie\!\!\Leftarrow R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	2

$\Rightarrow\!\!\bowtie\!\!\Leftarrow$

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

=

a	b	c	d
1	2	5	1
1	2	8	3
5	6	3	2
7	2	5	1
7	2	8	3
3	4	NULL	NULL
NULL	9	6	7
NULL	7	4	5

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$\deg(R_3) = 4$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = (2+3) - 1 = 5 - 1 = 4$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

R_1

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 3 و 4 است، به صورت زیر:

$\deg(R_1) = 3$

$\deg(R_2) = 4$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c
1	2	3
3	6	7
5	2	3
8	4	5

$\Rightarrow\!\!\!\bowtie\!\!\!\Leftarrow$

b	c	d	e
2	3	5	1
6	7	8	4
2	3	9	1
9	8	1	2

=

a	b	c	d	e
1	2	3	5	1
1	2	3	9	1
3	6	7	8	4
5	2	3	5	1
5	2	3	9	1
8	4	5	NULL	NULL
NULL	9	8	1	2

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$\deg(R_3) = 5$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \big(\deg(R_1) + \deg(R_2)\big) - k = (3+4) - 2 = 7 - 2 = 5$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه

مقصد حاصل از عملگر =⋈= همواره کوچکتر یا مساوی حاصل ضرب کاردینالیتی روابط مبدا و بزرگتر یا مساوی کاردینالیتی الحاق طبیعی روابط مبدا می‌باشد، زیرا در الحاق خارجی کامل علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت چپ و راست نیز در خروجی قرار می‌گیرند، به صورت زیر:

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 5 است، به صورت زیر:

$$\text{card}(R_1) = 4$$
$$\text{card}(R_2) = 5$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 =\bowtie= R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	=⋈=	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							3	4	NULL	NULL
							NULL	9	6	7
							NULL	7	4	5

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 8 است، به صورت زیر:

$$\text{card}(R_3) = 8$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$
$$5 \le 8 \le 4 \times 5$$
$$5 \le 8 \le 20$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	8

R_1

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 4 و 4 است، به صورت زیر:

$$\text{card}(R_1) = 4$$
$$\text{card}(R_2) = 4$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\Leftarrow R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4
5	6
7	8

$\Rightarrow\!\Leftarrow$

b	c	d
2	5	1
4	3	2
6	8	3
8	7	9

=

a	b	c	d
1	2	5	1
3	4	3	2
5	6	8	3
7	8	7	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 4 است، به صورت زیر:

$$\text{card}(R_3) = 4$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$
$$4 \le 4 \le 4 \times 4$$
$$4 \le 4 \le 16$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	1
2	1

R_1

b	c	d
1	5	7
1	3	8
1	6	9

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$card(R_1) = 2$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	1
2	1

$\Rightarrow\!\Leftarrow$

b	c	d
1	5	7
1	3	8
1	6	9

=

a	b	c	d
1	1	5	7
1	1	3	8
1	1	6	9
2	1	5	7
2	1	3	8
2	1	6	9

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 6 است، به صورت زیر:

$card(R_3) = 6$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$6 \le 6 \le 2 \times 3$

$6 \le 6 \le 6$

پس، **بیشترین** کاردینالیتی حاصل از عملگر الحاق خارجی کامل زمانی رخ می‌دهد که ستون‌های مشترک، در دو رابطه‌ای که در عمل الحاق طبیعی شرکت کرده‌اند، دقیقا مشابه یکدیگر باشند و تنها از یک مقدار استفاده کرده باشند.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4

R_1

b	c	d
5	6	7
8	9	10
11	12	13

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 3 است، به صورت زیر:

$card(R_1) = 2$

$card(R_2) = 3$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\!\!\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	4

$\Rightarrow\!\!\!\!\Leftarrow$

b	c	d
5	6	7
8	9	10
11	12	13

=

a	b	c	d
1	2	NULL	NULL
3	4	NULL	NULL
NULL	5	6	7
NULL	8	9	10
NULL	11	12	13

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 5 است، به صورت زیر:

$card(R_3) = 5$

بنابراین رابطه زیر برقرار خواهد بود:

$card(R_1 \bowtie R_2) \le card(R_3) \le card(R_1) \times card(R_2)$

$0 \le 5 \le 2 \times 3$

$0 \le 5 \le 6$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b

R_1

b	c	d

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 0 و 0 است، به صورت زیر:

$card(R_1) = 0$

$card(R_2) = 0$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \Rightarrow\!\!\!\!\Leftarrow R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b

$\Rightarrow\!\!\!\!\Leftarrow$

b	c	d

=

a	b	c	d

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار صفر است، به صورت زیر:

$$\text{card}(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_3) \le \text{card}(R_1) \times \text{card}(R_2)$$

$$0 \le 0 \le 0$$

$$0 \le 0 \le 0$$

پس، **کمترین** کاردینالیتی حاصل از عملگر الحاق خارجی کامل زمانی رخ می‌دهد که جدول سمت چپ و راست تهی باشد.

همچنین در یک قاعده کلی روابط زیر برقرار است:

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_1 \Rightarrow\!\bowtie R_2) \le \text{card}(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$$

$$\text{card}(R_1 \bowtie R_2) \le \text{card}(R_1 \bowtie\!\Leftarrow R_2) \le \text{card}(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$$

زیرا عملگر الحاق خارجی کامل علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت چپ و راست را نیز در خروجی قرار می‌دهد.

همچنین در یک قاعده کلی رابطه زیر برقرار است:

$$(R_1 \bowtie R_2) \subseteq (R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$$

یعنی الحاق طبیعی دو رابطه همواره زیر مجموعه الحاق خارجی کامل دو رابطه است.

شرط لازم برای زیر مجموعه بودن، رعایت شروط سازگاری است، در جبر رابطه‌ای دو شرط به عنوان شروط سازگاری مطرح است:

شرط اول: تعداد ستون‌های دو جدول یکسان باشد، به عبارت دیگر دو رابطه (جدول) هم درجه باشند.

شرط دوم: نوع یا دامنه ستون‌های متناظر در دو جدول یکسان باشد.

اگر بخواهیم دو شرط فوق را در یک جمله بیان بیان کنیم، اینطور خواهد بود، شروط سازگاری یعنی تیترها در دو رابطه (جدول) یکسان باشد.

و شرط کافی برای زیر مجموعه بودن، قرار داشتن کلیه سطرهای رابطه سمت چپ در رابطه سمت راست است.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

R_1

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

R_2

پرس و جوی زیر را در نظر بگیرید:

$(R_1 \bowtie R_2)$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⋈	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3

همچنین پرس و جوی زیر را در نظر بگیرید:

$(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$

a	b	⇒⋈⇐	b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							3	4	NULL	NULL
							NULL	9	6	7
							NULL	7	4	4

از آنجا که شروط سازگاری مابین رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ و رابطه سمت راست یعنی $(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$ برقرار است، و همچنین همه سطرهای رابطه سمت چپ یعنی $(R_1 \bowtie R_2)$ در رابطه سمت راست یعنی $(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$ قرار دارد، پس رابطه $(R_1 \bowtie R_2)$ زیر مجموعه رابطه $(R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$ می‌باشد، یعنی:

$(R_1 \bowtie R_2) \subseteq (R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$

همچنین در یک قاعده کلی روابط زیر برقرار است:

$(R_1 \Rightarrow\!\bowtie R_2) \subseteq (R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$

یعنی الحاق خارجی چپ دو رابطه همواره زیر مجموعه الحاق خارجی کامل دو رابطه است.

$(R_1 \bowtie\!\Leftarrow R_2) \subseteq (R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$

یعنی الحاق خارجی راست دو رابطه همواره زیر مجموعه الحاق خارجی کامل دو رابطه است.

$$((R_1 \Rightarrow\!\bowtie R_2) \cup (R_1 \bowtie\!\Leftarrow R_2)) = (R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2)$$

$$((R_1 \Rightarrow\!\bowtie R_2) \cap (R_1 \bowtie\!\Leftarrow R_2)) = (R_1 \bowtie R_2)$$

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر $\Rightarrow\!\bowtie\!\Leftarrow$ در شرایط کلی وجود ندارد، زیرا مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد.

خاصیت جابه‌جایی

عملگر $\Rightarrow\!\bowtie\!\Leftarrow$ داری خاصیت جابه‌جایی است. زیرا عملگر الحاق خارجی کامل علاوه بر سطرهای پیوندپذیر دو جدول، سطرهای پیوندناپذیر جدول سمت چپ و راست را نیز در خروجی قرار می‌دهد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر $\Rightarrow\!\bowtie\!\Leftarrow$ بیانگر عمل الحاق خارجی کامل باشد، رابطه زیر برقرار است:

$$R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2 = R_2 \Rightarrow\!\bowtie\!\Leftarrow R_1$$

موارد خاص

اگر تمامی ستون‌های دو رابطه R_1 و R_2 یکسان باشند، یعنی همه ستون‌های آنها ستون مشترک باشد، همواره تساوی زیر برقرار است:

$$R_3 = R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2 = R_1 \cup R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c
1	2	3
4	5	6
7	8	6

R_1

a	b	c
1	2	3
4	5	6
7	9	6

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\bowtie\!\Leftarrow R_2$$

خروجی پرس و جوی پرانتز داخلی به صورت زیر است:

a	b	c
1	2	3
4	5	6
7	8	6

$\Rightarrow\!\bowtie\!\Leftarrow$

a	b	c
1	2	3
4	5	6
7	9	6

=

a	b	c
1	2	3
4	5	6
7	8	6
7	9	6

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$$R_3 = R_1 \Rightarrow\!\!\!\!\Leftarrow R_2 = R_1 \cup R_2$$

همانطور که گفتیم، اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق طبیعی، دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد. دقت کنید که عملگر الحاق طبیعی یک عملگر اصلی نیست، بلکه یک عملگر فرعی است، پس در ضمیرناخودآگاه خود روح و ذات عملگرهای اصلی را دارد، تاکید می‌کنیم که عملگرهای فرعی صرفا جهت ساده‌نویسی پرس و جوها مورد استفاده قرار می‌گیرند، درحالیکه روح و ذات عملگرهای اصلی که از آنها ساخته شده‌اند را درون خود به صورت نهفته به ارث برده‌اند. در عملگر الحاق خارجی کامل اگر دو جدولی که با یکدیگر الحاق طبیعی می‌شوند دارای ستون یا ستون‌های مشترک نباشند، آنگاه عملگر الحاق خارجی کامل دقیقا مانند عملگر ضرب دکارتی رفتار خواهد کرد.

$$R_1 \Rightarrow\!\!\!\!\Leftarrow R_2 = R_1 \times R_2$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
3	2

R_1

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

R_2

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \Rightarrow\!\!\!\!\Leftarrow R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	⇒⇐	c	d	e	=	a	b	c	d	e
1	2		5	6	7		1	2	5	6	7
3	2		8	9	10		1	2	8	9	10
			11	12	13		1	2	11	12	13
			14	6	7		1	2	14	6	7
							3	2	5	6	7
							3	2	8	9	10
							3	2	11	12	13
							3	2	14	6	7

همچنین پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \times R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
3	2

×

c	d	e
5	6	7
8	9	10
11	12	13
14	6	7

=

a	b	c	d	e
1	2	5	6	7
1	2	8	9	10
1	2	11	12	13
1	2	14	6	7
3	2	5	6	7
3	2	8	9	10
3	2	11	12	13
3	2	14	6	7

بنابراین رابطه زیر در شرایط مذکور برقرار خواهد بود:

$R_1 \Rightarrow\!\!\bowtie\!\!\Leftarrow R_2 = R_1 \times R_2$

شبیه‌سازی

عملگر الحاق خارجی کامل یک عملگر فرعی است که می‌توان توسط پنج عملگر اصلی انتخاب، پرتو، تفاضل، ضرب دکارتی و اجتماع آنرا شبیه‌سازی نمود، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های $R=\{a,b\}$ و $S=\{b,c,d\}$ و ستون b به عنوان ستون مشترک باشد، آنگاه $r \Rightarrow\!\!\bowtie\!\!\Leftarrow s$ به صورت زیر قابل شبیه‌سازی است:

$$r \Rightarrow\!\!\bowtie\!\!\Leftarrow s = (r \Rightarrow\!\!\bowtie s) \cup (r \bowtie\!\!\Leftarrow s)$$

$$r \Rightarrow\!\!\bowtie s = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right) \cup \left(\left(r - \Pi_{a,r.b}\left(\sigma_{r.b=s.b}(r \times s)\right)\right) \times \{NULL, NULL\}\right)$$

تعداد NULL به صورت زیر محاسبه می‌گردد:

$S-R = \{b, c, d\} - \{a, b\} = \{c, d\}$

$$r \bowtie\!\!\Leftarrow s = \left(\Pi_{a,s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right) \cup \left(\{NULL\} \times \left(s - \Pi_{s.b,c,d}\left(\sigma_{r.b=s.b}(r \times s)\right)\right)\right)$$

تعداد NULL به صورت زیر محاسبه می‌گردد:

$R-S = \{a, b\} - \{b, c, d\} = \{a\}$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
1	2
3	4
5	6
7	2

r

b	c	d
2	5	1
6	3	2
2	8	3
9	6	7
7	4	5

s

پرس و جوی زیر را در نظر بگیرید:

$R_3 = r \Rightarrow\!\!\!\!\leftarrow s$

خروجی پرس و جوی فوق به صورت زیر است:

a	b		b	c	d	=	a	b	c	d
1	2		2	5	1		1	2	5	1
3	4		6	3	2		1	2	8	3
5	6		2	8	3		5	6	3	2
7	2		9	6	7		7	2	5	1
			7	4	5		7	2	8	3
							3	4	NULL	NULL
							NULL	9	6	7
							NULL	7	4	5

همچنین پرس و جوی زیر را در نظر بگیرید:

$(r \Rightarrow\!\!\triangleleft\ s) \cup (r \triangleright\!\!\leftarrow s)$

مطابق آنچه پیش از این درباره شبیه‌سازی عملگر الحاق خارجی چپ و راست گفتیم، خروجی پرس و جوی فوق به صورت زیر خواهد بود:

a	b	c	d	∪	a	b	c	d	=	a	b	c	d
1	2	5	1		1	2	5	1		1	2	5	1
1	2	8	3		1	2	8	3		1	2	8	3
5	6	3	2		5	6	3	2		5	6	3	2
7	2	5	1		7	2	5	1		7	2	5	1
7	2	8	3		7	2	8	3		7	2	8	3
3	4	NULL	NULL		NULL	9	6	7		3	4	NULL	NULL
					NULL	7	4	5		NULL	9	6	7
										NULL	7	4	5

بنابراین رابطه زیر برقرار خواهد بود:

$r \Rightarrow\!\!\!\!\leftarrow s = (r \Rightarrow\!\!\triangleleft\ s) \cup (r \triangleright\!\!\leftarrow s)$

عملگر تقسیم (Division)

این عملگر توسط نماد ÷ نمایش داده می‌شود. عملگر ÷ یک عملگر فرعی است. عملگر تقسیم گاها با نماد DIV یا DIVIDEBY نیز نشان داده می‌شود. عملگر ÷ زمانی مورد استفاده قرار می‌گیرد که پرس و جو بخواهد **همه** حالت‌های یک اتفاق را بررسی کند. مانند، شماره تولیدکنندگانی که همه قطعات را تولید کرده‌اند. در جبر رابطه‌ای تقسیم هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای تقسیم دارد. فقط روابطی را می‌توان برهم تقسیم کرد که تمام خصیصه‌های مقسوم علیه در مقسوم باشد. به عبارت دیگر فقط روابطی را می‌توان برهم تقسیم کرد که مجموعه ستون‌های مقسوم علیه، زیر مجموعه، مجموعه ستون‌های مقسوم باشد. اگر دو جدولی که بر یکدیگر تقسیم می‌شوند دارای شرط تقسیم‌پذیری باشند، آنگاه سطرهایی از جدول مقسوم در خروجی قرار می‌گیرد که همه سطرهای جدول مقسوم علیه را در کنار خود داشته باشد. اما اگر دو جدولی که بر یکدیگر تقسیم می‌شوند دارای شرط تقسیم‌پذیری نباشند، آنگاه عمل تقسیم، قابل انجام نخواهد بود.

فرم کلی عملگر به صورت زیر است:

$$R_3 = R_1 \div R_2$$

در بخش R_1 و R_2 نام جدول یا جداول یا خروجی یک پرس و جو یعنی یک عبارت جبر رابطه‌ای دیگر می‌تواند قرار گیرد.

اگر رابطه R_1 به عنوان رابطه مقسوم و رابطه R_2 به عنوان رابطه مقسوم علیه و رابطه R_3 را به عنوان رابطه خارج قسمت، حاصل از عملگر ÷ در نظر بگیریم، آنگاه قواعد زیر را داریم:

درجه رابطه مقصد

به تعداد ستون‌های یک رابطه، درجه رابطه گفته می‌شود. تعداد ستون‌های یک رابطه همواره بزرگتر از صفر می‌باشد، یعنی جدول بدون ستون نداریم. درجه رابطه مقصد حاصل از عملگر ÷ همواره برابر درجه رابطه مقسوم منهای درجه رابطه مقسوم علیه می‌باشد، به صورت زیر:

$$\deg(R_3) = \deg(R_1) - \deg(R_2)$$

همچنین ستون‌های رابطه مقصد یعنی خارج قسمت حاصل از عملگر ÷ همواره برابر تفاضل مجموعه ستون‌های رابطه مقسوم و مجموعه ستون‌های رابطه مقسوم علیه است. به عبارت دیگر، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های R={a,b} و S={b} باشد، آنگاه ستون‌های r÷s به صورت زیر خواهد بود:

$$r \div s = \Pi_{R-S}(r \div s) = \Pi_{a}(r \div s)$$

R-S به صورت زیر محاسبه می‌گردد:

R-S = {a, b}-{b} = {a}

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	2
2	4
2	3
1	3
1	4
5	3
7	8

R_1

b
3
4

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 2 و 1 است، به صورت زیر:

$$\deg(R_1) = 2$$
$$\deg(R_2) = 1$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \div R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
1	2
2	4
2	3
1	3
1	4
5	3
7	8

÷

b
3
4

=

a
1
2

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 1 است، به صورت زیر:

$$\deg(R_3) = 1$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) - \deg(R_2) = 2 - 1 = 1$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c	d	e
5	6	7	1	2
5	6	7	3	4
8	2	3	6	7
6	4	5	3	4
7	8	9	1	2
6	4	5	1	2
5	6	7	8	9
5	6	8	1	2
5	6	8	3	4
6	2	5	1	2
6	2	5	3	4
2	6	8	1	2
2	6	8	3	4

R_1

d	e
1	2
3	4

R_2

درجه روابط فوق به عنوان روابط مبدا برابر مقدار 5 و 2 است، به صورت زیر:

$\deg(R_1) = 5$

$\deg(R_2) = 2$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \div R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c	d	e
5	6	7	1	2
5	6	7	3	4
8	2	3	6	7
6	4	5	3	4
7	8	9	1	2
6	4	5	1	2
5	6	7	8	9
5	6	8	1	2
5	6	8	3	4
6	2	5	1	2
6	2	5	3	4
2	6	8	1	2
2	6	8	3	4

÷

d	e
1	2
3	4

=

a	b	c
5	6	7
6	4	5
5	6	8
6	2	5
2	6	8

درجه رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$\deg(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$\deg(R_3) = \deg(R_1) - \deg(R_2) = 5 - 2 = 3$$

کاردینالیتی رابطه مقصد

به تعداد سطرهای یک رابطه، کاردینالیتی رابطه گفته می‌شود. تعداد سطرهای یک رابطه برابر صفر یا بزرگتر از صفر می‌باشد، یعنی جدول بدون سطر داریم که جدول تهی است. کاردینالیتی رابطه مقصد حاصل از عملگر ÷ همواره کوچکتر یا مساوی حاصل تقسیم کاردینالیتی روابط مبدا یعنی مقسوم و مقسوم علیه و بزرگتر یا مساوی صفر می‌باشد، به صورت زیر:

$$0 \le \text{card}(R_3) \le \left\lfloor \frac{\text{card}(R_1)}{\text{card}(R_2)} \right\rfloor$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	3
1	4
2	3
2	4
3	3
3	4

R_1

b
3
4

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 6 و 2 است، به صورت زیر:

$$\text{card}(R_1) = 6$$
$$\text{card}(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \div R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	÷	b	=	a
1	3		3		1
1	4		4		2
2	3				3
2	4				
3	3				
3	4				

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$0 \le card(R_3) \le \left\lfloor \frac{card(R_1)}{card(R_2)} \right\rfloor$$

$$0 \le 3 \le \left\lfloor \frac{6}{2} \right\rfloor$$

$$0 \le 3 \le 3$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	3
1	4
2	3
2	4
3	3
3	4
5	6

R_1

b
3
4

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 7 و 2 است، به صورت زیر:

$$card(R_1) = 7$$

$$card(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \div R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	÷	b	=	a
1	3		3		1
1	4		4		2
2	3				3
2	4				
3	3				
3	4				
5	6				

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 3 است، به صورت زیر:

$$card(R_3) = 3$$

بنابراین رابطه زیر برقرار خواهد بود:

$$0 \le card(R_3) \le \left\lfloor \frac{card(R_1)}{card(R_2)} \right\rfloor$$

$$0 \le 3 \le \left\lfloor \frac{7}{2} \right\rfloor$$

$$0 \le 3 \le 3$$

پس، **بیشترین** کاردینالیتی حاصل از عملگر تقسیم زمانی رخ می‌دهد که همه سطرهای مقسوم علیه به عنوان یک گروه کامل در بیشترین حالت ممکن، کنار سطرهای مقسوم به طور کامل تکرار گردد.

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b
1	3
1	5
2	3
2	5
3	3
3	5

R_1

b
3
4

R_2

کاردینالیتی روابط فوق به عنوان روابط مبدا برابر مقدار 6 و 2 است، به صورت زیر:

$$card(R_1) = 6$$

$$card(R_2) = 2$$

پرس و جوی زیر را در نظر بگیرید:

$$R_3 = R_1 \div R_2$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	÷	b	=	a
1	3		3		
1	5		4		
2	3				
2	5				
3	3				
3	5				

کاردینالیتی رابطه حاصل به عنوان رابطه مقصد برابر مقدار 0 است، به صورت زیر:

$$card(R_3) = 0$$

بنابراین رابطه زیر برقرار خواهد بود:

$$0 \le card(R_3) \le \left\lfloor \frac{card(R_1)}{card(R_2)} \right\rfloor$$

$$0 \le 0 \le \left\lfloor \frac{6}{2} \right\rfloor$$

$$0 \le 0 \le 3$$

پس، **کمترین** کاردینالیتی حاصل از عملگر تقسیم زمانی رخ می‌دهد که همه سطرهای مقسوم علیه به عنوان یک گروه کامل در کنار هیچ یک از سطرهای مقسوم به طور کامل تکرار نگردد.

کلید کاندید رابطه مقصد

کلید کاندید رابطه مقصد حاصل از عملگر ÷ به دو فرم زیر وجود دارد:

فرم اول: اگر کلید کاندید رابطه مبدا مقسوم یعنی R_1 در رابطه مقصد خارج قسمت یعنی R_3 باشد، آنگاه کلید کاندید رابطه مقصد یعنی R_3 برابر همان کلید کاندید رابطه مبدا مقسوم یعنی R_1 خواهد بود، به صورت زیر:

$$C.K.(R_3) = C.K.(R_1)$$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c	d	e
1	1	4	1	2
1	2	4	1	2
2	1	5	1	2
3	2	4	1	2
4	4	6	2	1

R_1

d	e
1	2

R_2

کلید کاندید رابطه مبدا مقسوم برابر ab است، به صورت زیر:

$C.K.(R_1) = ab$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \div R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c	d	e
1	1	4	1	2
1	2	4	1	2
2	1	5	1	2
3	2	4	1	2
4	4	6	2	1

÷

d	e
1	2

=

a	b	c
1	1	4
1	2	4
2	1	5
3	2	4

از آنجا که کلید کاندید رابطه مبدا مقسوم یعنی R_1 در رابطه مقصد خارج قسمت یعنی R_3 وجود دارد، پس کلید کاندید رابطه مقصد خارج قسمت یعنی R_3 برابر همان کلید کاندید رابطه مبدا مقسوم یعنی R_1 خواهد بود، به صورت زیر:

$C.K.(R_3) = ab$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = C.K.(R_1)$

فرم دوم: اما اگر کلید کاندید رابطه مبدا مقسوم یعنی R_1 در رابطه مقصد خارج قسمت یعنی R_3 نباشد، آنگاه کلید کاندید رابطه مقصد تمام کلید خواهد بود، به صورت زیر:

$C.K.(R_3) = All\ Key$

مثال: جداول R_1 و R_2 را به صورت زیر در نظر بگیرید:

a	b	c	d	e
5	6	7	1	2
5	6	7	3	4
8	2	3	6	7
6	4	5	3	4
7	8	9	1	2
6	4	5	1	2
5	6	7	8	9
5	6	8	1	2
5	6	8	3	4
6	2	5	1	2
6	2	5	3	4
2	6	8	1	2
2	6	8	3	4

R_1

d	e
1	2
3	4

R_2

کلید کاندید رابطه مبدا مقسوم برابر abcd است، به صورت زیر:

$C.K.(R_1) = abcd$

پرس و جوی زیر را در نظر بگیرید:

$R_3 = R_1 \div R_2$

خروجی پرس و جوی فوق به صورت زیر است:

a	b	c	d	e
5	6	7	1	2
5	6	7	3	4
8	2	3	6	7
6	4	5	3	4
7	8	9	1	2
6	4	5	1	2
5	6	7	8	9
5	6	8	1	2
5	6	8	3	4
6	2	5	1	2
6	2	5	3	4
2	6	8	1	2
2	6	8	3	4

÷

d	e
1	2
3	4

=

a	b	c
5	6	7
6	4	5
5	6	8
6	2	5
2	6	8

از آنجا که کلید کاندید رابطه مبدا یعنی R_1 در رابطه مقصد یعنی R_3 وجود ندارد، پس کلید کاندید رابطه مقصد خارج قسمت یعنی R_3 برابر abc خواهد بود، به صورت زیر:

$C.K.(R_3) = abc$

بنابراین رابطه زیر برقرار خواهد بود:

$C.K.(R_3) = All\ Key$

خاصیت جابه‌جایی

عملگر ÷ داری خاصیت جابه‌جایی نیست. زیرا در جبر رابطه‌ای تقسیم هر دو رابطه دلخواه امکان‌پذیر نیست و شرط خاصی برای تقسیم دارد. فقط روابطی را می‌توان برهم تقسیم کرد که تمام خصیصه‌های مقسوم علیه در مقسوم باشد. به عبارت دیگر فقط روابطی را می‌توان برهم تقسیم کرد که مجموعه ستون‌های مقسوم علیه، زیر مجموعه، مجموعه ستون‌های مقسوم باشد. به طور کلی اگر R_1 و R_2 دو رابطه و عملگر ÷ بیانگر عمل تقسیم باشد، رابطه زیر برقرار است:

$R_1 \div R_2 \neq R_2 \div R_1$

شبیه‌سازی

عملگر تقسیم یک عملگر فرعی است که می‌توان توسط سه عملگر اصلی پرتو، تفاضل و ضرب

دکارتی آنرا شبیه‌سازی نمود، اگر r و s دو رابطه (جدول) به ترتیب با مجموعه ستون‌های R={a,b} و S={b} باشد، آنگاه r ÷ s به صورت زیر قابل شبیه‌سازی است:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}\left(\left(\Pi_{R-S}(r) \times S\right) - r\right)$$

همچنین رابطه زیر برقرار است:

$$\Pi_{R-S,S}(r) = \Pi_{a,b}(r) = r$$

R-S به صورت زیر محاسبه می‌گردد:

$$R\text{-}S = \{a, b\} - \{b\} = \{a\}$$

بنابراین r÷s به صورت زیر نیز قابل شبیه‌سازی است:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}\left(\left(\Pi_{R-S}(r) \times S\right) - \Pi_{R-S,S}(r)\right)$$

مثال: جداول r و s را به صورت زیر در نظر بگیرید:

a	b
5	7
5	1
4	2
4	3
5	2

r

b
1
2

s

پرس و جوی زیر را در نظر بگیرید:

$$r \div s$$

خروجی پرس و جوی فوق به صورت زیر است:

a	b
5	7
5	1
4	2
4	3
5	2

÷

b
1
2

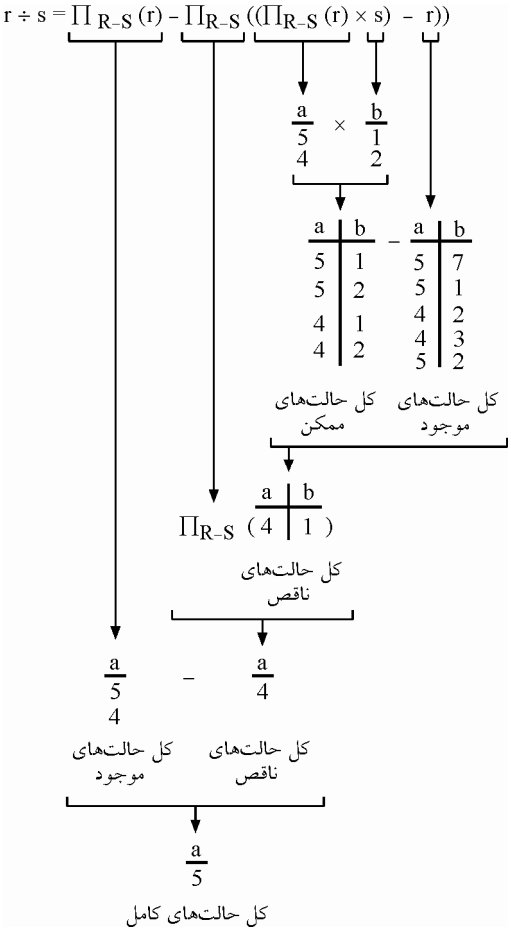
=

a
5

همچنین پرس و جوی زیر را در نظر بگیرید:

$$\Pi_{R-S}(r) - \Pi_{R-S}\left(\left(\Pi_{R-S}(r) \times S\right) - r\right)$$

خروجی پرس و جوی فوق به صورت زیر است:



بنابراین رابطه زیر برقرار خواهد بود:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}\left((\Pi_{R-S}(r) \times S) - r\right)$$

پرس و جو نویسی

در ادامه به بررسی چند پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#,P\#}(SP) \div \Pi_{P\#}(P)$$

در ادامه نحوه شبیه‌سازی عملگر ÷، توسط سه عملگر اصلی پرتو، تفاضل و ضرب دکارتی بررسی شده‌است.

$$\Pi_{S\#,P\#}(SP) \div \Pi_{P\#}(P) = \Pi_{R-S}(r) - \Pi_{R-S}\left((\Pi_{R-S}(r) \times S) - r\right)$$

همچنین مولفه‌های r و s به صورت زیر است:

$$r = \Pi_{S\#,P\#}(SP)$$

$$s = \Pi_{P\#}(P)$$

R-S به صورت زیر محاسبه می‌گردد:

$$R\text{-}S = \{S\#, P\#\} - \{P\#\} = \{S\#\}$$

پس از جایگذاری مولفه‌های فوق در رابطه شبیه‌سازی عملگر تقسیم داریم:

$$\Pi_{S\#}(\Pi_{S\#,P\#}(SP)) - \Pi_{S\#}\left((\Pi_{S\#}(\Pi_{S\#,P\#}(SP)) \times \Pi_{P\#}(P)) - \Pi_{S\#,P\#}(SP)\right)$$

پس از ساده‌سازی پرس و جوی فوق، بر اساس قوانین عملگر پرتو داریم:

$$\Pi_{S\#}(SP) - \Pi_{S\#}\left((\Pi_{S\#}(SP) \times \Pi_{P\#}(P)) - \Pi_{S\#,P\#}(SP)\right)$$

در پرانتز داخلی رابطه سمت چپ عملگر ضرب دکارتی یعنی داخل جدول SP شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2

همچنین در پرانتز داخلی رابطه سمت راست عملگر ضرب دکارتی یعنی داخل جدول P مشخصات کلیه قطعات موجود قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی پرس و جو به صورت زیر خواهد بود:

P#
P1
P2

همچنین در پرانتز داخلی پس از انجام عملگر ضرب دکارتی، خروجی پرس و جو به صورت زیر خواهد بود:

S#	×	P#	=	S#	P#
S1		P1		S1	P1
S2		P2		S1	P2
				S2	P1
				S2	P2

در خروجی فوق مجموعه همه زوج شماره تولیدکننده و شماره قطعه **ممکن** قرار دارد، یعنی چه آن قطعه واقعا توسط آن تولیدکننده تولید شده باشد، و چه تولید نشده باشد. در ادامه و حرکت به سمت خارج و اجرای پرس و جوی سمت راست عملگر تفاضل داریم:

S#	P#
S1	P1
S1	P2
S2	P1

در خروجی فوق مجموعه همه زوج شماره تولیدکننده و شماره قطعه **واقعی** قرار دارد، یعنی آن قطعه واقعا توسط آن تولیدکننده تولید شده باشد. در ادامه و حرکت به سمت خارج و اجرای عملگر تفاضل داریم:

S#	P#	−	S#	P#	=	S#	P#
S1	P1		S1	P1		S2	P2
S1	P2		S1	P2			
S2	P1		S2	P1			
S2	P2						

در خروجی فوق مجموعه همه زوج شماره تولیدکننده و شماره قطعه **غیر واقعی** قرار دارد. در ادامه و حرکت به سمت خارج و اجرای عملگر پرتو داریم:

S#
S2

در خروجی فوق شماره تولیدکنندگانی قرار دارد که حداقل یک قطعه وجود دارد که تولید نکرده‌اند. در ادامه و حرکت به سمت خارج و اجرای پرس و جوی سمت چپ عملگر تفاضل داریم:

S#
S1
S2

در خروجی فوق شماره تولیدکنندگانی قرار دارد که حداقل یک قطعه تولید کرده‌اند. که در نهایت پس از انجام عملگر تفاضل خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#
S1
S2

\-

S#
S2

=

S#
S1

در خروجی فوق شماره تولیدکنندگانی قرار دارد، که هیچ قطعه‌ای نباشد که آنرا تولید نکرده باشد، به عبارت دیگر شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند استخراج شده است.

پرس و جو: شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{S\#,P\#}(SP) \div \Pi_{P\#}(P)$$

توسط عملگر پرتو شرط تقسیم‌پذیری دو رابطه برقرار شده‌است، در پرس و جوی سمت چپ یعنی داخل جدول SP شماره تولیدکنندگانی که حداقل یک قطعه تولید کرده‌اند قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی پرس و جوی سمت چپ به صورت زیر خواهد بود:

S#	P#
S1	P1
S1	P2
S2	P1

در پرس و جوی سمت راست یعنی داخل جدول P مشخصات کلیه قطعات موجود قرار دارد، که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی پرس و جوی سمت راست به صورت زیر خواهد بود:

P#
P1
P2

که در نهایت پس از انجام عملگر تقسیم، خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	P#
S1	P1
S1	P2
S2	P1

÷

P#
P1
P2

=

S#
S1

پرس و جو: مشخصات تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$S \bowtie \left(\Pi_{S\#,P\#}(SP) \div \Pi_{P\#}(P)\right)$$

در پرانتز داخلی مطابق پرس و جوی قبل شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند قرار دارد، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1

÷

P#
P1
P2

=

S#
S1

که در نهایت پس از حرکت به سمت خارج و انجام عملگر الحاق طبیعی خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

⋈

S#
S1

=

S#	Sname	City
S1	Sn1	C1

همچنین مطابق پرس و جوی مطرح شده در جبر رابطه‌ای به شکل دیگری توسط عملگر نیم پیوند چپ نیز داریم:

$$S \ltimes \left(\Pi_{(S\#,P\#)}(SP) \div \Pi_{(P\#)}(P)\right)$$

در پرانتز داخلی مطابق پرس و جوی قبل شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند قرار دارد، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1

÷

P#
P1
P2

=

S#
S1

که در نهایت پس از حرکت به سمت خارج و انجام عملگر نیم پیوند چپ خروجی نهایی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\ltimes$

S#
S1

=

S#	Sname	City
S1	Sn1	C1

پرس و جو: نام تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند:
مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\Pi_{Sname}\left(S \bowtie \left(\Pi_{S\#,P\#}(SP) \div \Pi_{P\#}(P)\right)\right)$$

در پرانتز داخلی مطابق پرس و جوی قبل شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند قرار دارد، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1

÷

P#
P1
P2

=

S#
S1

که در ادامه پس از حرکت به سمت خارج و انجام عملگر الحاق طبیعی خروجی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\bowtie$

S#
S1

=

S#	Sname	City
S1	Sn1	C1

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو خروجی نهایی پرس و جو به صورت زیر خواهد بود:

Sname
Sn1

همچنین مطابق پرس و جوی مطرح شده در جبر رابطه‌ای به شکل دیگری توسط عملگر نیم پیوند چپ داریم:

$$\Pi_{Sname}\left(S \ltimes \left(\Pi_{(S\#,P\#)}(SP) \div \Pi_{(P\#)}(P)\right)\right)$$

در پرانتز داخلی مطابق پرس و جوی قبل شماره تولیدکنندگانی که **همه** قطعات را تولید کرده‌اند قرار دارد، به صورت زیر:

S#	P#
S1	P1
S1	P2
S2	P1

÷

P#
P1
P2

=

S#
S1

که در ادامه پس از حرکت به سمت خارج و انجام عملگر نیم پیوند چپ خروجی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$\ltimes$

S#
S1

=

S#	Sname	City
S1	Sn1	C1

که در نهایت پس از حرکت به سمت خارج و انجام عملگر پرتو خروجی نهایی پرس و جو به صورت زیر خواهد بود:

Sname
Sn1

عملگر نام‌گذاری مجدد (Rename)

این عملگر توسط نماد **ρ** نمایش داده می‌شود. عملگر **ρ** یک عملگر اصلی است. عملگر **ρ** جهت نام‌گذاری مجدد یک جدول یا نام‌گذاری مجدد ستون‌های یک جدول **به طور موقت** مورد استفاده قرار می‌گیرد. محدوده اعتبار عملگر **ρ** در همان پرس وجوی مربوطه است، و پس از پایان پرس و جو نام‌گذاری جدید دیگر اعتبار ندارد. این عملگر فقط یک نام‌گذاری مجدد برای یک جدول است و نه ذخیره‌سازی مجدد جدول مورد نظر.

فرم کلی عملگر **ρ** برای نام‌گذاری مجدد یک جدول به طور موقت به صورت زیر است:

$\rho_E(R)$

فرم کلی عملگر **ρ** برای نام‌گذاری مجدد ستون‌های یک جدول به طور موقت به صورت زیر است:

$\rho_{E(L)}(R)$

در بخش E نام‌گذاری مجدد برای رابطه R مشخص می‌شود. و در بخش L نام‌گذاری مجدد برای ستون‌های رابطه R مشخص می‌شود.

در بخش R نام جدول یا جداول به عنوان مکان یا محل پرس و جو مشخص می‌گردد، این مکان جستجو می‌تواند خروجی یک پرس و جوی دیگر یعنی یک عبارت جبر رابطه‌ای دیگر باشد.

پرس و جو نویسی

در ادامه به بررسی یک پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: جفت شماره تولیدکنندگانی که با هم همشهری هستند:

مطابق قوانین جبر رابطه‌ای، ضرب دکارتی یک جدول در خودش با همان نام جدول مجاز نمی‌باشد، بنابراین در این مواقع یعنی هنگامی که قرار است سطرهای یک جدول در سطرهای همان جدول ضرب دکارتی و بعد باهم مقایسه شود، کافیست نام یکی از دو جدول توسط عملگر نام‌گذاری مجدد به طور موقت نام‌گذاری مجدد گردد.

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\sigma_{S.City=SX.City \wedge S.S\#\neq SX.S\#}\left(\Pi_{S\#,City}(S)\times\rho_{SX}\left(\Pi_{S\#,City}(S)\right)\right)$$

در پرانتز داخلی و داخلی‌ترین پرانتز، رابطه سمت راست عملگر ضرب دکارتی یعنی داخل جدول S مشخصات تولیدکنندگان قرار دارد، که پس از حرکت به سمت خارج و انجام عملگر پرتو و نام‌گذاری مجدد، جدول S با نام SX نام‌گذاری مجدد می‌شود، خروجی پرس و جو به صورت زیر خواهد بود:

S#	City
S1	C1
S2	C2
S3	C2

SX

در پرانتز داخلی رابطه سمت چپ عملگر ضرب دکارتی یعنی داخل جدول S مجددا مشخصات تولیدکنندگان قرار دارد، که پس از حرکت به سمت خارج و انجام عملگر پرتو، خروجی پرس و جو به صورت زیر خواهد بود:

S#	City
S1	C1
S2	C2
S3	C2

همچنین در پرانتز داخلی پس از انجام عملگر ضرب دکارتی، خروجی پرس و جو به صورت زیر خواهد بود:

S#	City
S1	C1
S2	C2
S3	C2

×

S#	City
S1	C1
S2	C2
S3	C2

SX

=

S#	City	SX.S#	SX.City
S1	C1	S1	C1
S1	C1	S2	C2
S1	C1	S3	C2
S2	C2	S1	C1
S2	C2	S2	C2
S2	C2	S3	C2
S3	C2	S1	C1
S3	C2	S2	C2
S3	C2	S3	C2

در نهایت پس از انجام عملگر ضرب دکارتی و انجام عملگر انتخاب، خروجی پرس و جو به صورت زیر خواهد بود:

S#	City	SX.S#	SX.City
S2	C2	S3	C2
S3	C2	S2	C2

عملگر انتساب (Assignment)

این عملگر توسط نماد ← نمایش داده می‌شود. عملگر انتساب گاها با نماد **:=** یا GIVING نیز نشان داده می‌شود. عملگر ← جهت قرار دادن خروجی یک عبارت جبر رابطه‌ای در یک رابطه مورد استفاده قرار می‌گیرد.

فرم کلی عملگر ← به صورت زیر است:

$R_2 \leftarrow R_1$

در بخش R_1 نام جدول یا جداول به عنوان مکان یا محل پرس و جو مشخص می‌گردد، این مکان جستجو می‌تواند خروجی یک پرس و جوی دیگر یعنی یک عبارت جبر رابطه‌ای دیگر باشد.

پرس و جو نویسی

در ادامه به بررسی یک پرس و جوی کاربردی می‌پردازیم:

جداول S، SP و P را به صورت زیر در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

پرس و جو: درج سطر {('S2', 'P2',40')} به جدول SP.

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$SP \leftarrow SP \cup \{('S2','P2',40)\}$$

پرس و جوی فوق رکورد {('S2', 'P2',40')} را به جدول SP اضافه می‌کند. در این پرس و جو ابتدا عملگر اجتماع اجرا می‌شود و سپس نتیجه حاصل در جدول SP قرار می‌گیرد، در نهایت خروجی پرس و جو به صورت زیر خواهد بود:

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30
S2	P2	40

←

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

∪

S#	P#	QTY
S2	P2	40

دقت کنید که در انجام عمل اجتماع برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

پرس و جو: حذف سطر {('S2', 'P2',40')} از جدول SP.

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$SP \leftarrow SP - \{('S2','P2',40)\}$$

پرس و جوی فوق رکورد {('S2', 'P2',40')} را از جدول SP حذف می‌کند. در این پرس و جو ابتدا عملگر تفاضل اجرا می‌شود و سپس نتیجه حاصل در جدول SP قرار می‌گیرد، در نهایت خروجی پرس و جو به صورت زیر خواهد بود:

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

$\leftarrow$

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30
S2	P2	40

$-$

S#	P#	QTY
S2	P2	40

دقت کنید که در انجام عمل تفاضل برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

پرس و جو: حذف تولیدکنندگان ساکن شهر C2 از جدول S.

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$S \leftarrow S - \sigma_{S.City='C2'}(S)$$

پرس و جوی فوق تولیدکنندگان ساکن شهر C2 را از جدول S حذف می‌کند. در این پرس و جو ابتدا عملگر انتخاب و سپس عملگر تفاضل اجرا می‌شود و سپس نتیجه حاصل در جدول S قرار می‌گیرد، در نهایت خروجی پرس و جو به صورت زیر خواهد بود:

S#	Sname	City
S1	Sn1	C1

$\leftarrow$

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2

$-$

S#	Sname	City
S2	Sn2	C2
S3	Sn3	C2

دقت کنید که در انجام عمل تفاضل برقراری شروط سازگاری الزامی است که در پرس و جوی فوق این الزام برقرار است.

پرس و جو: به تعداد همه قطعات موجود در جدول SP، 0.5 اضافه شود:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\sigma_{QTY \leftarrow QTY+(QTY \times 0.5)}(SP)$$

پرس و جو: نام شهر C2 به C3 در جدول S تغییر کند:

مطابق پرس و جوی مطرح شده در جبر رابطه‌ای داریم:

$$\sigma_{City \leftarrow 'C3'}\left(\sigma_{City='C2'}(S)\right)$$

پرس و جو: نام تولیدکنندگانی که قطعه P2 را تولید کرده‌اند:

$$Temp1 \leftarrow S \bowtie SP$$
$$Temp2 \leftarrow \sigma_{P\#='P2'}(Temp1)$$
$$\Pi_{Sname}(Temp2)$$

فرم دیگر پرس و جوی فوق به صورت زیر است:

$$\Pi_{Sname}(\sigma_{P\#='P2'}(S \bowtie SP))$$

۵ SQL دستورات DDL ، DCL و TCL

مقدمه

زبان پرس‌وجوی SQL سرواژه عبارت Structured Query Language می‌باشد. SQL زبان استاندارد سیستم‌های رابطه‌ای است، در واقع SQL یک پیاده‌سازی از جبر رابطه‌ای و حساب رابطه‌ای تاپلی است. امروزه این زبان توسط تمامی DBMS های تجاری موجود در بازار پشتیبانی می‌شود. SQL زبانی بیانی است که کاربر فقط نتیجه‌ای را که می‌خواهد، مشخص می‌کند. تمام مسائل مربوط به بهینه‌سازی و چگونگی اجرا به نرم‌افزار DBMS واگذار شده‌است. اولین نسخه SQL در سال 1976 توسط مرکز تحقیقات و توسعه شرکت IBM طراحی و پیاده‌سازی شد و Sequel نام گرفت. این نسخه برای سیستم رابطه‌ای آزمایشی بنام System R طراحی شده بود. 10 سال بعد در سال 1986 همکاری موسسه استانداردهای ملی آمریکا (ANSI) و سازمان استانداردهای بین‌المللی (ISO) منجر به ساخت اولین نسخه استاندارد SQL شد و SQL-86 نام گرفت. در ادامه توسعه SQL در سال 1989 نسخه استاندارد توسعه‌یافته‌تری به نام SQL-89 عرضه شد و SQL1 نام گرفت. سپس در سال 1992 نسخه استاندارد توسعه‌یافته‌تر دیگری به نام SQL-92 عرضه شد و SQL2 نام گرفت. در ادامه همین مسیر در سال 1999 نسخه استاندارد توسعه‌یافته‌تر دیگری به نام SQL-1999 عرضه شد و SQL3 نام گرفت. در سال 2003 و 2006 دو بهبود روی نسخه استاندارد SQL3 انجام شد و به نام SQL-2003 و SQL-2006 ارائه شد. در سال 2008 بهبود دیگری روی نسخه استاندارد SQL3 انجام شد و به نام SQL-2008 ارائه شد. در این نسخه ویژگی‌های پایگاه داده شی‌گرا به آن اضافه شد. این بهبود تا به امروز ادامه داشته است.

زبان‌های دیگری نیز در ارتباط با پرس‌وجو و کار با پایگاه داده وجود دارد، از جمله این زبان‌ها می‌توان به QBE، Quel و Datalog اشاره کرد. زبان QBE بر مبنای حساب رابطه‌ای دامنه‌ای، زبان Quel بر مبنای حساب رابطه‌ای تاپلی و زبان Datalog بر مبنای زبان برنامه‌نویسی prolog ایجاد شده‌اند. هر سه زبان مذکور در سیستم‌های پایگاه داده تحقیقاتی کاربرد دارند.

زبان‌های پیاده‌سازی

همانطور که گفتیم یک محصول نرم‌افزاری از دو وجه **عملکـرد** (برنامـه کـاربردی) و **داده** (بانـک اطلاعات) تشکیل می‌شود. در ادامه به معرفی انواع زبان‌های برنامه‌سازی می‌پردازیم.

زبان پیاده‌سازی برنامه کاربردی (وجه عملکرد)

برنامه کاربردی نیز مانند بخش داده، حاصل مراحل تحلیل، طراحی و پیاده‌سازی می‌باشد که شـرح این مراحل مربوط به درس مهندسی‌نرم‌افزار می‌باشد. مرحله پیاده‌سازی برنامه کاربردی توسط یکی از زبان‌های برنامه‌نویسی سطح بالا انجام می‌شود مانند زبان C#. در زبان‌هـای رویـه‌ای، چگـونگی، راه حل و مراحل رسیدن به جواب بیان می‌شود. در این بخش مسائل مربوط به اجرای الگوریتم‌ها، محاسبات و تعامل با کاربر پیاده‌سازی می‌شود.

توجه: به زبان‌های سطح بالا، زبـان میزبـان (Host Language) یـا زبـان روالـی یـا زبـان رویـه‌ای (Procedural) نیز گفته می‌شود.

زبان پیاده‌سازی بانک اطلاعات (وجه داده)

در بانک اطلاعات از زبان‌هـای بیـانی یـا توصـیفی (Declarative) کـه بـه آنهـا زبـان پـرس‌وجـو (Query Language) نیز گفته می‌شود، اسـتفاده مـی‌شـود ماننـد SQL. در زبـان‌هـای بیـانی کـاربر برنامه ساز کافیست بگوید چه چیزی لازم دارد تا سیستم برای او ایجاد (مثل جداول) یـا اسـتخراج (مثل پرس‌وجوها) کند. در واقع چگونگی ایجاد جداول یا اسـتخراج پـرس وجوهـا از دیـد کـاربر برنامه‌ساز وکاربر نهایی مخفی است. در زبان‌های بیانی، چگـونگی، راه حـل و مراحـل رسـیدن بـه جواب بیان نمی‌شود، بلکه خود جواب مورد توصیف قرار مـی‌گیـرد. بنـابراین زبـان‌هـای بیـانی از توانایی بالایی در توصیف پرس‌وجو برخوردارند.

برای مثال در یک سیستم کامپیوتری بخش آموزش یـک دانشـگاه بـرای اسـتخراج «نـام و شـماره دانشجویانی که معدل آنها بالای 18 است» کافی است در ابتدا نام جدول یا جداول به عنوان مکـان یا محل پرس‌وجو، سپس شرط انتخاب سطر به عنوان ملاک انتخاب سطرها و در نهایت ستون‌های نام و شماره دانشجو به عنوان ستون‌های مورد نیاز بیان شود. و به همین دلیل است که کلمه «بیانی» به این نوع زبان‌ها اطلاق می‌شود، زیرا کاربر برنامه‌ساز فقط مشخصات کامل وکلی آنچه را که لازم دارد بیان می‌کند و به جزئیات چگونگی و نحوه استخراج کاری ندارد، این مشخصات کامل و کلی به ترتیب شامل سه بخش نام جدول یا جداول به عنوان مکان یا محل پـرس‌وجـو، شـرط انتخـاب سطر به عنوان ملاک انتخاب سطرها و مدل انتخاب ستون به عنوان ستون‌های مورد نیـاز اسـت. در مثال فوق لازم نیست از تعداد سطرهای جدول، حلقه تکرار و غیره حرفی به میان آید. اصولاً حلقه تکرار با زبان‌های بیانی بیگانه است. انجام هرگونه تعامل با DBMS بر عهـده SQL اسـت. در ایـن حالت در طراحی نرم‌افزار هر زمان نیاز به تبادل اطلاعات با پایگاه داده باشد، در بدنه برنامه‌ای کـه

به یک زبان برنامه‌نویسی سطح بالا نوشته شده‌است از دستورات SQL به صورت توکار (Embedded) یا روال‌های ذخیره شده (Stored Procedures) استفاده می‌شود.

توجه: به زبان‌های بیانی، زبان‌های میهمان (Data Sublanguage) نیز گفته می‌شود.

توجه: یک برنامه کاربردی نوشته شده به یک زبان روالی سطح بالا پس از ارتباط با یک سرویس‌دهنده زبان بیانی مانند SQL Server از طریق مکانیزم‌های ویژه (Connection String) اقدام به استفاده از بانک اطلاعات می‌نماید. همچنین برای اتصال به پایگاه داده، در رشته اتصال (Connection String) نحوه احراز هویت (Authentication) کاربر مشخص می‌شود.

توجه: زبان SQL در ابتدا فقط یک زبان میهمان (Data Sublanguage) بود، اما بعدها برخی امکانات محاسباتی زبان‌های برنامه‌نویسی نظیر Call، Return، Set، Case، If، Loop، While و Repeat و ویژگی‌هایی نظیر تعریف متغیرها به آن اضافه شد. بنابراین SQL هم می‌تواند به طور مستقل از زبان‌های روالی استفاده شود. و هم به طور وابسته به زبان‌های روالی مورد استفاده قرار گیرد. اما همچنان امکانات برنامه‌نویسی کامل در SQL محدود است و به همین دلیل و همچنین اصول طراحی مطلوب نرم‌افزار، مدل SQL وابسته به زبان‌های روالی رایج‌تر است.

توجه: هر مدل داده‌ای، زبان بیانی خاص خود را دارد. به طور مثال SQL برای جبر رابطه‌ای و مدل رابطه‌ای ایجاد گردیده است. البته این زبان همه بخش‌های جبر رابطه‌ای و مدل رابطه‌ای را پوشش نمی‌دهد، به عنوان مثال عملگر تقسیم به عنوان یکی از پرکاربردترین عملگرهای فرعی در جبر رابطه‌ای در زبان SQL به صورت مستقیم پیاده‌سازی نشده است. به عنوان مثالی دیگر در مدل رابطه‌ای مطابق قانون جامعیت درون رابطه‌ای امکان تعریف ستون مرکب (مثل تاریخ) و چند مقداری (مثل شماره تلفن) وجود ندارد، اما در SQL ستون مرکب می‌توان تعریف کرد.

انواع دستورات در زبان SQL

در مرحله پیاده‌سازی بانک اطلاعات چهار مقوله «**ایجاد جداول، دیدها و شاخص‌ها**»، «**ایجاد پرس‌وجوها**»، «**ایجاد سطوح امنیتی کاربران**» و «**حفظ جامعیت داخلی و خارجی پایگاه داده**» انجام می‌گردد. بنابراین دستورات SQL به چهار طبقه زیر تقسیم می‌گردد:

۱ـ دستورات تعریف داده‌ها (DDL: Data Definition Language)

این دستورات کارهای مربوط به ساختارسازی و ظرف‌سازی پایگاه داده از قبیل ایجاد، حذف و تغییرات جداول، ستون‌ها، دیدها و شاخص‌ها را انجام می‌دهد. اجرای این دستورات اثری بر روی محتوای پایگاه داده ندارد.

توجه: دستورات Create، Drop، Alter در این طبقه قرار دارند.

مثال: دستور Create Table برای ایجاد جداول و دستور Drop Table برای حذف جداول است.

مثال: دستور Create View برای ایجاد دیدها و دستور Drop View برای حذف دیدها است.

مثال: دستور Create Index برای ایجاد شاخص‌ها و دستور Drop Index برای حذف شاخص‌ها است.

توجه: دستورات Create assertion ، Check ، Primery key ، Foreign key ، Unique ، NOT NULL ، on delete option و on delete option برای تعریف و حفظ محدودیت‌های جامعیت داخلی و خارجی ساختار پایگاه داده موسوم به دستورات جامعیتی (Integrity) در کنار دستورات DDL مورد استفاده قرار می‌گیرند.

۲ـ دستورات تغییر داده‌ها (DML: Data Manipulation Language)

این دستورات کارهای مربوط به تغییرات محتوای جداول پایگاه داده از قبیل ذخیره داده‌ها در جداول، بازیابی داده‌ها از جداول، بروزرسانی داده‌ها در جداول و حذف داده‌ها از جداول را انجام می‌دهد. اجرای این دستورات اثری بر روی ساختار پایگاه داده ندارد.

توجه: دستورات Select، Insert، Delete وUpdate به عنوان نمونه در این طبقه قرار دارند.

۳ـ دستورات کنترل داده‌ها (DCL: Data Control Language)

این دستورات کارهای مربوط به تعریف محدودیت‌های امنیتی جداول پایگاه داده از قبیل ایجاد سطوح دسترسی برای کاربران را انجام می‌دهد.

توجه: دستور Grant برای ایجاد و اعطای حق و مجوز دسترسی و دستور Revoke برای حذف و بازپس‌گیری حق و مجوز دسترسی در این طبقه قرار دارند.

۴ـ دستورات کنترل تراکنش‌ها (TCL: Transaction Control Language)

این دستورات کارهای مربوط به کنترل اجرای تراکنش‌های پایگاه داده از قبیل اعلام اجرای موفق یا ناموفق تراکنش‌ها را انجام می‌دهد. در SQL برای نمایش ابتدای یک تراکنش از دستور Begin Transaction و برای نمایش خاتمه یک تراکنش از دستور End Transaction استفاده می‌شود. تراکنش با اجرای Begin Transaction شروع می‌گردد و در صورت اجرای موفق commit و در صورت عدم موفقیت یعنی عدم اجرای همه بخش‌های مختلف تراکنش با اجرای دستور Rollback خاتمه می‌یابد. با اجرای این دستور کلیه تغییراتی که تراکنش روی پایگاه داده اعمال نموده است، ابطال می‌شود و وضعیت پایگاه داده به آخرین وضعیت قبل از اجرای تراکنش برگردانده می‌شود.

توجه: دستور commit برای اعلام اجرای موفق تراکنش و دستور Rollback برای اعلام ناموفق تراکنش در این طبقه قرار دارند.

توجه: اجتماع مجموعه دستورات DDL، DML ، DCL و TCL را DSL می‌گویند.

توجه: DSL سرواژه عبارت Data Sub Language است.

توجـــه: بـــه اجتمـــاع مجموعـــه دســـتورات DDL، DML ، DCL و TCL، زبـــان پـــرس‌وجـــو (QL: Query Language) نیز گفته می‌شود.
همانطور که گفتیم در مرحله پیاده‌سـازی بانـک اطلاعـات چهـار مقولـه ایجـاد جـداول، دیـدها و شاخص‌ها، ایجاد پرس‌وجوها، ایجاد سـطوح امنیتـی کـاربران نهـایی و حفـظ جامعیـت داخلـی و خارجی پایگاه داده انجام می‌گردد. بنابراین دستورات SQL به چهـار طبقـه DDL، DML، DCL و TCL تقسیم می‌شوند که در ادامه به بررسی دقیق‌تر دستورات هر طبقه می‌پردازیم:

دستورات تعریف داده‌ها (DDL: Data Definition Language)

در این بخش به بررسی دقیق‌تر دستورات DDL می‌پردازیم.

دستور Create Database

این دستور جهت ایجاد یک پایگاه داده مورد استفاده قرار می‌گیرد.
ساختار کلی این دستور به صورت زیر است:

Create Database name

مثال:

Create Database prodsuppdb

توجه: اجرای دستور فوق منجر به ایجاد پایگاه داده تولیدکنندگان و قطعات بـا نـام prodsuppdb می‌شود.
توجه: درحال حاضر این پایگاه داده خالی و فاقد هرگونه ساختار و جدولی است.
توجه: بعد از ایجاد پایگاه داده، می‌توان جداول و ساختارهای بعدی را به پایگاه داده اضافه کرد.

دستور Create Table

این دستور جهت ایجاد یک جدول در یک پایگاه داده مورد استفاده قرار می‌گیرد.
ساختار کلی این دستور به صورت زیر است:

Create Table name
(attr domain [NOT NULL],
.
.
.
attr …,
Primary key (…) ,
[**Unique (**…**)**],
[**Foreign key** (…) **References** …(…)]
[**on delete option**]
[**on update option**],
[**Check** (…)]
)

توجه: هنگام ایجاد یک جدول، نوع داده ستون‌ها و تمامی محدودیت‌های جامعیتی اجباری مربوط به ساختار یک جدول مثل تعریف کلید اصلی باید مشخص شود.

مثال: جداول زیر را در سیستم تولیدکنندگان و قطعات در نظر بگیرید.

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

(تولیدکنندگان)

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

(تولید)

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

(قطعات)

مثال: پیاده‌سازی جدول تولیدکنندگان (S)

```
Create Table S
    (
    S# char (5),
    Sname char (20) NOT NULL,
    City char (15),
    Primary key (S#)
    )
```

توجه: اجرای دستور فوق منجر به ایجاد جدول S در پایگاه داده prodsuppdb می‌شود.

توجه: #S در جدول S، کلید کاندید است، پس عناصرش یکتاست.

مثال: پیاده‌سازی جدول قطعات (P)

```
Create Table P
    (
    P# char (5),
    Pname char (20) NOT NULL,
    Color char (10),
    Primary key (P#)
    )
```

توجه: اجرای دستور فوق منجر به ایجاد جدول P در پایگاه داده prodsuppdb می‌شود.

توجه: #P در جدول P، کلید کاندید است، پس عناصرش یکتاست.

مثال: پیاده‌سازی جدول تولید (SP)

```
Create Table SP
            (
      S# char (5),
      P# char (5),
      QTY numeric (10),
      Primary key (S#, P#),
      Foreign key (S#) References S(S#)
        on delete cascade
        on update cascade,
      Foreign key (P#) References P(P#)
        on delete cascade
        on update cascade,
      Check (QTY>1 AND QTY<1000)
      )
```

توجه: اجرای دستور فوق منجر به ایجاد جدول SP در پایگاه داده prodsuppdb می‌شود.

توجه: ترکیب ستون‌های (S#,P#) باهم در جدول SP، کلید کاندید است، پس عناصرش یکتاست.

توجه: S# در جدول SP، به عنوان کلید خارجی، خاصیت کلیدی ندارد، پس ممکن است، مقادیر تکراری داشته باشد.

توجه: P# در جدول SP، به عنوان کلید خارجی، خاصیت کلیدی ندارد، پس ممکن است، مقادیر تکراری داشته باشد.

توجه: به طور کلی **جامعیت** در سیستم‌های بانکی به دو طبقه‌ی جامعیت **داخلی** و **خارجی** تقسیم می‌گردد. قوانین داخلی بانک شامل قانون جامعیت درون رابطه‌ای، قانون جامعیت موجودیت، قانون جامعیت ارجاعی و قانون جامعیت دامنه‌ای است و قوانین خارجی بانک هم شامل هر قانونی است که طراحان و برنامه‌نویسان بانک آنرا وضع می‌کنند، مانند تعریف بازه 1 تا 1000 برای ستون QTY در جدول SP.

توجه: در سیستم‌های بانکی کنترل جامعیت داخلی و خارجی به صورت خودکار توسط مکانیزم‌های موجود در DBMS انجام می‌گردد.

Primary key

این دستور برای تعریف کلید اصلی مورد استفاده قرار می‌گیرد. مطابق قانون جامعیت درون رابطه‌ای، هر رابطه (جدول) باید حتماً حداقل دارای یک کلید کاندید باشد، همچنین مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد. بنابراین

اگر در تعریف یک جدول ستون (یا ستون‌هایی) به عنوان کلید اصلی تعریف شود، مطابق قانون جامعیت درون رابطه‌ای علاوه بر اینکه آن ستون نمی‌تواند مقادیر تکراری داشته باشد، مطابق قانون جامعیت موجودیت، آن ستون نمی‌تواند مقدار NULL هم داشته باشد و به طور پیش فرض NOT NULL در نظر گرفته می‌شود.

NOT NULL

هر ستونی می‌تواند حاوی مقدار NULL باشد، مگر اینکه در تعریف آن NOT NULL به کار رفته باشد. کلید اصلی به طور پیش فرض NOT NULL تعریف می‌شود.

Unique

این دستور برای تعریف کلید فرعی مورد استفاده قرار می‌گیرد. با قرار دادن نام یک ستون در جلوی دستور Unique می‌توان کلید فرعی تعریف کرد، که در این حالت آن ستون نمی‌تواند مقدار تکراری داشته باشد. کلید فرعی نوعی کلید کاندید است که مقدار یکتا دارد.

Foreign key

این دستور برای تعریف کلید خارجی مورد استفاده قرار می‌گیرد. کلید خارجی برای ارتباط میان جداول مورد استفاده قرار می گیرد، مطابق قانون جامعیت ارجاعی، هیچگاه نباید کلید خارجی، دچار ارجاع NULL گردد. برای این منظور باید تعریف ساختاری و محتوایی کلید خارجی برقرار باشد. کلید خارجی در دو دیگاه ساختاری و محتوایی مورد بررسی قرار می‌گیرد:

دیدگاه ساختاری کلید خارجی

تعریف: اگر صفت(هایی) در یک جدول به عنوان کلید خارجی تعریف شود،
اول اینکه: این صفت(ها) در جدول خودش شرط خاصی ندارد یعنی الزاما خاصیت کلیدی ندارد.
و دوم اینکه: این صفت(ها) در همان جدول یا جدول دیگری، **باید کلید کاندید** (اصلی یا فرعی) باشد.

دیدگاه محتوایی کلید خارجی

به ازای هر مقدار موجود در یک کلید خارجی، باید دقیقاً یک مقدار متناظر در کلید کاندید متناظر آن وجود داشته باشد، در غیر این صورت می‌گوییم، کلید خارجی دارای ارجاع NULL است. به بیان دیگر، مقادیر کلید خارجی همواره باید زیرمجموعه مقادیر کلید کاندید باشد.
توجه: در جدول SP ستون #S به عنوان کلید خارجی تعریف شده است، و توسط دستور References S(S#) به جدول اصلی S و ستون #S ارجاع داده شده است، در این ارجاع نوع و درجه ستون #S در جدول SP به عنوان جدول فرعی با نوع و درجه ستون #S در جدول S به عنوان جدول اصلی باید یکسان و سازگار باشد.

توجه: در جدول SP ستون #P بـه عنـوان کلیـد خـارجی تعریـف شـده اسـت، و توسـط دسـتور References P(P#) به جدول اصلی P و ستون #P ارجاع داده شده اسـت، در ایـن ارجـاع نـوع و درجه ستون #P در جدول SP به عنوان جدول فرعی با نـوع و درجـه سـتون #P در جـدول P بـه عنوان جدول اصلی باید یکسان و سازگار باشد.

مثال:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30
(S4)	P3	40

× ارجاع NULL

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue
P3	Pn3	Green

جدول P

درج رکورد (S4,P3,40) در جدول SP، غیرمجاز و غیرممکن است، زیـرا سـبب ارجـاع NULL می‌گردد.

توجه: هر مقداری که در کلید خارجی وجود دارد، باید دارای مقدار متناظر در کلید کاندید مقصـد باشد ولی عکس آن صادق نیست.

مثال: مانند S3 که در جدول S قرار دارد ولی لزومی ندارد در جدول SP هم باشد. امـا بایـد همـه مقادیر کلید خارجی در کلید کاندید باشد، از بالا به پایین می‌بارد، از پایین به بالا که نمی‌بارد!

توجه: اگرچه کلید خارجی هیچگاه نباید، ارجاع NULL داشته باشد، اما مـی‌توانـد مقـدار NULL داشته باشد، البته به شرطی که کلید خارجی در شرایط قوانین بالا دستی قرار نگیرد.

مثال:

کلید کاندید — کلید خارجی

(a)	b	c
1	4	7
2	6	5
3	4	5

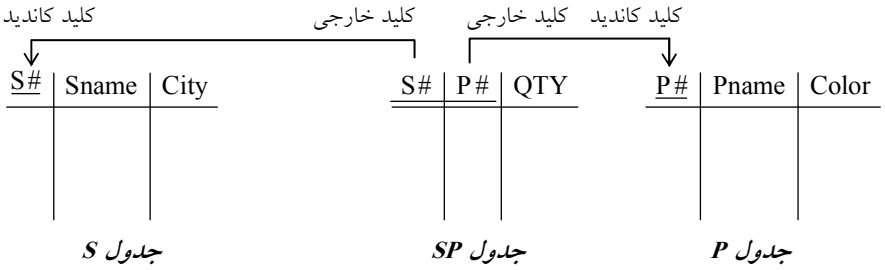
R_1

(a)	d
2	1
3	2
2	3
1	4

R_2

توجه: می‌توان مقادیر ستون a در جدول R_2 را NULL قرار داد. کلید کاندید (کلید اصلی) جدول R_2 ستون d است. ستون a در جدول R_2 کلید خارجی است.

مثال:



توجه: می‌توان مقادیر کلید خارجی را برابر NULL قرار داد، البته به شرطی که قوانین بالا دستی بر کلید خارجی حاکم نباشد. در مثال فوق S# و P# در جدول SP به عنوان کلید خارجی نمی‌توانند NULL باشند، زیرا (S#,P#) با هم قبل از اینکه به تنهایی کلید خارجی باشند، در جدول SP کلید اصلی جدول SP بوده‌اند. بنابراین قانون بالا دستی **جامعیت موجودیت**، بدین معنی که هیچگاه نباید تمام یا بخشی از کلید اصلی NULL باشد، جلوی NULL شدن S# و P# به عنوان کلید خارجی را در جدول SP می‌گیرد.

توجه: برای رفتار ستون کلید خارجی در یک جدول فرعی، در قبال تغییرات کلید کاندید از یک جدول اصلی گزینه‌های زیر وجود دارد:

```
Create Table name
        (attr domain [NOT NULL],
         .
         .
         .
        attr …,
        Primary key (…),
        [Unique (…)],
        [Foreign key (…) References …(…)]
             [on delete option]
             [on update option],
        [Check (…)]
       )
```

فیلد option می‌تواند یکی از موارد زیر باشد:

(restrict) no action

گزینه پیش فرض است و هیچ عملی انجام نمی‌شود. اگر بر اثر عملیات **حذف** یـا **بروزرسـانی** در جدول اصلی، قانون جامعیت ارجـاعی در جـدول فرعـی نقـض گـردد، آنگـاه ایـن اعمـال انجـام نمی‌گردد. در واقع زمانی عملیات حذف یا بروزرسانی در جدول اصلی انجام می‌گـردد کـه قـانون جامعیت ارجاعی در جـدول فرعـی نقـض نگـردد. بـه عبـارت دیگـر زمـانی عملیـات حـذف یـا بروزرسانی در سطری از جدول اصلی انجام می‌گردد کـه سـطری از جـدول فرعـی بـه آن ارجـاع نکرده باشد. برای مثال مقدار S3 در جدول S می‌تواند به S33 بروزرسانی شـود و یـا از جـدول S حذف شود، زیرا هیچ ارجاعی روی مقدار S3 از جدول SP به جدول S وجود ندارد. اما مقـدار S1 در جدول S مطابق رابطه no action نمی‌تواند به S11 بروزرسانی شـود و یـا از جـدول S حـذف شود، زیرا ارجاعی روی مقدار S1 از جدول SP به جدول S وجود دارد.

Cascade

اگر سطرهای جدول اصلی **حذف** یا **بروزرسانی** شود، آنگاه سطرهای جدول فرعی که توسط کلید خارجی به آن ارجاع کرده است نیز حذف یا بروزرسانی مـی‌شـود. بـرای مثـال اگـر مقـدار S1 در جدول S به S11 بروزرسانی شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطـابق رابطـه Cascade مقدار S1 در جدول SP به مقدار S11 نیز بروزرسانی می‌شود. برای مثالی دیگر اگر سـطر S1 از جدول S حذف شود و این مقـدار در جـدول SP نیـز موجـود باشـد، آنگـاه مطـابق رابطـه Cascade سطر S1 از جدول SP نیز حذف می‌شود.

(NULLIFY) Set NULL

اگر سطرهای جدول اصلی **حذف** یا **بروزرسانی** شود، آنگاه ستون‌های جـدول فرعـی کـه توسـط کلید خارجی به آن ارجاع کرده است با مقدار NULL پُر مـی‌شـود. بـرای مثـال اگـر مقـدار S1 در جدول S به S11 بروزرسانی شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطـابق رابطـه Set NULL مقدار S1 در جدول SP به مقدار NULL تغییر می‌کند. برای مثالی دیگر اگر مقـدار S1 از جـدول S حـذف شـود و ایـن مقـدار در جـدول SP نیـز موجـود باشـد، آنگـاه مطـابق رابطـه Set NULL مقدار S1 در جدول SP به مقدار NULL تغییر می‌کند.

کارکرد قطعه کد زیر از کد تعریف جدول SP به صورت زیر است:

```
Foreing key (S#) References S(S#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییـری اعـم از حـذف یـا بروزرسـانی در سطرهای ستون S# در جدول S به سطرهای ستون S# در جـدول SP نیـز اعمـال گـردد. بنـابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

یا به طور مشابه، کارکرد قطعه کد زیر از کد تعریف جدول SP به صورت زیر است:

```
Foreing key (P#) References P(P#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری اعم از حذف یا بروزرسانی در سطرهای ستون #P در جدول P به سطرهای ستون #P در جدول SP نیز اعمال گردد. بنابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

Check

این دستور برای بیان قوانین **جامعیت خارجی** مورد استفاده قرار می‌گیرد. این کنترل‌ها هنگام وارد کردن اطلاعات، اعمال می‌گردد.

کارکرد قطعه کد زیر از کد تعریف جدول SP به صورت زیر است:

```
Check (QTY>=1 AND QTY<=1000)
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه مقداردهی در ستون QTY از جدول SP در بازه 1 تا 1000 باشد، بنابراین **جامعیت خارجی** نقض نمی‌گردد.

توجه: می‌توان بخش Check را توسط دستور زیر و به طور جداگانه و خارج از تعریف جدول، تعریف کرد:

```
Create Assertion name
        Check (…)
```

توجه: دقت کنید که ASSERTIONها دقیقا همانطور که خوانده می‌شوند عمل می‌کنند، و دقیقا به همان شکل که خوانده می‌شوند اجازه ذخیره‌سازی داده‌ها را به جداول می‌دهند.

مثال:

```
Create Assertion SP-QTY
        Check (NOT EXISTS (SELECT *
                           FROM SP
                           WHERE QTY<1 OR QTY>1000)
```

دستور Assertion زمانی اعمال می‌شود که دستور Check برابر TRUE باشد. در مثال فوق دستور Check زمانی برابر TRUE خواهد بود که دستور NOT EXISTS برابر TRUE باشد و دستور NOT EXISTS زمانی برابر TRUE خواهد بود که جلوی دستور NOT EXISTS تهی باشد و جلوی دستور NOT EXISTS زمانی تهی خواهد بود که مقادیر QTY برابر 1 تا 1000 باشد. بنابراین این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه مقداردهی در ستون QTY از

جدول SP در بازه 1 تا 1000 باشد.

توجه: همانطور که واضح است DBMS در حفظ جامعیت داخلی و خارجی به دقت نظارت دارد.

انواع دامنه‌های استاندارد در زبان SQL

انواع دامنه‌هایی که در زبان SQL پشتیبانی می‌شود، به صورت زیر است:

Integer

مقادیر اعداد صحیح بزرگ را ذخیره می‌کند که اندازه آن بستگی به ساختار سخت‌افزار دارد.

Smallint

مقادیر اعداد صحیح کوچک را ذخیره می‌کند که اندازه آن بستگی به ساختار سخت‌افزار دارد.

Double ، Real ، Float

مقادیر اعداد اعشاری را ذخیره می‌کند که اندازه آن بستگی به ساختار سخت‌افزار دارد.

Numeric(p,q)

مقادیر اعداد اعشاری را ذخیره می‌کند که اندازه آن توسط کاربر مشخص می‌شود. عددی ممیز ثابت با دقت معین که شامل p رقم و q رقم از p رقم که در سمت راست ممیز قرار دارد. برای مثال، در دامنه Numeric(5,2) عددی مثل 123.45 می‌گنجد اما اعدادی مثل 123.456 یا 1234.56 نمی‌گنجد.

Char(n)

مقادیر رشته کاراکتر با طول ثابت و کوچک تا حداکثر 254 کاراکتر را ذخیره می‌کند. طول ثابت یعنی رشته کوچکتر از مقدار n با فضای خالی پُر می‌شود.

Varchar(n)

مقادیر رشته کاراکتر با طول متغیر و بزرگ تا حداکثر 32767 کاراکتر را ذخیره می‌کند. طول متغیر یعنی رشته کوچکتر از مقدار n با فضای خالی پُر نمی‌شود. به عبارت دیگر فضای حافظه به اندازه طول رشته پُر می‌شود. ستون رشته کاراکتر توسط تک کوتیشن مقداردهی می‌شود و بین حروف کوچک و بزرگ تفاوت وجود دارد. هنگام مقایسه رشته‌ها، حروف بر اساس کد اسکی مقایسه می‌شوند. پس رشته aa از ab کوچکتر است، زیرا حرف a با کد اسکی (97) از حرف b با کد اسکی (98) کوچکتر است. جهت یادآوری کد اسکی حروف بزرگ A تا Z به صورت 65 تا 90 و کد اسکی حروف کوچک a تا z به صورت 97 تا 122 است.

Date

مقادیر تاریخ به فرمت سال، ماه و روز (YYYY-MM-DD) را ذخیره می‌کند. در فیلدهای مذکور فقط مقادیر مجازشان قابل ذخیره‌سازی است. ستون تاریخ'2021-05-07' توسط تک کوتیشن مقداردهی می‌شود. تاریخ را می‌توان با عملگرهای رابطه‌ای مقایسه کرد و به تبع تاریخ قدیم‌تر، کوچکتر از تاریخ جدیدتر است.

Time

مقادیر زمان به فرمت ساعت، دقیقه و ثانیه (HH:MM:SS) را ذخیره می‌کند. در فیلدهای مذکور فقط مقادیر مجازشان قابل ذخیره‌سازی است. ستون زمان'09-12-47' توسط تک کوتیشن مقداردهی می‌شود. زمان را می‌توان با عملگرهای رابطه‌ای مقایسه کرد و به تبع زمان قدیم‌تر، کوچکتر از زمان جدیدتر است.

Timestamp

مقادیر تاریخ و زمان به فرمت سال، ماه، روز، ساعت، دقیقه، ثانیه و 6 مکان کسر ثانیه بیشتر به صورت (YYYY-MM-DD-HH:MM:SS:NNNNNN) را ذخیره می‌کند.

Logical

مقادیر FALSE و TRUE را ذخیره می‌کند. در SQL بخاطر وجود مقدار NULL و منطق سه ارزشی، مقدار سوم نوع Boolean برابر UNKNOWN است.

Blob

مقادیر داده‌های حجیم مثل فایل‌های تصویری، صوتی و ویدیویی را ذخیره می‌کند. این نوع داده‌ها ساختار ناشناخته‌ای برای DBMS رابطه‌ای دارند، بنابراین پایگاه داده رابطه‌ای فقط می‌تواند آنها را ذخیره و بازیابی نماید و امکان پردازش دیگری روی آنها در پایگاه داده رابطه‌ای وجود ندارد.

توجه: مقایسه انواع داده در محیط SQL فقط در دامنه‌های سازگار امکان‌پذیر است. برای مثال انواع integer و smallint باهم سازگارند و یا انواع char و varchar نیز باهم سازگارند هرچند طول آنها متفاوت است. به عنوان مثال چون هر smallint خود یک integer است، بنابراین اگر x یک smallint و y یک integer باشد در آن صورت مقایسه x < y یک مقایسه ممکن و صحیح است. در چنین مواردی مقایسه‌هایی از این قبیل، با همسان‌سازی (Casting) نوع‌ها صورت می‌پذیرد. برای نمونه در این مثال x تبدیل به integer می‌شود و سپس با y مقایسه می‌شود. تبدیلاتی به این ترتیب را همنوع‌سازی (Type Coercion) می‌نامند. همنوع‌سازی معمولا در زبان‌های برنامه‌نویسی متداول و سیستم‌های پایگاه داده مورد استفاده قرار می‌گیرد.

دستور Create Domain

این دستور جهت ایجاد یک دامنه جدید علاوه بر دامنه‌های استاندارد مورد استفاده قرار می‌گیرد. ساختار کلی این دستور به صورت زیر است:

```
Create Domain domain_name AS DataType
        Default …
        Constraint constraint_name
        Check (Value…)
```

مثال:

```
Create Domain mynumbers AS Integer
        Default 1
        Constraint myn
        Check (Value >= 1 and Value <=1000 )
```

پس از تعریف دامنه جدید می‌توان از mynumbers برای تعریف دامنه یک ستون از یک جدول استفاده کرد، دامنه جدید mynumbers فقط اعداد بازه 1 تا 1000 را به عنوان مقدار می‌پذیرد. Constraint myn نام محدودیت جامعیتی است که بعد آن توسط Check تعریف شده است.

مثال:

```
Create Table SP
                (
        S# char (5),
        P# char (5),
        QTY mynumbers,
        Primary key (S#, P#),
          …
        )
```

توجه: در جدول SP برای ستون QTY دامنه جدید mynumbers مورد استفاده قرار گرفته است.

دستور Create Type

این دستور همانند دستور Create Domain است با این تفاوت که نمی‌توان مقدار پیش فرض و محدودیت‌های جامعیتی برای آن تعریف کرد.

دستور Drop Table

این دستور جهت حذف فیزیکی یک جدول از یک پایگاه داده به طور کامل و با تمام اطلاعات مورد استفاده قرار می‌گیرد. پس از حذف فیزیکی یک جدول امکان دسترسی به آن دیگر وجود

نخواهد داشت.
ساختار کلی این دستور به صورت زیر است:

Drop Table name

مثال: حذف جدول تولید (SP)

Drop Table SP

توجه: اجرای دستور فوق منجر به حذف فیزیکی جدول SP از پایگاه داده prodsuppdb می‌شود.
مثال: حذف جدول تولیدکنندگان (S)

Drop Table S

توجه: اجرای دستور فوق منجر به حذف فیزیکی جدول S از پایگاه داده prodsuppdb نمی‌شود و با خطای عدم اجرای دستور مواجه می‌شود، زیرا اگر جدولی از طریق تعریف کلید خارجی، محل رجوع جدول دیگری باشد امکان حذف فیزیکی آن وجود ندارد، چون حذف آن منجر به نقض قانون جامعیت ارجاعی می‌شود. مطابق تعریف کلید خارجی، ستون #S به عنوان کلید کاندید از جدول اصلی S ، محل رجوع سطرهای ستون #S به عنوان کلید خارجی از جدول فرعی SP است. بنابراین امکان حذف فیزیکی جدول S وجود ندارد. همین مثال برای جدول P نیز صادق است.
توجه: از لحظه ایجاد یک جدول توسط دستور Create Table تا لحظه حذف فیزیکی یک جدول توسط دستور Drop Table ، تمامی اطلاعات سیستمی مربوط به یک جدول در کاتالوگ سیستم قرار دارد.

دستور Alter Table

این دستور جهت درج ستون جدید به یک جدول، تغییر نوع داده‌ای یک ستون و حذف ستون از جدول در شرایط خاص مورد استفاده قرار می‌گیرد.

درج ستون

می‌توان ستون یا ستون‌هایی به یک جدول پایه اضافه کرد.

مثال:

Alter Table S **add** Status integer

توجه: دستور فوق ستون Status را به جدول S اضافه می‌کند. اگر قبل از اجرای دستور فوق، سطرهایی در جدول S وجود داشته باشد، پس از اجرای دستور مذکور، مقدار تمامی سطرها در ستون Status برابر NULL خواهد شد.

تغییر ستون

می‌توان ستون یا ستون‌هایی از یک جدول پایه را تغییر نوع داده‌ای داد.

مثال:

Alter Table S **Modify** S# char(10)

توجه: دستور فوق دامنه ستون #S از جدول S را از char(5) به char(10) تغییر و افزایش می‌دهد. اگر قبل از اجرای دستور فوق، سطرهایی در جدول S وجود داشته باشد، پس از اجرای دستور مذکور، دامنه تمامی سطرها در ستون #S به طور خودکار به دامنه جدید تبدیل می‌شود.

توجه: اگر قبل از اجرای دستور فوق، سطرهایی در جدول S وجود داشته باشد، تغییر نوع به درجه بالاتر، سازگار و از یک خانواده فقط امکان‌پذیر است. به عبارت دیگر دامنه جدید حتما باید در برگیرنده دامنه قدیم باشد. در غیر این صورت اجرای دستور با خطا مواجه می‌شود. زیرا دامنه جدید باید مقادیر دامنه قدیم را پوشش دهد. تغییر کلی دامنه جدید به دامنه ناسازگار با دامنه قدیم امکان‌پذیر نیست، مگر اینکه ستون مربوطه فاقد داده باشد.

حذف ستون

می‌توان ستون یا ستون‌هایی از یک جدول پایه را حذف کرد.

مثال:

```
Alter Table S Drop City
```

توجه: دستور فوق ستون City را از جدول S حذف می‌کند. حذف ستون نباید منجر به حذف کلید اصلی، بخشی از کلید اصلی و یا نقض جامعیت ارجاعی شود.

دستور Create Trigger

عملیاتی که داخل این دستور تعریف می‌شود در شرایط خاص و به طور خودکار در صورت بروز برخی رخدادها در پایگاه داده توسط DBMS اجرا می‌شود. برای مثال هر موقع رکوردی از جدول مورد نظر آپدیت شد، تاریخ و ساعت لحظه آپدیت به طور خودکار از سیستم توسط دستور Getdate اخذ شود و در ستون مورد نظر هر رکورد درج شود. برای مثال اگر ستون ModifyDateTime به جدول S اضافه شود، آنگاه توسط trigger می‌تواند مقداردهی شود. تریگرها بعد از **درج**، **حذف** و **آپدیت** یک رکورد می‌تواند عملیاتی را انجام دهد.

ساختار کلی این دستور به صورت زیر است:

```
Create Trigger triger_name
   on table_name
   After  [Insert] , [Update] , [Delete]
   As
   sql_statements
```

دستور Create View

این دستور جهت ایجاد یک دید (جدول مجازی) در یک پایگاه داده مورد استفاده قرار می‌گیرد.

ساختار کلی این دستور به صورت زیر است:

Create View name
AS Select …

دید (View) یا جدول مجازی، خودش به خودی خود وجود خارجی و فیزیکی ندارد. اما می‌توان به آن جهت بازیابی رکوردهای اطلاعاتی از جداول پایه و ذخیره‌سازی رکوردهای اطلاعاتی روی جداول پایه دسترسی داشت. هدف اصلی از مفهوم دید، نمایشی محدود، خلاصه و فیلترشده از اطلاعات جداول پایه است. دید به جداول پایه متصل می‌شود و همانند یک فیلتر روی جداول پایه عمل می‌کند. هنگامی که توسط دید قصد بازیابی اطلاعات را داریم، از آنجاکه دید خودش به خودی خود وجود خارجی و فیزیکی ندارد، بنابراین ابتدا دید به جداول پایه متصل می‌شود و همانند یک فیلتر روی جداول پایه عمل می‌کند و در نهایت فقط بخشی از اطلاعات جداول پایه (سطر و ستون مورد نظر) را استخراج و به نمایش می‌گذارد. همچنین هنگامی که توسط دید قصد ذخیره‌سازی اطلاعات را داریم، از آنجاکه دید خودش به خودی خود وجود خارجی و فیزیکی ندارد، بنابراین ابتدا دید به جداول پایه متصل می‌شود و در نهایت رکوردهای ورودی روی خود جداول پایه ذخیره‌سازی می‌شوند. به بیان دیگر جداول مجازی، استقلال وجودی ندارند و به جداول پایه متکی و متصل هستند، به گونه‌ای که هر عملیاتی که روی جدول مجازی انجام می‌شود، در واقع در نهایت روی جداول پایه مرتبط و متصل به آن اعمال می‌گردد.

جداول مجازی (دیدها) تصویری از جداول حقیقی (پایه) هستند، یعنی توسط سیستم به جداول پایه متصل می‌شوند و وجود خارجی ندارند. دسترسی به آنها از دید کاربر مستقیم ولی از دید سیستم غیرمستقیم است، یعنی سیستم، هرگونه استخراج اطلاعات را از روی جداول پایه انجام می‌دهد. محتوای دید در لحظه اجرای دید، تولید می‌شود. این محتوای بازیابی شده از سوی دید، در جایی ذخیره نمی‌گردد، بلکه در هر لحظه بر اساس ساختار دید از روی جداول پایه، استخراج می‌گردد و به نمایش گذاشته می‌شود.

به طور کلی دو عمل **ذخیره‌سازی** (درج، حذف و بروزرسانی) و **بازیابی** بر روی دیدها انجام می‌شود. که عمل ذخیره‌سازی با قید و شرط انجام می‌شود یعنی نه همیشه، اما عمل بازیابی بی‌قید و شرط انجام می‌شود یعنی همیشه امکان‌پذیر است. بنابراین مشکل اصلی در دیدها، عمل ذخیره‌سازی (درج، حذف و بروزرسانی) است که قید و شرط دارد و همیشه قابل انجام نیست. هر عمل ذخیره‌سازی (درج، حذف و بروزرسانی) که روی جداول مجازی انجام می‌گیرد در واقع روی جداول پایه متصل به آن اعمال می‌شود، بنابراین فقط و فقط در صورتی می‌توان عمل ذخیره‌سازی را روی جداول مجازی انجام داد که عمل مورد نظر (درج، حذف و بروزرسانی) قوانین جامعیت داخلی و خارجی بانک اطلاعات را نقض نکند. به طور کلی، پذیرا بودن ذخیره‌سازی (درج، حذف و بروزرسانی) بر روی دیدها به سه دسته زیر تقسیم می‌گردد:

پذیرا از نظر تئوری و عملی

در این حالت دید از نظر **تئوری** پذیرای ذخیره‌سازی است، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ می‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ می‌گردد. و از نظر **عملی** هم پذیرای ذخیره‌سازی است به دلیل رعایت قوانین جامعیت بانک در مدل رابطه‌ای در حین اجرا.

موارد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر تئوری تضمین می‌کند:

– در تعریف دید فقط از یک جدول پایه با حداقل یک کلید کاندید استفاده شده باشد.

اگر رعایت نشود حفظ قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و حفظ تناظر سطر به سطر مابین دید و جدول پایه نقض می‌شود.

توجه مهم: عملیاتی همچون الحاق طبیعی، اشتراک، اجتماع و تفاضل میان دو جدول که خروجی آن یک جدول با کلید کاندید واضح و مشخص است، پذیرا بودن ذخیره‌سازی بر روی دید را از نظر تئوری تضمین می‌کند.

– در تعریف دید از دستور select distinct استفاده نشده باشد.

اگر رعایت نشود حفظ تناظر سطر به سطر مابین دید و جدول پایه نقض می‌شود.

– در تعریف دید از دستور group by استفاده نشده باشد.

اگر رعایت نشود حفظ تناظر سطر به سطر مابین دید و جدول پایه نقض می‌شود.

– در تعریف دید از توابع آماری استفاده نشده باشد.

اگر رعایت نشود حفظ قوانین مدل رابطه‌ای همچون داشتن ستون‌های واقعی و نه ستون مجازی (محاسبه شدنی) نقض می‌شود.

– در تعریف دید از زیر پرس‌وجوهای تو در تو استفاده نشده باشد.

اگر رعایت نشود حفظ تناظر سطر به سطر مابین دید و جدول پایه نقض می‌شود.

– در عمل Select کلید اصلی جدول حذف نشده باشد.

– در تعریف دید از عمل تقسیم استفاده نشده باشد.

مورد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر عملی تضمین می‌کند:

– در هنگام اجرای ذخیره‌سازی بر روی دید، قوانین جامعیت داخلی و خارجی بانک نقض نگردد.

سه جدول S، P و SP در سیستم تولیدکنندگان و قطعات با مقادیر زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S
(تولیدکنندگان)

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP
(تولید)

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P
(قطعات)

مثال: دید زیر را در نظر بگیرید:

```
Create View Report1
AS Select S# , Sname
    From S
```

توجه: اجرای دستور فوق منجر به ایجاد دید Report1 در پایگاه داده prodsuppdb می‌شود.

در تعریف دید Report1 از یک جدول پایه استفاده شده است، بدنه دید Report1 به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

آنچه واضح است، این است که ساختار دید Report1 از نظر تئوری پذیرای ذخیره‌سازی است، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ می‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ می‌گردد.

فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report1 درج نماید:

```
Insert into Report1
Values ('S4', 'Sn4')
```

از آنجا که دید Report1 استقلال وجودی ندارد و به جدول پایه S متصل است، عملیات درج فوق در واقع روی جدول پایه S انجام می‌گردد.

در واقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into S
Values ('S4' , 'Sn4' , NULL)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report1 قوانین جامعیت بانک رعایت می‌شود. پس اجرای دستور ذخیره‌سازی بر روی دید Report1 به صورت عملی از سوی DBMS پذیرفته می‌شود. البته به شرطی که ستون City در هنگام تعریف جدول S به صورت NOT NULL تعریف نشده باشد.

مقادیر کنونی جدول S پس از درج رکورد مذکور به صورت زیر است:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3
S4	Sn4	NULL

توجه: می‌توان ستون‌های جدول مجازی را با نام‌های جدیدی نام‌گذاری کرد. برای مثال در دید بالا S# به SID و Sname به SN می‌تواند نام‌گذاری مجدد شود، برای این منظور دید Report1 به صورت زیر مجدد بازنویسی می‌شود:

```
Create View Report1 (SID , SN)
AS Select S# , Sname
    From S
```

بدنه دید Report1 پس از نام‌گذاری مجدد به صورت زیر است:

SID	SN
S1	Sn1
S2	Sn2
S3	Sn3
S4	Sn4

توجه: جداول مجازی به جداول پایه مرتبط به خود متصل هستند، بنابراین در پرس‌وجوها می‌توان از دیدها نیز استفاده کرد، بنابراین هنگامی که در یک پرس‌وجو از جدول مجازی استفاده می‌شود، ابتدا تعریف جدول مجازی مربوطه در آن پرس‌وجو جاگذاری می‌شود و سپس پرس‌وجو اجرا می‌شود.

مثال: پرس‌وجوی زیر را روی دید Report1 در نظر بگیرید:

```
Select *
From Report1
Where Sname <> 'Sn1'
```

در واقع DBMS دستور فوق را به دستور زیر تبدیل می‌کند:

```
Select S# , Sname
From S
Where Sname <> 'Sn1'
```

خروجی پرس‌وجو به صورت زیر است:

S#	Sname
S2	Sn2
S3	Sn3
S4	Sn4

مثال: دید زیر را در نظر بگیرید:

```
Create view Report2
AS Select S# , P#
    From SP
    Where QTY>10
```

در تعریف دید Report2 از یک جدول پایه استفاده شده است، بدنه دید Report2 به صورت زیر است:

S#	P#
S_1	P_2
S_2	P_1

آنچه واضح است، این است که ساختار دید Report2 از نظر تئوری پذیرای ذخیره‌سازی است.
فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report2 درج نماید:

```
Insert into Report2
Values ('S3', 'P1')
```

از آنجا که دید Report2 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جدول پایه SP انجام می‌گردد.
در واقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into SP
Values ('S3', 'P1', NULL)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report2 قوانین جامعیت بانک رعایت می‌شود. پس اجرای دستور ذخیره‌سازی بر روی دید Report2 به صورت عملی از سوی DBMS پذیرفته می‌شود. البته به شرطی که ستون QTY در هنگام تعریف جدول SP به صورت NOT NULL تعریف نشده باشد. قانون جامعیت ارجاعی هم نقض نمی‌شود.
مقادیر کنونی جدول SP پس از درج رکورد مذکور به صورت زیر است:

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30
S3	P1	NULL

پذیرا از نظر تئوری و غیرپذیرا از نظر عملی

در این حالت دید از نظر تئوری پذیرای ذخیره‌سازی است، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ می‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ می‌گردد. اما از نظر عملی پذیرای ذخیره‌سازی نیست به دلیل عدم رعایت قوانین جامعیت بانک در مدل رابطه‌ای در حین اجرا.

موارد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر تئوری تضمین می‌کند:

– که در بخش قبل بررسی شد.

مورد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر عملی تضمین می‌کند:

– که در بخش قبل بررسی شد.

سه جدول S، P و SP در سیستم تولیدکنندگان و قطعات با مقادیر زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

مثال: دید زیر را در نظر بگیرید:

```
Create View Report3
AS Select S# , City
    From S
```

در تعریف دید Report3 از یک جدول پایه استفاده شده است، بدنه دید Report3 به صورت زیر است:

S#	City
S1	C1
S2	C2
S3	C3

آنچه واضح است، این است که ساختار دید Report3 از نظر تئوری پذیرای ذخیره‌سازی است.
فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report3 درج نماید:

```
Insert into Report3
Values ('S4', 'C4')
```

از آنجا که دید Report3 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جدول پایه S انجام می‌گردد. اما جدول S دارای سه ستون است، درحالیکه، فقط دو ستون در دستور فوق مربوط به جدول S مقداردهی شده‌است. بنابراین برای ستون Sname مقدار NULL در نظر گرفته می‌شود. فرض کنید در هنگام تعریف جدول S برای ستون Sname خاصیت NOT NULL در نظر گرفته شده باشد، یعنی حتما می‌بایست ستون Sname دارای مقدار باشد و مقدار NULL برای آن قابل پذیرش نیست. بنابراین مطابق قانون جامعیت بانک نباید ستون Sname با NULL مقداردهی شود. درواقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into S
Values ('S4', NULL , 'C4')
```

در هنگام ذخیره‌سازی عملی بر روی دید Report3 مطابق قانون جامعیت بانک در جداول نمی‌توان در ستون Sname از جدول S که خاصیت NOT NULL برای آن در نظر گرفته شده‌است، مقدار NULL را درج کرد، پس اجرای دستور ذخیره‌سازی بر روی دید Report3 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود.

سه جدول S، P و SP در سیستم تولیدکنندگان و قطعات با مقادیر زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

مثال: دید زیر را در نظر بگیرید:

```
Create view Report4
AS Select S# , P# , QTY
    From SP
    Where QTY>10
```

در تعریف دید Report4 از یک جدول پایه استفاده شده است، بدنه دید Report4 به صورت زیر است:

S#	P#	QTY
S1	P2	20
S2	P1	30

آنچه واضح است، این است که ساختار دید Report4 از نظر تئوری پذیرای ذخیره‌سازی است.
فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report4 درج نماید:

```
Insert into Report4
Values ('S10', 'P10' , 100)
```

از آنجا که دید Report4 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جدول پایه SP انجام می‌گردد.
درواقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into SP
Values ('S10', 'P10' , 100)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report4 هر چند هر سه ستون جدول SP به واسطه دستور درج، مقداردهی شده‌اند، اما این‌بار **قانون جامعیت ارجاعی** نقض شده است، زیرا ستون‌های S# و P# به عنوان کلیدهای خارجی در جدول SP تعریف شده‌اند و مطابق تعریف محتوایی کلید خارجی و قانون جامعیت ارجاعی، مجموعه مقادیر کلید خارجی باید همواره زیر مجموعه، مجموعه مقادیر کلید کاندید معادل خود باشد تا ارجاع NULL ایجاد نگردد، یعنی مجموعه مقادیر کلید خارجی S# در جدول SP باید همواره زیر مجموعه، مجموعه مقادیر کلید کاندید معادل خود یعنی S# در جدول S باشد و مجموعه مقادیر کلید خارجی P# در جدول SP باید همواره زیر مجموعه، مجموعه مقادیر کلید کاندید معادل خود یعنی P# در جدول P باشد، که

درج مقادیر S10 و P10 در جدول SP این قاعده را نقض می‌کند، پس اجرای دستور ذخیره‌سازی بر روی دید Report4 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود.
همچنین فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report4 درج نماید:

```
Insert into Report4
Values ('S1', 'P1', 90)
```

از آنجا که دید Report4 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جدول پایه SP انجام می‌گردد.
درواقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into SP
Values ('S1', 'P1', 90)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report4 هر چند هر سه ستون جدول SP به واسطه دستور درج، مقداردهی شده‌اند، اما این‌بار **قانون جامعیت درون رابطه‌ای** نقض شده است، زیرا ستون‌های S# و P# باهم به عنوان کلید کاندید در جدول SP تعریف شده‌اند و مطابق تعریف محتوایی کلید کاندید، مقادیر کلید کاندید باید یکتا باشد، که درج **مجدد** مقادیر S1 و P1 به طور همزمان در قالب یک سطر در جدول SP این قاعده را نقض می‌کند، پس اجرای دستور ذخیره‌سازی بر روی دید Report4 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود.
سه جدول S، P و SP در سیستم تولیدکنندگان و قطعات با مقادیر زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

مثال: دید زیر را در نظر بگیرید:

```
Create View Report5
    AS Select S.S#, Sname, P#
    From S JOIN SP
```

در تعریف دید Report5 از الحاق طبیعی میان دو جدول S و SP استفاده شده است، بدنه دید Report5 به صورت زیر است:

S.S#	Sname	P#
S1	Sn1	P1
S1	Sn1	P2
S2	Sn2	P1

آنچه واضح است، این است که ساختار دید Report5 از نظر تئوری پذیرای ذخیره‌سازی است، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ می‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ می‌گردد. کلید کاندید جدول حاصل از الحاق طبیعی همواره برابر کلید کاندید جدولی است که در آن کلید خارجی تعریف شده است، کلید کاندید جدول SP که در آن کلید خارجی تعریف شده است برابر (S#,P#) می‌باشد. بنابراین کلید کاندید جدول حاصل از الحاق طبیعی هم (S#,P#) است.

توجه مهم: شرط «**در تعریف دید فقط از یک جدول پایه با حداقل یک کلید کاندید استفاده شده باشد.**» برای پذیرا بودن ذخیره‌سازی بر روی دید را از نظر تئوری در الحاق طبیعی برقرار است، چون در نهایت خروجی الحاق طبیعی دو جدول یک جدول است که کلید کاندید آن نیز واضح و مشخص است.

فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report5 درج نماید:

```
Insert into Report5
Values ('S4', 'Sn4', 'P3')
```

از آنجا که دید Report5 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جداول پایه S و SP و P انجام می‌گردد.

درواقع دستور فوق به هنگام اجرا، به دستورات زیر می‌بایست تبدیل گردد:

```
 Insert into S
Values ('S4', 'Sn4', NULL)

Insert into SP
Values ('S4', 'P3', NULL)

Insert into P
Values ('P3', NUUL, NULL)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report5، برای مثال برای حفظ قوانین جامعیت بانک می‌بایست در چند جای مختلف از بانک درج‌هایی براساس سه دستور فوق انجام گردد، که SQL در عمل توانایی انجام چنین کاری را ندارد، شناخت و اجرای دستورات فوق برای SQL امکان‌پذیر نیست، از آنجاکه عدم اجرای هر یک از دستورات فوق جامعیت بانک را برهم می‌زند، و SQL هم توانایی شناخت و اجرای همه دستورات فوق را ندارد، پس اجرای دستور فوق یعنی ذخیره‌سازی بر روی دید Report5 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود.

غیر پذیرا از نظر تئوری و عملی

در این حالت دید از نظر تئوری پذیرای ذخیره‌سازی نیست، یعنی قوانین مدل رابطه‌ای همچون

داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ نمی‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ نمی‌گردد. همچنین از نظر عملی هم پذیرای ذخیره‌سازی نیست به دلیل عدم پذیرا بودن ذخیره‌سازی در دید از نظر تئوری.

موارد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر تئوری تضمین می‌کند:

- که در بخش قبل بررسی شد.

مورد زیر پذیرا بودن ذخیره‌سازی بر روی دید را از نظر عملی تضمین می‌کند:

- که در بخش قبل بررسی شد.

سه جدول S، P و SP در سیستم تولیدکنندگان و قطعات با مقادیر زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

مثال: دید زیر را در نظر بگیرید:

```
Create view Report6
AS Select S# , QTY
    From SP
    Where QTY>10
```

در تعریف دید Report6 از یک جدول پایه استفاده شده است، بدنه دید Report6 به صورت زیر است:

S#	QTY
S1	20
S2	30

آنچه واضح است، این است که ساختار دید Report6 از نظر تئوری پذیرای ذخیره‌سازی نیست، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید حفظ نشده است و بخشی از کلید اصلی (S#,P#) از جدول SP حذف شده است.

فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report6 درج نماید:

```
Insert into Report6
Values ('S3', 50)
```

از آنجا که دید Report6 استقلال وجودی ندارد، عملیات درج فوق در واقع روی جدول پایه SP انجام می‌گردد. اما جدول SP دارای سه ستون است، درحالیکه، فقط دو ستون در دستور فوق مربوط به جدول SP مقداردهی شده‌است. بنابراین برای ستون P# مقدار NULL در نظر گرفته

می‌شود. در هنگام تعریف جدول SP ترکیب ستون‌های (S#,P#) باهم کلید اصلی معرفی شده‌است. که به طور پیش فرض ستون‌های کلید اصلی NOT NULL در نظر گرفته می‌شود، یعنی حتما می‌بایست ستون‌های کلید اصلی (S#,P#) دارای مقدار باشد و مقدار NULL برای آن قابل پذیرش نیست. بنابراین مطابق **قانون جامعیت موجودیت** بانک نباید ستون S# و P# در جدول SP با NULL مقداردهی شود.

درواقع دستور فوق به هنگام اجرا، به دستور زیر تبدیل می‌گردد:

```
Insert into SP
Values ('S3', NULL , 50)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report6 مطابق قانون جامعیت موجودیت بانک در جداول نمی‌توان در ستون‌های (S#,P#) به عنوان کلید اصلی که به طور پیش فرض خاصیت NOT NULL برای آن در نظر گرفته شده است، مقدار NULL را درج کرد، پس اجرای دستور ذخیره‌سازی بر روی دید Report6 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود. بنابراین ذخیره‌سازی بر روی دید Report6 به صورت عملی قابل انجام نیست به دلیل عدم پذیرا بودن ذخیره‌سازی در دید Report6 از نظر تئوری. دقت کنید که وقتی تئوری ذخیره‌سازی روی دید امکان‌پذیر نباشد به تبع قطعا ذخیره‌سازی عملی آن هم امکان‌پذیر نیست.

مثال: دید زیر را در نظر بگیرید:

```
Create View Report7
    AS Select S#, AVG (QTY)
From SP
Group by S#
```

در تعریف دید Report7 از یک جدول پایه، دستور group by و تابع آماری AVG استفاده شده است، بدنه دید Report7 به صورت زیر است:

S#	no column name
S1	15
S2	30

آنچه واضح است، این است که دید Report7 از نظر تئوری پذیرای ذخیره‌سازی نیست، یعنی قوانین مدل رابطه‌ای همچون داشتن حداقل یک کلید کاندید و داشتن ستون‌های واقعی و نه مجازی حفظ نمی‌گردد و همچنین تناظر سطر به سطر مابین دید و جدول پایه حفظ نمی‌گردد.

فرض کنید کاربر با اجرای دستور زیر بخواهد یک سطر جدید را در دید Report7 درج نماید:

```
Insert into Report7
Values ('S3', 12)
```

در هنگام ذخیره‌سازی عملی بر روی دید Report7، برای مثال، مقصد مقدار 12 مشخص نیست،

زیرا مقصد آن یک ستون مجازی است و نه یک ستون واقعی، اگر به فرض محال هم درج شود میانگین تعداد قطعات S3 برابر 12 است حال آنکه #P آن مشخص نیست، پس اجرای دستور ذخیره‌سازی بر روی دید Report7 به صورت عملی از سوی DBMS پذیرفته نمی‌شود و رد می‌شود. بنابراین ذخیره‌سازی بر روی دید Report7 به صورت عملی قابل انجام نیست به دلیل عدم پذیرا بودن ذخیره‌سازی در دید Report7 از نظر تئوری. ضمن اینکه در تعریف دید Report7 همانند Report6 بازهم بخشی از کلید اصلی حذف شده‌است و قانون جامعیت موجودیت نیز نقض شده است.

افزایش استقلال داده‌ای به کمک Veiw

معماری ANSI برای پایگاه داده‌ها شامل سه لایه زیر است:

۱- لایه خارجی.

۲- لایه ادراکی شامل زیر لایه‌های مدل تحلیل (طراحی ادراکی یا ادراکی عام) و مدل طراحی (طراحی منطقی یا ادراکی خاص).

۳- لایه داخلی (فیزیکی).

یک محصول نرم افزاری از دو وجه عملکرد (برنامه کاربردی) و داده (بانک اطلاعات) تشکیل می‌شود. یکی از مهم‌ترین مزایای تکنولوژی پایگاه داده‌ها (مدل مفهومی پایگاه داده)، بلکه مهمترین هدف آن تأمین و افزایش استقلال داده‌ای است، به معنی وابسته نبودن برنامه‌های کاربردی به داده‌های ذخیره شده.

استقلال داده‌ای بر دو نوع می‌باشد:

۱- استقلال فیزیکی داده‌ها

به معنی مصونیت برنامه های کاربردی در قبال تغییراتی که در سطح فیزیکی (رسانه ذخیره سازی) پایگاه داده‌ها بروز می‌کند. یعنی اگر تغییری در رسانه ذخیره‌سازی داده‌ها انجام گیرد (برای مثال نوع دیسک عوض شود) برنامه‌های کاربردی هیچ تغییری نکند.

۲- استقلال منطقی داده‌ها

به معنی مصونیت برنامه‌های کاربردی در قبال تعاریف و تغییراتی که در سطح مدل طراحی (مدل رابطه‌ای) پایگاه داده بروز می‌کند. یعنی تعریف و تغییر مدل طراحی بانک (ادراکی خاص یا طراحی منطقی) از دید برنامه‌های کاربردی آنها مخفی بماند.

برای مثال مدل رابطه‌ای از تجریدی به نام جدول استفاده می‌کند و داده‌ها هر چه باشند در قالب چند جدول ریخته می‌شوند و نحوه ذخیره‌سازی داده‌ها روی رسانه‌ها از دید برنامه کاربردی مخفی است. در حالی که در روش فایلینگ تعاریف مربوط به فایل های داده‌ای، در فایل برنامه کاربردی می‌آمد. از آنجاکه برنامه‌های کاربردی براساس مدل طراحی بانک (ادراکی خاص یا طراحی

منطقی) تعریف می‌شوند، بنابراین به طور بالقوه در معرض تأثیرپذیری از **تغییرات** در مدل طراحی بانک (ادراکی خاص یا طراحی منطقی) قرار دارند.

توجه: در سیستم‌های امروزی، این نوع استقلال هم تا حدی (و نه صددرصد) تأمین شده است.

انواع تغییر در مدل طراحی (طراحی منطقی یا ادراکی خاص)

۱- **رشد پایگاه داده ها به دلیل مطرح شدن نیازهای جدید مشتری:** مانند درج جدول جدید، ترکیب جداول، تجزیه جداول.

۲- **سازماندهی مجدد:** مانند تغییر در نوع صفات خاصه، تغییر در اندازه صفات.

مثال: اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیز به آن اضافه گردد، در صورتی که برنامه کاربردی سابق نیاز به دستکاری و تغییر نداشته باشد، استقلال منطقی داده‌ها براساس تغییرات نیز لحاظ شده است.

توجه: از آنجاکه با حذف جداول، داده‌ها هم از بین می‌رود، بنابراین برنامه‌های کاربردی نسبت به **حذف** جداول، **هیچگاه** استقلال منطقی نخواهند داشت.

همانطور که گفتیم یک محصول نرم‌افزاری از دو وجه **عملکرد** (برنامه کاربردی) و **داده** (بانک اطلاعات) تشکیل شده است، **بخش داده** (بانک اطلاعات) که با SQL پیاده‌سازی می‌شود به مفاهیم **استقلال داده‌ای** مرتبط است. **ساختار وجه داده** توسط دستورات DDL نظیر Create Table، Create View و Create Index و دیگر دستورات آن ایجاد و مدیریت می‌گردد. و **مقادیر وجه داده** توسط دستورات DML نظیر Insert، Delete و Update و دیگر دستورات آن ایجاد و مدیریت می‌گردد.

دستور Create Table با ساخت مفهوم جدول، کمک به برقراری استقلال داده‌ای از نوع **استقلال فیزیکی داده‌ها** میان یک برنامه کاربردی و داده‌ها می‌کند، به معنی وابسته نبودن برنامه‌های کاربردی به داده‌های ذخیره شده، یعنی همانطور که گفتیم، مدل رابطه‌ای از تجریدی به نام جدول استفاده می‌کند و داده‌ها هر چه باشند در قالب چند جدول ریخته می‌شوند و نحوه ذخیره‌سازی داده‌ها روی رسانه‌ها از دید برنامه کاربردی مخفی است. دقت کنید که Table بخشی از وجه داده است. در واقع **ساختار وجه داده** از بخش‌های مختلف Table، View و Index تشکیل شده است.

دستور Create View با ساخت مفهوم دید، تا حدی کمک به برقراری استقلال داده‌ای از نوع **استقلال منطقی داده‌ها** میان یک برنامه کاربردی و داده‌ها می‌کند، به معنی وابسته نبودن برنامه‌های کاربردی به داده‌های ذخیره شده، یعنی همانطور که گفتیم، اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیز به آن اضافه گردد، در صورتی که برنامه کاربردی سابق نیاز به دستکاری و تغییر نداشته باشد، استقلال منطقی داده‌ها براساس تغییرات نیز لحاظ شده است. از آنجاکه View روی ساختار قدیم شامل نام جدول قدیم و ستون‌های قدیم ایجاد می‌شود، اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیز به آن اضافه گردد، آنگاه بدون تغییرات در ساختار View بخش

داده و به تبع تغییرات در ساختار بخش عملکرد(برنامه کاربردی)، امکان حیات برنامه کاربردی بدون اشکال همچنان وجود دارد و این یعنی View حافظ استقلال داده‌ای از نوع استقلال منطقی داده‌ها است. دقت کنید که View بخشی از وجه داده است. در واقع **ساختار وجه داده** از بخش‌های مختلف Table، View و Index تشکیل شده است.

مثال: پیاده‌سازی جدول S به صورت زیر را در نظر بگیرید:

```
Create Table S
        (
        S# char (5),
        Sname char (20),
        Primary key (S#)
        )
```

جدول S با مقادیر زیر را در نظر بگیرید:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

پرس‌وجوی زیر را در نظر بگیرید:

```
Select *
From S
```

خروجی پرس‌وجو به صورت زیر است:

S#	Sname
S1	Sn1
S2	Sn2
S3	Sn3

پیاده‌سازی View با نام Report8 را روی جدول S به صورت زیر در نظر بگیرید:

```
Create View Report8 (SID , SN)
AS Select S# , Sname
    From S
```

پرس‌وجوی زیر بر روی View با نام Report8 را به صورت زیر در نظر بگیرید:

```
Select *
From Report8
```

خروجی پرس‌وجو بر روی View با نام Report8 به صورت زیر است:

SID	SN
S1	Sn1
S2	Sn2
S3	Sn3

اضافه‌شدن ستون City به جدول S را به صورت زیر در نظر بگیرید:

Alter Table S **add** City char(15)

ساختار و مقادیر جدول S پس از اضافه‌شدن ستون City به صورت زیر است:

S#	Sname	City
S1	Sn1	NULL
S2	Sn2	NULL
S3	Sn3	NULL

مجددا پرس‌وجوی زیر بر روی View با نام Report8 را به صورت زیر در نظر بگیرید:

```
Select *
From Report8
```

خروجی پرس‌وجو بر روی View با نام Report8 به صورت زیر است:

SID	SN
S1	Sn1
S2	Sn2
S3	Sn3

توجه: همانطور که واضح است، از آنجاکه View روی ساختار قدیم شامل نام جدول قدیم و ستون‌های جدول قدیم ایجاد می‌شود، اگر جدولی دارای دو ستون باشد و ستون سومی نیز به آن اضافه گردد، آنگاه بدون تغییرات در ساختار View در بخش داده و به تبع تغییرات در ساختار بخش عملکرد (برنامه کاربردی)، امکان حیات برنامه کاربردی بدون اشکال همچنان وجود دارد و این یعنی **ساختار View حافظ استقلال داده‌ای از نوع استقلال منطقی داده‌ها است.**

دستور Create Index با ساخت مفهوم شاخص، هیچ کمکی به برقراری استقلال داده‌ای از نوع **استقلال فیزیکی داده‌ها و استقلال منطقی داده‌ها** میان یک برنامه کاربردی و داده‌ها نمی‌کند، به معنی وابسته نبودن برنامه‌های کاربردی به داده‌های ذخیره شده. مهمترین کاربرد شاخص، افزایش سرعت جستجو و بازیابی اطلاعات است و این مفهوم و کاربرد هیچ ارتباطی به مفهوم استقلال داده‌ای ندارد. دقت کنید که Index بخشی از وجه داده است. در واقع **ساختار وجه داده** از بخش‌های مختلف Table، View و Index تشکیل شده است.

دستور Drop View

این دستور جهت حذف فیزیکی یک دید از یک پایگاه داده به طور کامل و با تمام اطلاعات، مورد استفاده قرار می‌گیرد. پس از حذف فیزیکی یک دید امکان دسترسی به آن دیگر وجود نخواهد داشت.

ساختار کلی این دستور به صورت زیر است:

Drop View name

مثال: حذف دید (Report1)

Drop View Report1

توجه: اجرای دستور فوق منجر به حذف فیزیکی دید Report1 از پایگاه داده prodsuppdb می‌شود.

توجه: از لحظه ایجاد یک دید توسط دستور Create view تا لحظه حذف ساختار یک دید توسط دستور Drop View ، تمامی اطلاعات سیستمی مربوط به یک دید در کاتالوگ سیستم قرار دارد.

دستور Create Index

این دستور جهت ایجاد یک شاخص در یک پایگاه داده مورد استفاده قرار می‌گیرد.

ساختار کلی این دستور به صورت زیر است:

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] **INDEX** index_name
ON table_name (column1 [ASC | DESC], column2 [ASC | DESC], ...);

توجه: اجرای دستور فوق منجر به ایجاد یک شاخص در پایگاه داده می‌شود.

دستور Drop Index

این دستور جهت حذف فیزیکی یک شاخص از یک پایگاه داده به طور کامل و با تمام اطلاعات، مورد استفاده قرار می‌گیرد. پس از حذف فیزیکی یک شاخص امکان دسترسی به آن دیگر وجود نخواهد داشت.

ساختار کلی این دستور به صورت زیر است:

DROP INDEX index_name
Drop View Report1

توجه: اجرای دستور فوق منجر به حذف فیزیکی یک شاخص از پایگاه داده می‌شود.

توجه: از لحظه ایجاد یک شاخص توسط دستور Create Index تا لحظه حذف ساختار یک شاخص توسط دستور Drop Index ، تمامی اطلاعات سیستمی مربوط به یک شاخص در کاتالوگ سیستم قرار دارد.

توجه: مبحث شاخص‌گذاری اطلاعات (Index) در فصل شاخص به طور کامل تشریح می‌گردد.

دستورات تغییر داده‌ها (DML: Data Manipulation Language)

این دستورات در فصل DML طور کامل تشریح می‌گردد.

دستورات کنترل داده‌ها (DCL: Data Control Language)

همانطور که گفتیم این دستورات کارهای مربوط به تعریف محدودیت‌های امنیتی جداول پایگاه داده از قبیل ایجاد سطوح دسترسی برای کاربران را انجام می‌دهد. دستورات DCL به دستورات Authorization نیز موسوم هستند.

توجه: دستور Grant برای ایجاد و اعطای حق و مجوز دسترسی و دستور Revoke برای حذف و بازپس‌گیری حق و مجوز دسترسی در این طبقه قرار دارند.

در این بخش به بررسی دقیق‌تر دستورات DCL می‌پردازیم.

توجه: در SQL هر کاربر شناسه مختص به خود را دارد، کاربران جهت دستیابی به اطلاعات جداول نیاز به مجوز دارند.

در SQL می‌توان پنج نوع مجوز را به کاربران اعطاء نمود:

۱- **Select** : مجوز خواندن و بازیابی اطلاعات

۲- **Insert**: مجوز درج اطلاعات جدید

۳- **Update**: مجوز بروزرسانی اطلاعات

۴- **Delete**: مجوز حذف اطلاعات

۵- **References**: مجوز ارجاع کلید خارجی یک جدول فرعی به کلید کاندید یک جدول اصلی، کاربری که قصد ارجاع به کلید کاندید جدول اصلی دارد این مجوز باید برای او فعال باشد.

توجه: این مجوزها به کاربران داده می‌شود تا بتوانند روی جداول پایگاه داده عملیات خواندن، درج، بروزرسانی، حذف و ارجاع را به شکل محدود و کنترل‌شده انجام دهند.

دستور Grant

این دستور جهت ایجاد و اعطای حق و مجوز دسترسی به کاربران پایگاه داده مورد استفاده قرار می‌گیرد.

ساختار کلی این دستور به صورت زیر است:

Grant <فهرست مجوزها> **on** <نام جدول> **to** <نام کاربر> [**With Grant Option**]

توجه: شناسه Public همه کاربران را شامل می‌شود، اگر در بخش نام کاربر در دستور Grant، شناسه Public استفاده شود، مجوز تعیین شده به تمامی کاربران تعریف شده تا آن لحظه و کلیه کاربرانی که در آینده تعریف می‌شوند اعطا خواهد شد.

دستور Revoke

این دستور جهت حذف و بازپس‌گیری حق و مجوز دسترسی از کاربران پایگاه داده مورد استفاده قرار می‌گیرد.

ساختار کلی این دستور به صورت زیر است:

Revoke <فهرست مجوزها> **on** <نام جدول> **from** <نام کاربر>

مثال: فرض کنید مدیر اصلی پایگاه داده (DBA) چهار حساب کاربری با شناسه‌های A_1، A_2، A_3 و A_4 روی پایگاه داده prodsuppdb تعریف و ایجاد کند، همچنین طبق این تعریف فقط و فقط کاربر A_1 حق ایجاد جداول پایه (S، SP و P) را داشته باشد. (کاربر A_1 در برگه Server Roles در SQL Server از نوع sysadmin تعریف شده باشد).

اکنون کاربر A_1 می‌تواند جداول S، SP و P را درون پایگاه داده prodsuppdb ایجاد کند. فرض کنید کاربر A_1 سه جدول پایه S، SP و P را با ساختار مطرح شده در ابتدای فصل ساخته است. در اینصورت A_1 مالک این سه جدول است و مجوز انجام تمام عملیات‌ها (خواندن، درج، بروزرسانی، حذف و ارجاع) را به طور پیش فرض روی این سه جدول دارد.

توجه: اگر کاربری جدولی را به عنوان مالک آن ایجاد کند، آنگاه مجوز انجام تمام عملیات‌ها (خواندن، درج، بروزرسانی، حذف و ارجاع) را روی جدول مورد نظر خواهد داشت.

حال فرض کنید A_1 می‌خواهد مجوز **درج** و **حذف** را روی سه جدول پایه (S، SP و P) به A_2 اعطا کند، ولی نمی‌خواهد A_2 این مجوزها را به کاربر دیگری بدهد. پس دستور زیر را صادر می‌کند:

Grant Insert , Delete **on** S, SP, P **to** A_2

توجه: کاربری که مالک و خالق جدولی باشد (کاربر A_1) به طور پیش فرض حق واگذاری مجوزهای دسترسی را به کاربران دیگر دارد. همچنین A_2 نمی‌تواند مجوزهای دریافتی از سوی A_1 را به کاربران دیگر اعطا کند، چون A_1 مجوز دسترسی را به صورت With Grant Option و منتشر شونده به کاربر A_2 نداده است.

در مرحله بعد، A_1 می‌خواهد مجوز **خواندن و بازیابی اطلاعات** از سه جدول S، SP و P را به A_3 بدهد و البته در این مرحله A_1 مجوز دسترسی را به صورت With Grant Option و منتشر شونده به A_3 می‌دهد، در این شرایط A_3 می‌تواند مجوزهای دریافتی از سوی A_1 را به کاربران دیگر نیز اعطا کند. پس دستور زیر را صادر می‌کند:

Grant Select **on** S, SP, P **to** A_3 **With Grant Option**

اکنون، A_3 اگر بخواهد می‌تواند مجوز خواندن و بازیابی اطلاعات از سه جدول S، SP و P را به A_4 هم بدهد. پس دستور زیر را صادر می‌کند:

Grant Select **on** S, SP, P **to** A_4

اما A_4 نمی‌تواند مجوزهای دریافتی از سوی A_3 را به کاربران دیگر اعطا کند، چون A_3 مجوز دسترسی را به صورت With Grant Option و منتشر شونده به A_4 نداده است.

در مرحله بعد، فرض کنید کاربر A_1 تصمیم بگیرید، مجوز **خواندن و بازیابی اطلاعات** از سه جدول S، SP و P را از کاربر A_3 سلب کند. پس دستور زیر را صادر می‌کند:

Revoke Select **on** S, SP, P **from** A_3

البته DBMS پس از اجرای دستور فوق، در نهایت مجوز خواندن را علاوه بر A_3 از A_4 هم سلب می‌کند، زیرا مجوز A_4 توسط A_3 اعطا شده بود.

در مرحله بعد، فرض کنید A_1 تصمیم بگیرد، مجوز **خواندن و بازیابی اطلاعات** از فقط و فقط جدول S را مجددا به A_3 بدهد اما اینبار به شکل محدود شده و فقط روی ستون‌های #S و Sname و البته در این مرحله A_1 مجوز دسترسی را به صورت With Grant Option و منتشر شونده به A_3 می‌دهد، در این شرایط A_3 می‌تواند مجوزهای دریافتی از سوی A_1 را به کاربران دیگر نیز اعطا کند.

روش اول:

Grant Select (S# , Sname) **on** S **to** A_3 **With Grant Option**

توجه: در نسخه‌های جدید SQL امکان تعریف صفت خاص مثل روش اول برای مجوزهای Select و Delete وجود ندارد، زیرا با تعریف مفهوم View روی صفت‌های مورد نظر و دادن مجوز روی View می‌توان این نیاز را تامین کرد. بنابراین امروزه روش دوم مورد استفاده قرار می‌گیرد.

روش دوم:

توجه مهم: هر کاربری در پایگاه داده قدرت ساخت دید یعنی Create View را ندارد، فقط و فقط کاربری حق ساخت View را دارد که یا خالق و مالک همه جداول پایه دخیل در ساخت View باشد و یا حق Select توسط مالکان آن و دستور Grant به آن داده شده باشد.

توجه: کاربری که توسط دستور Create Table یک جدول ایجاد می‌کند، مالک و خالق آن جدول محسوب می‌شود. و تمام مجوزهای دسترسی خواندن، درج، حذف، بروزرسانی و ارجاع را به طور پیش فرض دارد، حتی این کاربر حق دادن این مجوزها به کاربران بعدی را نیز دارد.

توجه: ساختار View یک روش دسترسی محدود، کنترل‌شده و فیلترشده به جداول پایگاه داده می‌باشد که نتیجه آن تامین امنیت برای داده‌های جداول پایگاه داده است.

بنابراین در ادامه حل روش دوم ابتدا یک View ایجاد می‌کنیم:

Create View A3S
AS Select S# , Sname
From S

حال بعد از تعریف View، مجوز **خواندن و بازیابی اطلاعات** از جدول S و فقط روی ستون‌های #S و Sname با استفاده از ساختار View به A_3 اعطا می‌شود:

Grant Select **on** A3S **to** A_3 **With Grant Option**

در مرحله بعد، فرض کنید A_1 تصمیم بگیرد، مجوز **بروزرسانی** از فقط و فقط جدول S را مجددا به A_4 بدهد، اما اینبار به شکل محدود شده و فقط روی ستون‌های #S و Sname ، همچنین A_4 نمی‌تواند مجوزهای دریافتی از سوی A_1 را به کاربران دیگر اعطا کند، چون A_1 مجوز دسترسی را به صورت With Grant Option و منتشر شونده به A_4 نداده است.

Grant Update (S# , Sname) **on** S **to** A_4

و در نهایت، فرض کنید A_1 تصمیم بگیرید، مجوز **بروزرسانی** از جدول S و فقط ستون #S را از A_4 سلب کند. پس دستور زیر را صادر می‌کند:

Revoke Update(S#) **on** S **from** A_4

دستورات کنترل تراکنش‌ها (TCL: Transaction Control Language)

در این بخش به بررسی دقیق‌تر دستورات TCL می‌پردازیم.

تراکنش

هر برنامه ای که در محیط بانک اطلاعاتی توسط کاربر اجرا گردد یک تراکنش نام دارد. (مانند عملیات کارت به کارت در یک دستگاه خودپرداز بانک) تراکنش واحد کار DBMS است. به طور کلی هر عملیاتی در پایگاه داده در قالب یک تراکنش تعریف و اجرا می‌شود. هر تراکنش شامل دو یا چند دستور SQL است. تفاوت اصلی یک تراکنش با یک برنامه معمولی در محیط غیربانکی این است که تراکنش همواره به DBMS تسلیم می شود و DBMS در اعمال هر گونه کنترل و حتی به تعویق انداختن و ساقط کردن آن آزادی عمل دارد. هدف اصلی از اینگونه کنترل‌ها، حذف‌ها و تعویق‌ها، حفظ جامعیت داخلی و خارجی بانک اطلاعات است.

تضمین جامعیت داخلی و خارجی بانک اطلاعات

چهار کنترل زیر لازم است روی تمامی تراکنش ها در بانک اطلاعات اعمال گردد تا جامعیت داخلی و خارجی یعنی رعایت اصل سازگاری آن تضمین شود.

این کنترل‌ها به خواص ACID معروفند:

۱- یکپارچگی یا تجزیه‌ناپذیری (Atomicity)

این خاصیت به همه یا هیچ موسوم است. منظور این است که یا تمام دستورات یک تراکنش باید اجرا شود یا هیچکدام از آنها نباید اجرا شود. این به معنی تجزیه‌ناپذیر بودن بخش‌های مختلف یک تراکنش است.

برای مثال تراکنش انتقال پول از حساب A به حساب B از دو بخش جداگانه تشکیل یافته است:

الف) بخش اول (برداشت پول)
پول را از حساب A برداشت می‌کند.

ب) بخش دوم (واریز پول)
همان پول را به حساب B واریز می‌کند.
بخش اول حساب A را بدهکار و بخش دوم حساب B را بستانکار می‌کند. در شروع و پایان یک تراکنش سیستم باید سازگار باشد ولی در اثنای اجرای تراکنش ممکن است موقتاً نیاز به ناسازگاری باشد. برای مثال هنگام واریز پول از حساب A به B، پس از برداشت پول از حساب A سیستم به طور موقت ناسازگار است و پس از واریز آن به حساب B دوباره سیستم سازگار می‌شود. لذا برنامه برداشت از حساب A یا واریز به حساب B به تنهایی تراکنش نیستند.
این دو بخش ممکن است روی دو کامپیوتر جداگانه اجرا شوند. فرض کنید بخش اول تراکنش اجرا شود اما ناگهان ارتباط با ماشین دوم قطع گردد. و بخش دوم قابل انجام نباشد. بدیهی است که در این حالت باید پول برداشت شده دوباره به همان حساب اول بازگردانده شود تا جامعیت بانک اطلاعات حفظ شود. این عمل معادل این است که بگوییم هیچ دستورالعملی از تراکنش انجام نشده است.
به عنوان مثالی دیگر، هنگام خرید اینترنتی با کارت عضو شتاب ممکن است پول از حساب شما کسر گردد، اما به حساب فروشگاه مورد نظر واریز نگردد، بنابراین خرید شما ناموفق اعلام می‌گردد، که در این حالت پول حداکثر تا 48 ساعت دیگر به حساب شما بازمی‌گردد.
در بیانی دیگر تراکنش را می توان اینگونه تعریف کرد، تراکنش مجموعه‌ای از دستورات تعریف و دستکاری داده‌هاست که DBMS تضمین می‌کند یا همه آن دستورات اجرا شوند و یا هیچکدام از آن دستورات اجرا نشوند. برای محقق‌کردن خاصیت یکپارچگی (Atomicity) هر تراکنش می‌بایست بین دو دستور Begin Transaction و End Transaction قرار گیرد. به طور کلی هر عملیاتی در پایگاه داده در قالب یک تراکنش تعریف و اجرا می‌شود. پس تراکنش واحد کار DBMS است. هر تراکنش شامل دو یا چند دستور SQL است. بر این اساس می‌توان گفت که هیچ پرس و جویی برای پایگاه داده هویت مستقل ندارد، بلکه DBMS فقط تراکنش‌ها را می‌شناسد و اجرا می‌کند. به این ترتیب باید گفت که هر پرس‌وجویی در پایگاه داده ابتدا به یک تراکنش تبدیل می‌شود و سپس اجرا می‌گردد. در SQL برای نمایش ابتدای یک تراکنش از دستور Begin Transaction و برای نمایش خاتمه یک تراکنش از دستور End Transaction استفاده می‌شود. تراکنش با اجرای Begin Transaction شروع می‌گردد و در صورت اجرای موفق commit و در صورت عدم موفقیت یعنی عدم اجرای همه بخش‌های مختلف تراکنش با اجرای دستور Rollback خاتمه می‌یابد. با اجرای این دستور کلیه تغییراتی که تراکنش روی پایگاه داده

اعمال نموده است، ابطال می‌شود و وضعیت پایگاه داده به آخرین وضعیت قبل از اجرای تـراکنش برگردانده می‌شود.

۲- سازگاری (Consistency)

به طور کلی **جامعیت** در سیستم‌هـای بـانکی بـه دو طبقـه‌ی جامعیـت داخلـی و خـارجی تقسـیم می‌گردد. به حفظ قوانین مطـرح شـده از سـوی مـدل رابطـه‌ای و DBMS در سـطح پیـاده‌سـازی، جامعیت داخلی و به حفظ قوانین مطرح شده از سوی طراحـان و برنامـه‌نویسـان بانـک در سـطح پیاده‌سازی، جامعیت خارجی گفته می‌شود. در صورتی که در یک بانک جامعیت داخلی و خارجی هر دو توام باهم برقرار باشد، در آن بانک، **اصل سازگاری** برقـرار شـده اسـت. بـه عبـارت دیگـر سازگاری، به معنی رعایت قوانین داخلی و خارجی در بانک است. DBMS مسئول کنتـرل قـوانین داخلی و خارجی در بانک است و هرگونه عاملی که باعث نقض قوانین داخلی و خارجی و به تبع سازگاری بانک گردد را رد می‌کند. قوانین داخلی بانک شامل قانون جامعیت درون رابطه‌ای، قـانون جامعیت موجودیت، قانون جامعیت ارجاعی و قانون جامعیت دامنه‌ای است، ایـن مـوارد در فصـل مدل رابطه‌ای بررسی خواهد شد. قوانین خارجی بانک هم شامل هر قـانونی اسـت کـه طراحـان و برنامه‌نویسان بانک آنرا وضع می‌کنند، مانند تعریف زیردامنه برای ورود اطلاعات، تعریف بازه 0 تا 20 برای نمرات، تعریف بازه 0 تا بینهایت برای حساب‌های بانکی به معنی عـدم وجـود موجـودی منفی در حساب‌های بانکی...

در سیستم‌های بانکی کنترل جامعیت داخلی و خارجی به صـورت خودکـار توسـط مکـانیزم‌هـای موجود در DBMS انجام می‌گردد.

حال یکبار دیگر کد تعریف جدول SP را در نظر بگیرید:

```
Create Table SP
        (
        S# char (5),
        P# char (5),
        QTY numeric (10),
        Primary key (S#, P#),
        Foreign key (S#) References S(S#)
          on delete cascade
          on update cascade,
        Foreign key (P#) References P(P#)
          on delete cascade
          on update cascade,
        Check (QTY>1 AND QTY<1000)
        )
```

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (S#) References S(S#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #S در جدول S به ستون #S در جدول SP نیز اعمال گردد. بنابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

یا به طور مشابه، کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Foreing key (P#) References P(P#)
on delete cascade
on update casecade
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه تغییری در ستون #P در جدول P به ستون #P در جدول SP نیز اعمال گردد. بنابراین **جامعیت داخلی** از نوع جامعیت ارجاعی نقض نمی‌گردد.

کارکرد قطعه کد زیر از کد تعریف جدول SP فوق به صورت زیر است:

```
Check (QTY>1 AND QTY<1000)
```

این قطعه کد، سبب می‌گردد تا به طور خودکار هرگونه مقداردهی در ستون QTY از جدول SP در بازه 1 تا 1000 باشد، بنابراین **جامعیت خارجی** نقض نمی‌گردد.

توجه: همانطور که واضح است DBMS در حفظ جامعیت داخلی و خارجی به دقت نظارت دارد.

خاصیت سازگاری (Consistency) بیانگر این است که اگر یک تراکنش در محیط بانک اطلاعات انجام شود باید بانک اطلاعات را از حالتی سازگار به حالت سازگار دیگری منتقل کند.

به بیان دیگر هر تراکنش باید تمامی قوانین جامعیت داخلی و خارجی بانک اطلاعات را رعایت کند. خاصیت سازگاری می‌گوید انجام تراکنشی از سوی DBMS باید پذیرفته شود که جامعیت داخلی و خارجی به معنی حفظ قوانین داخلی و خارجی پایگاه داده را رعایت کند، یعنی پس از انجام تراکنش‌ها اصل سازگاری برقرار باشد، در غیراینصورت انجام تراکنش رد شود.

بنابراین تا به اینجا مشاهده شد که تراکنش ممکن است دو نوع پایان داشته باشد:

الف) پایان موفق که آن را انجام (commit) می‌نامند.

ب) پایان ناموفق که آن را سقوط (abort) می‌نامند.

۳- انزوا یا جداسازی (Isolation)

همزمانی در دسترسی به داده‌ها موجب بهبود کارایی و کاهش زمان پاسخ‌گویی سیستم می‌گردد. و این امر تسریع عملکرد برنامه‌ها را در پی دارد. بسیاری از سیستم‌ها اجازه می‌دهند که چندین کاربر به صورت همروند به داده ها دسترسی یابند و تغییرات مورد نظر خود را بر روی آنها اعمال نمایند.

در چنین محیط هایی تغییرات همروند ایجاد شده بر روی داده ممکن است منجر به ایجاد ناسازگاری در داده‌ها گردد.
در سیستم‌های بانکی کاربران مختلف می‌توانند به صورت همزمان با بانک کار کنند. بنابراین اگر یک داده **خاص** بین کاربران مختلف به صورت اشتراکی مورد بازیابی و دستکاری قرار گرفت سیستم پایگاه داده باید محیطی را ایجاد نماید که مانع از بروز مشکلات و یا ایجاد نتایج نامطلوب گردد، مانند مطالب مربوط به ناحیه بحرانی در فرآیندهای همروند در درس سیستم عامل.
مثال : فرض کنید کاربر A و B به ترتیب در تهران و شیراز همزمان قصد برداشت وجه از حساب آقای 6037 از طریق برگ چک را دارند. بنابراین روال‌های زیر را خواهیم داشت، از آنجا که برداشت وجه از یک رکورد مشترک (عامل مشترک) صورت می‌گیرد، به تبع وقوع پدیده همزمانی برای هر دو تراکنش رخ می دهد. روال کار بدین صورت است که هر دو تراکنش مبلغ موجودی حساب که برابر مقدار 500 هزارتومان می‌باشد را خوانده و با توجه به مبالغ برگ چک A و B به ترتیب مبالغ 100 و 50 هزارتومان را از حساب کسر می کنند و بر حسب اینکه کدام تراکنش آخرین بروزرسانی را انجام دهد مبلغ مانده حساب 400 تا 450 هزارتومان ذخیره می‌گردد. که در هر دو صورت اطلاعات نادرستی ذخیره شده است. در حالی که مبلغ 350 هزار تومان باید جهت مبلغ مانده حساب ذخیره می‌شد!

شماره مشتری	شماره حساب	نام	نام‌خانوادگی	موجودی
6037	0313	Ali	Alavi	500

Read (A)	Read (B)
A □ 500	B □ 500
R=500-100	R=500-50
R □ 400	R □ 450
Write (R) □ 400	Write (R) □ 450

بدین‌ترتیب مقادیر نادرستی توسط برنامه‌ها ذخیره شده است که این نادرستی به دلیل تداخل عملیات دو برنامه همروند است. برای جلوگیری از ایجاد چنین نتایج نامطلوبی سیستم باید نوعی نظارت بر عملکرد برنامه‌های همروند داشته باشد. مانند روش قفل‌گذاری.
در بانک اطلاعات ممکن است تراکنش‌های همروند وجود داشته باشد (مثل سیستم‌های چند برنامه‌ای) بر طبق خاصیت انزوا همروندی تراکنش‌ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند به بیان دیگر اثر تراکنش های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا انجام می‌شود.
به تعریفی دیگر تراکنش‌ها جدا از یکدیگر هستند. اگر چند تراکنش به طور همزمان اجرا شوند، به هنگام‌سازی‌های هر کدام از یکدیگر مخفی می‌مانند تا به اتمام برسند. به عبارتی دیگر، برای دو تراکنش مجزای A و B، تراکنش A می‌تواند بهنگام سازی‌های B را پس از پذیرفته شدن آن

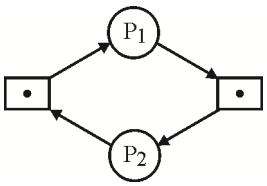
(commit) یا B می‌تواند بهنگام سازی‌های A را پس از پذیرفته شدن A ببیند. اما این دو تراکنش به طور همزمان نمی‌توانند، بهنگام‌سازی‌های یکدیگر ببینند.

توجه: کنترل هم‌روندی توسط بخشی از DBMS به نام واحد کنترل همروندی (concurrency control) انجام می‌شود.

روش قفل‌گذاری

یکی از روش‌های اعمال خاصیت جداسازی در تراکنش‌ها و جلوگیری از اثر مخرب تراکنش‌های همروند بر روی یکدیگر، روش قفل‌گذاری است. در این روش هنگامی که تراکنش به داده‌ای نیاز داشته باشد. تقاضای قفل کردن آن را می‌دهد و در این حالت بقیه تراکنش‌ها تا اتمام کار آن نمی‌توانند از آن استفاده کنند. در واقع استفاده از انواع قفل‌ها در مکانیسم قفل‌گذاری، روش‌های دسترسی به فیلدها و یا داده‌های اشتراکی را در کاربردهای دیگر، امکان‌پذیر می‌سازد. وجود قفل‌ها سبب می‌شود که در اجرای همروند چند تراکنش، یک رکورد مشترک (عامل مشترک) به صورت همزمان توسط دو تراکنش مورد استفاده قرار نگیرد. این کار سرعت عملیات را افزایش می‌دهد زیرا داده اشتراکی به صورت انحصاری فقط در اختیار یک تراکنش است. ولی احتمال بروز بن‌بست را به دلیل برقراری شرط انحصار متقابل زیاد می‌کند.

بر اساس قاعده کافمن شرایط وقوع بن‌بست به صورت زیر است:



۱ـ انحصار متقابل

۲ـ انحصاری بودن

۳ـ نگهداری و انتظار

۴ـ انتظار چرخشی

۴- پایائی یا ماندگاری (Durabiltiy)

براساس این خاصیت تراکنش هایی که به مرحله انجام (commit) برسند اثرشان ماندنی است و هرگز به طور تصادفی از بین نمی‌روند. برای مثال اگر مبلغی به حسابی واریز شود و تراکنش مربوطه انجام یافته (commit) اعلام شود حتی در صورت وقوع حادثه در آن شعبه بانک، مشتری نباید متضرر شود، برای مثال عمل واریز قبل از اعلام انجام موفق (commit) باید در جای دیگر نیز ثبت شده باشد، مثل دیسک اصلی یا دیسک پشتیبان. یعنی تأثیرات تراکنش در پایگاه داده، ماندگار باشد. هنگامی‌که یک تراکنش دستور commit را اجرا می‌کند نتایج اجرای آن به نسخه‌ی اصلی پایگاه داده در دیسک منتقل می‌شود. بنابراین می‌توان گفت تا زمانی‌که یک تراکنش دستور commit را اجرا نکرده است یا به اصطلاح تثبیت نشده است، ویژگی ماندگاری در مورد آن تراکنش تضمین شده نیست. اما پس از اجرای دستور ماندگاری نتایج تراکنش تضمین می‌شود.

توجه: دو خاصیت یکپارچگی و پایایی توسط واحدی از DBMS به نام واحد مدیریت بازگرد (Recovery management) کنترل می‌گردد.

۶ شاخص‌گذاری اطلاعات

مقدمه

یک پایگاه داده عملیاتی با جداول بدون شاخص، تنها زمانی که تعداد کمی رکورد در جداول وجود داشته باشد، سرعت قابل قبولی در اجرای پرس و جوها خواهد داشت، اما با افزایش تدریجی رکوردها، عملا با کندی سرعت جستجو و بازیابی اطلاعات مواجه خواهد شد. ممکن است شما هم این مورد را تجربه کرده باشید که در زمان توسعه‌ی یک برنامه کاربردی، کلیه‌ی عملیات جستجو و بازیابی اطلاعات با سرعت خوبی اجرا می‌شود، اما بعد از تحویل به مشتری و گذشت یک مدت زمان، با گله و شکایت کندی سرعت از سوی مشتری مواجه می‌شوید. در انتهای اغلب کتاب‌ها شاخص (Index) وجود دارد، به این معنا که لیست کلمات و اصطلاحات مهم (کلیدها) کتاب به ترتیب حروف الفبا، به همراه شماره صفحاتی که آن کلمات استفاده شده (آدرس رکوردها)، آورده می‌شود. مهمترین کاربرد شاخص، افزایش سرعت جستجو و بازیابی اطلاعات است. به نحوی که در صورت عدم وجود فهرست مطالب در یک کتاب، می‌بایست تک تک صفحات کتاب را جهت یافتن مطلب مورد نظر، برگ می‌زدیم. پس شاخص‌ها بر مبنای کلیدهای جستجو و آدرس رکوردها ساخته می‌شوند. شاخص باعث بالا رفتن سرعت دستیابی به اطلاعات می‌گردد. تکنیک شاخص‌بندی یا شاخص‌گذاری (Indexing) در اکثر نرم‌افزارهای امروزی استفاده می‌شود. شاخص‌ها برای بهبود فرآیند جستجو و بازیابی اطلاعات در جداول ایجاد می‌شوند. به عبارت دیگر شاخص‌ها به فرآیند جستجو و بازیابی اطلاعات، سرعت می‌بخشند. شاخص‌ها باعث می‌شوند موتور جستجوی پایگاه داده کل یک جدول را برای پیداکردن رکورد یا رکوردهای مورد نظر به طور کامل نگردد. اساسا شاخص‌ها بر روی ستون‌هایی باید تنظیم شود که بیشتر مورد جستجو قرار می‌گیرند.

دو هدف اصلی سیستم ذخیره و بازیابی اطلاعات در پایگاه داده‌ها، اول سرعت عملیات در ذخیره و بازیابی اطلاعات و دوم صرفه‌جویی در مصرف حافظه است. برای مثال کاهش افزونگی

محتوایی (طبیعی) توسط نرمال‌سازی جداول منجر به کاهش میزان حافظه مصرفی می‌شود. عمل واکشی تک تک رکوردها وقت‌گیر است، برای رفع این عیب، شاخص یا Index ابداع شد. برای اینکه جستجو و بازیابی داده‌ها با سرعت و کارایی بیشتر صورت گیرد، از شاخص استفاده می‌شود. شاخص ساختمان داده‌ای است که سیستم مدیریت پایگاه داده‌ها به کمک آن رکوردهای مورد نظر را در یک فایل با سرعت بسیار زیاد پیدا می‌کند و به این ترتیب سرعت پاسخ به پرس و جوها افزایش می‌یابد. هر ساختار شاخص، حاوی رکوردهایی است که در هر رکورد یک مقدار کلیدی (کلید جستجو) و یک اشاره‌گر شامل آدرس فیزیکی رکوردهای جدول پایه در آن نگهداری می‌شود. مطابق شکل زیر:

Search Key	Pointer

اغلب سیستم‌های مدیریت پایگاه داده‌ها، از ساختار درخت برای ایجاد شاخص‌ها استفاده می‌کنند. عمق درخت، بیشترین تعداد سطوح از ریشه به برگ است. عمق ممکن است در مسیرهای مختلف از ریشه تا برگ متفاوت باشد. و همچنین عمق ممکن است در مسیرهای مختلف از ریشه تا برگ یکسان باشد، که در این شرایط با درخت متوازن و $B^+ - Tree$ مواجه هستیم. هرچه درجه‌ی گره‌های درخت بیشتر شود، درخت پهن‌تر و کم‌عمق‌تری ایجاد می‌شود. از آنجاییکه زمان دسترسی در یک درخت، بیشتر وابسته به عمق درخت است تا پهنای آن، پس ساخت درخت پهن و کم‌عمق در ایجاد شاخص باعث افزایش سرعت جستجو می‌شود، ساختارهای $B^+ - Tree$ درخت‌هایی با عمق کم و پهنای زیاد هستند. ساختار B^+ Tree index برای پاسخ به Range Query و Equality Query مناسب است. بنابراین اعمال سیاست شاخص‌گذاری توسط ساختار B^+ Tree باعث می‌شود Range Queryها و Equality Queryهای مرتبط با ستون مورد نظر، با سرعت بیشتری انجام شود.

توجه: یک شاخص روی مقادیر یک یا چند ستون از جدول تعریف می‌شود. و به ستون‌هایی که شاخص روی آنها تعریف می‌شود، **کلید جستجو** (Search Key) گفته می‌شود. در واقع در شاخص به جای نگهداری کلیه اطلاعات یک جدول، فقط مقادیر کلید جستجو نگهداری می‌شود.

توجه: اغلب شاخص روی کلید اصلی تعریف می‌شود، که در این حالت کلید جستجو همان کلید اصلی خواهد بود. اما هیچ الزامی برای این امر وجود ندارد و می‌توان با تعریف شاخص روی هر یک از ستون‌ها آنرا به عنوان کلید جستجو در نظر گرفت. بنابراین امکان تعریف بیش از یک شاخص روی یک جدول وجود دارد.

توجه: امکان تعریف یک شاخص روی چندین ستون به صورت ترکیبی نیز وجود دارد، در اینصورت کلید جستجو شامل چندین ستون است که به آن کلید جستجوی مرکب

(Composite Search Key) می‌گویند. در این حالت مبنای مرتب‌سازی رکوردها ترکیب ستون‌هاست. برای مثال اگر شاخص روی ترکیب ستون‌های (X,Y) تعریف شده باشد، آنگاه $(x_1,y_1)<(x_2,y_2)$ است، اگر $x_1<x_2$ و یا $x_1=x_2$ و $y_1<y_2$

توجه: اگر شاخص روی یک ستون یا چندستون(ترکیبی) تعریف شده باشد، فقط سرعت جستجوهای مبتنی بر آن ستون یا چندستون را افزایش می‌دهد.

توجه: تعریف و نگهداری شاخص موجب تحمیل سربار حافظه‌ای به سیستم می‌شود. شاخص بر روی هارد دیسک نگهداری می‌شود و هنگام استفاده به حافظه اصلی آورده می‌شود، بنابراین اعمال سیاست شاخص‌گذاری، بر افزایش حجم اطلاعات ذخیره شده بر روی حافظه اصلی و هارد دیسک تاثیر دارد.

توجه: عملیات درج، حذف و بروزرسانی در جدولی که شاخص دارد یعنی خود جدول پایه، نسبت به جدولی که شاخص ندارد زمان بیشتری مصرف می‌کند و به تبع این عملیات کندتر خواهد بود. زیرا شاخص‌ها نیز همگام با جداول پایه خود نیاز به بروزرسانی دارند. بنابراین تنها روی ستون‌هایی شاخص ایجاد می‌گردد، که به تناوب روی آنها جستجو انجام می‌شود. بنابراین هرچه تعداد شاخص‌های یک جدول بیشتر باشد، سرعت Insert، Update و Delete در آن جدول کمتر می‌شود. اما خود شاخص عامل جستجو و بازیابی سریع اطلاعات از یک جدول پایه است.

توجه: شاخص‌گذاری، افزونگی تکنیکی دارد و مقداری از فضای حافظه اصلی و هارد دیسک را اشغال می‌کند. به عبارت دیگر تعریف هر شاخص روی یک جدول هزینه‌ی زیادی را به پایگاه داده تحمیل می‌کند و اگر نرخ تغییرات محتوایی پایگاه داده زیاد باشد این هزینه به صورت قابل توجهی افزایش می‌یابد. بنابراین طراحان پایگاه داده ترجیح می‌دهند که روی هر جدول بیش از یک شاخص تعریف نکنند.

توجه: شاخص‌گذاری، منجر به افزایش سرعت جستجو و بازیابی اطلاعات و به تبع کاهش زمان جستجو و بازیابی اطلاعات می‌شود، نبود شاخص باعث کندی سرعت جستجو و زیادی آن باعث کندی فرآیندهای درج، حذف بروزرسانی رکوردها در جداول پایه می‌شود. بنابراین راز نه در افراط است و نه در تفریط بلکه راز در تعادل است، گاهی هم استفاده‌ی مکرر از گزینه‌های خوب، نتیجه‌ی بد هم می‌تواند به همراه داشته باشد! و به قول معروف در حوزه‌ی سلامتی نیاکان ما گفته‌اند که کم بخور، همیشه بخور...

توجه: شاخص (Index) با هدف افزایش سرعت جستجو و بازیابی اطلاعات و به تبع کاهش زمان جستجو و بازیابی اطلاعات ایجاد می‌گردد. و به عنوان نمونه‌ی دیگری از فراداده‌ها، مشخصات سیستمی شاخص در کاتالوگ سیستم نگهداری می‌شود.

پیاده‌سازی مفهوم شاخص در SQL-DDL

برای **ایجاد** یک شاخص روی یک جدول پایه، از دستور CREATE INDEX استفاده می‌شود. با اجرای این دستور، تعریف مشخصات شاخص در کاتالوگ نیز درج می‌شود.
فرم کلی ایجاد شاخص به صورت زیر است:

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX index_name
ON table_name (column1 [ASC | DESC], column2 [ASC | DESC], ...);
```

برای **حذف** یک شاخص تعریف شده روی یک جدول پایه، از دستور DROP INDEX استفاده می‌شود. با اجرای این دستور، تعریف مشخصات شاخص از کاتالوگ نیز حذف می‌شود.
فرم کلی حذف شاخص به صورت زیر است:

```
DROP INDEX index_name
```

به طور کلی دو نوع شاخص وجود دارد:
(۱) شاخص از نوع مرتب‌شده و (۲) شاخص از نوع Hash که در ادامه به بررسی آن می‌پردازیم.

۱- شاخص از نوع مرتب‌شده (Ordered Index)

همانطور که گفتیم، یک شاخص روی مقادیر یک یا چند ستون از جدول تعریف می‌شود. و به ستون‌هایی که شاخص روی آنها تعریف می‌شود، **کلید جستجو** (Search Key) گفته می‌شود. در واقع در شاخص به جای نگهداری کلیه اطلاعات یک جدول پایه، فقط مقادیر کلید جستجو نگهداری می‌شود. جهت جستجوی سریع‌تر، در شاخص مرتب‌شده مقادیر کلید جستجو به صورت مرتب‌شده نگهداری می‌شود، که علت نامگذاری نیز همین بوده است.

توجه: شاخص مرتب‌شده به کمک دو ساختار آرایه و ساختار B^+ Tree قابل پیاده‌سازی است.

توجه: در SQL Server شاخص مرتب‌شده به کمک ساختار B^+ Tree پیاده‌سازی شده است. به عبارت دیگر مقادیر شاخص در ساختار B^+ Tree نگهداری می‌شود.

توجه: به طور کلی، درخت یک ساختمان داده غیرخطی است که برای عملیاتی مانند **جستجو** مورد استفاده قرار می‌گیرد.

توجه: انجام عملیات در B^+ Tree، نسبت به سایر درخت‌ها سریعتر است، بنابراین در SQL Server ساختار B^+ Tree جهت نگهداری مقادیر شاخص مورد استفاده قرار می‌گیرد.

توجه: شاخص مرتب‌شده به دو صورت clustered Index و Nonclustered Index وجود دارد.

شاخص مرتب‌شده به صورت clustered Index

توجه: هر جدول می‌تواند حداکثر **یک** clustered Index داشته باشد.

توجه: زمانی که کلید اصلی یک جدول در SQL Server تعریف می‌شود اولین شاخص بر روی آن جدول ایجاد می‌شود که به آن clustered Index گفته می‌شود.

توجه: در clustered Index ترتیب منطقی رکوردها در جدول پایه با ترتیب فیزیکی چیدمان آن‌ها بر روی هارد دیسک یکسان است. بنابراین هر جدول می‌تواند حداکثر یک clustered Index داشته باشد. حفظ ترتیب منطقی رکوردها در جدول پایه با ترتیب فیزیکی چیدمان آن‌ها بر روی هارد دیسک توسط بخشی بنام Row Offset Array و توسط اشاره‌گرها انجام می‌گردد.

توجه: به شاخص مرتب‌شده به صورت clustered Index، شاخص اصلی یا اولیه نیز می‌گویند.

توجه: مقادیر شاخص مرتب‌شده به صورت clustered Index باید یکتا و منحصر به فرد باشد و UNIQUE تعریف گردد.

توجه: شاخص مرتب‌شده به صورت clustered Index، منجر به این می‌شود که رکوردهای یک جدول پایه به صورت ترتیبی و **صعودی** بر روی هارد دیسک ذخیره شوند.

قطعه کد ایجاد clustered Index به صورت زیر است:

```
CREATE UNIQUE CLUSTERED INDEX index_name
ON table_name (column1 ASC, column2 ASC, ...);
```

توجه: زمانی که کلید اصلی یک جدول در SQL Server تعریف می‌شود قطعه کد فوق به طور خودکار اجرا می‌شود.

توجه: به طور پیش فرض و غیرقابل تغییر در ایجاد clustered Index دستور UNIQUE استفاده می‌گردد، تا یکتایی و منحصر به فرد بودن مقادیر شاخص **گارانتی** گردد.

توجه: به طور پیش فرض و غیرقابل تغییر در ایجاد clustered Index دستور Ascending یا ASC استفاده می‌گردد، تا **صعودی** بودن مقادیر شاخص **گارانتی** گردد.

مثال: با اجرای قطعه کد زیر جدول S با کلید اصلی S# و به تبع یک clustered Index روی ستون S# ایجاد می‌گردد.

```
CREATE TABLE S (
    S# int NOT NULL,
    SName varchar(255),
    City varchar(255),
    PRIMARY KEY (S#)
);
```

قطعه کد ایجاد clustered Index که بطور خودکار اجرا می‌شود، به صورت زیر است:

```
CREATE UNIQUE CLUSTERED INDEX PK_S
ON S (S# ASC);
```

جداول زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

با استفاده از clustered Index پیش فرض و موجود روی ستون #S، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس #S در جدول S انجام می‌دهند، افزایش می‌یابد.

مثال:

```
SELECT S#
FROM S
WHERE S#='S9'
```

اما در مورد پرس و جوهایی که جستجو را بر اساس ستون #S انجام نمی‌دهند. تعریف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

توجه: در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

ساختار B^+ Tree

B^+ Tree به معنای درخت متوازن (Balanced Tree) است و نه درخت دودویی (Binary Tree) و یا درخت جستجوی دودویی (BST-Binary Search Tree).

توجه: ساختار B^+ – Tree یک درخت جستجو است. در این ساختار سه نوع گره ریشه (Root Node)، گره میانی (Intermediate Node) و گره برگ (Leaf Node) وجود دارد که همه برگ‌هایش در یک سطح هستند. به عبارت دیگر ساختار B^+ – Tree از سطوح ریشه، میانی و برگ تشکیل شده است.

توجه: با توجه به متوازن بودن ساختار B^+-Tree و فاصله یکسان همه گره‌های موجود در سطح برگ تا ریشه، جهت دسترسی به هر یک از گره‌های موجود در سطح برگ مراحل یکسانی لازم است. به عبارت دیگر تمامی گره‌های موجود در سطح برگ (leaf) دارای فاصله (عمق) یکسانی تا ریشه هستند. به دلیل اینکه در هر عملیاتی تعداد مراجعه به دیسک متناسب با تعداد سطوح درخت است. بنابراین زمانی که همه گره‌های سطح برگ دارای عمق یکسانی باشند، انجام عملیات در هر گره هزینه یکسانی با سایر گره‌ها دارد.

توجه: به دلیل وجود ارتباط (اشاره‌گر) بین گره‌ها در این ساختار، پیمایش در B^+ Tree به سرعت انجام می‌شود.

توجه: اصطلاحا B^+-Tree را درجه t می‌گویند $(t \geq 3)$ که داخل هر گره‌اش حداقل $\left\lceil \left(\frac{t}{2}\right) \right\rceil - 1$ و حداکثر $t-1$ کلید جستجو است. همچنین تعداد فرزندان (اشاره‌گرها) یکی از تعداد کلیدها بیشتر است، بنابراین هر گره غیر برگ حداقل $\left\lceil \left(\frac{t}{2}\right) \right\rceil$ فرزند و حداکثر t فرزند دارد. البته ریشه استثنا است و می‌تواند حداقل شامل یک کلید جستجو باشد، یعنی حداقل دو فرزند داشته باشد، اما ریشه حداکثر همان $t-1$ کلید جستجو را می‌تواند داشته باشد، یعنی حداکثر t فرزند. در یک قاعده‌ی کلی گره‌ای که xتا کلید داشته باشد، اگر برگ نباشد، دقیقا $x+1$ فرزند دارد.

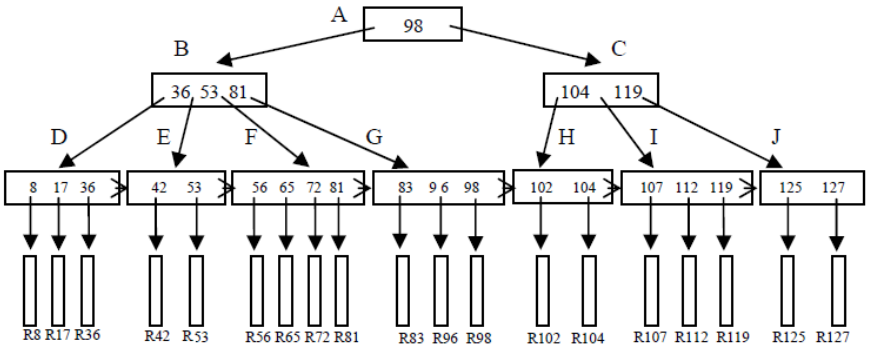
نتیجه: $\left\lceil \left(\frac{t}{2}\right) \right\rceil - 1 \leq$ **تعداد کلید جستجو در** B^+-Tree **درجه** $t \leq t-1$

نتیجه: $t \leq$ **تعداد فرزند(اشاره‌گر) در** B^+-Tree **درجه** $t \leq \left\lceil \left(\frac{t}{2}\right) \right\rceil$

توجه: در ساختار B^+-Tree اشاره‌گرهای موجود در ریشه و گره‌های میانی به گره‌های فرزند اشاره می‌کنند، اما اشاره‌گرهای موجود در برگ‌ها به سطرها یا رکوردهای مربوطه در دیسک اشاره می‌کنند.

توجه: در B^+ Tree تمام کلیدهای جدول پایه در برگ‌ها نگهداری شده و برگ‌ها به یکدیگر متصل می‌شوند تا یک مسیر سریالی و ترتیبی برای پیمایش کلیدها در درخت را فراهم سازد.

توجه: استفاده از B^+ Tree سرعت جستجو و بازیابی اطلاعات مورد نظر را افزایش می‌دهد. برای جستجو و بازیابی اطلاعات مورد نظر با استفاده از B^+ Tree عملیات جستجو از ریشه آغاز می‌شود و تا رسیدن به یکی از برگ‌ها ادامه پیدا می‌کند. برای مثال شکل زیر یک B^+ Tree است:



در شکل فوق برای جستجو و بازیابی (lookup) رکورد وابسته به کلید 53 یعنی R53 ابتدا کلید 53 با 98 یعنی ریشه مقایسه می‌شود، چون کمتر از آن است کنترل به سراغ گره B می‌رود. کلید 53، از 36 بزرگتر و از 53 کوچکتر یا مساوی است لذا به سراغ گره E می‌رود. در ساختار B^+ Tree فقط کلیدهای موجود در برگ‌ها حاوی اشاره‌گرهایی به رکوردهای جدول پایه هستند. لذا عمل جستجو وقتی تمام می‌شود که کلید مورد جستجو در برگ‌ها پیدا شود. به لیست پیوندی برگ‌ها مجموعه توالی یا مجموعه دنباله می‌گویند.

توجه: در ساختار B^+-Tree به دلیل وجود چندین کلید جستجو در هر گره، رشد تعداد سطوح به کندی انجام می‌شود. به عنوان مثال جهت شاخص‌گذاری روی اطلاعات چندین میلیون رکورد از یک جدول پایه و درج و نگهداری مقادیر شاخص در ساختار B^+-Tree، سطح میانی تنها دارای دو یا سه سطح است و به ندرت یک B^+-Tree با شش سطح میانی ایجاد می‌شود. در واقع با توجه به وابستگی هزینه عملیات جستجو به تعداد سطوح، کم بودن تعداد سطوح از نقاط قوت ساختار B^+-Tree محسوب می‌شود.

عمل درج در ساختار B^+ Tree

مثال: درج رشته زیر را در نظر بگیرید:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

توجه: فرض کنید درجه t=5 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 5-1=4 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	4	
Number of Pointers	5	

قوانین درج در درخت B^+ Tree به صورت زیر است:

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node
NO	YES	**Index node** **Split into two nodes** 1- key < middle key go to the left index node 2- key >= middle key go to the right leaf node 3-**move** the smallest key of right leaf node to the parent node

توجه: اگر تعداد مجموعه کلیدهای جستجو **فرد** بود آنگاه عدد وسط برابر middle key خواهد بود. اما اگر تعداد مجموعه کلیدهای جستجو **زوج** بود آنگاه کلیدهای جستجو به طور مساوی در در دو گره سمت چپ و راست قرار می‌گیرند. و middle key کوچکترین عضو مجموعه گره سمت راست خواهد بود.

ابتدا رکورد اول با کلید 5 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

5			

سپس رکوردهای بعدی با کلیدهای 10، 25 و 30 به ترتیب اما به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

5	10	25	30

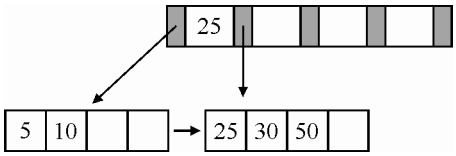
در ادامه رکورد بعدی با کلید 50 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 50 چون گره مورد نظر پر است (5,10,25,30,50) عمل تقسیم گره صورت می‌گیرد:

Leaf node Full	Index node Full	**Action**
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node

left leaf node	**right leaf node**
(5,10)	(25,30,50)
leaf node **Split into two nodes** 1- keys < middle key go to the left leaf node 2- key >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (50 ,30, **25**) یعنی عدد 25 به گره بالاتر فرستاده (**Copy**) می‌شود.	
(**25**)	
(5,10)	(**25**,30,50)



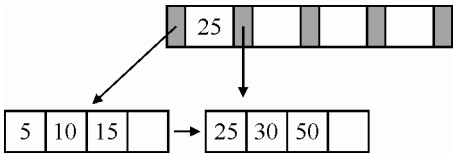
در ادامه رکورد بعدی با کلید 15 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 15 چون گره مورد نظر پر نیست (5,10,15) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	right leaf node
(5,10,15)	(25,30,50)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 15 به صورت مرتب شده درج و جایگذاری می‌شود.	



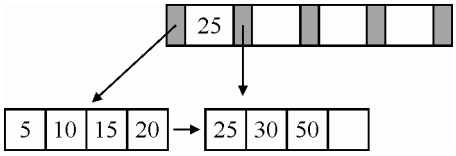
در ادامه رکورد بعدی با کلید 20 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 20 چون گره مورد نظر پر نیست (5,10,15,20) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	right leaf node
(5,10,15,20)	(25,30,50)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 20 به صورت مرتب شده درج و جایگذاری می‌شود.	



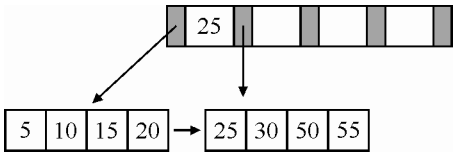
در ادامه رکورد بعدی با کلید 55 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 55 چون گره مورد نظر پر نیست (25,30,50,55) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	right leaf node
(5,10,15,20)	(25,30,50,55)
Place the record in sorted position in the appropriate leaf node توجه: فقط رکورد بعدی با کلید 55 به صورت مرتب شده درج و جایگذاری می‌شود.	



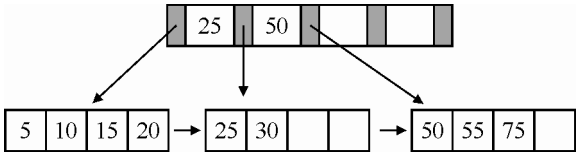
در ادامه رکورد بعدی با کلید 75 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 75 چون گره مورد نظر پر است (25,30,50,55,75) عمل تقسیم گره صورت می‌گیرد:

Leaf node Full	Index node Full	**Action**
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node

left leaf node	right leaf node
(25,30)	(50,55,75)
leaf node **Split into two nodes** 1- keys < middle key go to the left leaf node 2- key >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (**50**,55,75) یعنی عدد 50 به گره بالاتر فرستاده (**Copy**) می‌شود.	
(**50**)	
(25,30)	(**50**,55,75)



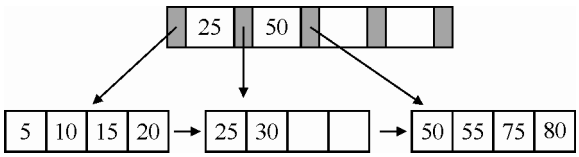
در ادامه رکورد بعدی با کلید 80 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 80 چون گره مورد نظر پر نیست (50,55,75,80) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	**right leaf node**
(25,30)	(50,55,75,80)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 80 به صورت مرتب شده درج و جایگذاری می‌شود.	



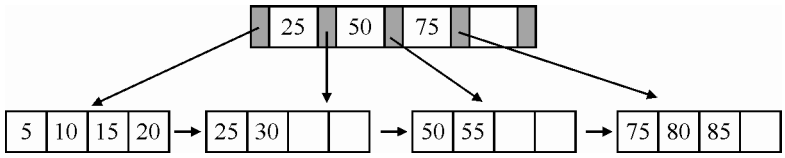
در ادامه رکورد بعدی با کلید 85 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 85 چون گره مورد نظر پر است (50,55,75,80,85) عمل تقسیم گره صورت می‌گیرد:

Leaf node Full	Index node Full	**Action**
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node

left leaf node	right leaf node
(50,55)	(75,80,85)
leaf node **Split into two nodes** 1- keys < middle key go to the left leaf node 2- key >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (**75**,80,85) یعنی عدد 75 به گره بالاتر فرستاده (**Copy**) می‌شود.	
(**75**)	
(50,55)	(**75**,80,85)



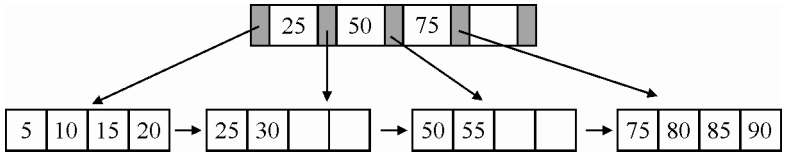
در ادامه رکورد بعدی با کلید 90 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 90 چون گره مورد نظر پر نیست (75,80,85,90) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	right leaf node
(50,55)	(75,80,85,90)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 90 به صورت مرتب شده درج و جایگذاری می‌شود.	



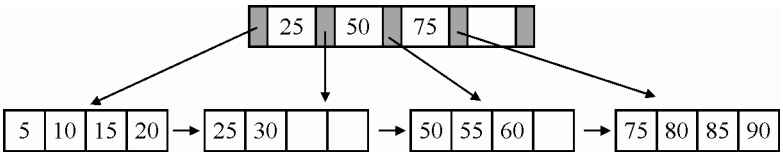
در ادامه رکورد بعدی با کلید 60 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 60 چون گره مورد نظر پر نیست (50,55,60) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	**right leaf node**
(50,55,60)	(75,80,85,90)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 60 به صورت مرتب شده درج و جایگذاری می‌شود.	



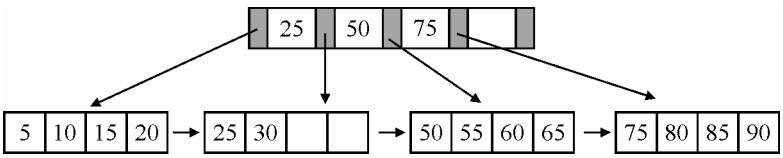
در ادامه رکورد بعدی با کلید 65 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 65 چون گره مورد نظر پر نیست (50,55,60,65) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	**right leaf node**
(50,55,60,65)	(75,80,85,90)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 65 به صورت مرتب شده درج و جایگذاری می‌شود.	



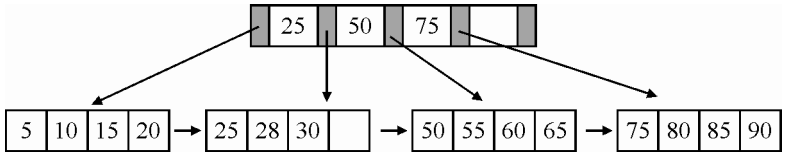
در ادامه رکورد بعدی با کلید 28 به صورت مرتب شده درج و جایگذاری می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 28 چون گره مورد نظر پر نیست (25,28,30) عمل تقسیم گره صورت نمی‌گیرد.

Leaf node Full	Index node Full	**Action**
NO	NO	Place the record in sorted position in the appropriate leaf node

left leaf node	**right leaf node**
(5,10,15,20)	(25,28,30)
Place the record in sorted position in the appropriate leaf node **توجه:** فقط رکورد بعدی با کلید 28 به صورت مرتب شده درج و جایگذاری می‌شود.	



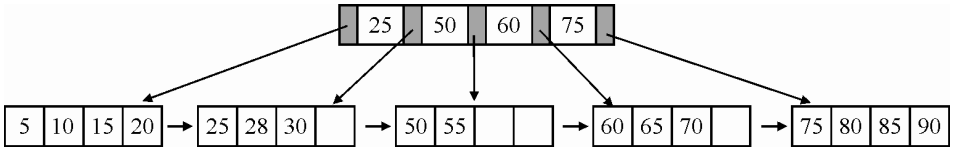
در ادامه رکورد بعدی با کلید 70 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 70 چون گره مورد نظر پر است (50,55,60,65,70) عمل تقسیم گره صورت می‌گیرد:

Leaf node Full	Index node Full	**Action**
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node

left leaf node	right leaf node
(50,55)	**(60,65,70)**
leaf node **Split into two nodes** 1- keys < middle key go to the left leaf node 2- key >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (**60**,65,70) یعنی عدد 60 به گره بالاتر فرستاده (**Copy**) می‌شود.	
(60)	
(50,55)	(**60**,65,70)



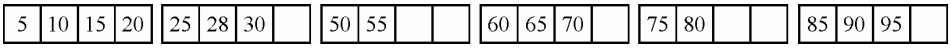
در ادامه رکورد بعدی با کلید 95 درج می‌شود:

5,10,25,30,50,15,20,55,75,80,85,90,60,65,28,70,95

با درج کلید 95 چون گره مورد نظر پر است (75,80,85,90,95) عمل تقسیم گره صورت می‌گیرد:

Leaf node Full	Index node Full	**Action**
YES	NO	**leaf node** **Split into two nodes** 1- keys < middle key go to the left leaf node 2- keys >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node

left leaf node	right leaf node
(75,80)	**(85,90,95)**
leaf node **Split into two nodes** 1- keys < middle key go to the left leaf node 2- key >= middle key go to the right leaf node 3-**copy** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (**85**,90,95) یعنی عدد 85 به گره بالاتر فرستاده (**Copy**) می‌شود.	
(**85**)	
(75,80)	(**85**,90,95)

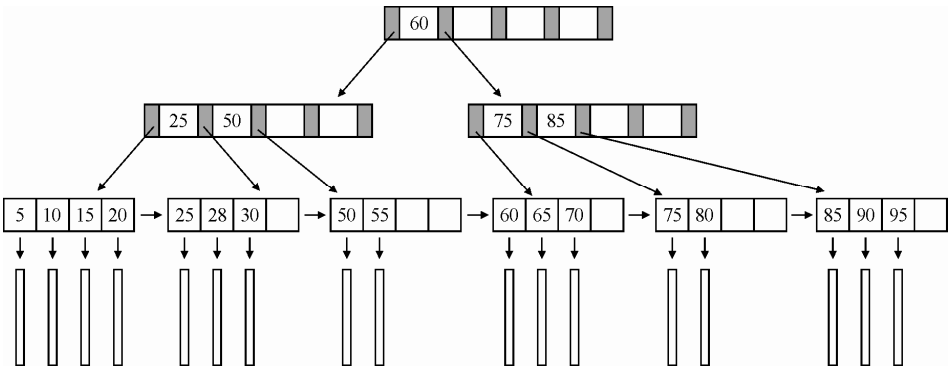


در ادامه کلید 85 در سطح بالاتر یعنی ریشه درج می‌شود.

با درج کلید85 چون گره ریشه پر است (25,50,60,75,85) عمل تقسیم گره ریشه صورت می‌گیرد:

Leaf node Full	Index node Full	Action
NO	YES	**Index node** **Split into two nodes** 1- key < middle key go to the left index node 2- key >= middle key go to the right leaf node 3-**move** the smallest key of right leaf node to the parent node

left leaf node	right leaf node
(25,50)	**(60,75,85)**
Index node **Split into two nodes** 1- key < middle key go to the left index node 2- key >= middle key go to the right leaf node 3-**move** the smallest key of right leaf node to the parent node **توجه:** کوچکترین عدد در right leaf node یعنی مجموعه اعداد (**60**,75,85) یعنی عدد 60 به گره بالاتر فرستاده (**Move**) می‌شود، به کلمه‌ی **Move** و تفاوت آن با **Copy** در حالت‌های قبل دقت کنید.	
(**60**)	
(25,50)	(75,85)



به طور کلی دو نوع پرس و جو در جداول پایگاه داده انجام می‌گردد:
(۱) پرس و جوی نقطه‌ای و (۲) پرس و جوی بازه‌ای که در ادامه به بررسی آن می‌پردازیم.

۱- پرس و جوی نقطه‌ای(Point query یا Equality query)

در پرس و جوی نقطه‌ای، هدف شناسایی و بازیابی سطرهایی است که مقدار ستون مورد جستجو در **یک مقدار** مشخص قرار دارد.

با اجرای قطعه کد زیر جدول S با کلید اصلی S# و به تبع یک clustered Index روی ستون S# ایجاد می‌گردد.

```
CREATE TABLE S (
    S# int NOT NULL,
    SName varchar(255),
    City varchar(255),
    PRIMARY KEY (S#)
);
```

قطعه کد ایجاد clustered Index که بطور خودکار اجرا می‌شود، به صورت زیر است:

```
CREATE UNIQUE CLUSTERED INDEX PK_S
ON S (S# ASC);
```

جداول زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

توجه: با استفاده از clustered Index پیش فرض و موجود روی ستون #S، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس #S در جدول S انجام می‌دهند، افزایش می‌یابد.
توجه: اما در مورد پرس و جوهایی که جستجو را بر اساس ستون #S انجام نمی‌دهند. تعریف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.
مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

S1,S2,S3,S4,S5,S6,S7,S8,S9

پرس و جوی زیر را در نظر بگیرید:

```
SELECT S#
FROM S
WHERE S#='S9'
```

توجه: شرط جلوی where یعنی S#='S9' **یک مقدار** مشخص است و این یعنی **پرس و جوی نقطه‌ای**. خروجی پرس و جوی فوق به صورت زیر است:

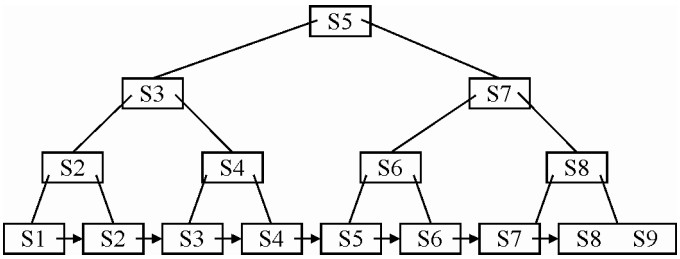
S#
S9

پیاده‌سازی شاخص فوق با ساختار B^+ Tree

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.
توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 3-1=2 است.
ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree به صورت زیر است:



نتیجه: رکورد S9 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(\text{Log}_t\, n)$است.

توجه: به طور کلی اگر n تعداد حداکثر سطرها (رکوردها) باشد و t درجه درخت و h تعداد سطوح در ساختار $B^+ - \text{Tree}$ باشد، آنگاه روابط زیر برقرار است:

For a t-order B+ tree with h levels of index

-The minimum number of records stored is: $n_{min} = 2\left\lceil \frac{t}{2} \right\rceil^{h-1} - 2\left\lceil \frac{t}{2} \right\rceil^{h-2}$

-The maximum number of records stored is: $n_{max} = t^h - t^{h-1}$

-The minimum number of keys is: $n_{key-min} = 2\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1$

-The maximum number of keys is: $n_{max-key} = t^h - 1$

-The space required to store the tree is: $O(n)$

-Inserting a record requires: $O(\text{Log}_t\, n)$

-Performing a **point query**: $O(\text{Log}_t\, n)$

-Performing a **range query** with k elements: $O(\text{Log}_t\, n + k)$

نتیجه: جستجو و بازیابی رکورد مورد نظر یعنی **پرس و جوی نقطه‌ای (Point query یا Equality query)** در ساختار B^+ Tree از مرتبه $O(\text{Log}_t\, n)$ است. و پیدا کردن رکورد بعدی در ساختار B^+ Tree از مرتبه $o(1)$ است. همچنین جستجو و بازیابی رکوردهای مورد نظر بازه‌ای یعنی **پرس و جوی بازه‌ای (Range query)** در ساختار B^+ Tree از مرتبه $O(\text{Log}_t\, n + k)$ است. پرس و جوی بازه‌ای جلوتر شرح داد می‌شود.

توجه: همانطور که گفتیم اصطلاحا $B^+ - \text{Tree}$ را درجه t می‌گویند $(t \geq 3)$که داخل هر گره غیر ریشه‌اش حداقل $\left\lceil \frac{t}{2} \right\rceil - 1$ و حداکثر $t-1$ کلید جستجو دارد. همچنین تعداد فرزندان (اشاره‌گرها) یکی از تعداد کلیدها بیشتر است، بنابراین هر گره غیربرگ و غیرریشه‌اش حداقل $\left\lceil \frac{t}{2} \right\rceil$ فرزند و حداکثر t فرزند دارد. ریشه استثنا است و می‌تواند حداقل شامل یک کلید جستجو باشد، یعنی حداقل دو فرزند داشته باشد، اما ریشه حداکثر همان $t-1$ کلید جستجو را می‌تواند داشته باشد، یعنی حداکثر t فرزند. در یک قاعده‌ی کلی گره‌ای که xتا کلید داشته باشد، اگر برگ نباشد، دقیقا $x+1$ فرزند دارد.

نتیجه: $t-1 \leq$ تعداد کلید در $B^+ - \text{Tree}$ درجه t در گره ریشه ≤ 1

نتیجه: $t \le$ **تعداد فرزند در** B^+-Tree **درجه t در گره ریشه** ≤ 2

نتیجه: $t-1 \le$ **تعداد کلید در** B^+-Tree **درجه t در گره غیرریشه** $\le \left\lceil \frac{t}{2} \right\rceil -1$

نتیجه: $t \le$ **تعداد فرزند در** B^+-Tree **درجه t در گره غیرریشه و غیربرگ** $\le \left\lceil \frac{t}{2} \right\rceil$

توجه: اگر یک B^+-Tree دارای h سطح (Level) باشد، پس ارتفاع (عمق) آن h-1 است.

توجه: به طور کلی اگر t درجه درخت (فاکتور انشعاب) و h تعداد سطوح در ساختار B^+-Tree باشد، آنگاه روابط زیر برقرار است:

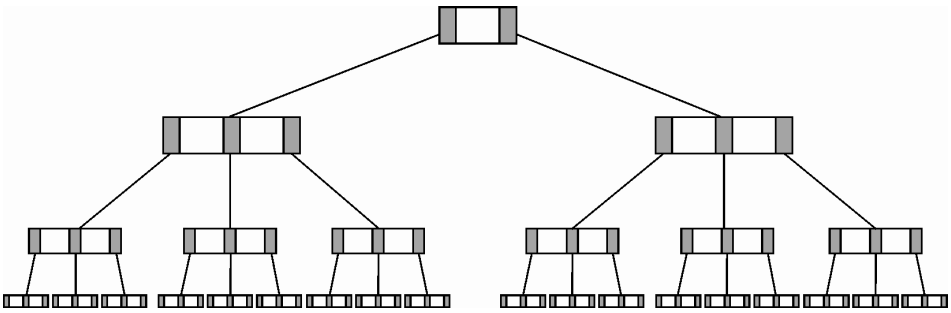
حداقل تعداد گره‌ها در یک درخت

توجه: درحالت کلی در محاسبه حداقل تعداد گره‌ها در یک درخت، گره ریشه **همواره** حداقل یک کلید و حداقل دو فرزند دارد.

توجه: درحالت کلی در محاسبه حداقل تعداد گره‌ها در یک درخت، گره غیرریشه **همواره** حداقل $\left\lceil \frac{t}{2} \right\rceil -1$ کلید و گره غیرریشه و غیربرگ حداقل $\left\lceil \frac{t}{2} \right\rceil$ فرزند دارد.

نتیجه: در یک B^+-Tree در حالت درخت **حداقل**، سطح اول که مربوط به ریشه است همواره یک گره و یک کلید دارد، و سطح دوم همواره دو گره و $2\times\left(\left\lceil \frac{t}{2} \right\rceil -1\right)$ کلید دارد. بنابراین در محاسبه حداقل تعداد گره‌ها، ابتدا گره ریشه را کنار می‌گذاریم، سپس تعداد گره‌های زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه می‌کنیم و حاصل را با یک گره ریشه جمع می‌کنیم.

مثال: برای یک درخت دارای h=4 سطح و ارتفاع (عمق) h-1=3 و فاکتور انشعاب t=5 روابط زیر در حالت حداقلی برقرار است.



توجه: حداقل تعداد گره‌های زیر درخت چپ که ریشه را ندارد به صورت زیر محاسبه می‌شود:

$$\text{node}^{\min}_{\text{Total}}(h:\text{Left}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 = \frac{\left\lceil \frac{t}{2} \right\rceil^3 - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} =$$

$$\text{node}^{\min}_{\text{Total}}(h:\text{Left}) = \left\lceil \frac{5}{2} \right\rceil^0 + \left\lceil \frac{5}{2} \right\rceil^1 + \left\lceil \frac{5}{2} \right\rceil^2 = \frac{\left\lceil \frac{5}{2} \right\rceil^3 - 1}{\left\lceil \frac{5}{2} \right\rceil - 1} =$$

$$\text{node}^{\min}_{\text{Total}}(h:\text{Left}) = 3^0 + 3^1 + 3^2 = \frac{3^3 - 1}{3 - 1} =$$

$$\text{node}^{\min}_{\text{Total}}(h:\text{Left}) = 1 + 3 + 9 = \frac{27 - 1}{2} = \frac{26}{2} = 13$$

نتیجه: بنابراین در حالت کلی حداقل تعداد گره‌ها در زیر درخت چپ که ریشه را ندارد از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$\text{node}^{\min}_{\text{Total}}(h:\text{Left}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 + \ldots + \left\lceil \frac{t}{2} \right\rceil^{h-2} = \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1}$$

نتیجه: همچنین در حالت کلی حداقل تعداد گره‌ها در زیر درخت راست که ریشه را ندارد از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$\text{node}^{\min}_{\text{Total}}(h:\text{Right}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 + \ldots + \left\lceil \frac{t}{2} \right\rceil^{h-2} = \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1}$$

نتیجه: همانطور که گفتیم در حالت کلی در محاسبه حداقل تعداد گره‌ها در یک درخت، ابتدا گره ریشه را کنار می‌گذاریم، سپس تعداد گره‌های زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه می‌کنیم و حاصل را با یک گره ریشه جمع می‌کنیم، بنابراین رابطه‌ی زیر برقرار است:

$$\text{node}^{\min}_{\text{Total}}(h) = \text{node}^{\min}_{\text{Total}}(h:\text{Left}) + \text{node}^{\min}_{\text{Total}}(h:\text{Right}) + \text{node}^{\min}_{\text{Total}}(h:\text{Root})$$

$$node_{Total}^{min}(h) = \left(\frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} \right) + \left(\frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} \right) + 1 =$$

$$node_{Total}^{min}(h) = 2 \times \left(\frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} \right) + 1 =$$

$$node_{Total}^{min}(h) = 2 \times \left(\frac{\left\lceil \frac{5}{2} \right\rceil^{4-1} - 1}{\left\lceil \frac{5}{2} \right\rceil - 1} \right) + 1 = 2 \times \left(\frac{3^3 - 1}{3 - 1} \right) + 1 = 2 \times \left(\frac{27 - 1}{2} \right) + 1 = 2 \times \left(\frac{26}{2} \right) = 2 \times 13 + 1 = 27$$

حداقل تعداد گره‌ها (برگ‌ها) در سطح آخر یک درخت

$$node_{Total}^{min}(h : last\ level) = \left(2 \times \left(\frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} \right) + 1 \right) - \left(2 \times \left(\frac{\left\lceil \frac{t}{2} \right\rceil^{h-2} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} \right) + 1 \right) =$$

$$node_{Total}^{min}(h : last\ level) = \left(2 \times \left(\frac{\left\lceil \frac{5}{2} \right\rceil^{4-1} - 1}{\left\lceil \frac{5}{2} \right\rceil - 1} \right) + 1 \right) - \left(2 \times \left(\frac{\left\lceil \frac{5}{2} \right\rceil^{4-2} - 1}{\left\lceil \frac{5}{2} \right\rceil - 1} \right) + 1 \right) =$$

$$node_{Total}^{min}(h : last\ level) = \left(2 \times \left(\frac{3^3 - 1}{3 - 1} \right) + 1 \right) - \left(2 \times \left(\frac{3^2 - 1}{3 - 1} \right) + 1 \right) =$$

$$node_{Total}^{min}(h : last\ level) = \left(2 \times \left(\frac{27 - 1}{2} \right) + 1 \right) - \left(2 \times \left(\frac{9 - 1}{2} \right) + 1 \right) =$$

$$node_{Total}^{min}(h : last\ level) = \left(2 \times \left(\frac{26}{2} \right) + 1 \right) - \left(2 \times \left(\frac{8}{2} \right) + 1 \right) =$$

$$node_{Total}^{min}(h : last\ level) = (2 \times 13 + 1) - (2 \times 4 + 1) = 27 - 9 = 18$$

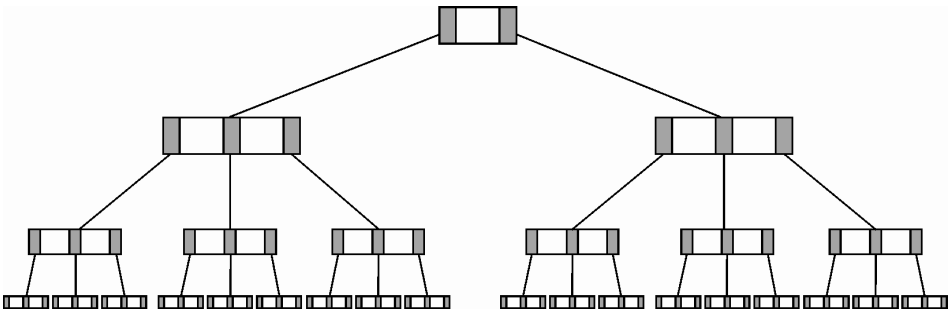
حداقل تعداد کلیدها در یک درخت

توجه: درحالت کلی در محاسبه حداقل تعداد کلیدها در یک درخت، گره ریشه **همواره** حداقل یک کلید و حداقل دو فرزند دارد.

توجه: درحالت کلی در محاسبه حداقل تعداد کلیدها در یک درخت، گره غیرریشه **همواره** حداقل $\left\lceil \frac{t}{2} \right\rceil - 1$ کلید و گره غیرریشه و غیربرگ حداقل $\left\lceil \frac{t}{2} \right\rceil$ فرزند دارد.

نتیجه: در یک $B^+ - \text{Tree}$ در حالت درخت **حداقل**، سطح اول که مربوط به ریشه است همواره یک گره و یک کلید دارد، و سطح دوم همواره دو گره و $2\times\left(\left\lceil \frac{t}{2} \right\rceil - 1\right)$ کلید دارد. بنابراین در محاسبه حداقل تعداد کلیدها، ابتدا گره ریشه را کنار می‌گذاریم، سپس تعداد کلیدهای زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه می‌کنیم و حاصل را با یک کلید گره ریشه جمع می‌کنیم.

مثال: برای یک درخت دارای h=4 سطح و ارتفاع (عمق) h-1=3 و فاکتور انشعاب t=5 روابط زیر در حالت حداقلی برقرار است.



توجه: حداقل تعداد گره‌های زیر درخت چپ که ریشه را ندارد به صورت زیر محاسبه می‌شود:

$$\text{node}_{\text{Total}}^{\min}(h:\text{Left}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 = \frac{\left\lceil \frac{t}{2} \right\rceil^3 - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} =$$

$$\text{node}_{\text{Total}}^{\min}(h:\text{Left}) = \left\lceil \frac{5}{2} \right\rceil^0 + \left\lceil \frac{5}{2} \right\rceil^1 + \left\lceil \frac{5}{2} \right\rceil^2 = \frac{\left\lceil \frac{5}{2} \right\rceil^3 - 1}{\left\lceil \frac{5}{2} \right\rceil - 1} =$$

$$\text{node}_{\text{Total}}^{\min}(h:\text{Left}) = 3^0 + 3^1 + 3^2 = \frac{3^3 - 1}{3 - 1} =$$

$$\text{node}_{\text{Total}}^{\min}(h:\text{Left}) = 1 + 3 + 9 = \frac{27 - 1}{2} = \frac{26}{2} = 13$$

نتیجه: بنابراین در حالت کلی حداقل تعداد گره‌ها در زیر درخت چپ که ریشه را ندارد از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$\text{node}_{\text{Total}}^{\min}(h:\text{Left}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 + \ldots + \left\lceil \frac{t}{2} \right\rceil^{h-2} = \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1}$$

نتیجه: همچنین در حالت کلی حداقل تعداد گره‌ها در زیر درخت راست که ریشه را ندارد از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$\text{node}_{\text{Total}}^{\min}(h:\text{Right}) = \left\lceil \frac{t}{2} \right\rceil^0 + \left\lceil \frac{t}{2} \right\rceil^1 + \left\lceil \frac{t}{2} \right\rceil^2 + \ldots + \left\lceil \frac{t}{2} \right\rceil^{h-2} = \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1}$$

توجه: وقتی درجه‌ی (فاکتور انشعاب) یک گره غیرریشه حداقل $\left\lceil \frac{t}{2} \right\rceil$ است، پس حداقل تعداد کلیدهای ذخیره شده در آن، $\left\lceil \frac{t}{2} \right\rceil - 1$ است.

نتیجه: حداقل تعداد کلیدها در زیردرخت چپ که ریشه را ندارد از رابطه‌ی زیر بدست می‌آید:

$$\text{keys}_{\text{Total}}^{\min}(h:\text{Left}) = \left(\left\lceil \frac{t}{2} \right\rceil - 1 \right) \times \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} = \left\lceil \frac{t}{2} \right\rceil^{h-1} - 1$$

نتیجه: حداقل تعداد کلیدها در زیردرخت راست که ریشه را ندارد از رابطه‌ی زیر بدست می‌آید:

$$\text{keys}_{\text{Total}}^{\min}(h:\text{Right}) = \left(\left\lceil \frac{t}{2} \right\rceil - 1 \right) \times \frac{\left\lceil \frac{t}{2} \right\rceil^{h-1} - 1}{\left\lceil \frac{t}{2} \right\rceil - 1} = \left\lceil \frac{t}{2} \right\rceil^{h-1} - 1$$

نتیجه: همانطور که گفتیم در حالت کلی در محاسبه حداقل تعداد کلیدها در یک درخت، ابتدا گره ریشه را کنار می‌گذاریم، سپس تعداد کلیدهای زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه می‌کنیم و حاصل را با یک کلید گره ریشه جمع می‌کنیم، بنابراین رابطه‌ی زیر

برقرار است:

$$keys_{Total}^{min}(h) = keys_{Total}^{min}(h:Left) + keys_{Total}^{min}(h:Right) + keys_{Total}^{min}(h:Root)$$

$$keys_{Total}^{min}(h) = \left(\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1\right) + \left(\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1\right) + 1$$

$$keys_{Total}^{min}(h) = 2\times\left(\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1\right) + 1 = 2\times\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1$$

$$keys_{Total}^{min}(h) = 2\times\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1 = 2\times\left\lceil\frac{5}{2}\right\rceil^{4-1} - 1 = 2\times 3^3 - 1 = 2\times 27 - 1 = 54 - 1 = 53$$

حداقل تعداد کلیدها در سطح آخر یک درخت

$$keys_{Total}^{min}(last\ level:h) = \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-1} - 1\right) - \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-2} - 1\right)$$

$$keys_{Total}^{min}(last\ level:h) = \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-1}\right) - 1 - \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-2}\right) + 1$$

$$keys_{Total}^{min}(last\ level:h) = \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-1}\right) - \left(2\times\left\lceil\frac{t}{2}\right\rceil^{h-2}\right)$$

$$keys_{Total}^{min}(last\ level:h) = \left(2\times\left\lceil\frac{5}{2}\right\rceil^{4-1}\right) - \left(2\times\left\lceil\frac{5}{2}\right\rceil^{4-2}\right)$$

$$keys_{Total}^{min}(last\ level:h) = \left(2\times 3^3\right) - \left(2\times 3^2\right)$$

$$keys_{Total}^{min}(last\ level:h) = (2\times 27) - (2\times 9) = 54 - 18 = 36$$

حداکثر تعداد گره‌ها در یک درخت

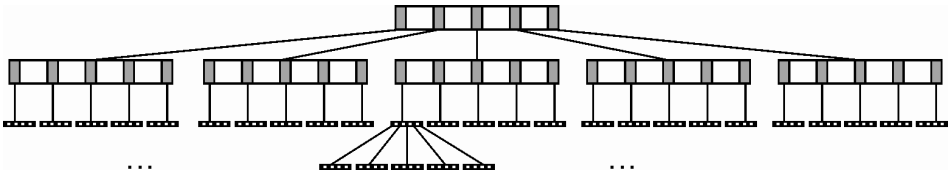
توجه: درحالت کلی در محاسبه حداکثر تعداد گره‌ها در یک درخت، گره ریشه **همواره** حداکثر $t-1$ کلید و حداکثر t فرزند دارد.

توجه: درحالت کلی در محاسبه حداکثر تعداد گره‌ها در یک درخت، گره غیرریشه **همواره** حداکثر $t-1$ کلید و گره غیرریشه و غیربرگ حداکثر t فرزند دارد.

نتیجه: در یک B^+-Tree در حالت درخت **حداکثر**، سطح اول که مربوط به ریشه است همواره

یک گره و $t-1$ کلید دارد، و سطح دوم همواره t گره و $t\times(t-1)$ کلید دارد. بنابراین در محاسبه حداکثر تعداد گره‌ها، دیگر نیاز نیست مانند درخت حداقل، گره ریشه را کنار بگذاریم، و سپس تعداد گره‌های زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه کنیم و حاصل را با یک گره ریشه جمع کنیم، بلکه کافیست از یک سری هندسی برای کل درخت استفاده کنیم.

مثال: برای یک درخت دارای h=4 سطح و ارتفاع (عمق) h-1=3 و فاکتور انشعاب t=5 روابط زیر در حالت حداکثری برقرار است.



توجه: حداکثر تعداد گره در هر سطح از رابطه‌ی حداکثر فاکتور انشعاب به توان عمق به دست می‌آید، به صورت زیر:

$$node^{max}_{Level}(i) = t^i$$

حداکثر تعداد گره در عمق صفر(سطح یک) به صورت زیر محاسبه می‌شود:

$$node^{max}_{Level=1}(i=0) = t^i = 5^0 = 1$$

حداکثر تعداد گره در عمق یک (سطح دو) به صورت زیر محاسبه می‌شود:

$$node^{max}_{Level=2}(i=1) = t^i = 5^1 = 5$$

حداکثر تعداد گره در عمق دو (سطح سه) به صورت زیر محاسبه می‌شود:

$$node^{max}_{Level=3}(i=2) = t^i = 5^2 = 25$$

حداکثر تعداد گره در عمق سه (سطح چهار) به صورت زیر محاسبه می‌شود:

$$node^{max}_{Level=4}(i=3) = t^i = 5^3 = 125$$

بنابراین حداکثر تعداد گره‌ها در درخت فوق به صورت زیر بدست می‌آید:

$$node^{max}_{Total}(h) = 5^0 + 5^1 + 5^2 + 5^3 = 1+5+25+125 = \frac{5^4-1}{5-1} = \frac{625-1}{5-1} = \frac{624}{4} = 156$$

نتیجه: بنابراین در حالت کلی حداکثر تعداد گره‌ها در یک درخت از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$node^{max}_{Total}(h) = t^0 + t^1 + t^2 + t^3 + ... + t^{h-1} = \frac{t^h - 1}{t-1}$$

حداکثر تعداد گره‌ها (برگ‌ها) در سطح آخر یک درخت

$$\text{node}^{\max}_{\text{Total}}(\text{h : last level}) = \left(\frac{t^{h}-1}{t-1}\right) - \left(\frac{t^{h-1}-1}{t-1}\right) =$$

$$\text{node}^{\max}_{\text{Total}}(\text{h : last level}) = \left(\frac{5^{4}-1}{5-1}\right) - \left(\frac{5^{4-1}-1}{5-1}\right) =$$

$$\text{node}^{\max}_{\text{Total}}(\text{h : last level}) = \left(\frac{625-1}{5-1}\right) - \left(\frac{125-1}{5-1}\right) =$$

$$\text{node}^{\max}_{\text{Total}}(\text{h : last level}) = \left(\frac{624}{4}\right) - \left(\frac{124}{4}\right) =$$

$$\text{node}^{\max}_{\text{Total}}(\text{h : last level}) = (156) - (31) = 125$$

همچنین حداکثر تعداد گره در عمق سه (سطح چهار) یعنی حداکثر تعداد گره‌ها (برگ‌ها) در سطح آخر یک درخت به صورت زیر محاسبه می‌شود:

$$\text{node}^{\max}_{\text{Level=4}}(i = 3) = t^{i} = 5^{3} = 125$$

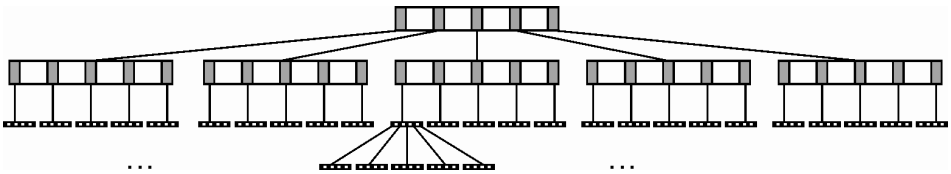
حداکثر تعداد کلیدها در یک درخت

توجه: درحالت کلی در محاسبه حداکثر تعداد کلیدها در یک درخت، گره ریشه **همواره** حداکثر $t-1$ کلید و حداکثر t فرزند دارد.

توجه: درحالت کلی در محاسبه حداکثر تعداد کلیدها در یک درخت، گره غیرریشه **همواره** حداکثر $t-1$ کلید و گره غیرریشه و غیربرگ حداکثر t فرزند دارد.

نتیجه: در یک $B^{+}-\text{Tree}$ در حالت درخت **حداکثر**، سطح اول که مربوط به ریشه است همواره یک گره و $t-1$ کلید دارد، و سطح دوم همواره t گره و $t\times(t-1)$ کلید دارد. بنابراین در محاسبه حداکثر تعداد کلیدها، دیگر نیاز نیست مانند درخت حداقل گره ریشه را کنار بگذاریم، و سپس تعداد کلیدهای زیر درخت چپ و زیر درخت راست را به طور جداگانه محاسبه کنیم و حاصل را با یک کلید گره ریشه جمع کنیم، بلکه کافیست از یک سری هندسی برای کل درخت استفاده کنیم.

مثال: برای یک درخت دارای h=4 سطح و ارتفاع (عمق) h-1=3 و فاکتور انشعاب t=5 روابط زیر در حالت حداکثری برقرار است.



نتیجه: در حالت کلی حداکثر تعداد گره‌ها در یک درخت از رابطه‌ی زیر به صورت یک سری هندسی بدست می‌آید:

$$node_{Total}^{max}(h) = t^0 + t^1 + t^2 + t^3 + ... + t^{h-1} = \frac{t^h - 1}{t - 1}$$

توجه: وقتی درجه‌ی (فاکتور انشعاب) یک گره ریشه و غیرریشه حداکثر t است، پس حداکثر تعداد کلیدهای ذخیره شده در آن، $t-1$ است.

توجه: حداکثر تعداد کلیدها در کل درخت از رابطه‌ی زیر بدست می‌آید:

$$keys_{Total}^{max}(h) = (t-1) \times \frac{t^h - 1}{t - 1} = t^h - 1$$

$$keys_{Total}^{max}(h) = (5-1) \times \frac{5^4 - 1}{5 - 1} = 5^4 - 1$$

$$keys_{Total}^{max}(h) = 4 \times \frac{625 - 1}{5 - 1} = 625 - 1$$

$$keys_{Total}^{max}(h) = 4 \times \frac{624}{4} = 624$$

$$keys_{Total}^{max}(h) = 4 \times 156 = 624$$

$$keys_{Total}^{max}(h) = 624$$

حداکثر تعداد کلیدها در سطح آخر یک درخت

$$keys_{Total}^{max}(\text{last level} : h) = \left(t^h - 1\right) - \left(t^{h-1} - 1\right)$$

$$keys_{Total}^{max}(\text{last level} : h) = t^h - 1 - t^{h-1} + 1$$

$$keys_{Total}^{max}(\text{last level} : h) = t^h - t^{h-1}$$

$$keys_{Total}^{max}(\text{last level} : h) = 5^4 - 5^{4-1}$$

$$keys_{Total}^{max}(\text{last level} : h) = 625 - 125 = 500$$

همچنین حداکثر تعداد کلید در عمق سه (سطح چهار) یعنی حداکثر تعداد گره‌ها (برگ‌ها) در سطح آخر یک درخت به صورت زیر محاسبه می‌شود:

$$keys^{max}_{Total}(\text{last level} : h) = (t-1)\times t^{h-1}$$
$$keys^{max}_{Total}(\text{last level} : h) = (5-1)\times 5^{4-1}$$
$$keys^{max}_{Total}(\text{last level} : h) = 4\times 5^{3}$$
$$keys^{max}_{Total}(\text{last level} : h) = 4\times 125 = 500$$

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

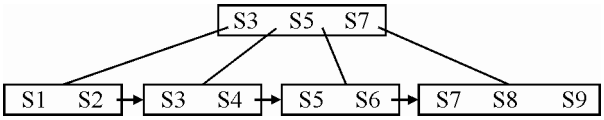
S1,S2,S3,S4,S5,S6,S7,S8,S9

توجه: فرض کنید درجه t=4 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 4-1=3 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=3	
Number of Pointers	t=4	

ساختار نهایی B^+ Tree **به صورت زیر است:**



مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

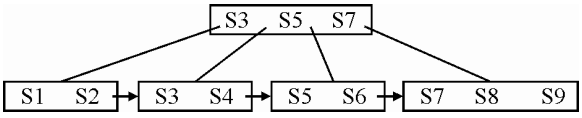
S1,S2,S3,S4,S5,S6,S7,S8,S9

توجه: فرض کنید درجه t=5 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 5-1=4 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=4	
Number of Pointers	t=5	

ساختار نهایی B^+ Tree **به صورت زیر است:**



نتیجه: دو رشته‌ی یکسان از نظر مقادیر در دو مثال قبل؛ و یکسان از نظر ترتیب درج و متفاوت از نظر درجه، دو شکل یکسان در ساختار، چیدمان و تعداد سطوح داشتند.

پیاده‌سازی شاخص فوق با ساختار آرایه

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C3
S6	Sn6	C4
S7	Sn7	C5
S8	Sn8	C6
S9	Sn9	C7

جدول S

S#
S1
S2
S3
S4
S5
S6
S7
S8
S9

Index

توجه: رکورد S9 پس از جستجو در ساختار آرایه و به صورت خطی در 9 حرکت در رکورد نهم یافت می‌شود، که مرتبه اجرایی آن $O(n)$ است.

توجه: از آن‌جا که **طول رکورد** در ساختار شاخص نسبت به جدول پایه کوچکتر است، واضح است که سرعت جستجو و بازیابی روی ساختار شاخص نسبت به جدول پایه بیش‌تر است. به بیان دیگر زمان اجرای جستجو و بازیابی کاهش می‌یابد. مطابق رابطه‌ی زیر:

$$\text{زمان انتقال رکورد } (T_f) = \frac{\text{طول رکورد } (L)}{\text{نرخ انتقال } (R)}$$

توجه: رابطه‌ی فوق در شاخص‌های با ساختار آرایه و ساختار B^+ Tree صادق است.

۲- پرس و جوی بازه‌ای(Range query)

در پرس و جوی بازه‌ای، هدف شناسایی و بازیابی سطرهایی است که مقدار ستون مورد جستجو در **یک بازه** مشخص قرار دارد.

مثال: درج رشته مقابل را بر اساس جدول S در نظر بگیرید: S1,S2,S3,S4,S5,S6,S7,S8,S9

پرس و جوی زیر را در نظر بگیرید:

```
SELECT S#
FROM S
WHERE S#>'S4' AND S#<='s9'
```

شرط جلوی where یعنی S#>'S4' AND S#<='s9' **یک بازه** مشخص است و این یعنی **پرس و جوی بازه‌ای**. خروجی پرس و جوی فوق به صورت زیر است:

S#
S5
S6
S7
S8
S9

توجه: رکوردهای S5، S6، S7، S8 و S9 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(Log_t\, n+k)$ است.

پیاده‌سازی شاخص فوق با ساختار B^+ Tree

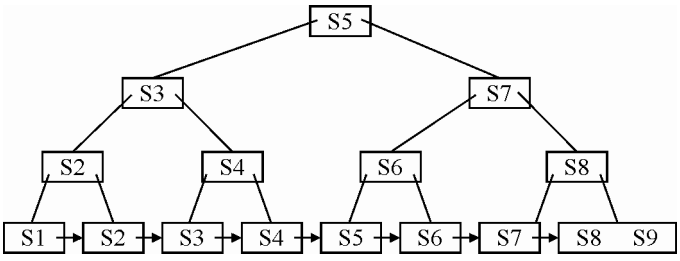
در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 3-1=2 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree به صورت زیر است:



نتیجه: رکوردهای S5، S6، S7، S8 و S9 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت و چند قدم در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن

$O(Log_t n + k)$ است.

پیاده‌سازی شاخص فوق با ساختار آرایه

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه بـه دلیـل مزایـای سـاختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C3
S6	Sn6	C4
S7	Sn7	C5
S8	Sn8	C6
S9	Sn9	C7

جدول S

S#
S1
S2
S3
S4
S5
S6
S7
S8
S9

Index

توجه: رکوردهای S5، S6، S7، S8 و S9 پس از جستجو در ساختار آرایه و به صورت خطـی در 9 حرکت در رکورد نهم یافت می‌شود، که مرتبه اجرایی آن $O(n)$ است.

شاخص مرتب‌شده به صورت Nonclustered Index

توجه: هر جدول می‌تواند حداکثر نهصد و نود و نه Nonclustered Index داشته باشد.

توجه: در Nonclustered Index ترتیب منطقی رکوردها در جدول پایه با ترتیب فیزیکـی چیـدمان آن‌ها بر روی هارد دیسک لزوما یکسان نیست والزامی هم برای آن وجود ندارد. و تنها یک مقـدار و اشاره‌گر به رکوردی که حاوی مقدار مورد نظر در جـدول پایـه اسـت در Nonclustered Index نگهداری می‌شود.

توجه: به شاخص مرتب‌شـده بـه صـورت Nonclustered Index، شـاخص فرعـی یـا ثانویـه نیـز می‌گویند.

توجه: مقادیر شاخص مرتب‌شده به صورت Nonclustered Index می‌تواند یکتا و منحصر به فـرد باشد اگر UNIQUE تعریف گردد و می‌تواند یکتا و منحصر به فرد نباشد اگـر UNIQUE تعریـف نگردد.

قطعه کد ایجاد Nonclustered Index به صورت زیر است:

```
CREATE [UNIQUE] [NONCLUSTERED] INDEX index_name
ON table_name (column1 [ASC | DESC], column2 [ASC | DESC], ...);
```

توجه: به طور پیش فرض و قابل تغییر در ایجاد Nonclustered Index دستور UNIQUE استفاده نمی‌گردد، که می‌شود **Non-Unique, Non-Clustered** در صورت نیاز به یکتایی و منحصر به فرد بودن مقادیر شاخص دستور UNIQUE استفاده می‌گردد، که می‌شود **Unique, Non-Clustered**.

توجه: به طور پیش فرض و قابل تغییر در ایجاد Nonclustered Index دستور Ascending یا ASC استفاده می‌گردد، تا **صعودی** بودن مقادیر شاخص **گارانتی** گردد. در صورت نیاز به **نزولی** بودن مقادیر شاخص دستور Descending یا DESC استفاده می‌گردد.

توجه: اگر پشت دستور INDEX نه دستور clustered و نه دستور Nonclustered باشد، حالت پیش فرض Nonclustered است.

توجه: اگر پشت دستور INDEX دستور Unique نباشد، حالت پیش فرض Non-Unique است که نوشته هم نمی‌شود.

مثال: با اجرای قطعه کد زیر جدول S با کلید اصلی #S و به تبع یک clustered Index روی ستون #S ایجاد می‌گردد.

```
CREATE TABLE S (
    S# int NOT NULL,
    SName varchar(255),
    City varchar(255),
    PRIMARY KEY (S#)
);
```

قطعه کد ایجاد clustered Index که بطور خودکار اجرا می‌شود، به صورت زیر است:

```
CREATE UNIQUE CLUSTERED INDEX PK_S
ON S (S# ASC);
```

جداول زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

با استفاده از clustered Index پیش فرض و موجود روی ستون #S، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس #S در جدول S انجام می‌دهند، افزایش می‌یابد.

مثال:

```
SELECT S#
FROM S
WHERE S#='S9'
```

اما در مورد پرس و جوهایی که جستجو را بر اساس ستون #S انجام نمی‌دهند. تعریف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

مثال: اجرای دستور زیر موجب ایجاد یک Nonclustered Index و Non-Unique به صورت **صعودی**، به نام CityX روی ستون City در جدول S می‌شود.

```
CREATE NONCLUSTERED INDEX CityX
ON S (City ASC);
```

با توجه به دستورات پیش فرض، قطعه کد زیر، معادل قطعه کد فوق است:

```
CREATE INDEX CityX
ON S (City);
```

با استفاده از Nonclustered Index، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس City در جدول S انجام می‌دهند، افزایش می‌یابد.

مثال:

```
SELECT S#
FROM S
WHERE City='C2'
```

اما در مورد پرس و جوهایی که جستجو را بر اساس ستون City انجام نمی‌دهند. تعریف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

مثال: اجرای دستور زیر موجب ایجاد یک Nonclustered Index و Unique به صورت **صعودی**، به نام SnameX روی ستون Sname در جدول S می‌شود.

```
CREATE UNIQUE NONCLUSTERED INDEX SnameX
ON S (Sname ASC);
```

با توجه به دستورات پیش فرض، قطعه کد زیر، معادل قطعه کد فوق است:

```
CREATE UNIQUE INDEX SnameX
ON S (Sname);
```

توجه: دستور فوق یعنی ایجاد شاخص یکتا توسط دستور Unique فقط در صورتی قابل اجرا توسط DBMS روی ستون مورد نظر است، که از قبل ستون مورد نظر مقادیر تکراری نداشته باشد، در غیر اینصورت یعنی وجود مقادیر تکراری در ستون مورد نظر، دستور فوق از سوی DBMS رد می‌شود. که این می‌شود پیشا اجرای دستور فوق، همچنین پس از ایجاد دستور فوق یعنی ایجاد شاخص یکتا بر روی ستون حائز شرایط ایجاد شاخص یکتا، یکتا بودن ستون مورد نظر، تا مادامی که شاخص یکتا وجود دارد، **گارانتی** می‌شود که این می‌شود پسااجرای دستور فوق.

با استفاده از Nonclustered Index، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس Sname در جدول S انجام می‌دهند، افزایش می‌یابد.

مثال:

```
SELECT S#
FROM S
WHERE Sname='Sn9'
```

اما در مورد پرس و جوهایی که جستجو را بـر اسـاس سـتون Sname انجـام نمـی‌دهنـد. تعریـف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

به طور کلی دو نوع پرس و جو در جداول پایگاه داده انجام می‌گردد:

(۱) پرس و جوی نقطه‌ای و (۲) پرس و جوی بازه‌ای که در ادامه به بررسی آن می‌پردازیم.

۱- پرس و جوی نقطه‌ای(Point query یا Equality query)

در پرس و جوی نقطه‌ای، هدف شناسایی و بازیابی سطرهایی است که مقدار ستون مورد جستجو در **یک مقدار** مشخص قرار دارد.

اجرای دستور زیر موجب ایجاد یک Nonclustered Index و Non-Unique به صورت **صـعودی**، به نام CityX روی ستون City در جدول S می‌شود.

```
CREATE NONCLUSTERED INDEX CityX
ON S (City ASC);
```

با توجه به دستورات پیش فرض، قطعه کد زیر، معادل قطعه کد فوق است:

```
CREATE INDEX CityX
ON S (City);
```

جداول زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

توجه: با استفاده از Nonclustered Index، سرعت پاسـخگویی بـه تمـامی پـرس و جوهـایی کـه جستجو را بر اساس City در جدول S انجام می‌دهند، افزایش می‌یابد.

توجه: اما در مورد پرس و جوهایی که جستجو را بر اساس ستون City انجام نمـی‌دهنـد. تعریـف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:(ascending یعنی صعودی)
C1,C2,C2,C3,C4,C5,C6,C7,C7

پرس و جوی زیر را در نظر بگیرید:

```
SELECT S#
FROM S
WHERE City='C7'
```

توجه: شرط جلوی where یعنی City='C7' **یک مقدار** مشخص است و این یعنی **پرس و جوی نقطه‌ای**. خروجی پرس و جوی فوق به صورت زیر است:

S#
S8
S9

پیاده‌سازی شاخص فوق با ساختار B^+ Tree

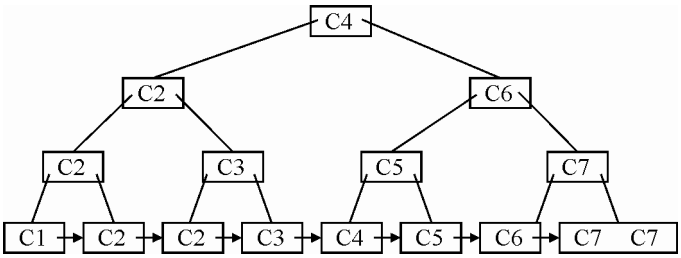
در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 3-1=2 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree **به صورت زیر است:**



نتیجه: رکورد C7 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(Log_t n)$ است.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید: (descending یعنی نزولی)

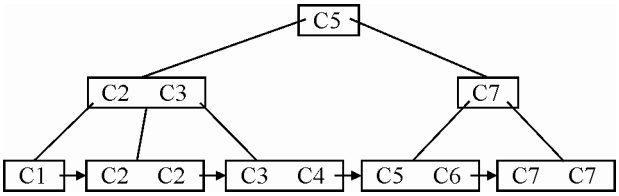
C7,C7,C6,C5,C4,C3,C2,C2,C1

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 3-1=2 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree **به صورت زیر است:**



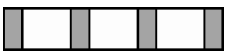
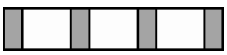
نتیجه: دو رشته‌ی یکسان از نظر مقادیر؛ و متفاوت از نظر ترتیب درج و یکسان از نظر درجه، دو شکل متفاوت در ساختار، چیدمان و تعداد سطوح داشتند.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

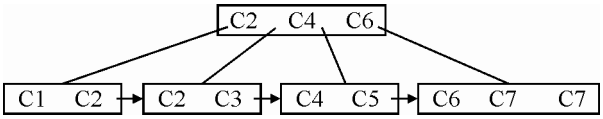
C1,C2,C2,C3,C4,C5,C6,C7,C7

توجه: فرض کنید درجه t=4 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 4-1=3 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=3	
Number of Pointers	t=4	

ساختار نهایی B^+ Tree **به صورت زیر است:**



مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

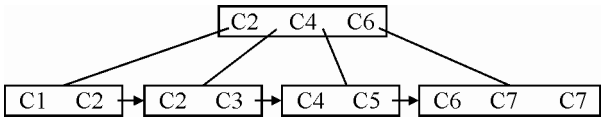
C1,C2,C2,C3,C4,C5,C6,C7,C7

توجه: فرض کنید درجه t=5 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 5-1=4 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=4	
Number of Pointers	t=5	

ساختار نهایی B^+ Tree به صورت زیر است:



نتیجه: دو رشته‌ی یکسان از نظر مقادیر در دو مثال قبل؛ و یکسان از نظر ترتیب درج و متفاوت از نظر درجه، دو شکل یکسان در ساختار، چیدمان و تعداد سطوح داشتند.

پیاده‌سازی شاخص فوق با ساختار آرایه

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

S#	Sname	City	City
S1	Sn1	C1	C1
S2	Sn2	C2	C2
S3	Sn3	C2	C2
S4	Sn4	C3	C3
S5	Sn5	C4	C4
S6	Sn6	C5	C5
S7	Sn7	C6	C6
S8	Sn8	C7	C7
S9	Sn9	C7	C7
	جدول S		***Index***

توجه: رکورد C7 پس از جستجو در ساختار آرایه و به صورت خطی در 9 حرکت در رکورد هشتم و نهم یافت می‌شود، که مرتبه اجرایی آن $O(n)$ است.

اجرای دستور زیر موجب ایجاد یک Nonclustered Index و Unique به صورت **صعودی**، به نام

SnameX روی ستون Sname در جدول S می‌شود.

```
CREATE UNIQUE NONCLUSTERED INDEX SnameX
ON S (Sname ASC);
```

با توجه به دستورات پیش فرض، قطعه کد زیر، معادل قطعه کد فوق است:

```
CREATE UNIQUE INDEX SnameX
ON S (Sname);
```

جداول زیر را در نظر بگیرید:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue

جدول P

توجه: دستور فوق یعنی ایجاد شاخص یکتا توسط دستور Unique فقط در صورتی قابل اجرا توسط DBMS روی ستون مورد نظر است، که از قبل ستون مورد نظر مقادیر تکراری نداشته باشد، در غیر اینصورت یعنی وجود مقادیر تکراری در ستون مورد نظر، دستور فوق از سوی DBMS رد می‌شود. که این می‌شود پیشا اجرای دستور فوق، همچنین پس از ایجاد دستور فوق یعنی ایجاد شاخص یکتا بر روی ستون حائز شرایط ایجاد شاخص یکتا، یکتا بودن ستون مورد نظر، تا مادامی که شاخص یکتا وجود دارد، گارانتی می‌شود که این می‌شود پسااجرای دستور فوق.

توجه: با استفاده از Nonclustered Index، سرعت پاسخگویی به تمامی پرس و جوهایی که جستجو را بر اساس Sname در جدول S انجام می‌دهند، افزایش می‌یابد.

توجه: اما در مورد پرس و جوهایی که جستجو را بر اساس ستون Sname انجام نمی‌دهند. تعریف شاخص فوق هیچ تاثیری در سرعت پاسخگویی به پرس و جو ندارد.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید: (ascending یعنی صعودی)

Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9

پرس و جوی زیر را در نظر بگیرید:

```
SELECT S#
FROM S
WHERE Sname='Sn9'
```

توجه: شرط جلوی where یعنی Sname='Sn9' **یک مقدار** مشخص است و این یعنی **پرس و جوی نقطه‌ای**. خروجی پرس و جوی فوق به صورت زیر است:

S#
S9

پیاده‌سازی شاخص فوق با ساختار B^+ Tree

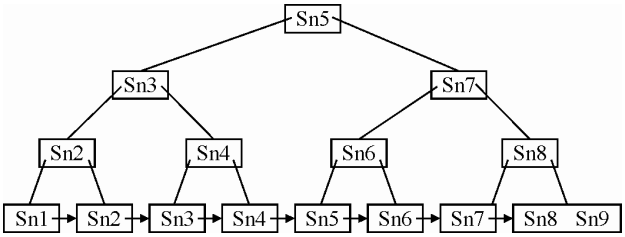
در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه بـه دلیـل مزایـای سـاختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جسـتجو در هـر گـره t-1 یعنـی 2=1-3 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree به صورت زیر است:



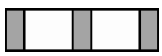
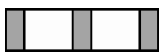
نتیجه: رکورد Sn9 پس از جستجو در سـاختار B^+ Tree و بـه صـورت درختـی در 4 حرکـت در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(Log_t\ n)$ است.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید: (descending یعنی نزولی)

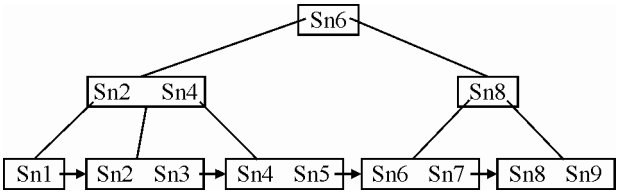
Sn9,Sn8,Sn7,Sn6,Sn5,Sn4,Sn3,Sn2,Sn1

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلیـد جسـتجو در هـر گـره t-1 یعنـی 2=1-3 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree **به صورت زیر است:**



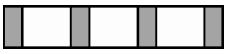
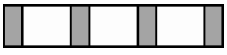
نتیجه: دو رشته‌ی یکسان از نظر مقادیر؛ و متفاوت از نظر ترتیب درج و یکسان از نظر درجه، دو شکل متفاوت در ساختار، چیدمان و تعداد سطوح داشتند.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

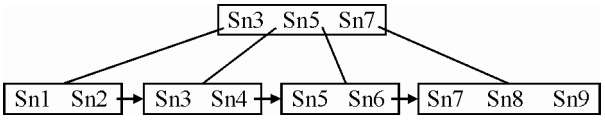
Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9

توجه: فرض کنید درجه t=4 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 4-1=3 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=3	
Number of Pointers	t=4	

ساختار نهایی B^+ Tree **به صورت زیر است:**

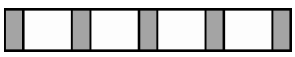


مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

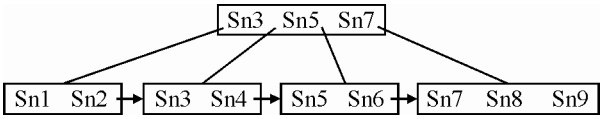
Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9

توجه: فرض کنید درجه t=5 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 5-1=4 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=4	
Number of Pointers	t=5	

ساختار نهایی B^+ Tree **به صورت زیر است:**



نتیجه: دو رشته‌ی یکسان از نظر مقادیر در دو مثال قبل؛ و یکسان از نظر ترتیب درج و متفاوت از نظر درجه، دو شکل یکسان در ساختار، چیدمان و تعداد سطوح داشتند.

پیاده‌سازی شاخص فوق با ساختار آرایه

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C2
S4	Sn4	C3
S5	Sn5	C4
S6	Sn6	C5
S7	Sn7	C6
S8	Sn8	C7
S9	Sn9	C7

جدول S

Sname
Sn1
Sn2
Sn3
Sn4
Sn5
Sn6
Sn7
Sn8
Sn9

Index

توجه: رکورد Sn9 پس از جستجو در ساختار آرایه و به صورت خطی در 9 حرکت در رکورد نهم یافت می‌شود، که مرتبه اجرایی آن $O(n)$ است.

۲- پرس و جوی بازه‌ای(Range query)

در پرس و جوی بازه‌ای، هدف شناسایی و بازیابی سطرهایی است که مقدار ستون مورد جستجو در **یک بازه** مشخص قرار دارد.

مثال: درج رشته زیر را بر اساس جدول S در نظر بگیرید:

Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9

پرس و جوی زیر را در نظر بگیرید:

```
SELECT S#
FROM S
WHERE Sname>'Sn4' AND Sname<='Sn9'
```

شرط جلوی where یعنی Sname>'Sn4' AND Sname<='Sn9' **یک بازه** مشخص است و این

یعنی **پرس و جوی بازه‌ای**. خروجی پرس و جوی فوق به صورت زیر است:

S#
Sn5
Sn6
Sn7
Sn8
Sn9

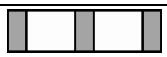
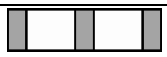
توجه: رکوردهای Sn5، Sn6، Sn7، Sn8 و Sn9 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(Log_t\, n+k)$ است.

پیاده‌سازی شاخص فوق با ساختار B^+ Tree

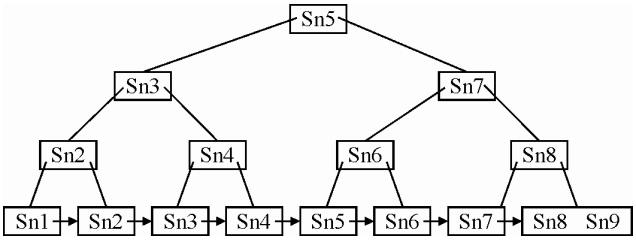
در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

توجه: فرض کنید درجه t=3 و به تبع حداکثر تعداد کلید جستجو در هر گره t-1 یعنی 3-1=2 است.

ساختار حداکثر درجه (اشاره‌گرها) و حداکثر کلید جستجو به صورت زیر است:

Number of Keys	key=2	
Number of Pointers	t=3	

ساختار نهایی B^+ Tree به صورت زیر است:



نتیجه: رکوردهای Sn5، Sn6، Sn7، Sn8 و Sn9 پس از جستجو در ساختار B^+ Tree و به صورت درختی در 4 حرکت و چند قدم در سطح چهارم یافت می‌شود، که مرتبه اجرایی آن $O(Log_t\, n+k)$ است.

پیاده‌سازی شاخص فوق با ساختار آرایه

در گذشته ساختار شاخص، به کمک آرایه پیاده‌سازی می‌شد. اما امروزه به دلیل مزایای ساختار

درختی و به تبع سرعت بیشتر در جستجو و بازیابی اطلاعات، ساختار B^+ Tree جهت پیاده‌سازی شاخص مورد استفاده قرار می‌گیرد.

S#	Sname	City	Pointer	Sname
S1	Sn1	C1	□	Sn1
S2	Sn2	C2	□	Sn2
S3	Sn3	C2	□	Sn3
S4	Sn4	C3	□	Sn4
S5	Sn5	C3	□	Sn5
S6	Sn6	C4	□	Sn6
S7	Sn7	C5	□	Sn7
S8	Sn8	C6	□	Sn8
S9	Sn9	C7	□	Sn9

جدول S ***Index***

توجه: رکوردهای Sn5، Sn6، Sn7، Sn8 و Sn9 پس از جستجو در ساختار آرایه و به صورت خطی در 9 حرکت در رکورد نهم یافت می‌شود، که مرتبه اجرایی آن $O(n)$ است.

۲- شاخص از نوع Hash (Hash Index)

شاخص از نوع Hash به کمک ساختار فایل مستقیم(تکنیک درهم‌سازی) قابل پیاده‌سازی است.

توجه: در SQL Server شاخص از نوع Hash پیاده‌سازی نشده است.

درهم‌سازی (Hashing)

منظور از درهم‌سازی، تبدیل کلید به آدرس (Key to Address Transformation=KAT) توسط پردازشی است که بر روی کلید انجام می‌گیرد. به عبارتی دیگر تابع درهم‌ساز (hashing function) تابعی است که هنگام **درج** یک رکورد، کلید جستجو که یکی از صفات است را گرفته، پردازش و یا محاسباتی را بر روی آن انجام داده و آدرس معادل آنرا بر می‌گرداند. این آدرس همان جایی است که رکورد مورد نظر باید در آن قرار گیرد و درج شود و به آن آدرس طبیعی (Natural Address)، حفره طبیعی (Natural Slot) و یا آدرس خانگی (Home Address) نیز می‌گویند. هنگام جستجو و بازیابی رکورد مورد نظر همین عمل مجددا صورت می‌گیرد، یعنی مجددا کلید مربوطه به تابع درهم‌ساز داده می‌شود و سیستم آدرس ذخیره‌سازی رکورد مرتبط با آنرا همانند قبل تولید می‌کند و با این تکنیک، دستیابی مستقیم به رکوردها امکان‌پذیر می‌گردد. به تابع درهم‌ساز، تابع مبدل یا نگاشتگر (Mapping Function) نیز گفته می‌شود.

توجه: همانطور که پیش‌تر گفتیم، در ساختار آرایه جستجو و بازیابی نقطه‌ای و بازه‌ای از مرتبه $O(n)$ بود. همچنین در ساختار B^+ Tree جستجو و بازیابی نقطه‌ای (Equality Query) از

مرتبه $O(Log_t n)$ و جستجو و بازیابی بازه‌ای (Range Query) از مرتبه $O(Log_t n + k)$ بود. اما در ساختار فایل مستقیم (تکنیک درهم‌سازی) جستجو و بازیابی نقطه‌ای (Equality Query) از مرتبه $O(1)$ است. دستیابی از مرتبه $O(1)$ به جدول پایه، به این معناست که بدون توجه، به اندازه جدول پایه، جهت دسترسی به یک رکورد دلخواه، همواره تعداد اندکی پیگرد نیاز است. در حالت ایده‌آل با یک حرکت می‌توان به رکورد مورد نظر دسترسی پیدا کرد. همچنین در ساختار فایل مستقیم (تکنیک درهم‌سازی) جستجو و بازیابی بازه‌ای (Range Query) می‌بایست برای تک تک اعضای بازه به طور مستقل تابع hash اجرا شود که این امر مستلزم صرف وقت و هزینه زیادی است. فایل مستقیم بی‌نظم است و امکان پردازش سریالی و ترتیبی را ندارد، زیرا فایل مستقیم بر حسب کلید مرتب نشده است. به عبارت دیگر به علت درهم بودن فایل، واکشی رکورد بعدی مشابه واکشی یک رکورد جدید است، پس امکان پردازش سریالی و ترتیبی در آن وجود ندارد، ساختار فایل مستقیم مناسب محیط‌هایی است که دستیابی سریع به رکوردها مورد نیاز است و پردازش‌ها ترتیبی و سریالی مد نظر نباشد. بنابراین برای جستجو و بازیابی بازه‌ای(Range Query)، شاخص‌گذاری مرتب شده با ساختار B^+ Tree با مرتبه $O(Log_t n + k)$ مناسب‌تر است. و شاخص‌های از نوع Hash برای پاسخ به Range Query ها مفید نیست.

توجه: در ساختار شاخص از نوع Hash، شاخص دارای یک فضای آدرس (Address Space) است با m آدرس از آدرس 1 تا m یا از صفر تا m-1 و هر آدرس مربوط است به یک حفره و هر حفره، مکان ذخیره‌سازی یک رکورد است. اگر تعداد رکوردها n باشد، آنگاه m>=n است. توابع درهم‌ساز(hashing) رکوردها را به صورت تصادفی و نامنظم در فضای آدرس، پخش می‌کنند.

پدیده تصادم (برخورد Collision)

اگر دو کلید متمایز، پس از اعمال تابع مبدل، آدرس‌های مساوی تولید کنند، یعنی $k_i \neq k_j \Rightarrow a_i = a_j$، آنگاه تصادم رخ داده است. که پس از وقوع پدیده برخود می‌بایست مساله برخورد به نحوی حل شود.

مثال: نگهداری آدرس n=6 رکورد در شاخص، کلید رکوردها **بخش عددی** ستون S# از جدول پایه S و تعداد سطرهای آدرس دهی در شاخص برابر m=14 است. تابع مولد یعنی h به صورت مقابل است:

$$h(S\#) \Rightarrow address = S\# \bmod 13$$

$R_1 : h(S100) \Rightarrow address = 100 \bmod 13 = 9$

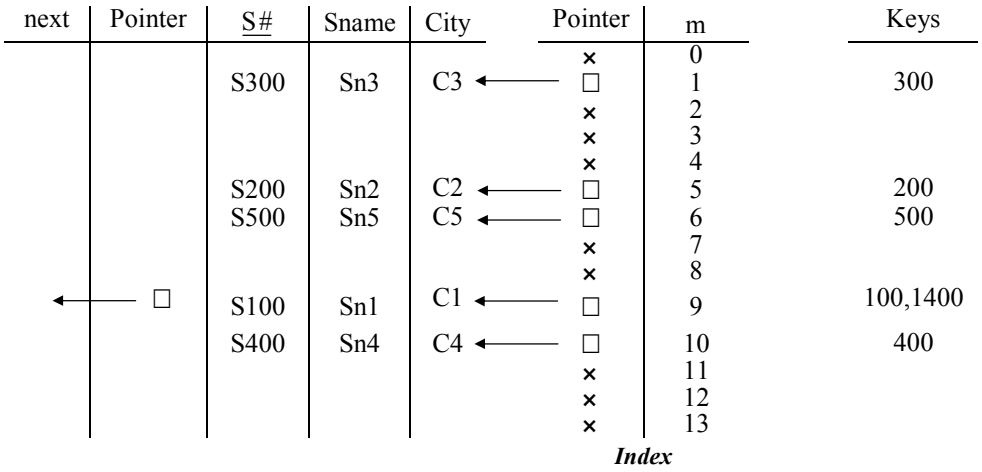
$R_2 : h(S200) \Rightarrow address = 200 \bmod 13 = 5$

$R_3 : h(S300) \Rightarrow address = 300 \bmod 13 = 1$

$R_4 : h(S400) \Rightarrow address = 400 \bmod 13 = 10$

$R_5 : h(S500) \Rightarrow address = 500 \bmod 13 = 6$

$R_6 : h(S1400) \Rightarrow address = 1400 \bmod 13 = 9$



توجه: رکورد R_6 هنگام درج با رکورد R_1 تصادم دارد، که پس از وقوع پدیده برخود می‌بایست مساله برخورد به نحوی حل شود. برای مثال رکورد R_1 توسط اشاره‌گر به رکورد R_6 اشاره کند که این موضوع می‌تواند برای یک حفره مدام تکرار شود و یک زنجیره تشکیل شود. و یا رکورد R_6 در اولین حفره خالی مثل حفره 11 درج شود.

توجه: راه حل ایده‌آل برای حل مساله برخورد، آن است که از الگوریتم و تابعی استفاده شود که به طور کلی از تصادم جلوگیری کند. به چنین تابعی، **تابع درهم‌ساز کامل** گفته می‌شود. در عمل پیدا کردن چنین تابعی بسیار پیچیده و زمان‌بر است. یعنی با آنکه پیدا کردن تابع درهم‌سازی کامل امکان‌پذیر است ولی هزینه‌ی یافتن آن به قدری زیاد است که در عمل هیچگاه از این روش استفاده نمی‌گردد.

راه حل عمومی آن است که:

۱- ابتدا با روش‌هایی سعی کنیم تعداد برخوردها به میزان قابل توجهی کاهش پیدا کند.

۲- سپس راه حل‌های مناسبی را بیابیم که اگر برخوردی رخ داد، آنرا بر طرف سازد.

توجه: تابع مولد (درهم‌ساز) همواره باید، به ازای یک کلید معین، همان آدرسی را تولید کند که در بار اولیه ساخته است. به عبارت دیگر تابع درهم‌ساز باید توانایی **تکرارشدنی** داشته باشد. همچنین تابع مولد ایده‌آل باید اول اینکه رکوردها را به طور یکنواخت توزیع کند، دوم اینکه باید از تمام اجزای کلید استفاده کند و سوم اینکه تعداد برخوردها را نیز کاهش دهد.

خصوصیات توابع مبدل ایده‌آل

۱- پراکنده کردن رکوردها: توابع مبدل (درهم‌ساز) باید رکوردها رو به صورت اتفاقی و تصادفی بین آدرس‌ها توزیع کند. بدیهی است که اگر احتمال قرارگیری کلیدها در خانه‌های مشخصی بیشتر از سایر خانه‌ها باشد، در آنجا پدیده تصادم بیشتر رخ می‌دهد. ولی هرچقدر که کلیدها در فضای

آدرس، بیشتر پخش شوند آنگاه احتمال برخورد نیز کمتر می‌شود. برای مثال اگر تابع مولد جهت درهم‌سازی فقط حرف اول کلید جستجو را معیار درهم‌سازی قرار دهد، آنگاه از آنجا که برای مثال نام‌های متعددی با حرف A یا M آغاز شده و اسامی کمی با حرف Z و G شروع می‌شوند، پس این تابع مولد به طور مناسب رکوردها را پراکنده نمی‌سازد. بنابراین همانطور که گفتیم، تابع مولد ایده‌آل باید رکوردها را به طور یکنواخت توزیع کند، باید از تمام اجزای کلید استفاده کند و تعداد برخوردها را نیز کاهش دهد.

۲- رشد خطی فضای آدرس: بدیهی است که اگر تعداد کمی رکورد را بخواهیم در بین تعداد زیادی آدرس قرار دهیم، تابع درهم‌ساز ساده‌تر بوده و احتمال برخورد نیز کاهش می‌یابد. برای مثال اگر بخواهیم 25 رکورد را در یک فضای 1000 خانه‌ای ذخیره کنیم (n=25 و m=1000). در این حالت احتمال بروز تصادم کاهش می‌یابد ولی روشن است که در این روش، فضای زیادی به هدر خواهد رفت. در بعضی سیستم‌ها m به صورت پویا رشد می‌کند. به عبارت دیگر فضای آدرس به صورت خطی گسترش داده می‌شود. برای مثال یک روش آن است که به ازای هر بار تصادم یک حفره به فضای آدرس اضافه شود. (m=m+1).

۳- ذخیره بیش از یک رکورد در یک آدرس (تکنیک باکت‌بندی): می‌توان فایل شاخص را به گونه‌ای پیاده‌سازی کرد که هر آدرس آن بتواند چند رکورد را در خود ذخیره کند. برای مثال اگر اندازه یک باکت برابر 512 بایت باشد و اندازه هر رکورد برابر 128 بایت باشد، آنگاه می‌توان در هر باکت 4 رکورد را جای داد. به چنین فضاهایی که می‌توانند چند رکورد را به این شیوه ذخیره کنند، باکت (bucket) گفته می‌شود. اگر فایل مستقیم باکت‌بندی شود، آنگاه به جای آدرس حفره، آدرس باکت خواهیم داشت. از صفر تا M-1.

مثال: نگهداری آدرس n=6 رکورد در شاخص، کلید رکوردها **بخش عددی** ستون S# از جدول پایه S و تعداد رکوردهای آدرس‌دهی در شاخص برابر m=14 است. همچنین اندازه یک باکت برابر 512 بایت و اندازه هر رکورد برابر 256 بایت است یعنی فاکتور باکت‌بندی برابر 2 رکورد در یک باکت است. تابع مولد یعنی h به صورت زیر است:

$$h(S\#) \Rightarrow \text{address} = S\# \bmod 13$$

$$R_1 : h(S100) \Rightarrow \text{address} = 100 \bmod 13 = 9$$

$$R_2 : h(S200) \Rightarrow \text{address} = 200 \bmod 13 = 5$$

$$R_3 : h(S300) \Rightarrow \text{address} = 300 \bmod 13 = 1$$

$$R_4 : h(S400) \Rightarrow \text{address} = 400 \bmod 13 = 10$$

$$R_5 : h(S500) \Rightarrow \text{address} = 500 \bmod 13 = 6$$

$$R_6 : h(S1400) \Rightarrow \text{address} = 1400 \bmod 13 = 9$$

next	Pointer	S#	Sname	City	Pointer	M	Keys
					× ×	0	
		S300	Sn3	C3	□ ×	1	300
					× ×	2	
					× ×	3	
					× ×	4	
		S200	Sn2	C2	□ ×	5	200
		S500	Sn5	C5	□ ×	6	500
					× ×	7	
					× ×	8	
		S100	Sn1	C1	□	9	100
		S1400	S14	C14	□		1400
		S400	Sn4	C4	□ ×	10	400
					× ×	11	
					× ×	12	
					× ×	13	

Index

توجه: رکورد R_6 هنگام درج بدون تصادم با رکورد R_1 در باکت شماره 9 قرار می‌گیرد.

توجه: در باکت‌بندی ممکن است فضای به هدر رفته زیاد باشد.

توجه: هنگام واکشی یک رکورد، تابع درهم‌ساز شماره حفره باکت را مشخص می‌کند. و کل آن باکت در حافظه واکشی شده و سپس رکوردهای موجود در آن جهت یافتن رکورد مورد نظر در حافظه جستجو می‌شود.

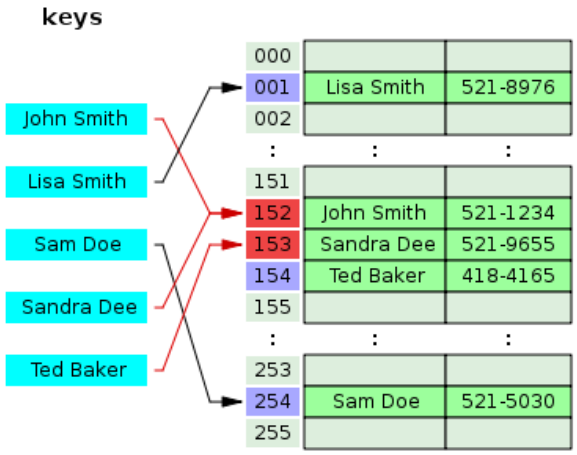
توجه: در حالت باکت‌بندی نیز همچنان مشکل سرریزی وجود دارد که برای مثال می‌توان از روش زنجیره اشاره‌گرها استفاده کرد. برای مثال حفره شماره 9 در مثال فوق دیگر جا ندارد ولی بدیهی است که تعداد سرریزها در حالت باکت‌بندی کمتر از حالت بدون باکت‌بندی است.

توجه: از نظر سیستم ورودی و خروجی، زمان انتقال یک رکورد با زمان انتقال یک باکت، تفاوت چندانی ندارد. بنابراین باکت‌بندی ایده مناسبی در جهت کاهش زمان انتقال و تصادم است.

روش‌های رفع مشکل برخورد

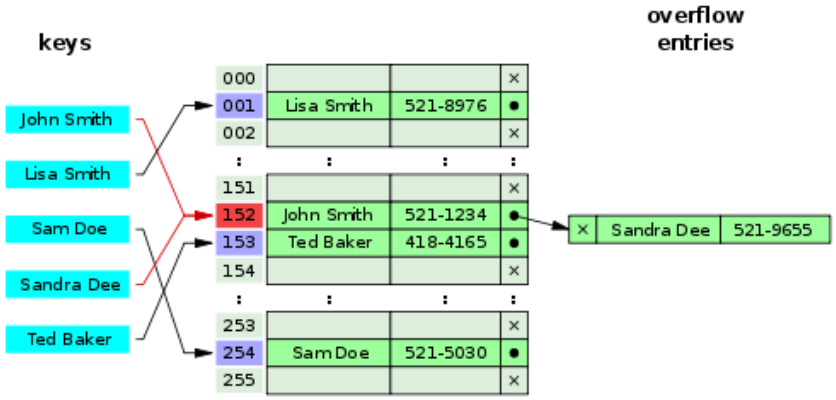
جستجوی خطی و درج در اولین حفره جادار (Linear Probing)

در این روش هنگام اضافه کردن رکورد در صورتی که برخورد رخ دهد، از نقطه‌ای که تصادم رخ داده است به سمت انتهای فایل به صورت خطی دنبال اولین حفره خالی جهت درج رکورد سرریز می‌گردیم. اگر تا انتهای فایل حفره‌ای پیدا نشد، بصورت چرخشی از ابتدای فایل تا نقطه تصادم جستجو را ادامه می‌دهیم. شکل زیر گویای مطلب است:

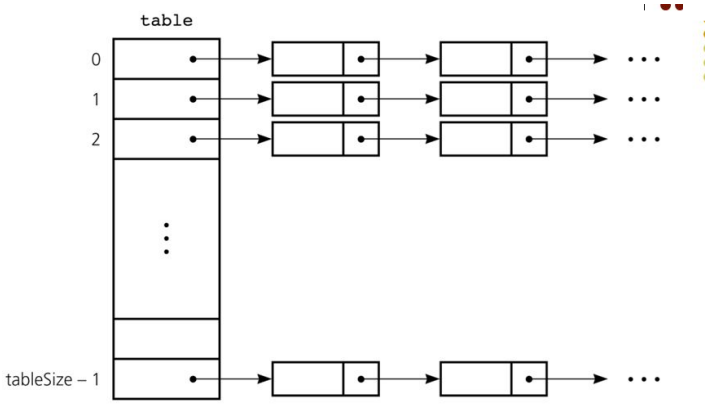


روش ساخت زنجیره اشاره‌گرها

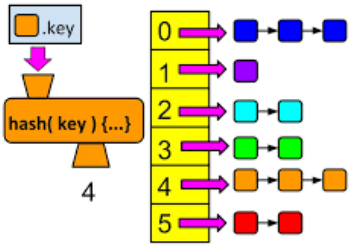
در این روش هنگام اضافه کردن رکورد در صورتی که برخورد رخ دهد، اشاره‌گر جهت اتصال رکوردها به عنوان زنجیره اتصال مورد استفاده قرار می‌گیرد. شکل زیر گویای مطلب است:



دو تصویر زیر نیز گویای مطلب است:



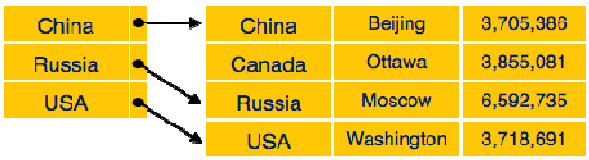
Each location of the hash table contains a pointer to a linked list



توجه: در گذشته شاخص مرتب شده توسط آرایه پیاده‌سازی می‌شد و به دو شکل زیر نیز وجود داشت:

شاخص متراکم (Dense Index): در شاخص متراکم به ازای هر مقدار **متمایز** برای کلید جستجو یک مدخل متمایز در فایل شاخص در نظر گرفته می‌شود.

شاخص پراکنده یا تنک (Sparse Index): در شاخص پراکنده به ازای برخی از مقادیر کلید جستجو یک مدخل در فایل شاخص در نظر گرفته می‌شود.



توجه: سرعت جستجو و بازیابی اطلاعات مورد نظر با استفاده از شاخص متراکم بیشتر از شاخص پراکنده است، همچنین سربار حافظه‌ای در شاخص متراکم از شاخص پراکنده بیشتر است.

شاخص Bitmap

در شاخص Bitmap به ازای تعداد سطرهای ستونی از جدول پایه که قصد تعریف شاخص داریم، یک بیت برای هر سطر در نظر گرفته می‌شود.
جدول زیر را در نظر بگیرید:

EmpNo	EmpName	Job	New_Emp	Salary
1	Alice	Analyst	Yes	15000
2	Joe	Salesperson	No	10000
3	Katy	Clerk	No	12000
4	Annie	Manager	Yes	25000

توجه: برای تعریف شاخص Bitmap روی ستون New_Emp رشته دودویی به طول چهار بیت در نظر گرفته می‌شود، زیرا تعداد سطرهای ستون New_Emp برابر چهار است، به صورت زیر:

New-Emp Values	Bitmap Indices
yes	1001
No	0110

رشته 1001 یعنی مقدار Yes در سطرهای اول و چهارم از ستون New_Emp از جدول Employee است و رشته 0110 یعنی مقدار No در سطرهای دوم و سوم از ستون New_Emp از جدول Employee است.

توجه: برای تعریف شاخص Bitmap روی ستون Job رشته دودویی به طول چهار بیت در نظر گرفته می‌شود، زیرا تعداد سطرهای ستون Job برابر چهار است، به صورت زیر:

Job Values	Bitmap Indices
Analyst	1000
Salesperson	0100
Clerk	0010
Manager	0001

رشته 1000 یعنی مقدار Analyst در سطر اول از ستون Job از جدول Employee است، رشته 0100 یعنی مقدار Salesperson در سطر دوم از ستون Job از جدول Employee است، رشته 0010 یعنی مقدار Clerk در سطر سوم از ستون Job از جدول Employee است و رشته 0001 یعنی مقدار Manager در سطر چهارم از ستون Job از جدول Employee است.

قطعه کد ایجاد Bitmap Index به صورت زیر است:

```
CREATE BITMAP INDEX index_name
ON table_name (column_name);
```

مثال: اجرای دستور زیر موجب ایجاد یک Bitmap Index ، به نام index_New_Emp روی ستون New_Emp در جدول Employee می‌شود.

```
CREATE BITMAP INDEX index_New_Emp
ON Employee (New_Emp);
```

مثال: اجرای دستور زیر موجب ایجاد یک Bitmap Index ، به نام index_Job روی ستون Job در جدول Employee می‌شود.

```
CREATE BITMAP INDEX index_Job
ON Employee (Job);
```

مثال:

```
SELECT *
FROM Employee
WHERE New_Emp  ='NO' and Job= 'Salesperson'
```

Bitmap Index for "NO"	⟶	0 1 1 0
		AND
Bitmap Index for Salesperson	⟶	0 1 0 0
Result	⟶	0 1 0 0

توجه: رشته 0100 در Result بدین معنی است که سطر دوم از جدول Employee حائز شرایط پرس و جوی مطرح شده است. به عملگر AND دقت نمایید.

توجه: تعداد 1 در خروجی Result نشانه، تعداد اتفاق افتادن شرایط و رویداد مورد نظر است.

خروجی پرس و جوی فوق به صورت زیر است:

EmpNo	EmpName	Job	New_Emp	Salary
2	Joe	Salesperson	No	10000

توجه: از آنجا که هر سطر از جدول پایه در شاخص Bitmap فقط با یک بیت مشخص می‌شود، بنابراین فضای سربار توسط شاخص، بسیار کم است.

توجه: اگر در ساختار شاخص، مقادیر ستون‌هایی غیر از کلید جستجو نگهداری شود، به این نوع شاخص، **شاخص پوششی (Covering Index)** گفته می‌شود. شاخص پوششی، دسترسی به مقادیر ستون‌هایی غیر از کلید جستجو را نیز سرعت می‌بخشد.

بهترین منابع

کنکور ارشد فناوری اطلاعات

	عنوان درس	منابع
زبان عمومی و تخصصی	زبان انگلیسی	کلاس آنلاین یا فیلم استاد زارع یا کتاب راهیان ارشد
دروس مشترک	ریاضیات گسسته	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	ساختمان داده‌ها	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	طراحی الگوریتم	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	مهندسی نرم‌افزار	کلاس آنلاین یا فیلم استاد خلیلی‌فر یا کتاب راهیان ارشد
	شبکه‌های کامپیوتری	کلاس آنلاین یا فیلم استاد خلیلی‌فر یا کتاب راهیان ارشد
دروس تخصصی	پایگاه داده‌ها	کلاس آنلاین یا فیلم استاد خلیلی‌فر یا کتاب راهیان ارشد
	هوش مصنوعی	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	سیستم عامل	کلاس آنلاین یا فیلم استاد خلیلی‌فر یا کتاب راهیان ارشد
	اصول و مبانی مدیریت	مبانی مدیریت و مبانی رفتار سازمانی استیفن رابینز یا کتاب راهیان ارشد

گروه بابان
BABAN.IR

بهترین منابع
کنکور ارشد علوم کامپیوتر

	عنوان درس	منابع
زبان عمومی و تخصصی	زبان انگلیسی	کلاس آنلاین یا فیلم استاد زارع یا کتاب راهیان ارشد
دروس پایه	ریاضی عمومی ۱ و ۲	کلاس آنلاین یا فیلم استاد گیلک یا جزوه بابان
	مبانی علوم ریاضی	کتاب لیپن ولیپن ترجمه عمید رسولیان فصل های اول، پنجم و هفتم کلاس آنلاین ریاضیات گسسته یا فیلم استاد گیلک یا کتاب ریاضیات گسسته راهیان ارشد
	مبانی ماتریس ها و جبرخطی	کتاب جبرخطی هافمن یا کتاب جبرخطی اونان یا کتاب جبرخطی پیام نور
	مبانی آنالیز ریاضی	اصول آنالیز ریاضی رودین فصل یک تا شش یا کتاب آنالیز ریاضی یک پیام نور
	مبانی آنالیز عددی	کتاب راهیان ارشد بهزاد خداکرمی یا کتاب آنالیز عددی بابلیان
	مبانی احتمال	کلاس آنلاین یا فیلم استاد گیلک یا جزوه بابان کتاب آمار و احتمال شلدون راس
دروس مشترک	ساختمان داده	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	طراحی الگوریتم	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد
	مبانی نظریه محاسبه	کلاس آنلاین یا فیلم استاد گیلک یا کتاب علوم رایانه استاد شاپوری
	مبانی منطق و نظریه مجموعه ها	کلاس آنلاین یا فیلم استاد گیلک یا جزوه بابان
	ریاضیات گسسته و مبانی ترکیبیات	کلاس آنلاین یا فیلم استاد گیلک یا کتاب راهیان ارشد

گروه بابان
BABAN.IR

مشاوره تخصصی رشته کامپیوتر و IT

در راستای رسالت **مؤسسهٔ فرهنگی و انتشاراتی بابان** مبنی بر ارتقای سطح علـم و دانش کشور و کمک همه جانبه به دانشجویان و داوطلبان گرامـی، در جهـت قبـولی در کنکور کارشناسی ارشد و دکتری مهندسـی کـامپیوتر و IT دو طـرح زیـر را پایـه‌ریـزی کرده‌ایم:

۱) ارائه مشاوره تخصصی **حضوری** و **غیرحضوری** (تلفنی و آنلاین)

۲) برگزاری کلاس‌های **حضوری** و **غیرحضوری** (فیلم آموزشی و کلاس آنلاین)

برای آشنایی بیشتر با خدمات ارائـه شـده توسـط **موسسـه بابـان** بـه وب سـایت **khalilifar.ir** یا کانال تلگرام **@arastookhalilifar** مراجعه فرمایید.

تلفن دفتر مرکزی موسسه بابان: ۷۷۹۷۲۸۶۸-۰۲۱

تلفن دفتر فروشگاه انتشارات بابان: ۷۷۹۷۳۳۸۶-۰۲۱

پایگاه اطلاع رسانی موسسه بابان: www.baban.ir