موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

پایگاه داده‌ها

(حل تشریحی سوالات دولتی ۱۳۹۸)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

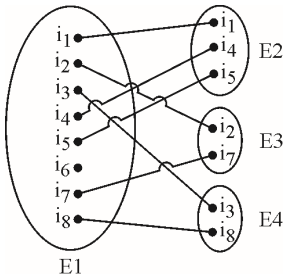
براساس کتب مرجع

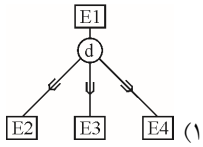
راما کریشنان، آبراهام سیلبرشاتز و رامز المصری

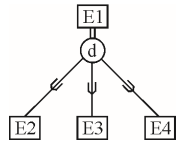
ارسطو خلیلی‌فر

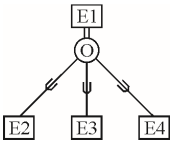
تست‌های فصل سوم: مدل رابطه‌ای

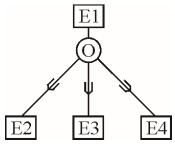
۱- نمونه‌هایی از چهـار نـوع موجودیـت E1، E2، E3 و E4 در شکل مقابل نمایش داده شده است. کدام مورد بهترین نمودار EER معرف محیط است؟ **(مهندسی کامپیوتر-دولتی ۹۸)**



۱)


۲)


۳)


۴)


۲- در نمودار E-R، اگر رابطه is-A از نوع منفصل و کامل باشد، کدام مورد بـرای تبـدیل نمـودار بـه مـدل رابطه‌ای مناسب‌تر است؟ **(مهندسی IT- دولتی ۹۸)**

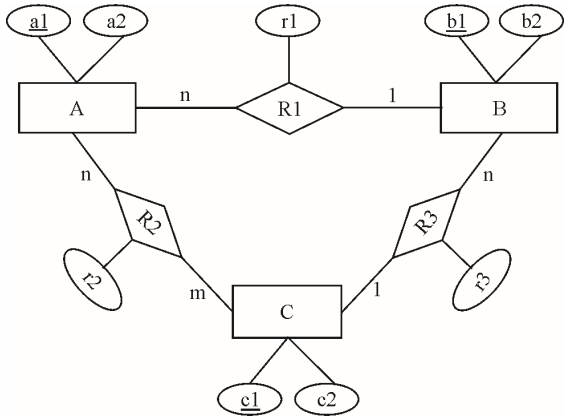
۱) کلید اصلی موجودیت پدر را به عنوان کلید خارجی به موجودیت‌های فرزند اضافه می‌کنیم.

۲) کلید اصلی موجودیت‌های فرزند را به عنوان کلید خارجی به موجودیت پدر اضافه می‌کنیم.

۳) برای موجودیـت پـدر، رابطـه جداگانـه ایجـاد نمـی‌کنیم و ویژگی‌هـای رابطـه پـدر را بـه موجودیت‌های فرزند اضافه می‌کنیم.

۴) موجودیت جداگانه‌ای ایجاد می‌شود که شامل کلید اصـلی موجودیـت پـدر و کلیـد اصـلی موجودیت‌های فرزند است.

۳- مدل رابطه‌ای متناظر با نمودار ER زیر کدام است؟ **(مهندسی IT- دولتی ۹۸)**



۱) A(a1, a2) B(b1, b2) C(c1, c2) R1(a1, b1, r1) R2(a1, c1, r2) R3(b1, c1, r3)

۲) A(a1, a2) B(b1, b2, a1, r1) C(c1, c2, b1, r3) R1(a1, b1, r1) R3(b1, c1, r3)

۳) A(a1, a2, b1) B(b1, b2, c1, r1) C(c1, c2, r3) R2(a1, c1, r2)

۴) A(a1, a2, b1, r1) B(b1, b2, c1, r3) C(c1, c2) R2(a1, c1, r2)

پاسخ تست‌های فصل سوم: مدل رابطه‌ای

۱- گزینه (۱) صحیح است.

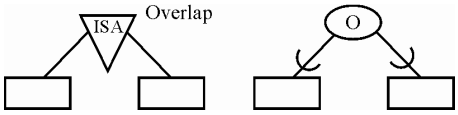
نگاشت رابطه ISA یا وراثت به مدل رابطه‌ای

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

توجه: در یک رابطه اجباری یا کلی (Total)، هر نمونه از موجودیت پدر حتماً می‌بایست با یکی از نمونه موجودیت‌های فرزند در ارتباط باشد.

توجه: در یک رابطه اختیاری یا جزئی (Partial)، هر نمونه از موجودیت پدر می‌تواند با یکی از نمونه موجودیت‌های فرزند در ارتباط باشد یا نباشد.

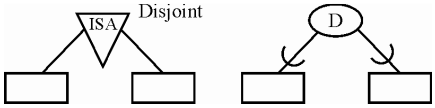
رابطه متصل یا پوشا یا تخصیص غیرمجزا (Overlap) مابین فرزندان و پدر به دو شیوه زیر نشان داده می‌شود:



توجه: در رابطه متصل یا پوشا یا تخصیص غیرمجزا (Overlap) ارتباط پدر با فرزندان می‌تواند یک به یک باشد و همچنین می‌تواند یک به چند باشد. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر می‌تواند تهی باشد و همچنین می‌تواند غیرتهی باشد.

توجه: در یک رابطه پوشا یا تخصیص غیرمجزا (Overlap)، نمونه موجودیت‌های فرزند می‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند که این یعنی همان رابطه یک به چند میان پدر و فرزندان.

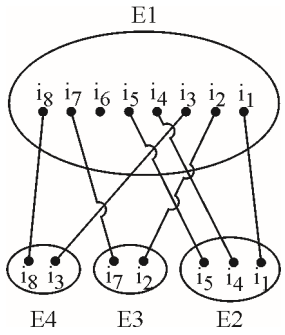
رابطه منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) مابین فرزندان و پدر به دو شیوه زیر نشان داده می‌شود:



توجه: در رابطه منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) ارتباط پدر با فرزندان فقط و فقط یک به یک است. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر همواره تهی است.

توجه: در یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint)، نمونه موجودیت‌های فرزند نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند که این یعنی همان رابطه یک به یک میان پدر و فرزندان.

صورت سوال به این شکل است:
نمونه‌هایی از چهار نوع موجودیت E1، E2، E3 و E4 در شکل مقابل نمایش داده شده است، کدام مورد بهترین نمودار EER معرف محیط است؟



توجه: در صورت سوال باید شرایط محیط عملیاتی به دقت گفته می‌شد، تعدد نمونه موجودیت‌ها بیان شده در نمودار صورت سوال، گویای شرایط فعلی محیط عملیاتی است و گویای آینده تعدد نمونه موجودیت‌های شرایط آتی محیط عملیاتی نیست، ضمن اینکه نمودار مطرح شده در صورت سوال نشانه تعدد لحظه‌ای یک محیط عملیاتی است و گویای آینده محیط عملیاتی نیست، و خود نمودار ER همواره گویای محیط عملیاتی است که طراح خواسته از تعدد به نمودار ER برسد آن هم بدون ذکر شرایط دقیق محیط عملیاتی. که تفکر نادرستی بوده است. سناریوی دقیق شرایط محیط عملیاتی می‌بایست در صورت سوال مطرح می‌شد. که نشده است.

بطور کلی خواص رابطه به سه شکل زیر وجود دارد:

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.

ب) کاردینالیتی ارتباط

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

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 رابطه ISA در نمودار EER با نماد خط مضاعف عمودی از موجودیت پدر به سمت موجودیت‌های

فرزند نشان داده می‌شود.

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 **رابطه** ISA در نمودار EER با نماد خط عمودی از موجودیت پدر به سمت موجودیت‌های فرزند نشان داده می‌شود.

همانطور که گفتیم در رابطه ISA رابطه پدر با فرزندان به دو صورت رابطه اختیاری یا جزئی یا بخشی (Partial) با نماد خط عمودی و رابطه اجباری یا کلی یا کامل (Total) با نماد خط مضاعف عمودی است. همچنین همانطور که گفتیم یک رابطه اختیاری است، اگر و تنها اگر حداقل یکی از نمونه‌های موجودیت پدر در رابطه شرکت نکرده باشد. اختیاری بودن **رابطه** ISA در نمودار EER با نماد خط عمودی نشان داده می‌شود.

همانطور که در شکل صورت سوال واضح و مشخص است، نمونه موجودیت j6 در موجودیت E1 یعنی پدر با هیچ یک از نمونه موجودیت‌های فرزند یعنی موجودیت‌های E2، E3 و E4 وارد رابطه نشده است. و از آنجاکه اگر و تنها اگر حداقل یکی از نمونه‌های موجودیت پدر در رابطه با فرزندان شرکت نکرده باشد، آن رابطه اختیاری است، بنابراین نمونه‌های موجودیت E1 به صورت **اختیاری** وارد رابطه با فرزندان یعنی موجودیت‌های E2، E3 و E4 شده‌اند. بنابراین گزینه‌های دوم و سوم پاسخ سوال نیستند.

توجه: اگر فرض کنیم نمودار مطرح شده در صورت سوال، یک لحظه‌ی خاص از تعریف محیط عملیاتی نباشد، بلکه حالت کلی، ایستا، پایدار و همیشگی محیط عملیاتی باشد، آنگاه از آنجا که ارتباط پدر با فرزندان یک به یک است. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر تهی است، می‌توان رابطه فرزندان با پدر را حالت منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) در نظر گرفت، که در این شرایط گزینه اول پاسخ سوال است. که طراح محترم هم همین گزینه را به عنوان بهترین نمودار EER برای نمودار صورت سوال در نظر گرفته است.

توجه: در یک رابطه غیرپوشا یا تخصیص مجزا (Disjoint)، نمونه موجودیت‌های فرزند نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند که این یعنی همان رابطه یک به یک میان پدر و فرزندان.

توجه: اما اگر فرض کنیم نمودار مطرح شده در صورت سوال، یک لحظه‌ی خاص از تعریف محیط عملیاتی باشد، یعنی حالت جزئی، پویا، ناپایدار و غیرهمیشگی محیط عملیاتی باشد، آنگاه از آنجا که ارتباط پدر با فرزندان می‌تواند یک به یک باشد و همچنین می‌تواند یک به چند باشد در آینده. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر می‌تواند تهی باشد و همچنین می‌تواند غیرتهی باشد، می‌توان رابطه فرزندان با پدر را حالت متصل یا پوشا یا تخصیص غیرمجزا (Overlap) در نظر گرفت، که در این شرایط گزینه چهارم پاسخ سوال است.

توجه: در یک رابطه پوشا یا تخصیص غیرمجزا (Overlap)، نمونه موجودیت‌های فرزند می‌توانند به طور همزمان با نمونه‌ای از موجودیت پدر در ارتباط باشند که این یعنی همان رابطه یک به چند میان پدر و فرزندان.

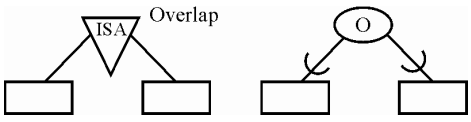
توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه اول را به عنوان پاسخ اعلام کرده بود.

۲- گزینه (۳) صحیح است.

نگاشت رابطه ISA یا وراثت به مدل رابطه‌ای

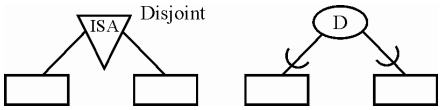
در رابطه ISA رابطه پدر با فرزندان به دو صورت رابطه اختیاری یا جزئی یا بخشی (Partial) با نماد خط عمودی و رابطه اجباری یا کلی یا کامل (Total) با نماد خط مضاعف عمودی است و رابطه فرزندان با پدر به دو صورت رابطه متصل یا پوشا یا تخصیص غیرمجزا (Overlap) و رابطه منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) می‌باشد.

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



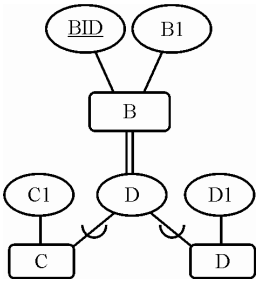
توجه: در رابطه متصل یا پوشا یا تخصیص غیرمجزا (Overlap) ارتباط پدر با فرزندان می‌تواند یک به یک باشد و همچنین می‌تواند یک به چند باشد. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر می‌تواند تهی باشد و همچنین می‌تواند غیرتهی باشد.

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



توجه: در رابطه منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) ارتباط پدر با فرزندان فقط و فقط یک به یک است. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت از پدر همواره تهی است.

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



مدل EER رسم شده در شکل فوق، یک رابطه اجباری یا کلی یا کامل (Total) مابین موجودیت B و موجودیت‌های C و D و یک رابطه منفصل یا 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 یا کامل (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 منفصل یا 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 بیان دیگر همانطور که گفتیم در رابطه منفصل یا غیرپوشا یا تخصیص مجزا (Disjoint) ارتباط پدر با فرزندان فقط و فقط یک به یک است. و همچنین اشتراک نمونه موجودیت‌ها میان موجودیت‌های فرزند با یک نمونه موجودیت پدر همواره تهی است.

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 یا کامل (Total).

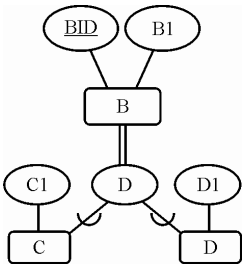
همچنین نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است نمی‌توانند به طور همزمان با نمونه‌ای از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد در ارتباط باشند، به عبارت دیگر نمی‌توان نمونه‌هایی از موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است داشت که به طور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

از آنجاکه رابطه مابین موجودیت B و موجودیت‌های C و D یک رابطه اجباری یا کلی یا کامل (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 منفصل یا 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. به عبارت دیگر **برای موجودیت پدر، رابطه جداگانه ایجاد**

نمی‌کنیم و ویژگی‌های رابطه پدر را به موجودیت‌های فرزند اضافه می‌کنیم. بنابراین گزینه‌های اول، دوم و چهارم را بطور کامل کنار می‌گذاریم، پس پُرواضح است که گزینه سوم پاسخ سوال است. بهمین سادگی.
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

۳- گزینه (۴) صحیح است.

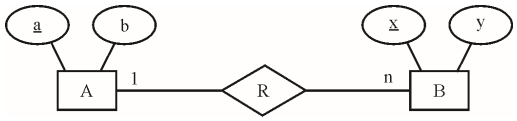
به طور کلی در مدل رابطه‌ای، هر موجودیت شناسایی شده در نمودار ER (مدل تحلیل) هنگام نگاشت به مدل رابطه‌ای (مدل طراحی) به یک جدول تبدیل می‌شود. همچنین صفت‌های موجودیت پس از نگاشت آن در مدل رابطه‌ای به صورت ستون‌های جدول بیان می‌شوند. همچنین ارتباط بین جدول از طریق کلید خارجی برقرار می‌گردد.

نگاشت رابطه یک به چند بین دو موجودیت به مدل رابطه‌ای

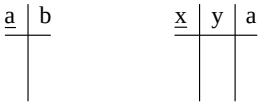
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد. و کلید کاندید جدول یک در جدول چند به عنوان کلید خارجی تعریف می‌گردد. همچنین صفات متصل به **رابطه**، درون جدول چند **مستتر** می‌شود.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:

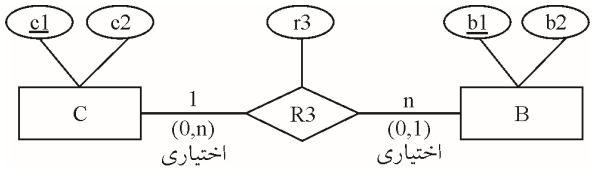


جدول A *جدول B*

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

حالت اختیاری

مدل تحلیل:

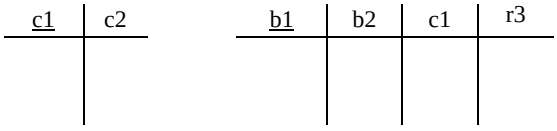


توجه: در شکل فوق صفت c1 کلید موجودیت C و صفت b1 کلید موجودیت B است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است.

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

مدل طراحی:



جدول C *جدول B*

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

توجه: همچنین صفات متصل (r3) به **رابطه**، درون جدول چند یعنی موجودیت B **مستتر** می‌شود.

توجه: کلید کاندید جدول یک یعنی **موجودیت C** برابر همان **کلید کاندید سابق** در جدول موجودیت C است. یعنی کلید کاندید جدول C برابر (c1) است.

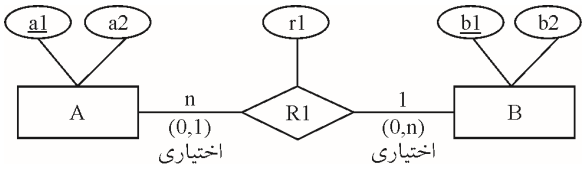
توجه: کلید کاندید جدول چند یعنی **موجودیت B** برابر همان **کلید کاندید سابق** در جدول موجودیت B است. یعنی کلید کاندید جدول B برابر (b1) است.

توجه: همانطور که واضح است، طراحی جدول B در گزینه چهارم به صورت B(b1,b2,c1,r3) در نظر گرفته شده است، همچنین طراحی جدول C در گزینه چهارم به صورت C(c1,c2) در نظر گرفته شده است

که مطابق آنچه بیان کردیم، طراحی درستی است. بنابراین گزینه‌های اول، دوم و سوم را کنار می‌گذاریم، پس تا همینجا پُرواضح است که گزینه چهارم پاسخ سوال است.
به طور مجدد بخشی از مدل EER رسم شده در صورت سوال، یک رابطه یک به چند بین دو موجودیت را نشان می‌دهد. که در ادامه فرآیند نگاشت آن به مدل رابطه‌ای را شرح می‌دهیم.

حالت اختیاری

مدل تحلیل:



توجه: در شکل فوق صفت a1 کلید موجودیت A و صفت b1 کلید موجودیت B است.
توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است.
توجه: قید (0,1) نشان می‌دهد که هر نمونه موجودیت از A حداقل با صفر و حداکثر با یک نمونه موجودیت از B ارتباط دارد و قید (0,N) نشان می‌دهد که هر نمونه موجودیت از B حداقل با صفر و حداکثر با N نمونه موجودیت از A ارتباط دارد.

مدل طراحی:

a1	a2	b1	r1

جدول A

b1	b2	c1	r3

جدول B

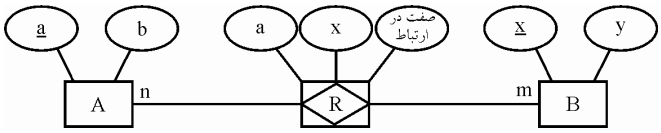
توجه: کلید کاندید جدول یک یعنی **موجودیت B** در جدول چند یعنی **موجودیت A** به عنوان کلید خارجی تعریف می‌گردد.
توجه: همچنین صفات متصل (r1) به **رابطه**، درون جدول چند یعنی موجودیت A **مستتر** می‌شود.
توجه: کلید کاندید جدول یک یعنی **موجودیت B** برابر همان **کلید کاندید سابق** در جدول موجودیت B است. یعنی کلید کاندید جدول B برابر (b1) است.
توجه: کلید کاندید جدول چند یعنی **موجودیت A** برابر همان **کلید کاندید سابق** در جدول موجودیت A است. یعنی کلید کاندید جدول A برابر (a1) 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: همانطور که واضح است، طراحی جدول B در گزینه چهارم به صورت $B(\underline{b1}, b2, c1, r3)$ در نظر گرفته شده است، همچنین طراحی جدول A در گزینه چهارم به صورت $A(\underline{a1}, a2, b1, r1)$ در نظر گرفته شده است که مطابق آنچه بیان کردیم، طراحی درستی است. بنابراین گزینه‌های اول، دوم و سوم را کنار می‌گذاریم، پس تا همینجا پُرواضح است که گزینه چهارم پاسخ سوال است.

نگاشت رابطه چند به چند بین دو موجودیت به مدل رابطه‌ای

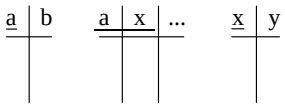
مستقل از اختیاری یا اجباری بودن موجودیت‌ها، هر موجودیت به یک جدول تبدیل می‌گردد و یک جدول پُل (Bridge) نیز به عنوان ارتباط دهنده دو جدول مورد استفاده قرار می‌گیرد. همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد. همچنین صفات متصل به **رابطه**، درون جدول پُل **مستتر** می‌شود.

روال کلی نگاشت در این حالت به صورت زیر است:

مدل تحلیل:



مدل طراحی:



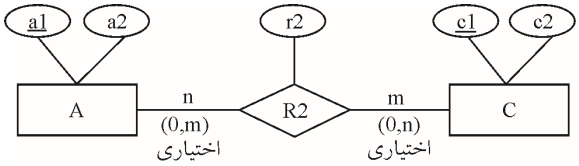
جدول A جدول AB جدول B

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.

حالت اختیاری

مدل تحلیل:

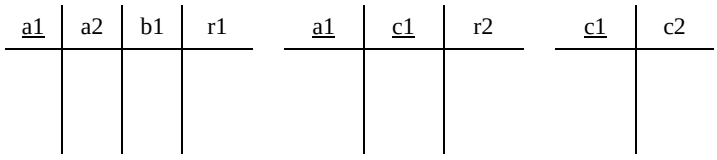


توجه: در شکل فوق صفت a1 کلید موجودیت A و صفت c1 کلید موجودیت C است.

توجه: نماد خط افقی نشانه اختیاری بودن موجودیت چسبیده به آن است.

توجه: قید (0,m) نشان می‌دهد که هر نمونه موجودیت از A حداقل با صفر و حداکثر با M نمونه موجودیت از C ارتباط دارد و قید (0,n) نشان می‌دهد که هر نمونه موجودیت از C حداقل با صفر و حداکثر با N نمونه موجودیت از A ارتباط دارد.

مدل طراحی:



جدول A *جدول r2* *جدول C*

توجه: ستون a1 در جدول r2 به عنوان کلید خارجی تعریف می‌گردد که به جدول A ارجاع می‌کند. همچنین ستون c1 در جدول r2 به عنوان کلید خارجی تعریف می‌گردد که به جدول C ارجاع می‌کند.

توجه: همچنین صفات متصل (r2) به **رابطه**، درون جدول پُل یعنی جدول r2 **مستتر** می‌شود.

توجه: همچنین کلید کاندید جدول پُل از ترکیب کلید کاندید دو جدول دیگر ایجاد می‌گردد. یعنی کلید کاندید جدول r2 برابر (a1,c1) است.

توجه: کلید کاندید جدول چند چپ یعنی **موجودیت A** برابر همان **کلید کاندید سابق** در جدول موجودیت A است. یعنی کلید کاندید جدول A برابر (a1) است.

توجه: کلید کاندید جدول چند راست یعنی **موجودیت C** برابر همان **کلید کاندید سابق** در جدول موجودیت C است. یعنی کلید کاندید جدول C برابر (c1) است.

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

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 C در گزینه چهارم به صورت C(c1,c2) در نظر گرفته شده است، همچنین طراحی جدول A در گزینه چهارم به صورت A(a1,a2,b1,r1) در نظر گرفته شده است، همچنین طراحی جدول r2 در گزینه چهارم به صورت R2(a1,c1,r2) در نظر گرفته شده است که مطابق آنچه بیان کردیم، طراحی درستی است. بنابراین گزینه‌های اول، دوم و سوم را به طور کامل کنار می‌گذاریم، پس پُرواضح است که گزینه چهارم پاسخ سوال است. به همین سادگی.

تست‌های فصل چهارم: جبر رابطه‌ای

۱- پایگاه داده زیر را در نظر بگیرید: **(مهندسی IT- دولتی ۹۸)**

گره‌ها: node(NID , Name , Color , Description)

اطلاعات موجود در جدول node شامل شماره، نام، رنگ و شرح مربوط به هر گره است.

یال‌ها: edge(NID1, NID2, EdgeType)

هر سطر از جدول edge، نشان دهنده وجود یک یال جهت‌دار از نوع EdgeType از گره با شماره NID1 به گره با شماره NID2 است.

عبارت جبر رابطه‌ای زیر معادل کدام مورد است؟ (عملگر $\rho_{R2}(R1)$، نام رابطه R1 را به R2 تغییر می‌دهد.)

$$\Pi_{E1.NID1}\left(\sigma_{E1.NID2 = E2.NID1}\left(\sigma_{E1.EdgeType = 'T2'}\left(\rho_{E1}(Edge)\right) \times \rho_{E2}(Edge)\right)\right)$$

۱) شماره گره‌هایی که حداقل یک یال از نوع T2 از آنها خارج شده است.

۲) شماره گره‌هایی که حداقل یک یال خروجی از نوع T2 به یک گره مانند g دارند و گره g حداقل یک یال خروجی دارد.

۳) شماره گره‌هایی که حداقل یک یال ورودی از نوع T2 از یک گره مانند g دریافت می‌کنند و گره g حداقل یک یال خروجی دارد.

۴) شماره گره‌هایی که حداقل یک یال خروجی از نوع T2 به یک گره مانند g دارند و گره g حداقل یک یال خروجی از نوع T2 دارد.

پاسخ تست‌های فصل چهارم: جبر رابطه‌ای

۱- گزینه (۲) صحیح است.

دو جدول Node و Edge با مقادیر زیر را در نظر بگیرید:

NID	Name	Color	Description
N1	NN1	C1	D1
N2	NN2	C2	D2
N3	NN3	C3	D3
N4	NN4	C4	D4

جدول Node

NID1	NID2	EdgeType
N1	N2	T1
N1	N3	T2
N4	N1	T3
N3	N2	T4
N4	N3	T5

جدول Edge

توجه: کلید کاندید جدول Node ستون NID است و جدول Edge دارای دو کلید خارجی NID1 و NID2 است که هر دو به کلید کاندید جدول Node ارجاع می‌کنند.

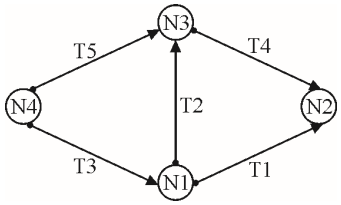
توجه: ستون NID1 در جدول Edge به عنوان کلید خارجی به ستون NID از جدول Node ارجاع می‌کند.

توجه: ستون NID2 در جدول Edge به عنوان کلید خارجی به ستون NID از جدول Node ارجاع می‌کند.

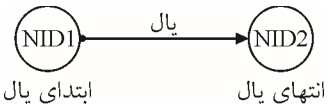
توجه: مقادیر کلید خارجی یعنی ستون‌های NID1 و NID2 از جدول Edge همواره باید زیرمجموعه مقادیر کلید کاندید یعنی ستون NID از جدول Node باشد.

توجه: هر تاپل یا سطر یا رکورد از جدول Edge نشانه یک ارتباط است، پس هر تاپل نشانه یک یال هم هست، که مجموع این یال‌ها یک گراف را ایجاد می‌کند. خطی که از یک گره به گره بعدی رسم می‌شود یک یال است. هر گراف G شامل دو مجموعه V و E است. V مجموعه محدود و غیرتهی از رئوس است و E مجموعه‌ای محدود و احتمالا تهی از لبه‌ها می‌باشد. V(G) و E(G) مجموعه رئوس و لبه‌های گراف G را نمایش می‌دهند. برای نمایش یک گراف می‌توانیم بنویسیم G=(V , E). در یک گراف بدون جهت، زوج رئوس، زوج مرتب نیستند. بنابراین زوج‌های (V_0,V_1) و (V_1,V_0) باهم یکسان هستند. گراف حداقل یک راس دارد و نمی‌تواند کاملا تهی باشد. در یک گراف جهت دار، هر لبه با زوج مرتب $<V_0,V_1>$ نمایش داده می‌شود، که پیکانی از V_0 به V_1 ترسیم می‌شود. بنابراین $<V_1,V_0>$ و $<V_0,V_1>$ دو لبه متفاوت را نمایش می‌دهند. گراف جهت دار را به صورت G=<V , E> نمایش می‌دهیم.

توجه: گراف جهت دار جدول Edge به صورت زیر است.



گراف جدول Edge



هر تاپل از جدول Edge نشانه یک یال از گراف جدول Edge است.

توجه: همانطور که واضح است گره‌های موجود در ستون NID1 از جدول Edge یا گراف Edge، **یال‌های خروجی** دارند.

NID1={N1 , N3 , N4}

یعنی از گره‌های N1، N3 و N4 یال **خارج** شده است.

توجه: همانطور که واضح است گره‌های موجود در ستون NID2 از جدول Edge یا گراف Edge، **یال‌های ورودی** دارند.

NID2={N1 , N2 , N3}

یعنی به گره‌های N1، N2 و N3 یال **وارد** شده است.

نتیجه:

استخراج گره‌هایی که هم یالی از آن **خارج** شده است و هم یالی به آن **وارد** شده است، به صورت زیر است:

$NID1 \cap NID2 = \{N1 , N3 , N4\} \cap \{N1 , N2 , N3\} = \{N1 , N3\}$

استخراج گره‌هایی که یالی از آن **خارج** شده است ولی یالی به آن **وارد** نشده است، به صورت زیر است:

$NID1 - NID2 = \{N1 , N3 , N4\} - \{N1 , N2 , N3\} = \{N4\}$

استخراج گره‌هایی که یالی به آن **وارد** شده است ولی یالی از آن **خارج** نشده است، به صورت زیر است:

$NID2 - NID1 = \{N1 , N2 , N3\} - \{N1 , N3 , N4\} = \{N2\}$

مطابق پرس و جوی مطرح شده در صورت سوال، داریم:

$$\Pi_{E1.NID1}\left(\sigma_{E1.NID2 = E2.NID1}\left(\sigma_{E1.EdgeType = 'T2'}\left(\rho_{E1}(Edge)\right) \times \rho_{E2}(Edge)\right)\right)$$

ابتدا در داخلی‌ترین پرانتز، عملوند سمت چپ عملگر ضرب دکارتی اجرا می‌شود، که خروجی آن بر اساس عملگر ρ_{E1} به جدول E1 نام‌گذاری می‌شود، به صورت زیر:

$$\sigma_{E1.EdgeType = 'T2'}\left(\rho_{E1}(Edge)\right)$$

معادل پرس و جوی جبر رابطه‌ای فوق به زبان SQL به صورت زیر است:

```
(select *
from edge
where EdgeType = 'T2')   E1
```

خروجی قطعه کد فوق به صورت زیر است:

NID1	NID2	EdgeType
N1	N3	T2

جدول E1

توجه: قطعه پرس و جوی فوق یال‌های نوع T2 را استخراج می‌کند.

در ادامه در داخلی‌ترین پرانتز، عملوند سمت راست عملگر ضرب دکارتی اجرا می‌شود، که خروجی آن بر اساس عملگر ρ_{E2} به جدول E2 نام‌گذاری می‌شود، به صورت زیر:

$$\rho_{E2}(Edge)$$

معادل پرس و جوی جبر رابطه‌ای فوق به زبان SQL به صورت زیر است:

```
(select *
from edge)   E2
```

خروجی قطعه کد فوق به صورت زیر است:

NID1	NID2	EdgeType
N1	N2	T1
N1	N3	T2
N4	N1	T3
N3	N2	T4
N4	N3	T5

جدول E2

در ادامه در داخلی‌ترین پرانتز، عملوند سمت چپ عملگر ضرب دکارتی (T1) و عملوند سمت راست عملگر ضرب دکارتی (T2) در هم ضرب دکارتی می‌شوند، که خروجی آن بر اساس عملگر $\sigma_{E1.NID2 = E2.NID1}$، به صورت زیر است:

$$\sigma_{E1.NID2 = E2.NID1}\left(\sigma_{E1.EdgeType = 'T2'}\left(\rho_{E1}(Edge)\right)\times\rho_{E2}(Edge)\right)$$

معادل پرس و جوی جبر رابطه‌ای فوق به زبان SQL به صورت زیر است:

```
select *
from (select *
      from edge
      where EdgeType= 'T2')  E1 ,
      (select *
       from edge)   E2
where  E1.NID2 = E2.NID1
```

خروجی قطعه کد فوق به صورت زیر است:

E1.NID1	E1.NID2	E1.EdgeType	E2.NID1	E2.NID2	E2.EdgeType
N1	N3	T2	N1	N2	T1
N1	N3	T2	N1	N3	T2
N1	N3	T2	N4	N1	T3
N1	**N3**	**T2**	**N3**	**N2**	**T4**
N1	N3	T2	N4	N3	T5

که خروجی آن پس از اجرای عملگر $\sigma_{E1.NID2 = E2.NID1}$، به صورت زیر است:

E1.NID1	E1.NID2	E1.EdgeType	E2.NID1	E2.NID2	E2.EdgeType
N1	**N3**	**T2**	**N3**	**N2**	**T4**

در نهایت و حرکت به سمت خارجی‌ترین پرانتز، عملگر Project اجرا می‌شود، که خروجی آن بر اساس عملگر $\Pi_{E1.NID1}$، به صورت زیر است:

$$\Pi_{E1.NID1}\left(\sigma_{E1.NID2 = E2.NID1}\left(\sigma_{E1.EdgeType = 'T2'}\left(\rho_{E1}(Edge)\right)\times\rho_{E2}(Edge)\right)\right)$$

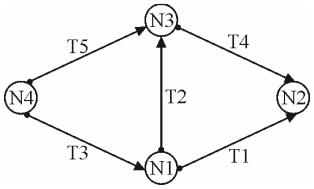
معادل پرس و جوی جبر رابطه‌ای فوق به زبان SQL به صورت زیر است:

```
select  E1.NID1
from (select *
      from edge
      where EdgeType= 'T2')  E1 ,
      (select *
       from edge)   E2
where  E1.NID2 = E2.NID1
```

که خروجی نهایی آن پس از اجرای عملگر $\Pi_{E1.NID1}$، به صورت زیر است:

E1.NID1
N1

یعنی «شماره گره‌هایی که حداقل یک یال خروجی از نوع T2 به یک گره مانند g دارند و گره g حداقل یک یال خروجی دارد.» بنابراین پُرواضح است که گزینه‌ی **دوم** پاسخ سوال است.
شکل زیر گویای مطلب است:



گراف جدول Edge

توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه دوم را به عنوان پاسخ اعلام کرده بود.

تست‌های فصل ششم: SQL دستورات DML

۱- با توجه به پایگاه داده زیر، کدام کد SQL داده شده، لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت گرفته شده است؟ **(مهندسی کامپیوتر- دولتی ۹۸)**

user (UID , name , Contact)
BOOK (BID , Title , Publisher , TotalNumber)
BORROWING (UID , BID , StartDate , EndDate)

وقتی کتابی هنوز در امانت است تاریخ خاتمه NULL است و به محض تحویل پُر می‌شود.

a)
```
with tb1 as (
  Select BORROWING.BID, COUNT(BORROWING.BID) as cnt
  From BORROWING, BOOK
  Where BOOK.BID = BORROWING.BID and EndDate is NULL
  Group by BORROWING.BID)
  Select tb1.BID
  From tb1, BOOK
  Where tb1.BID = BOOK.BID and BOOK.TotalNumber = cnt
```

b)
```
Select BID
  From BOOK b1
  Where (select COUNT (UID)
          From BORROWING
         Where BORROWING.BID = b1.BID and
               BORROWING.EndDate is NULL) = b1.TotalNumber
```

c)
```
Select BID, TotalNumber
  From BORROWING, BOOK
  Where BOOK.BID = BORROWING.BID and EndDate is NULL
  Group by BORROWING.BID
  Having COUNT(BID) = TotalNumber
```

۱) b , a ۲) c , a ۳) c , b ۴) c , b , a

۲- بانک اطلاعاتی ماشین‌ها و تصادفات روبه‌رو را در نظر بگیرید: **(مهندسی کامپیوتر- دولتی ۹۸)**

Person (SSN , name , address)
Car (License , year , model)
Accident (License , accident – date , driver , damage – amount)
Owns (SSN , License)

با توجه به بانک اطلاعاتی ماشین‌ها و تصادفات، کدام راننده پرهزینه‌ترین تصادف را داشته است؟ (نام راننده و میزان خسارت برگردانده شود به زبان SQL)

(۱)
```
(select driver, damage – amount
from Accident) Except (select a.driver, a.damage – amont
                       from Accident a, Accident b
                       where a.damage – amount < b.damage – amount and a.driver <> b.driver
```

(۲)
```
select driver, damage – amount
from Accident
where damage – amount in (select MAX(damage – amount)
                          from Accident)
```

(۳)
```
select driver, damage – amount
from Accident
where damage – amount = MAX(damage – amount)
```

۴) همه موارد صحیح است.

۳- بانک اطلاعاتی ماشین‌ها و تصادفات روبه‌رو را در نظر بگیرید: (مهندسی کامپیوتر- دولتی ۹۸)

Person (SSN , name , address)
Car (License , year , model)
Accident (License , accident – date , driver, damage – amount)
Owns (SSN , License)

با توجه به بانک اطلاعاتی ماشین‌ها و تصادفات، کدام شماره پلاک ماشین در بیش از یک تصادف، درگیر بوده است؟

(پاسخ به زبان SQL و سطرهای تکراری فقط یک‌بار نشان داده شوند.)

(۱)
```
select distinct A.License
from Accident A
where A.License in (select B.License
                    from Accident B
                    where A.accident – date <> B.accident – date)
```

(۲)
```
select License
from Accident
group by License
having count (accident – date) > 1
```

(۳)
```
select A.License
from Accident A, Accident B
where A.License = B.License and A.accident – date <> B.accident – date
```

۴) موارد ۱ و ۲ صحیح است.

۴- پایگاه داده زیر را در نظر بگیرید: (مهندسی IT- دولتی ۹۸)

گره‌ها: node (NID , Name , Color , Description)

اطلاعات موجود در جدول node شامل شماره، نام، رنگ و شرح مربوط به هر گره است.

یال‌ها: edge (NID1, NID2, EdgeType)

هر سطر از جدول edge، نشان دهنده وجود یک یال جهت‌دار از نوع EdgeType از گره با شماره NID1 به گره با شماره NID2 است.

در خصوص پرس و جوهای SQL زیر کدام گزینه درست است؟

پرس و جوی اول	select distinct NID from node, edge where NID=edge.NID2 AND not exists (select * from edge where edge.NID1= NID)
پرس و جوی دوم	select T1.NID from (select count (NID1) as cnt , NID from node left outer join edge on edge.NID1=NID group by NID) T1 , (select count (NID2) as cnt , NID from node left outer join edge on edge.NID2=NID group by NID) T2 where T1.NID = T2.NID and T1.cnt < T2.cnt

۱) پرس و جوی اول، شماره گره‌هایی را می‌دهد که یال خروجی دارند اما یال ورودی ندارند.

۲) پرس و جوی دوم، شماره گره‌هایی را می‌دهد که درجه ورودی آنها کمتر از درجه خروجی آنها است.

۳) پرس و جوی دوم، شماره گره‌هایی را می‌دهد که درجه خروجی آنها کمتر از درجه ورودی آنها است.

۴) گزینه‌های ۱ و ۳ صحیح هستند.

پاسخ تست‌های فصل ششم: SQL دستورات DML

۱- گزینه (۱) صحیح است.

سه جدول USER، BOOK و BORROWING با مقادیر زیر را در نظر بگیرید:

UID	NAME	…	…
U1	Un1		
U2	Un2		
U3	Un3		
U4	Un4		
U5	UN5		

جدول USER

BID	TITLE	PUBLISHER	TOTALNUMBER
B1	Tn1	Pub1	1
B2	Tn2	Pub2	2
B3	Tn3	Pub3	3

جدول BOOK

UID	BID	StartDate	EndDate
U1	B1	95-01-1	NULL
U2	B2	95-02-1	NULL
U3	B2	95-03-1	NULL
U4	B3	95-04-1	NULL
U5	B3	95-05-1	NULL

جدول BORROWING

مطابق پرس و جوی مطرح شده در گزاره‌ی a، داریم:

```
a) with tb1 as (
   Select BORROWING.BID, COUNT(BORROWING.BID) as cnt
   From BORROWING, BOOK
   Where BOOK.BID = BORROWING.BID and EndDate is NULL
   Group by  BORROWING.BID)
   Select tb1.BID
   From tb1, BOOK
   Where tb1.BID = BOOK.BID  and  BOOK.TotalNumber = cnt
```

توجه: دستور with کل پرس و جوی مقابلش را tb1 نام گذاری می‌کند. که در From دوم به طور خلاصه‌تر، ساده‌تر و خواناتر مورد استفاده قرار می‌گیرد.

با توجه به جداول فوق، خروجی بخش اول پرس و جوی گزاره‌ی a پس از انجام عملگر ضرب دکارتی به صورت زیر است:

UID	BID	StartDate	EndDate	BID	TITLE	PUBLISHER	TOTALNUMBER
U1	B1	95-01-1	NULL	B1	Tn1	Pub1	1
U2	B2	95-02-1	NULL	B2	Tn2	Pub2	2
U3	B2	95-03-1	NULL	B2	Tn2	Pub2	2
U4	B3	95-04-1	NULL	B3	Tn3	Pub3	3
U5	B3	95-05-1	NULL	B3	Tn3	Pub3	3

همچنین در ادامه، پس از انجام دستور Group By BORROWING.BID براساس ستون BORROWING.BID خروجی پرس و جو به صورت زیر گروه‌بندی می‌شود:

B1
U1 95-01-1 Tn1 ...
گروه اول

B2
U2 95-02-1 Tn2 ...
U3 95-03-1 Tn2 ...
گروه دوم

B3
U4 95-04-1 Tn3 ...
U5 95-05-1 Tn3 ...
گروه سوم

در ادامه دستور COUNT(BORROWING.BID) داخل دستور SELECT برای هر گروه به طور مستقل محاسبه می‌گردد و در خروجی پرس و جو قرار می‌گیرد.
بنابراین خروجی نهایی بخش اول پرس و جوی گزاره‌ی a به صورت زیر است:

BID	cnt
B1	1
B2	2
B3	2

توجه: دستور GROUP BY، سرگروه‌ها را، راهی خروجی می‌کند.
توجه: نتیجه اینکه خروجی بخش اول پرس و جوی گزاره‌ی a پس از انجام عملگر ضرب دکارتی و اجرای دستور Group By شامل دو ستون کد کتاب (BID) و تعداد نسخه‌های به امانت رفته کتاب است که توسط دستور with به جدول tb1 نام‌گذاری شده است.
بخش دوم پرس و جوی گزاره‌ی a را نظر بگیرید:

```
Select tb1.BID
From tb1, BOOK
Where tb1.BID = BOOK.BID and BOOK.TotalNumber = cnt
```

با توجه به جداول فوق، خروجی بخش دوم پرس و جوی گزاره‌ی a پس از انجام عملگر ضرب دکارتی به صورت زیر است:

BID	cnt	BID	TITLE	PUBLISHER	TOTALNUMBER
B1	**1**	B1	Tn1	Pub1	**1**
B2	**2**	B2	Tn2	Pub2	**2**
B3	2	B3	Tn3	Pub3	3

توجه: پس از انجام عمل ضرب دکارتی مابین جدول tb1 و جدول BOOK ، شرط اصلی پرس و جو اجرا می‌گردد که منجر به استخراج کد کتاب‌هایی می‌گردد که تعداد نسخه‌های به امانت رفته آنها برابر با تعداد کل نسخه‌های آن کتاب است. در واقع لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت

گرفته شده است.

در ادامه پس از اجرای شرط اصلی پرس و جو یعنی BOOK.TotalNumber = cnt خروجی پرس و جو به صورت زیر خواهد بود:

BID	cnt	BID	TITLE	PUBLISHER	TOTALNUMBER
B1	1	B1	Tn1	Pub1	1
B2	2	B2	Tn2	Pub2	2

و در نهایت پس از اجرای دستور Select tb1.BID خروجی پرس و جو به صورت زیر خواهد بود:

BID
B1
B2

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت گرفته شده است.»

مطابق پرس و جوی مطرح شده در گزاره b، داریم:

```
b) Select BID
  From BOOK b1
  Where (select COUNT (UID)
        From BORROWING
        Where  BORROWING.BID = b1.BID and
              BORROWING.EndDate is NULL) = b1.TotalNumber
```

با توجه به وجود عملگر تساوی، زیر پرس و جوی داخلی فوق یک Correlated Subquery است، یعنی به ازای حرکت در هریک از سطرهای پرس و جوی خارجی، یک بار به طور کامل از ابتدا تا انتها زیر پرس و جوی داخلی اجرا و بر اساس شرطی که زیر پرس و جوی داخلی را به پرس و جوی خارجی متصل می‌کند، بررسی انجام می‌شود. مانند دو حلقه تو در تو، که به ازای هربار اجرای حلقه خارجی، یک بار به طور کامل حلقه داخلی اجرا می‌گردد.

توسط دستور where در فرم زیر:

```
SELECT BID
FROM Book b1
WHERE (…)
```

برای هر سطر از جدول Book شرط تساوی که حاصل یک مقایسه می‌باشد، محاسبه می‌گردد، اگر برابر بود، شرط جلوی where که همان تساوی است، TRUE می‌گردد و سطر مورد نظر از جدول Book انتخاب می‌گردد و این رویه برای تک تک سطرهای جدول Book، تا به انتهای جدول Book ادامه پیدا می‌کند. به بیان دیگر این پرس و جو شماره کتاب‌هایی را می‌دهد که پرانتز مقابل where برای آن‌ها TRUE است. این پرانتز هنگامی TRUE می‌شود که حاصل مقایسه بیان شده در این پرانتز برابر شود. حاصل این مقایسه در صورتی برابر می‌شود که تعداد نسخه‌های به امانت رفته کتاب مورد بررسی برابر کل نسخه‌های کتاب مورد بررسی باشد. به عبارت دیگر پرس و جوی گزاره‌ی b لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت گرفته شده است. که مطابق پرس و جوی مطرح شده در صورت سوال است.

به بیان دیگر عبارت موجود در جلوی دستور where که به صورت زیر است:

```
Select BID
From BOOK b1
Where (select COUNT (UID)
      From BORROWING
      Where  BORROWING.BID = b1.BID and
             BORROWING.EndDate is NULL) = b1.TotalNumber
```

توسط دستور count (UID) تعداد کتاب‌های به امانت رفته را می‌شمارد و محاسبه می‌کند.
همچنین عبارت موجود در جلوی دستور where که به صورت زیر است:

```
Select BID
From BOOK b1
Where (select COUNT (UID)
      From BORROWING
      Where  BORROWING.BID = b1.BID and
             BORROWING.EndDate is NULL) = b1.TotalNumber
```

توسط دستور BORROWING.BID = b1.BID به محیط خارج یعنی جدول Book متصل می‌گردد. حال به ازای حرکت در هر سطر از جدول Book، یک بار به طور کامل سطرهای جدول BORROWING از ابتدا تا انتها با توجه به شرط اتصال بررسی می‌گردد. مطابق شکل زیر:

ابتدا برای سطر اول از جدول Book با توجه به شرط اتصال ~~BORROWING.BID~~ = b1.~~BID~~ داریم:

B1 B1

BID	...	**TOTALNUMBER**
B1		**1**
B2		2
B3		3

BOOK

UID	**BID**	…	EndDate
U1	**B1**		**NULL**
U2	B2		NULL
U3	B2		NULL
U4	B3		NULL
U5	B3		NULL

BORROWING

UID	…
U1	
U2	
U3	
U4	
U5	

USER

که پرس و جوی زیر برای آن اجرا می‌شود:

```
Select COUNT (UID)
      From BORROWING
      Where  BORROWING.BID = b1.BID and
             BORROWING.EndDate is NULL
```

توجه: مقدار count (UID) برای سطر اول جدول Book یعنی B1 برابر مقدار 1 است. که رابطه‌ی برابری برای آن برقرار است، به صورت زیر:

~~count (UID)~~ = b1.~~TotalNumber~~

1 1

شرط تساوی جلوی where برقرار است، بنابراین شرط where، TRUE می‌گردد. بنابراین سطر اول از جدول Book در خروجی نمایش داده می‌شود.

بنابراین کد کتاب **B1** جهت نمایش در خروجی انتخاب می‌شود.

حال برای سطر دوم از جدول Book با توجه به شرط اتصال ~~BID~~ = b1.~~BID~~ BORROWING. داریم:

B2 B2

BID	...	**TOTALNUMBER**
B1		1
B2		**2**
B3		3

BOOK

UID	**BID**	...	EndDate
U1	B1		NULL
U2	**B2**		**NULL**
U3	**B2**		**NULL**
U4	B3		NULL
U5	B3		NULL

BORROWING

UID	...
U1	
U2	
U3	
U4	
U5	

USER

که پرس و جوی زیر برای آن اجرا می‌شود:

```
Select COUNT (UID)
        From BORROWING
        Where  BORROWING.BID = b1.BID and
               BORROWING.EndDate is NULL
```

توجه: مقدار count (UID) برای سطر دوم جدول Book یعنی B2 برابر مقدار 2 است. که رابطه‌ی برابری برای آن برقرار است، به صورت زیر:

~~count (UID)~~ = b1.~~TotalNumber~~

2 2

شرط تساوی جلوی where برقرار است، بنابراین شرط where، TRUE می‌گردد. بنابراین سطر دوم از جدول Book در خروجی نمایش داده می‌شود.

بنابراین کد کتاب **B2** جهت نمایش در خروجی انتخاب می‌شود.

حال برای سطر سوم از جدول Book با توجه به شرط اتصال ~~BID~~ = b1.~~BID~~ BORROWING. داریم:

B3 B3

BID	...	**TOTALNUMBER**
B1		1
B2		2
B3		**3**

BOOK

UID	**BID**	...	EndDate
U1	B1		NULL
U2	B2		NULL
U3	B2		NULL
U4	**B3**		**NULL**
U5	**B3**		**NULL**

BORROWING

UID	...
U1	
U2	
U3	
U4	
U5	

USER

که پرس و جوی زیر برای آن اجرا می‌شود:

```
Select count (UID)
        From BORROWING
        Where  BORROWING.BID = b1.BID and
               BORROWING.EndDate is NULL
```

توجه: مقدار count (UID) برای سطر سوم جدول Book یعنی B3 برابر مقدار 2 است. که رابطه‌ی برابری برای آن برقرار نیست، به صورت زیر:

$$\cancel{\text{count (UID)}} \neq \text{b1.}\cancel{\text{TotalNumber}}$$
$$2 \qquad 3$$

شرط تساوی جلوی where برقرار نیست، بنابراین شرط where، FALSE می‌گردد. بنابراین سطر سوم از جدول Book در خروجی نمایش داده نمی‌شود.

بنابراین کد کتاب **B3** جهت نمایش در خروجی انتخاب نمی‌شود.
و در نهایت پس از اجرای کامل دستور Select BID خروجی پرس و جو به صورت زیر خواهد بود:

BID
B1
B2

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت گرفته شده است.»

مطابق پرس و جوی مطرح شده در گزاره c، داریم:

```
c) Select BID, TotalNumber
   From BORROWING, BOOK
   Where BOOK.BID = BORROWING.BID and EndDate is NULL
   Group by BORROWING.BID
   Having  COUNT(BID) = TotalNumber
```

که البته پرس و جوی مطرح شده در گزاره c کمی خطای نحوی دارد، که فرم اصلاح شده آن به صورت زیر است:

```
Select BORROWING.BID, TotalNumber
  From BORROWING, BOOK
  Where BOOK.BID = BORROWING.BID and EndDate is NULL
  Group by BORROWING.BID, TotalNumber
  Having  COUNT(BORROWING.BID) = TotalNumber
```

توجه: در فرم اصلاح شده‌ی گزاره c، نام جدول BORROWING پشت ستون BID در دستور select قرار گرفت، که کامپایلر متوجه شود ستون BID جدول BOOK یا BORROWING را انتخاب کند، پرس و جوی گزاره c تا به اینجا یک خطای نحوی داشت که اصلاح شد. اما یک **خطای نحوی پررنگ‌تری** هم دارد، اینکه همواره به غیر از توابع آماری، همه ستون‌های جلوی دستور Select باید زیرمجموعه یا مساوی ستون‌های دستور Group By باشد، بنابراین در فرم اصلاح شده‌ی گزاره c، ستون TotalNumber جلوی دستور Group By قرار گرفت. گزاره c تا به اینجا دو خطای نحوی داشت که اصلاح شد. همچنین نام جدول BORROWING پشت ستون BID در دستور having قرار گرفت، که کامپایلر متوجه شود ستون BID جدول BOOK یا BORROWING را انتخاب کند، نتیجه اینکه پرس و جوی گزاره c سه خطای نحوی داشت که اصلاح شد.

با توجه به جداول فوق، خروجی بخش اول پرس و جوی گزاره‌ی **C** پس از انجام عملگر ضرب دکارتی به صورت زیر است:

UID	BID	StartDate	EndDate	BID	TITLE	PUBLISHER	TOTALNUMBER
U1	B1	95-01-1	NULL	B1	Tn1	Pub1	1
U2	B2	95-02-1	NULL	B2	Tn2	Pub2	2
U3	B2	95-03-1	NULL	B2	Tn2	Pub2	2
U4	B3	95-04-1	NULL	B3	Tn3	Pub3	3
U5	B3	95-05-1	NULL	B3	Tn3	Pub3	3

همچنین در ادامه، پس از انجام دستور Group By BORROWING.BID , TotalNumber براساس ستون‌های BORROWING.BID و TotalNumber خروجی پرس و جو به صورت زیر گروه‌بندی می‌شود:

B1, 1
U1 95-01-1 NULL Tn1 Pub1
گروه اول

B2, 2
U2 95-02-1 NULL Tn2 Pub2
U3 95-03-1 NULL Tn2 Pub2
گروه دوم

B3, 3
U4 95-04-1 NULL Tn3 Pub3
U5 95-05-1 NULL Tn3 Pub3
گروه سوم

و در نهایت دستور having count(BORROWING.BID) = TotalNumber برای هر گروه به طور مستقل اعمال می‌گردد.

توجه: دستور HAVING بر روی گروه‌ها، اعمال می‌گردد.

B2, 2
U2 95-02-1 NULL Tn2 Pub2
U3 95-03-1 NULL Tn2 Pub2
گروه دوم

$2 = 2$

B1, 1
U1 95-01-1 NULL Tn1 Pub1
گروه اول

$1 = 1,$

B3, 3
U4 95-04-1 NULL Tn3 Pub3
U5 95-05-1 NULL Tn3 Pub3
گروه سوم

$2 \neq 3$

توجه: با توجه به شرط انتخاب گروه توسط دستور having count(BORROWING.BID) = TotalNumber ، فقط گروه **اول و دوم** جهت نمایش در خروجی انتخاب می‌شود.

و در نهایت دستور BORROWING.BID,TotalNumber داخل دستور select برای هر گروه انتخاب شده

توسط دستور Having به طور مستقل اعمال می‌گردد و در خروجی پرس و جو قرار می‌گیرد، بنابراین خروجی نهایی پرس و جو به صورت زیر است:

BID	TOTALNUMBER
B1	1
B2	2

توجه: دستور GROUP BY، سرگروه‌ها را، راهی خروجی می‌کند.

که مطابق پرس و جوی مطرح شده در صورت سوال نیست. یعنی «لیست تمام کتاب‌هایی را نمایش می‌هد که تمام نسخه‌های آنها امانت گرفته شده است.» در واقع پرس و جوی گزاره c به دو دلیل مطابق پرس و جوی مطرح شده در صورت سوال نیست؛ اول اینکه خطای نحوی داشت که جهت اجرا می‌بایست اصلاح می‌شد که ما به فرم اصلاح شده آنرا بررسی کردیم، هرچند که در My-SQL این فرم خطای نحوی صادر نمی‌شود. دوم اینکه در صورت سوال صرفا **شماره کتاب‌هایی** خواسته شده که همه نسخه‌های آنها به امانت گرفته شده است، درحالیکه در این پرس و جو علاوه بر شماره کتاب (BID)، تعداد کل نسخه‌های هر کتاب هم در خروجی ظاهر می‌شود که این بر خلاف صورت سوال است. بنابراین فقط و فقط گزاره‌های a و b مطابق پرس و جوی صورت سوال است که به تبع گزینه‌های دوم، سوم و چهارم را به طور کامل کنار می‌گذاریم، پس پُرواضح است که گزینه اول پاسخ سوال است.

۲- گزینه (۲) صحیح است.

چهار جدول Person، Car، Accident و Owns با مقادیر زیر را در نظر بگیرید:

SSN	name	address

جدول Person

License	year	model

جدول Car

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L2	95-02-1	d2	20
L3	95-03-1	d3	30

جدول Accident

SSN	License

جدول Owns

توجه: دقت کنید که مطابق فرض سوال، ترکیب دو ستون License و accident-date هردو باهم به عنوان کلید کاندید جدول Accident مشخص شده است، بنابراین نباید ترکیب این دو ستون باهم سطرهای تکراری داشته باشد، پس یک خودرو (License) در یک تاریخ تصادف (accident-date) نمی‌تواند بیش از

یکبار تکرار شود، یعنی نباید دوبار و در دو سطر و بیشتر تکرار شود.

مطابق پرس و جوی مطرح شده در گزینه‌ی اول، داریم:

```
(select  driver, damage – amount
from Accident) Except (select a.driver, a.damage – amont
                       from Accident a, Accident b
                       where a.damage – amount < b.damage – amount and a.driver <> b.driver
```

با توجه به جداول فوق، خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی به صورت زیر است:

License	accident-date	driver	damage-amount	License	accident-date	driver	damage-amount
L1	95-01-1	d1	10	L1	95-01-1	d1	10
L1	95-01-1	d1	10	L2	95-02-1	d2	20
L1	95-01-1	d1	10	L3	95-03-1	d3	30
L2	95-02-1	d2	20	L1	95-01-1	d1	10
L2	95-02-1	d2	20	L2	95-02-1	d2	20
L2	95-02-1	d2	20	L3	95-03-1	d3	30
L3	95-03-1	d3	30	L1	95-01-1	d1	10
L3	95-03-1	d3	30	L2	95-02-1	d2	20
L3	95-03-1	d3	30	L3	95-03-1	d3	30

خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی و اجرای شرط where a.damage – amount < b.damage – amount and a.driver <> b.driver به صورت زیر است:

License	accident-date	driver	damage-amount	License	accident-date	driver	damage-amount
L1	95-01-1	d1	10	L2	95-02-1	d2	20
L1	95-01-1	d1	10	L3	95-03-1	d3	30
L2	95-02-1	d2	20	L3	95-03-1	d3	30

خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی و اجرای شرط where و اجرای دستور select a.driver, a.damage – amont به صورت زیر است:

driver	damage-amount
d1	10
d1	10
d2	20

توجه: شرط where مطرح شده، نام رانندگانی را مشخص می‌کند که هزینه تصادف آنها از هزینه تصادف حداقل یک راننده دیگر کمتر باشد، به عبارت دیگر شرط where نام رانندگان با هزینه تصادف MAX را حذف می‌کند. در واقع نام رانندگانی که هزینه تصادف آنها MAX یعنی **بیشینه** نیست استخراج می‌شود.

همچنین در ادامه، خروجی بخش اول پرس و جوی گزینه‌ی اول به صورت زیر است:

```
select  driver, damage – amount
from Accident
```

driver	damage-amount
d1	10
d2	20
d3	30

توجه: در پرانتز سمت راست عملگر تفاضل (Except) نام رانندگانی مشخص می‌شود که هزینه تصادف آنها از هزینه تصادف حداقل یک راننده دیگر کمتر بوده است. همچنین در پرانتز سمت چپ عملگر تفاضل (Except) نام کلیه رانندگانی که هزینه تصادفی داشته‌اند، مشخص می‌شود. پس از انجام عملگر تفاضل، نام رانندگانی در خروجی پرس و جو استخراج می‌شوند که هزینه تصادفی داشته‌اند و هزینه‌ی تصادف آنها از هزینه تصادف حداقل یک راننده دیگر کمتر نبوده‌است. به عبارت دیگر در خروجی پرس و جوی گزینه‌ی اول نام رانندگانی استخراج می‌شود که هزینه تصادف آنها MAX و **بیشینه** بوده است.

در نهایت پس از انجام عملگر Except خروجی پرس و جو به صورت زیر خواهد بود:

driver	damage-amount
d1	10
d2	20
d3	30

Except

driver	damage-amount
d1	10
d1	10
d2	20

=

driver	damage-amount
d3	30

عملگر تفاضل (Except)

این عملگر توسط دستور Except نمایش داده می‌شود. عملگر Except یک عملگر اصلی است. عملگر Except جهت تفاضل سطرهای دو جدول مورد استفاده قرار می‌گیرد. اگر R_1 و R_2 دو رابطه باشند، منظور از R_1 Except R_2 مجموعه کلیه تاپل‌هایی است که عضو R_1 هستند اما در R_2 حضور ندارند. در جبر رابطه‌ای و SQL تفاضل هر دو رابطه دلخواه امکان‌پذیر نیست. مگر اینکه شروط سازگاری در مورد آنها برقرار باشد. در جبر رابطه‌ای و SQL 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 Except به صورت زیر است:

$R_3 = R_1$ Except R_2

توجه: در پرس و جوی گزینه‌ی اول شروط سازگاری در طرفین عملگر تفاضل (Except) برقرار است.

همچنین جدول Accident با مقادیر زیر را در نظر بگیرید:

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L1	95-02-1	d1	20
L1	95-03-1	d1	30

جدول Accident

توجه: در جدول فوق، فقط تصادف‌های یک راننده بررسی شده است.

مطابق پرس و جوی مطرح شده در گزینه‌ی اول، داریم:

```
(select  driver, damage – amount
from Accident) Except (select a.driver, a.damage – amont
                       from Accident a, Accident b
                       where a.damage – amount < b.damage – amount and a.driver <> b.driver
```

با توجه به جداول فوق، خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی به صورت زیر است:

License	accident-date	driver	damage-amount	License	accident-date	driver	damage-amount
L1	95-01-1	d1	10	L1	95-01-1	d1	10
L1	95-01-1	d1	10	L1	95-02-1	d1	20
L1	95-01-1	d1	10	L1	95-03-1	d1	30
L1	95-02-1	d1	20	L1	95-01-1	d1	10
L1	95-02-1	d1	20	L1	95-02-1	d1	20
L1	95-02-1	d1	20	L1	95-03-1	d1	30
L1	95-03-1	d1	30	L1	95-01-1	d1	10
L1	95-03-1	d1	30	L1	95-02-1	d1	20
L1	95-03-1	d1	30	L1	95-03-1	d1	30

خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی و اجرای شرط where a.damage – amount < b.damage – amount and a.driver <> b.driver به صورت زیر است:

License	accident-date	driver	damage-amount	License	accident-date	driver	damage-amount

توجه: علت تهی شدن خروجی فوق این است که در شرط where شاید a.damage – amount < b.damage – amount برقرار باشد، اما a.driver <> b.driver برقرار نیست، چون فقط یک راننده وجود دارد که مقدار driver آنها مخالف هم نیستند.

خروجی بخش دوم پرس و جوی گزینه‌ی اول پس از انجام عملگر ضرب دکارتی و اجرای شرط where و اجرای دستور select a.driver, a.damage – amont به صورت زیر است:

driver	damage-amount

همچنین در ادامه، خروجی بخش اول پرس و جوی گزینه‌ی اول به صورت زیر است:

```
select  driver, damage – amount
from Accident
```

driver	damage-amount
d1	10
d1	20
d1	30

در نهایت پس از انجام عملگر Except خروجی پرس و جو به صورت زیر خواهد بود:

driver	damage-amount
d1	10
d1	20
d1	30

Except

driver	damage-amount

=

driver	damage-amount
d1	10
d1	20
d1	30

توجه: نتیجه اینکه گزینه اول در بررسی هزینه‌های تصادف چند راننده مختلف، حالت هزینه بیشینه تصادف را میان رانندگان مختلف استخراج می‌کند، اما در بررسی هزینه‌های تصادف یک راننده، هزینه بیشینه تصادف را مشخص نمی‌کند. بنابراین گزینه اول پاسخ سوال نیست.

مطابق پرس و جوی مطرح شده در گزینه‌ی دوم، داریم:

```
select  driver, damage – amount
from Accident
where damage – amount in (select MAX(damage – amount)
                          from Accident)
```

با توجه به وجود دستور in، زیر پرس و جوی داخلی فوق یک Normal Subquery است، یعنی ابتدا زیر پرس‌وجوی داخلی یک بار و برای همیشه اجرا می‌گردد، سپس پرس‌وجوی خارجی به ازای حرکت در هر یک از سطرهای خود، از مقادیر زیر پرس‌وجوی داخلی استفاده می‌کند.

در پرس‌وجوی فوق به ازای حرکت در هر سطر از جدول Accident مقدار جلوی in بررسی می‌گردد که آیا damage-amount برابر MAX(damage-amount) است یا خیر. اگر برابر بود سطر مورد نظر از جدول Accident در خروجی نمایش داده می‌شود.

توسط دستور where در فرم زیر:

```
SELECT driver , damage-amount
FROM Accident
WHERE  damage-amount  in (…)
```

برای هر سطر از جدول Accident مقدار جلوی in که حاصل یک تابع آماری به صورت MAX(damage-amount) است، بررسی می‌گردد، اگر damage-amount موجود در هر سطر، برابر MAX(damage-amount) جلوی in بود، آنگاه شرط جلوی where که همان in است، TRUE می‌گردد و سطر مورد نظر از جدول Accident انتخاب می‌گردد و این رویه برای تک تک سطرهای جدول Accident، تا به انتهای جدول Accident ادامه پیدا می‌کند. به بیان دیگر این پرس و جو نام رانندگانی از جدول Accident را می‌دهد که در شرط پرانتز مقابل in قرار دارند.

به عبارت دیگر در خروجی پرس و جوی گزینه‌ی دوم نام رانندگانی استخراج می‌شود که هزینه تصادف آنها MAX و **بیشینه** بوده است. که مطابق پرس و جوی مطرح شده در صورت سوال است.
توجه: همان‌طور که مشاهده می‌کنید، هزینه تصادف راننده d3، از هزینه تصادف تمام رانندگان دیگر بیشتر است.

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L2	95-02-1	d2	20
L3	**95-03-1**	**d3**	**30**

با توجه به جداول فوق، ابتدا خروجی زیر پرس و جوی داخلی بر اساس جدول Accident به صورت زیر محاسبه می‌گردد:

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L2	95-02-1	d2	20
L3	**95-03-1**	**d3**	**30**

```
Select MAX(damage-amount)
From Accident
```

که مقدار آن برابر 30 می‌شود، به صورت زیر:

$$30 > 20 > 10$$

بنابراین در ادامه پرس و جوی زیر را خواهیم داشت:

```
SELECT driver , damage-amount
FROM Accident
WHERE  damage-amount  in (30)
```

همانطور که گفتیم برای هر سطر از جدول Accident مقدار جلوی in که حاصل یک تابع آماری به صورت MAX(damage-amount) است، بررسی می‌گردد، اگر damage-amount موجود در هر سطر، برابر MAX(damage-amount) جلوی in بود، آنگاه شرط جلوی where که همان in است، TRUE می‌گردد و سطر مورد نظر از جدول Accident انتخاب می‌گردد و این رویه برای تک تک سطرهای جدول Accident، تا به انتهای جدول Accident ادامه پیدا می‌کند. به بیان دیگر این پرس و جو نام رانندگانی از جدول Accident را می‌دهد که در شرط پرانتز مقابل in قرار دارند.
با توجه به جداول فوق، خروجی نهایی پرس و جوی فوق پس از انجام عملگر in به ازای هر سطر جدول Accident به صورت زیر است:

driver	damage-amount
d3	30

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «نام رانندگانی استخراج می‌شود که هزینه تصادف آنها MAX و **بیشینه** بوده است.»

همچنین جدول Accident با مقادیر زیر را در نظر بگیرید:

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L1	95-02-1	d1	20
L1	**95-03-1**	**d1**	**30**

جدول Accident

با توجه به جداول فوق، خروجی نهایی پرس و جوی فوق پس از انجام عملگر in به ازای هر سطر جدول Accident به صورت زیر است:

driver	damage-amount
d3	30

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «نام رانندگانی استخراج می‌شود که هزینه تصادف آنها MAX و **بیشینه** بوده است.»

توجه: نتیجه اینکه گزینه دوم در بررسی هزینه‌های تصادف چند راننده مختلف، حالت هزینه بیشینه تصادف را میان رانندگان مختلف استخراج می‌کند و همچنین در بررسی هزینه‌های تصادف یک راننده هم، هزینه بیشینه تصادف را مشخص می‌کند. بنابراین گزینه دوم پاسخ سوال است.

توجه: در پرس و جوی گزینه‌ی دوم چون خروجی select داخلی فقط یک مقدار و حاصل تابع آماری MAX است، می‌توان از عملگر = به جای عملگر in نیز استفاده نمود، بنابراین فرم زیر همانند فرم گزینه دوم است.

```
select  driver, damage – amount
from Accident
where damage – amount = (select MAX(damage – amount)
                         from Accident)
```

مطابق پرس و جوی مطرح شده در گزینه‌ی سوم، داریم:

```
select  driver, damage – amount
from Accident
where damage – amount = MAX(damage – amount)
```

در پرس و جوی گزینه‌ی سوم تابع عددی MAX در محل نادرست مورد استفاده قرار گرفته است، به طور کلی توابع عددی در جلوی Where باید داخل Select قرار بگیرند. بنابراین پرس‌وجوی گزینه‌ی سوم دارای خطای نحوی است و از سوی کامپایلر اجرا نمی‌گردد. فرم اصلاح شده‌ی گزینه‌ی سوم می‌تواند به فرم گزینه‌ی دوم باشد. بنابراین گزینه سوم پاسخ سوال نیست.

توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه دوم را به عنوان پاسخ اعلام کرده بود.

۳- گزینه (۴) صحیح است.

چهار جدول Person، Car، Accident و Owns با مقادیر زیر را در نظر بگیرید:

SSN	name	address

جدول Person

License	year	model

جدول Car

License	accident-date	driver	damage-amount
L1	95-01-1	d1	10
L1	95-01-2	d1	15
L2	95-02-1	d2	20

جدول Accident

SSN	License

جدول Owns

توجه: دقت کنید که مطابق فرض سوال، ترکیب دو ستون License و accident-date هردو باهم به عنوان کلید کاندید جدول Accident مشخص شده است، بنابراین نباید ترکیب این دو ستون باهم سطرهای تکراری داشته باشد، پس یک خودرو (License) در یک تاریخ تصادف (accident-date) نمی‌تواند بیش از یکبار تکرار شود، یعنی نباید دوبار و در دو سطر و بیشتر تکرار شود.

مطابق پرس و جوی مطرح شده در گزینه‌ی اول، داریم:

```
select distinct A.License
from Accident A
where A.License in (select B.License
                    from Accident B
                    where A.accident – date <> B.accident – date)
```

با توجه به وجود عملگر in و **البته شرط اتصال زیر پرس و جوی داخلی به پرس و جوی خارجی**، زیر پرس و جوی داخلی فوق یک Correlated Subquery است، یعنی به ازای حرکت در هریک از سطرهای پرس و جوی خارجی، یک بار به طور کامل از ابتدا تا انتها زیر پرس و جوی داخلی اجرا و بر اساس شرطی که زیر پرس و جوی داخلی را به پرس و جوی خارجی متصل می‌کند، بررسی انجام می‌شود. مانند دو حلقه تو در تو، که به ازای هربار اجرای حلقه خارجی، یک بار به طور کامل حلقه داخلی اجرا می‌گردد. توسط دستور where در فرم زیر:

```
SELECT distinct A.License
FROM Accident A
WHERE A.License IN (…)
```

برای هر سطر از جدول Accident شرط جلوی IN که حاصل یک مقایسه می‌باشد، محاسبه می‌گردد، اگر غیرتهی بود، شرط جلوی where که همان IN است، TRUE می‌گردد و سطر مورد نظر از جدول Accident انتخاب می‌گردد و این رویه برای تک تک سطرهای جدول Accident، تا به انتهای جدول Accident ادامه پیدا می‌کند. به بیان دیگر این پرس و جو شماره پلاک ماشین‌هایی را می‌دهد که پرانتز مقابل IN برای آن‌ها غیرتهی است. این پرانتز هنگامی غیرتهی می‌شود که حاصل مقایسه بیان شده در این پرانتز غیرتهی شود. حاصل این مقایسه در صورتی غیرتهی می‌شود که شماره پلاک ماشین مورد بررسی در بیش از یک تصادف، درگیر باشد. به عبارت دیگر پرس و جوی گزینه‌ی اول شماره پلاک ماشین‌هایی را استخراج می‌کند که در بیش از یک تصادف، درگیر هستند. که مطابق پرس و جوی مطرح شده در صورت سوال است.

به بیان دیگر عبارت موجود در جلوی دستور where که به صورت زیر است:

```
select distinct A.License
from Accident A
where A.License in (select B.License
                    from Accident B
                    where A.accident – date<>B.accident – date)
```

توسط دستور A.accident – date<>B.accident – date به محیط خارج یعنی جدول Accident A متصل می‌گردد. حال به ازای حرکت در هر سطر از جدول Accident A، یک بار به طور کامل سطرهای جدول Accident B از ابتدا تا انتها با توجه به شرط اتصال بررسی می‌گردد. مطابق شکل زیر:

ابتدا برای سطر اول از جدول Accident A با توجه به شرط اتصال $A.\underbrace{accident-date}_{L1} = B.\underbrace{accident-date}_{L1}$ داریم:

License	Accident.date	…
L1	95-01-1	
L1	95-01-2	
L2	95-02-1	

Accident A

License	Accident.date	…
L1	95-01-1	
L1	**95-01-2**	
L2	95-02-1	

Accident B

که پرس و جوی زیر برای آن اجرا می‌شود:

```
select distinct A.License
from Accident A
where A.License in (select B.License
                    from Accident B
                    where A.accident – date<>B.accident – date)
```

توجه: خروجی زیر پرس و جوی داخلی در تاریخ تصادف‌های مختلف به ازای سطر اول جدول Accident A، به صورت زیر:

B.License
L1

Accident B

جلوی in برابر غیرتهی است، بنابراین شرط where در پشت in، TRUE می‌گردد. بنابراین سطر اول از جدول Accident A در خروجی نمایش داده می‌شود.

بنابراین شماره پلاک ماشین **L1** از آنجاکه در بیش از یک تصادف در تاریخ‌های مختلف، درگیر بوده است، جهت نمایش در خروجی انتخاب می‌شود، به صورت زیر:

B.License
L1

Accident A

حال برای سطر دوم نیز دقیقا همان روال سطر اول تکرار می‌شود.

حال برای سطر سوم از جدول Accident با توجه به شرط اتصال A.accident – date = B.accident – date
L2 L2
داریم:

License	Accident.date	…
L1	95-01-1	
L1	95-01-2	
L2	95-02-1	

Accident A

License	Accident.date	…
L1	95-01-1	
L1	95-01-2	
L2	95-02-1	

Accident B

که پرس و جوی زیر برای آن اجرا می‌شود:

```
select distinct A.License
from Accident A
where A.License in (select B.License
                    from Accident B
                    where  A.accident – date <> B.accident – date)
```

توجه: خروجی زیر پرس و جوی داخلی در تاریخ تصادف‌های مختلف به ازای سطر سوم جدول Accident A، به صورت زیر:

B.License
تهی

Accident B

جلوی in برابر تهی است، بنابراین شرط where در پشت in، FALSE می‌گردد. بنابراین سطر سوم از جدول Accident A در خروجی نمایش داده نمی‌شود.

بنابراین شماره پلاک ماشین **L2** از آنجاکه در بیش از یک تصادف در تاریخ‌های مختلف، درگیر نبوده است، جهت نمایش در خروجی انتخاب نمی‌شود، به صورت زیر:

B.License
تهی

Accident A

و در نهایت پس از اجرای کامل دستور Select distinct A.License خروجی پرس و جو به صورت زیر خواهد بود:

A.License
L1
L1

Accident A

که پس از اجرای دستور distinct سطرهای تکراری حذف می‌شود، به صورت زیر:

A.License
L1

Accident A

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «شماره پلاک ماشین‌هایی کـه در بـیش از یک تصادف در تاریخ‌های مختلف درگیر بوده‌اند.»

فرم دوم گزینه اول با استفاده از دستور exists به صورت زیر است:

```
select distinct A.License
from Accident A
where exists(select B.License
             from Accident B
             where A.License=B.License and A.accident – date <> B.accident – date)
```

که پس از اجرای دستور distinct سطرهای تکراری حذف می‌شود، به صورت زیر:

A.License
L1

Accident A

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «شماره پلاک ماشین‌هایی کـه در بـیش از یک تصادف در تاریخ‌های مختلف درگیر بوده‌اند.»

مطابق پرس و جوی مطرح شده در گزینه‌ی دوم، داریم:

```
select License
from Accident
group by License
having count(accident – date)>1
```

با توجه به جداول فوق، پس از انجام دستور group by License براساس ستون License خروجی پـرس و جو به صورت زیر گروه‌بندی می‌شود:

L1

95 – 01 – 1 d1 10

95 – 01 – 2 d1 15 ← گروه اول

L2

95 – 02 – 1 d2 20 ← گروه دوم

توجه: هر گروه شامل تصادف‌های ثبت شده در تاریخ‌های مختلف برای یک شماره پلاک است.

و در نهایت دستور having count(accident – date)>1 برای هر گروه به طور مستقل اعمال می‌گردد.

توجه: دستور HAVING بر روی گروه‌ها، اعمال می‌گردد.

L1

95 – 01 – 1 d1 10

95 – 01 – 2 d1 15 ← گروه اول

2 > 1,

L2

95 – 02 – 1 d2 20 ← گروه دوم

1 ≯ 1

توجه: با توجه به شرط انتخاب گروه توسط دستور having count(accident – date)>1، فقـط گـروه **اول** جهت نمایش در خروجی انتخاب می‌شود.

و در نهایت دستور select License داخل دستور select برای هر گروه انتخاب شده توسط دستور Having به طور مستقل اعمال می‌گردد و در خروجی پرس و جو قرار می‌گیرد، بنابراین خروجی نهایی پرس و جو به صورت زیر است:

License
L1

توجه: دستور GROUP BY، سرگروه‌ها را، راهی خروجی می‌کند، که به تبع سرگروه، سطر تکراری ندارد.

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «شماره پلاک ماشین‌هایی که در بیش از یک تصادف در تاریخ‌های مختلف درگیر بوده‌اند.»

مطابق پرس و جوی مطرح شده در گزینه‌ی سوم، داریم:

```
select A.License
from Accident A, Accident B
where A.License = B.License and A.accident – date <> B.accident – date
```

با توجه به جداول فوق، خروجی پرس و جوی گزینه‌ی سوم پس از انجام عملگر ضرب دکارتی به صورت زیر است:

License	accident-date	driver	accident-amount	License	accident-date	driver	accident-amount
L1	95-01-1	d1	10	L1	95-01-2	d1	15
L1	95-01-2	d1	15	L1	95-01-1	d1	10

و در نهایت پس از اجرای دستور select A.License خروجی پرس و جو به صورت زیر خواهد بود:

A.License
L1
L1

Accident A

که مطابق پرس و جوی مطرح شده در صورت سوال نیست، زیرا در صورت سوال این فرض قید شده است که پاسخ به زبان SQL باشد و سطرهای تکراری **فقط یک‌بار** نشان داده شوند، که در پرس و جوی گزینه‌ی سوم به دلیل عدم استفاده از دستور distinct در خروجی دستور select جهت حذف سطرهای تکراری، فرض مطرح شده در صورت سوال نقض شده است. بنابراین گزینه‌ی سوم پاسخ سوال نیست.

توجه: البته اگر پرس و جوی گزینه‌ی سوم به فرم زیر اصلاح شود آنگاه خروجی گزینه‌ی سوم، همانند گزینه اول و دوم می‌شود، به عبارت دیگر فرم اصلاح شده گزینه‌ی سوم، همان فرم کلاسیک گزینه‌های اول و دوم با استفاده از ضرب دکارتی است، فرم اصلاح شده گزینه‌ی سوم به فرم زیر است:

```
select distinct A.License
from Accident A, Accident B
where A.License = B.License and A.accident – date <> B.accident – date
```

و در نهایت پس از اجرای کامل دستور Select distinct A.License خروجی پرس و جو به صورت زیر خواهد بود:

A.License
L1
L1
Accident A

که پس از اجرای دستور distinct سطرهای تکراری حذف می‌شود، به صورت زیر:

A.License
L1
Accident A

که مطابق پرس و جوی مطرح شده در صورت سوال است. یعنی «شماره پلاک ماشین‌هایی که در بیش از یک تصادف در تاریخ‌های مختلف درگیر بوده‌اند.»

توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه چهارم را به عنوان پاسخ اعلام کرده بود.

۴- گزینه (۳) صحیح است.

دو جدول Node و Edge با مقادیر زیر را در نظر بگیرید:

NID	Name	Color	Description
N1	NN1	C1	D1
N2	NN2	C2	D2
N3	NN3	C3	D3
N4	NN4	C4	D4

جدول Node

NID1	NID2	EdgeType
N1	N2	T1
N1	N3	T2
N4	N1	T3
N3	N2	T4
N4	N3	T5

جدول Edge

توجه: کلید کاندید جدول Node ستون NID است و جدول Edge دارای دو کلید خارجی NID1 و NID2 است که هر دو به کلید کاندید جدول Node ارجاع می‌کنند.

توجه: ستون NID1 در جدول Edge به عنوان کلید خارجی به ستون NID از جدول Node ارجاع می‌کند.

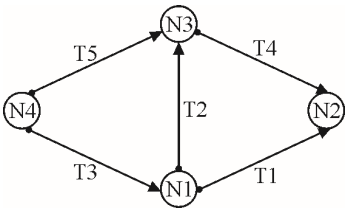
توجه: ستون NID2 در جدول Edge به عنوان کلید خارجی به ستون NID از جدول Node ارجاع می‌کند.

توجه: مقادیر کلید خارجی یعنی ستون‌های NID1 و NID2 از جدول Edge همواره باید زیرمجموعه مقادیر کلید کاندید یعنی ستون NID از جدول Node باشد.

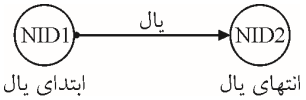
توجه: هر تاپل یا سطر یا رکورد از جدول Edge نشانه یک ارتباط است، پس هر تاپل نشانه یک یال هم هست، که مجموع این یال‌ها یک گراف را ایجاد می‌کند. خطی که از یک گره به گره بعدی رسم می‌شود یک یال است. هر گراف G شامل دو مجموعه V و E است. V مجموعه محدود و غیرتهی از رئوس است و E مجموعه‌ای محدود و احتمالا تهی از لبه‌ها می‌باشد. V(G) و E(G) مجموعه رئوس و لبه‌های گراف G را نمایش می‌دهند. برای نمایش یک گراف می‌توانیم بنویسیم G=(V , E). در یک گراف بدون جهت، زوج رئوس، زوج مرتب نیستند. بنابراین زوج‌های (V_0,V_1) و (V_1,V_0) باهم یکسان هستند. گراف حداقل یک راس دارد و نمی‌تواند کاملا تهی باشد. در یک گراف جهت دار، هر لبه با زوج مرتب $<V_0,V_1>$ نمایش

داده می‌شود، که پیکانی از V_0 به V_1 ترسیم می‌شود. بنابراین $<V_1,V_0>$ و $<V_0,V_1>$ دو لبه متفاوت را نمایش می‌دهند. گراف جهت دار را به صورت $G=<V , E>$ نمایش می‌دهیم.

توجه: گراف جهت دار جدول Edge به صورت زیر است.



گراف جدول Edge



هر تاپل از جدول Edge نشانه یک یال از گراف جدول Edge است.

توجه: همانطور که واضح است گره‌های موجود در ستون NID1 از جدول Edge یا گراف Edge، **یال‌های خروجی** دارند.

NID1={N1 , N3 , N4}

یعنی از گره‌های N1، N3 و N4 یال **خارج** شده است.

فرم اول پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  EXISTS (select *
                  from edge
                  where node.NID=edge.NID1)
```

فرم دوم پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  NID  IN (select  NID1
                  from edge)
```

فرم سوم پرس و جوی آن به صورت زیر است:

```
select distinct NID
from node , edge
where  node.NID=edge.NID1
```

توجه: در فرم سوم یا کلاسیک با استفاده از ضرب دکارتی، جهت حذف سطرهای تکراری می‌بایست از دستور distinct استفاده نمود.

خروجی پرس و جوها به صورت زیر است:

NID
N1
N3
N4

توجه: همانطور که واضح است گره‌های موجود در ستون NID2 از جدول Edge یا گراف Edge، **یال‌هـای ورودی** دارند.

NID2={N1 , N2 , N3}

یعنی به گره‌های N1، N2 و N3 یال **وارد** شده است.

فرم اول پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  EXISTS (select *
                from edge
                where node.NID=edge.NID2)
```

فرم دوم پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  NID  IN (select  NID2
                from edge)
```

فرم سوم پرس و جوی آن به صورت زیر است:

```
select distinct NID
from node , edge
where  node.NID=edge.NID2
```

توجه: در فرم سوم یا کلاسیک با استفاده از ضرب دکارتی، جهت حذف سـطرهای تکـراری می‌بایسـت از دستور distinct استفاده نمود.

خروجی پرس و جوها به صورت زیر است:

NID
N1
N2
N3

نتیجه: استخراج گره‌هایی که هم یالی از آن **خارج** شده است و هم یالی به آن **وارد** شده است، به صـورت زیر است:

$NID1 \cap NID2 = \{N1 , N3 , N4\} \cap \{N1 , N2 , N3\} = \{N1 , N3\}$

فرم اول پرس و جوی آن به صورت زیر است:

```
(select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID1))  INTERSECT  (select  NID
                                                          from node
                                                           where  EXISTS (select *
                                                                          from edge
                                                                         where node.NID=edge.NID2))
```

فرم دوم پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID1)  AND EXISTS  (select *
                                                         from edge
                                                            where node.NID=edge.NID2))
```

خروجی پرس و جوها به صورت زیر است:

NID
N1
N3

استخراج گره‌هایی که یالی از آن **خارج** شده است ولی یالی به آن **وارد** نشده است، به صورت زیر است:

NID1 – NID2 = {N1 , N3 , N4} – {N1 , N2 , N3} = {N4}

فرم اول پرس و جوی آن به صورت زیر است:

```
(select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID1))  EXCEPT  (select  NID
                                                      from node
                                                      where  EXISTS (select *
                                                                     from edge
                                                                     where node.NID=edge.NID2))
```

فرم دوم پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID1)  AND NOT  EXISTS  (select *
                                                              from edge
                                                                 where node.NID=edge.NID2)
```

خروجی پرس و جوها به صورت زیر است:

NID
N1
N3

استخراج گره‌هایی که یالی به آن **وارد** شده است ولی یالی از آن **خارج** نشده است، به صورت زیر است:

NID2 − NID1 = {N1 , N2 , N3} – {N1 , N3 , N4} = {N2}

فرم اول پرس و جوی آن به صورت زیر است:

```
(select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID2))  EXCEPT  (select  NID
                                                      from node
                                                      where  EXISTS (select *
                                                                     from edge
                                                                     where node.NID=edge.NID1))
```

فرم دوم پرس و جوی آن به صورت زیر است:

```
select  NID
from node
where  EXISTS (select *
                from edge
                 where node.NID=edge.NID2)  AND NOT  EXISTS  (select *
                                                              from edge
                                                               where node.NID=edge.NID1)
```

فرم سوم پرس و جوی آن به صورت زیر است:

```
select distinct NID
from node , edge
where node.NID=edge.NID2  AND NOT  EXISTS (select *
                                           from edge
                                           where  node.NID = edge.NID1)
```

توجه: می‌توان دستور Exists اول از فرم دوم را به صورت ضرب دکارتی و کلاسیک نیز نوشت.

توجه: که دقیقا همان فرم پرس و جوی اول مطرح شده در صورت سوال است.

خروجی پرس و جوها به صورت زیر است:

NID
N2

مطابق پرس و جوی اول مطرح شده در صورت سوال، داریم:

```
select distinct NID
from node , edge
where node.NID=edge.NID2  AND NOT  EXISTS (select *
                                           from edge
                                           where  node.NID = edge.NID1)
```

توجه: ازآنجاییکه ستون‌های NID و NID1 در دو جدول مختلف تکراری نیستند، بنابراین الزامی بر وجـود نام جداول پشت این ستون‌ها وجود ندارد.

با توجه به وجود عملگر not exists، زیر پرس و جوی داخلی فوق یک Correlated Subquery است، یعنی به ازای حرکت در هریک از سطرهای پرس و جوی خارجی، یک بار به طور کامل از ابتدا تا انتها زیر پرس و جوی داخلی اجرا و بر اساس شرطی که زیر پرس و جوی داخلی را به پرس و جوی خارجی متصل می‌کند، بررسی انجام می‌شود. مانند دو حلقه تو در تو، که به ازای هربار اجرای حلقه خارجی، یک بار به طور کامل حلقه داخلی اجرا می‌گردد.

توسط دستور where در فرم زیر:

```
SELECT distinct NID
FROM  node , edge
WHERE  node.NID=edge.NID2 AND NOT EXISTS (…)
```

برای هر سطر از جدول حاصل از ضرب دکارتی node , edge به طور همزمان شرط node.NID=edge.NID2 (شرط اول) و شرط جلوی NOT EXISTS (شرط دوم) بررسی می‌شود. اگر به طور همزمان شرط node.NID=edge.NID2 و همچنین شرط جلوی NOT EXISTS برقرار بود، آنگاه سطر مورد نظر از جدول حاصل از ضرب دکارتی node , edge انتخاب می‌گردد و این رویه برای تک تک سطرهای جدول حاصل از ضرب دکارتی node , edge، تا به انتهای جدول حاصل از ضرب دکارتی node , edge ادامه پیدا می‌کند. دقت کنید که برای هر سطر از جدول حاصل از ضرب دکارتی node , edge شرط جلوی NOT EXISTS که حاصل یک مقایسه می‌باشد، محاسبه می‌گردد، اگر تهی بود، شرط جلوی NOT EXISTS (شرط دوم) که همان NOT EXISTS است، TRUE می‌گردد.

به بیان دیگر این پرس و جو شماره گره‌هایی را می‌دهد که NID آن، عضو ستون NID2 یعنی مجموعه گره‌های دارای **یال وارد شونده** باشد و همچنین به طور همزمان پرانتز مقابل NOT EXISTS برای آن‌ها تهی باشد. این پرانتز هنگامی تهی می‌شود که حاصل مقایسه بیان شده در این پرانتز تهی شود. حاصل این مقایسه در صورتی تهی می‌شود که شماره گره مورد بررسی، NID آن، عضو ستون NID1 یعنی مجموعه گره‌های دارای **یال خارج شونده** نباشد. به عبارت دیگر پرس و جوی اول مطرح شده در صورت سوال شماره گره‌هایی را استخراج می‌کند که **یال ورودی** دارند اما **یال خروجی** ندارند. یعنی استخراج گره‌هایی که یالی به آن **وارد** شده است ولی یالی از آن **خارج** نشده است، که مطابق پرس و جوی اول مطرح شده در صورت سوال است که در هیچکدام از گزینه‌های صورت سوال، این حالت پرس و جو وجود ندارد.

به بیان دیگر عبارت موجود در جلوی دستور not exists که به صورت زیر است:

```
select distinct NID
from node , edge
where node.NID=edge.NID2  AND NOT  EXISTS (select *
                                           from edge
                                           where  node.NID = edge.NID1)
```

توسط دستور node.NID = edge.NID1 به محیط خارج یعنی جدول حاصل از ضرب دکارتی node , edge متصل می‌گردد. حال به ازای حرکت در هر سطر از جدول حاصل از ضرب دکارتی node , edge، یک بار به طور کامل سطرهای جدول edge از ابتدا تا انتها با توجه به شرط اتصال بررسی می‌گردد. مطابق شکل زیر:

ابتدا برای سطر اول از جدول حاصل از ضرب دکارتی node , edge البته با اعمال شرط node.NID=edge.NID2 با توجه به شرط اتصال ~~node.NID~~ = edge.~~NID1~~ داریم:

N2 ×

NID	…	NID1	NID2	EdgeType
N2		N1	**N2**	T1
N3		N1	N3	T2
N1		N4	N1	T3
N3		N4	N3	T5
N2		N3	N2	T4

Node , Edge

NID1	NID2	EdgeType
N1	N2	T1
N1	N3	T2
N4	N1	T3
N3	N2	T4
N4	N3	T5

Edge

که پرس و جوی زیر برای آن اجرا می‌شود:

```
select distinct NID
from node , edge
where node.NID=edge.NID2  AND NOT  EXISTS (select *
                                           from edge
                                           where  node.NID = edge.NID1)
```

توجه: خروجی زیر پرس و جوی داخلی در کنترل عضویت شماره گره مورد نظر یعنی N2 از NID در ستون NID1 یعنی مجموعه گره‌های دارای یال **خارج** شونده. به ازای سطر اول جدول حاصل از ضرب دکارتی node , edge البته با اعمال شرط node.NID=edge.NID2 به صورت زیر است:

NID1	NID2	EdgeType

تهی

Edge

جلوی not exists برابر تهی است، بنابراین شرط where در not exists (شرط دوم) ، TRUE می‌گردد. شرط where در node.NID=edge.NID2 (شرط اول) هم که از قبل برقرار و TRUE بود. که نتیجه کلی شرط where به دلیل وجود عملگر AND می‌شود TRUE.

بنابراین سطر اول از جدول حاصل از ضرب دکارتی node , edge با اعمال شرط node.NID=edge.NID2 با توجه به شرط اتصال در خروجی نمایش داده می‌شود.

بنابراین گره شماره **N2** از آنجاکه که یال ورودی دارد اما یال خروجی ندارد. یعنی یالی به آن **وارد** شده است ولی یالی از آن **خارج** نشده است، جهت نمایش در خروجی انتخاب می‌شود، به صورت زیر:

NID
N2

حال برای سطر دوم از جدول حاصل از ضرب دکارتی node , edge البته با اعمال شرط node.NID=edge.NID2 با توجه به شرط اتصال ~~node.NID~~ = ~~edge.NID1~~ داریم:

N2 N2

NID	…	NID1	NID2	EdgeType
N2		N1	N2	T1
N3		N1	**N3**	T2
N1		N4	N1	T3
N3		N4	N3	T5
N2		N3	N2	T4

Node , Edge

NID1	NID2	EdgeType
N1	N2	T1
N1	N3	T2
N4	N1	T3
N3	N2	T4
N4	N3	T5

Edge

که پرس و جوی زیر برای آن اجرا می‌شود:

```
select distinct NID
from node , edge
where node.NID=edge.NID2  AND NOT  EXISTS (select *
                                           from edge
                                           where  node.NID = edge.NID1)
```

توجه: خروجی زیر پرس و جوی داخلی در کنترل عضویت شماره گره مورد نظر یعنی N3 از NID در ستون NID1 یعنی مجموعه گره‌های دارای یال **خارج** شونده. به ازای سطر دوم جدول حاصل از ضرب دکارتی node , edge البته با اعمال شرط node.NID=edge.NID2 به صورت زیر است:

NID1	NID2	EdgeType
N3	N2	T4

Edge

جلوی not exists برابر غیرتهی است، بنابراین شرط where در not exists (شرط دوم) ، FALSE می‌گردد. شرط where در node.NID=edge.NID2 (شرط اول) هم که از قبل برقرار و TRUE بود. که نتیجه کلی شرط where به دلیل وجود عملگر AND می‌شود FALSE.

بنابراین سطر دوم از جدول حاصل از ضرب دکارتی node , edge با اعمال شرط node.NID=edge.NID2 با توجه به شرط اتصال در خروجی نمایش داده نمی‌شود.

بنابراین گره شماره **N3** از آنجاکه که یال ورودی دارد و یال خروجی هم دارد. یعنی یالی به آن **وارد** شده است و یالی هم از آن **خارج** شده است، جهت نمایش در خروجی انتخاب نمی‌شود، به صورت زیر:

NID
N2

حال برای سطر سوم (N1)، چهارم (N3) و پنجم (N2) از جدول حاصل از ضرب دکارتی node , edge با اعمال شرط node.NID=edge.NID2 نیز همین روال تکرار می‌شود، همچنین به واسطه وجود دستور distinct در دستور select سطرهای تکراری نیز در خروجی پرس و جو حذف می‌شود.

با توجه به جداول فوق، خروجی نهایی پرس و جوی فوق به صورت زیر است:

NID
N2

که مطابق پرس و جوی اول مطرح شده در صورت سوال است. یعنی «شماره گره‌هایی را استخراج می‌کند که **یال ورودی** دارند اما **یال خروجی** ندارند. یعنی استخراج گره‌هایی که یالی به آن **وارد** شده است ولی یالی از آن **خارج** نشده است، که در هیچکدام از گزینه‌های صورت سوال، این حالت پرس و جو وجود ندارد.

مطابق پرس و جوی دوم مطرح شده در صورت سوال، داریم:

```
select T1.NID
from (select count (NID1) as cnt , NID
      from node left outer join edge on edge.NID1=NID
      group by NID) T1 ,

      (select count (NID2) as cnt, NID
      from node left outer join edge on edge.NID2=NID
      group by NID) T2

      where T1.NID=T2.NID and T1.cnt < T2.cnt
```

عملگر الحاق خارجی چپ در جبر رابطه‌ای

این عملگر، مانند الحاق طبیعی، ستون‌های مشترک را **فقط یکبار** در خروجی قرار می‌دهد. همچنین کلیه سطرهای پیوندپذیر را در خروجی قرار می‌دهد. اما علاوه بر آن کلیه سطرهای پیوندناپذیر جدول سمت چپ را نیز در خروجی قرار می‌دهد و در این حالت برای ستون‌های غیر مشترک جدول سمت راست مقدار NULL قرار می‌دهد.

عملگر الحاق خارجی چپ در SQL (Left Outer Join)

این عملگر، ستون‌های مشترک را **دوبار** در خروجی قرار می‌دهد. همچنین کلیه سطرهای پیوندپذیر را در خروجی قرار می‌دهد. اما علاوه بر آن کلیه سطرهای پیوندناپذیر جدول سمت چپ را نیز در خروجی قرار می‌دهد و در این حالت برای تمام ستون‌های جدول سمت راست مقدار NULL قرار می‌دهد.

توجه: به تفاوت عملگر الحاق خارجی چپ در جبر رابطه‌ای و SQL دقت کنید.

توجه: برای شمارش **درجه خروجی** گره‌ها یعنی شمارش **یال‌های خارج شونده** از هر گره باید جدول Node در Edge الحاق خارجی چپ شود که نتیجه آن T1 نام‌گذاری شده است، تا تعداد تکرار یال‌های خارج شونده از هر گره شمارش شود، به صورت زیر:

```
(select count (NID1) as cnt , NID
from node left outer join edge on edge.NID1=NID
group by NID) T1
```

NID	name	color	Description	NID1	NID2	EdgeType
N1	NN1	C1	D1	N1	N2	T1
N1	NN1	C1	D1	N1	N3	T2
N2	NN2	C2	D2	NULL	NUUL	NULL
N3	NN3	C3	D3	N3	N2	T4
N4	NN4	C4	D4	N4	N1	T3
N4	NN4	C4	D4	N4	N3	T5

با توجه به جدول فوق، پس از انجام دستور group by NID براساس ستون NID خروجی پرس و جو به صورت زیر گروه‌بندی می‌شود:

N1
NN1 C1 D1 N1 N2 T1
NN1 C1 D1 N1 N3 T2
گروه اول

N2
NN2 C2 D2 NULL NULL NULL
گروه دوم

N3
NN3 C3 D3 N3 N2 T4
گروه سوم

N4
NN4 C4 D4 N4 N1 T3
NN4 C4 D4 N4 N3 T5
گروه چهارم

توجه: هـر گـروه شـامل تعـداد **یال‌هـای خـارج شـونده** از هـر گـره اسـت، چـون شـرط اتصـال روی edge.NID1=NID در node left outer join edge است.

و در نهایت دستور select count (NID1) as cnt , NID داخل دستور select برای هر گروه به طـور مسـتقل اعمال می‌گردد و در خروجی پرس و جو قرار می‌گیرد، بنابراین خروجی نهایی پرس و جو به صورت زیـر است:

T1.cnt	T1.NID
2	N1
0	N2
1	N3
2	N4

توجه: دستور GROUP BY، سرگروه‌ها را، راهی خروجی می‌کند، که به تبع سرگروه، سطر تکراری ندارد.

توجه: تابع آماری count (NID1) برای هر گروه به طور مستقل اعمال می‌گردد و در خروجی پرس و جـو قرار می‌گیرد.

توجه: تابع آماری count (NID1)، ستون NULL را نادیده می‌گیرد و نمی‌شمارد.

توجه: برای شمارش **درجه ورودی** گره‌ها یعنی شمارش **یال‌های وارد شونده** به هر گره باید جدول Node در Edge الحاق خارجی چپ شود که نتیجه آن T2 نام‌گذاری شـده اسـت، تـا تعـداد تکـرار یال‌هـای وارد شونده به هر گره شمارش شود، به صورت زیر:

```
(select count (NID2) as cnt , NID
from node left outer join edge on edge.NID2=NID
group by NID) T2
```

NID	name	color	Description	NID1	NID2	EdgeType
N1	NN1	C1	D1	N4	N1	T3
N2	NN2	C2	D2	N1	N2	T1
N2	NN2	C2	D2	N3	N2	T4
N3	NN3	C3	D3	N1	N3	T2
N3	NN3	C3	D3	N4	N3	T5
N4	NN4	C4	D4	NULL	NULL	NULL

با توجه به جدول فوق، پس از انجام دستور group by NID براساس ستون NID خروجی پرس و جـو بـه صورت زیر گروه‌بندی می‌شود:

N1
NN1 C1 D1 N4 N1 T3
گروه اول

N2
NN2 C2 D2 N1 N2 T1
NN2 C2 D2 N3 N2 T4
گروه دوم

N3

NN3 C3 D3 N1 N3 T2	N4
NN3 C3 D3 N4 N3 T5	NN4 C4 D4 NULL NULL NULL
گروه سوم	گروه چهارم

توجه: هر گروه شامل تعداد **یال‌های وارد شونده** به هر گره است، چون شرط اتصال روی edge.NID2=NID در node left outer join edge است.

و در نهایت دستور select count (NID2) as cnt , NID داخل دستور select برای هر گروه به طور مستقل اعمال می‌گردد و در خروجی پرس و جو قرار می‌گیرد، بنابراین خروجی نهایی پرس و جو به صورت زیر است:

T2.cnt	T2.NID
1	N1
2	N2
2	N3
0	N4

توجه: دستور GROUP BY، سرگروه‌ها را، راهی خروجی می‌کند، که به تبع سرگروه، سطر تکراری ندارد.

توجه: تابع آماری count (NID2) برای هر گروه به طور مستقل اعمال می‌گردد و در خروجی پرس و جو قرار می‌گیرد.

توجه: تابع آماری count (NID2)، ستون NULL را نادیده می‌گیرد و نمی‌شمارد.

توجه: در ادامه T1 و T2 در هم ضرب دکارتی می‌شوند که می‌شود 16 سطر که با اعمال شرط T1.NID=T2.NID در دستور where خروجی زیر را خواهیم داشت:

T1.cnt	T1.NID	T2.cnt	T2.NID
2	N1	1	N1
0	N2	2	N2
1	N3	2	N3
2	N4	0	N4

توجه: در ادامه با اعمال شرط T1.cnt < T2.cnt در دستور where خروجی زیر را خواهیم داشت:

T1.cnt	T1.NID	T2.cnt	T2.NID
0	N2	2	N2
1	N3	2	N3

توجه: در نهایت با اعمال دستور select T1.NID در دستور select خروجی زیر را خواهیم داشت:

T1.NID
N2
N3

که مطابق پرس و جوی دوم مطرح شده در صورت سوال است. یعنی «شماره گره‌هایی را می‌دهد که درجه خروجی آنها کمتر از درجه ورودی آنها است.» یعنی استخراج گره‌هایی که **یال‌های خارج شونده** از آن کمتر از **یال‌های وارد شونده** به آن است، که در گزینه سوم این حالت پرس و جو وجود دارد. بنابراین

پُرواضح است که گزینه‌ی **سوم** پاسخ سوال است.

توجه: اگر در شرط where به جای T1.cnt < T2.cnt از T1.cnt > T2.cnt استفاده می‌شد، آنگاه شماره گره‌هایی را می‌داد که درجه خروجی آنها بیشتر از درجه ورودی آنها است.» یعنی استخراج گره‌هایی که **یال‌های خارج شونده** از آن بیشتر از **یال‌های وارد شونده** به آن است.

توجه: اگر در شرط where به جای T1.cnt < T2.cnt از T1.cnt = T2.cnt استفاده می‌شد، آنگاه شماره گره‌هایی را می‌داد که درجه خروجی آنها برابر درجه ورودی آنها است.» یعنی استخراج گره‌هایی که **یال‌های خارج شونده** از آن برابر **یال‌های وارد شونده** به آن است.

توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه سوم را به عنوان پاسخ اعلام کرده بود.

تست‌های فصل هفتم: SQL دستورات DDL و DCL

۱- گزینه (۲) صحیح است.

صورت سوال به این شکل است:

چه تعداد از گزاره‌های داده شده درست است؟

الف) تعداد کلیدهای کاندید یک رابطه از تعداد ابرکلیدهای آن رابطه همواره کمتر یا مساوی است.

گزاره اول درست است، زیرا در حالت کلی، اگر رابطه R، دارای n خصیصه باشد، آنگاه تعداد ابرکلیدهای آن حداقل یک و حداکثر $2^n - 1$ است. در جدول تمام کلید، یک جدول فقط و فقط یک ابرکلید دارد و فقط و فقط هم یک کلید کاندید دارد. رابطه تمام کلید مثلا ممکن است سه ستون داشته باشد، در این حالت یک جدول فقط و فقط یک ابرکلید دارد و فقط و فقط هم یک کلید کاندید دارد. یعنی حداقل یک ابرکلید و یک کلید کاندید دارد و حداکثر هم یک ابرکلید و یک کلید کاندید دارد. در گزاره اول حالت کلی مورد بررسی قرار گرفته است. دقت کنید که حداقل یک ابرکلید و حداکثر $2^n - 1$ ابرکلید، حالت خاص جدول تمام کلید هم پوشش می‌دهد چون بیان حداکثر $2^n - 1$ مقادیر کوچکتر و برابر خودش را پوشش می‌دهد. اگر در یک رابطه با n خصیصه، تک تک خصیصه‌ها به تنهایی کلید کاندید باشد، آنگاه رابطه دارای n کلید کاندید است. بنابراین هر زیرمجموعه غیر تهی از خصیصه‌های این رابطه یک ابرکلید است. که در این حالت تعداد ابرکلیدهای یک رابطه با n خصیصه برابر با $2^n - 1$ است که بیشترین مقدار ممکن در تعداد ابرکلیدهای یک رابطه با n خصیصه است. ابرکلید بدون صفت نداریم، بنابراین حالت $\binom{n}{0}$ اضافه است. همانطور که گفتیم در حالت کلی، یک رابطه دارای n خصیصه، شرایط مختلفی را در تعداد ابرکلید می‌تواند تجربه کند، که حداکثر تعداد ابرکلیدهایی که می‌تواند تجربه کند برابر $\binom{n}{1}+\binom{n}{2}+...\binom{n}{n}$ یا $2^n - 1$ است. در واقع عبارت زیر برقرار است:

$$2^n - 1 = \binom{n}{1}+\binom{n}{2}+...\binom{n}{n}$$

توجه: ابرکلیدی که عضو زائد نداشته باشد، کلید کاندید **(Candidate key)** 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

نتیجه اینکه مجموعه کلیدهای کاندید همواره زیر مجموعه ابرکلیدها است، بنابراین تعداد کلیدهای کاندید یک رابطه از تعداد ابرکلیدهای آن رابطه همواره کمتر یا مساوی است.

ب) اگر رابطه R دارای n خصیصه باشد، آنگاه تعداد ابرکلیدهای این رابطه حداکثر 2^n است.

گزاره دوم نادرست است، زیرا در حالت کلی، اگر رابطه R، دارای n خصیصه باشد، آنگاه تعداد ابرکلیدهای آن حداقل یک و حداکثر $2^n - 1$ است.

ج) دستور ALTER Table در SQL، کاتالوگ سیستم را به روز می‌کند.

گزاره سوم درست است، زیرا 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 داده‌ها شامل تمامی اطلاعات سیستمی مربوط به پایگاه داده‌ها همچون مشخصات سیستمی جداول و سطوح دسترسی کاربران می‌باشد که به طور خودکار توسط DBMS

بروزرسانی می‌گردد. به بیان کامل‌تر هرگونه تغییرات حاصل از دستورات DDL در پایگاه داده همچون ایجاد جداول، تغییر 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.

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.

کاتالوگ سیستم به واسطه اجرای **تمامی** دستورات DDL و **برخی** دستورات DML مانند Insert وDelete و نه Select و Update دستخوش تغییر می‌گردد. دستور ALTER Table یکی از دستورات DDL برای اعمال تغییرات روی جداول پایگاه داده است، دستورات DDL در SQL دستورات ظرف ساز و ساختارساز در پایگاه داده هستند. از آنجا که **تمامی** دستورات DDL منجر به تغییرات در پایگاه داده می‌شود، بنابراین کاتالوگ سیستم به واسطه اجرای تمامی دستورات DDL دستخوش تغییر می‌گردد، نتیجه اینکه دستور ALTER Table در SQL، کاتالوگ سیستم را به روز می‌کند.

د) استفاده از Veiw و Index می‌تواند استقلال داده‌ای را افزایش دهد.

گزاره چهارم نادرست است، زیرا 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 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.

زبان‌های پیاده‌سازی

یک محصول نرم‌افزاری از دو وجه عملکرد (برنامه کاربردی) و داده (بانک اطلاعات) تشکیل می‌شود. انواع زبان‌های برنامه‌سازی به صورت زیر است:

زبان پیاده‌سازی برنامه کاربردی (وجه عملکرد)

برنامه کاربردی نیز مانند بخش داده، حاصل مراحل تحلیل، طراحی و پیاده‌سازی 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.

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 برنامه‌های کاربردی در قبال تغییراتی که در سطح فیزیکی (رسانه ذخیره‌سازی) 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 فایل‌های 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 داده‌ها 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، 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 وجـه داده توسـط دسـتورات DML نظیـر 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 **اسـتقلال فیزیکـی داده‌ها** میان یک برنامه کاربردی و داده‌ها می‌کند، به معنی وابسته نبودن برنامـه‌های کـاربردی بـه داده‌هـای 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 وجـه داده اسـت. در واقـع بخـش داده از بخش‌هـای مختلـف Table، View و Index تشکیل شده است.

دستور Create View با ساخت مفهوم دید، تا حدی کمک به برقـراری اسـتقلال داده‌ای از نـوع **اسـتقلال منطقی داده‌ها** میان یک برنامه کاربردی و داده‌ها می‌کند، به معنـی وابسـته نبـودن برنامـه‌های کـاربردی بـه داده‌های ذخیره شده، یعنی همانطور که گفتیم، اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیـز بـه آن اضافه گردد، در صورتی که برنامه کاربردی سابق نیاز به دستکاری و تغییر نداشته باشد، استقلال منطقـی داده‌ها براساس تغییرات نیز لحاظ شده است. از آنجاکه View روی ساختار قدیم شامل نام جدول قـدیم و ستون‌های قدیم ایجاد می‌شود، اگر جدولی دارای چهار ستون باشد و ستون پنجمی نیز به آن اضـافه گـردد، آنگاه بدون تغییرات در سـاختار View بخـش داده و بـه تبـع تغییـرات در سـاختار بخـش عملکرد(برنامـه کاربردی)، امکان حیات برنامه کاربردی بدون اشکال همچنان وجود دارد و این یعنی View حافظ اسـتقلال داده‌ای از نوع استقلال منطقی داده‌ها است. دقت کنید که View بخشی از وجه داده است. در واقـع بخـش داده از بخش‌های مختلف Table، View و Index تشکیل شده است.

دستور Create Index با ساخت مفهوم شاخص، هیچ کمکی به برقـراری اسـتقلال داده‌ای از نـوع **اسـتقلال فیزیکی داده‌ها و استقلال منطقی داده‌ها** میان یک برنامه کاربردی و داده‌ها نمی‌کند، به معنی وابسته نبـودن برنامه‌های کاربردی به داده‌های ذخیره شده. مهمترین کاربرد شاخص، افزایش سـرعت جسـتجو و بازیـابی اطلاعات است و این مفهوم و کاربرد هیچ ارتباطی به مفهوم استقلال داده‌ای ندارد.

شاخص فقط باعث بالا رفتن سرعت دستیابی به اطلاعات می‌گردد. شاخص‌ها برای بهبود فرآیند جستجو و بازیابی اطلاعات در جداول ایجاد می‌شوند. بـه عبـارت دیگـر شـاخص‌ها بـه فرآینـد جسـتجو و بازیـابی 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. اساسـا شـاخص‌ها بـر روی سـتون‌هایی بایـد تنظیم شود که بیشتر مورد جستجو قرار می‌گیرند.

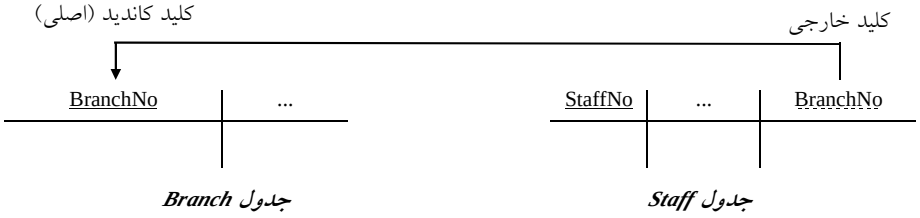
دو هدف اصلی سیستم ذخیره و بازیابی اطلاعات در پایگاه داده‌ها، اول سرعت عملیات در ذخیره و بازیابی اطلاعات و دوم صرفه‌جویی در مصرف حافظه است. برای مثال کاهش افزونگی محتوایی (طبیعـی) توسـط نرمال‌سازی جداول منجر به کاهش میزان حافظه مصرفی می‌شود. عمل واکشی تک تک رکوردها وقـت‌گیر است، برای رفع این عیب، شاخص یا Index ابداع شد. برای اینکه جستجو و بازیـابی داده‌هـا بـا سـرعت و کارایی بیشتر صورت گیرد، از شاخص استفاده می‌شود. شاخص ساختمان داده‌ای است که سیستم مـدیریت پایگاه داده‌ها به کمک آن رکوردهای مورد نظر را در یک فایل با سرعت بسیار زیاد پیدا می‌کنـد و بـه ایـن ترتیب سرعت پاسخ به پرس و جوها افزایش می‌یابد.

توجه: تعریف و نگهداری شاخص موجب تحمیل سربار حافظه‌ای به سیسـتم می‌شـود. شـاخص بـر روی هارد دیسک نگهداری می‌شود و هنگام استفاده به حافظه اصـلی آورده می‌شـود، بنـابراین اعمـال سیاسـت

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.
دقت کنید که Index بخشی از وجه داده است. در واقع بخش داده از بخش‌های مختلف Table، View و Index تشکیل شده است.
همانطور که گفتیم استفاده از Veiw می‌تواند استقلال داده‌ای را افزایش دهد. اما استفاده از Index به دلیل نامرتبط بودن آن با مفهوم استقلال داده‌ای نمی‌تواند استقلال داده‌ای را افزایش یا کاهش دهد و اثری بر استقلال داده‌ای ندارد.

۱- گزینه (۱) صحیح است.

دو جدول Branch و Staff را به صورت زیر در نظر بگیرید:



مطابق دستورات Create Table برای دو جدول Branch و Staff ستون BranchNo توسط دستور Primary key(BranchNo) به عنوان کلید اصلی برای جدول Branch انتخاب می‌گردد. همچنین ستون StaffNo توسط دستور Primary key(StaffNo) به عنوان کلید اصلی برای جدول Staff انتخاب می‌گردد. همچنین ستون BranchNo توسط دستور زیر:

Foreign key (BranchNo) References Branch (BranchNo)
On **delete** CASCADE
On **update** CASCADE

به عنوان کلید خارجی برای جدول Staff انتخاب می‌گردد. ستون BranchNo از جدول Staff به عنوان کلید خارجی به ستون BranchNo از جدول Branch مطابق ساختار تعریف جدول Staff، به شکل On Delete Cascade و On Update Cascade ارجاع می‌کند.

برای رفتار ستون کلید خارجی در یک جدول مقصد، در قبال تغییرات کلید کاندید از یک جدول مبداء گزینه‌های زیر وجود دارد:

```
Create Table      نام جدول
(
نام ستون‌ها
      ⋮
foreign key … references …
on delete option
on update option
      ⋮
)
```

فیلد option می‌تواند یکی از موارد زیر باشد:

(restrict) no action

گزینه پیش فرض است و هیچ عملی انجام نمی‌شود. یعنی اگر بر اثر عملیات حذف، درج و بروزرسانی، قوانین جامعیت نقض گردند، این اعمال انجام نمی‌گردند. در واقع زمانی عملیات حذف، درج و بروزرسانی انجام می‌گردند که قوانین جامعیت نقض نگردند.

Cascade

اگر سطرهای جداول مرجع **حذف** یا **بروزرسانی** شود، کلید خارجی جدولی که به آن ارجاع کرده است نیز حذف یا بروزرسانی خواهد شد.

Set NULL

همان روش NULLIFY است که اگر سطرهای جدول مرجع حذف یا بروزرسانی شود، کلید خارجی جدولی که به آن ارجاع کرده است با مقدار NULL پُر می‌شود.

توجه: کلید خارجی برای ارتباط میان جداول مورد استفاده قرار می گیرد.

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.

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

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

دستور زیر بر روی جدول Staff تعریف شده است:

Foreign key (BranchNo) References Branch (BranchNo)

On **delete** CASCADE

یعنی کلید خارجی جدول Staff یعنی ستون BranchNo به تغییرات (حذف) کلید کاندید جدول Branch یعنی ستون BranchNo به فرم cascade حساس باشد و واکنش نشان دهد، دقت کنید که درج در جدول Branch به خودش ربط دارد و نیاز به واکنش جدول دیگری نیست، اما حذف در جدول Branch به واسطه‌ی تعریف کلید خارجی در جدول Staff باعث می‌شود در جهت حفظ قانون جامعیت ارجاعی از جدول Staff به Branch، جدول Staff همواره به فرم cascade به تغییرات (حذف) در جدول Branch حساس باشد. اما این حساسیت در جدول Staff از جنس cascade است، یعنی اگر رکوردی در جدول Branch حذف شود که منجر به حذف در جدول Staff شود، آن حذف در جدول Staff نیز در جهت حفظ قانون جامعیت ارجاعی پذیرفته می‌شود.

همچنین دستور زیر بر روی جدول Staff تعریف شده است:

Foreign key (BranchNo) References Branch (BranchNo)

On **update** CASCADE

یعنی کلید خارجی جدول Staff یعنی ستون BranchNo به تغییرات (بروزرسانی) کلید کاندید جدول Branch یعنی ستون BranchNo به فرم cascade حساس باشد و واکنش نشان دهد، دقت کنید که درج در جدول Branch به خودش ربط دارد و نیاز به واکنش جدول دیگری نیست، اما بروزرسانی در جدول Branch به واسطه‌ی تعریف کلید خارجی در جدول Staff باعث می‌شود در جهت حفظ قانون جامعیت ارجاعی از جدول Staff به Branch، جدول Staff همواره به فرم cascade به تغییرات (بروزرسانی) در جدول Branch حساس باشد. اما این حساسیت در جدول Staff از جنس cascade است، یعنی اگر رکوردی در جدول Branch بروزرسانی شود که منجر به بروزرسانی در جدول Staff شود، آن بروزرسانی در جدول Staff نیز در جهت حفظ قانون جامعیت ارجاعی پذیرفته می‌شود.

اگر دستورات مطرح شده در صورت سوال به ترتیب زیر اجرا شوند، آنگاه کارمندان فعلی شعب 3 و 4 به شعبه 5 منتقل می‌گردند:

همانطور که گفتیم 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.

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

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

توجه: جهت جلوگیری از ایجاد ارجاع NULL ازجدول Staff به جدول Branch ابتدا در قدم اول می‌بایست مقدار شعبه 5 در جدول Branch درج گردد. سپس در قدم دوم کلیه سطرهایی که در جدول Staff به شعبه‌های 3 و 4 ارجاع می‌کنند توسط دستور بروزرسانی به شعبه 5 ارجاع کنند. و در نهایت در قدم سوم کلیه سطرهایی که در جدول Branch شامل شعبه‌های 3 و 4 هستند از جدول Branch حذف شوند.

توجه: دقت کنید که اگر در ابتدا دستور Insert انجام شود و سپس دستور Delete انجام شود، آنگاه از آنجایی که ستون BranchNo از جدول Staff به عنوان کلید خارجی به ستون BranchNo از جدول Branch مطابق ساختار تعریف جدول Staff، به شکل On Delete Cascade ارجاع می‌کند. پس از حذف سطرهای شعبه‌های 3 و4 از جدول Branch تمامی سطرهای شعبه‌های 3 و4 به دلیل خاصیت On Delete Cascade

از جدول Staff نیز حذف می‌شوند که دیگر امکان بروزرسانی و اجرای دستور Update برای سطرهای شعبه‌های 3 و 4 در جدول Staff وجود ندارد.

فرم جداول و تعداد رکوردهای جداول Branch و Staff **قبل از** انجام دستور Insert به صورت زیر است:

کلید کاندید (اصلی) — کلید خارجی

BranchNo	...
3	...
4	...
6	...
8	...

جدول Branch

StaffNo	...	BranchNo
S1	...	3
S2	...	4
S3	...	6
S4	...	8

جدول Staff

فرم جداول و تعداد رکوردهای جداول Branch و Staff **پس از** انجام دستور Insert به صورت زیر است:

نام دستور	دستور SQL
a	**Insert into Branch values (5, 'شادی', 021222324, 'تهران')**

کلید کاندید (اصلی) — کلید خارجی

BranchNo	...
3	…
4	…
6	…
8	…
5	…

جدول Branch

StaffNo	...	BranchNo
S1	...	3
S2	...	4
S3	...	6
S4	...	8

جدول Staff

فرم جداول و تعداد رکوردهای جداول Branch و Staff **پس از** انجام دستور Update به صورت زیر است:

نام دستور	دستور SQL
b	**Update Staff set BranchNo=5 where BranchNo=3**

کلید کاندید (اصلی) — کلید خارجی

BranchNo	...
3	…
4	…
6	…
8	…
5	…

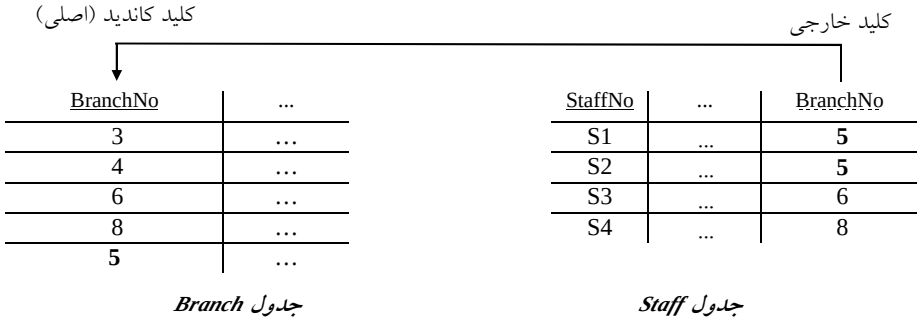
جدول Branch

StaffNo	...	BranchNo
S1	...	**5**
S2	...	4
S3	...	6
S4	...	8

جدول Staff

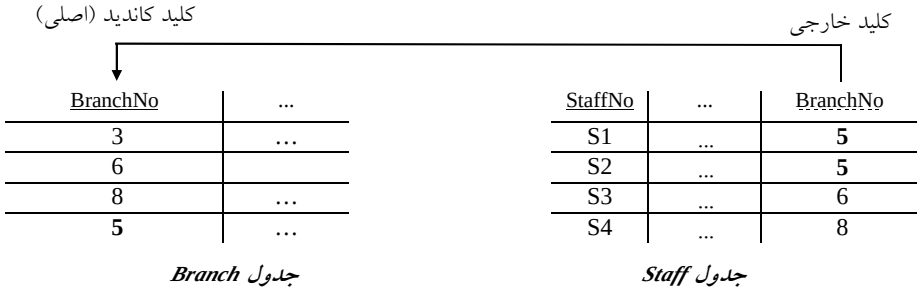
فرم جداول و تعداد رکوردهای جداول Branch و Staff **پس از** انجام دستور Update به صورت زیر است:

نام دستور	دستور SQL
d	**Update Staff set BranchNo=5 where BranchNo=4**



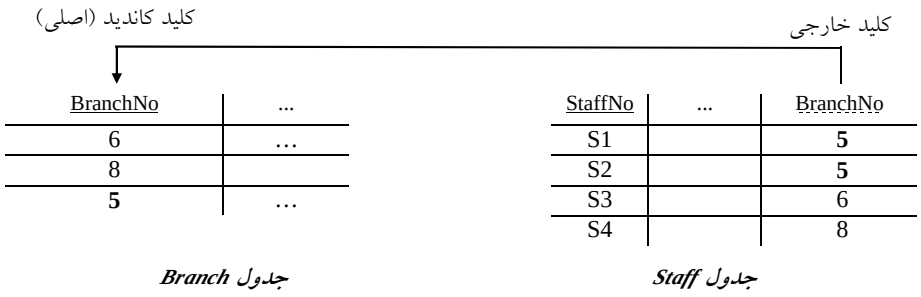
فرم جداول و تعداد رکوردهای جداول Branch و Staff **پس از** انجام دستور Delete به صورت زیر است:

نام دستور	دستور SQL
c	**Delete From Branch where BranchNo=4**



فرم جداول و تعداد رکوردهای جداول Branch و Staff **پس از** انجام دستور Delete به صورت زیر است:

نام دستور	دستور SQL
e	**Delete From Branch where BranchNo=3**



توجه: علاوه بر ترتیب abdce که در گزینه اول آمده است و پاسخ سوال نیز می‌باشد سایر ترتیب‌های abedc و adcbe و abdec و adbce و adbce نیز درست و امکان پذیر است. بنابراین گزینه‌های دوم، سوم و چهارم را به طور کامل کنار می‌گذاریم، پس پُرواضح است که گزینه اول پاسخ سوال است.

تست‌های فصل هشتم: وابستگی تابعی

۱- رابطه $R(A,B,C,D,E)$ و وابستگی‌های تابعی زیر را در نظر بگیرید:

$f = \{A \rightarrow B\ ,\ AB \rightarrow CD,\ D \rightarrow ABC\}$

کلید کاندید رابطه کدام است؟ (مهندسی کامپیوتر- دولتی ۹۸)

۱) AB ۲) AD ۳) AE ۴) BE

۲- با در نظر گرفتن رابطه $R(A,B,C,D,E)$ و مجموعه وابستگی‌های تابعی $F = \{AB \rightarrow C\ ,\ CD \rightarrow E,\ DE \rightarrow B\}$ ، این رابطه چند ابرکلید دارد؟ (مهندسی IT- دولتی ۹۸)

۱) 3 ۲) 7 ۳) 10 ۴) 12

پاسخ تست‌های فصل هشتم: وابستگی تابعی

۱- گزینه (۳) صحیح است.

مجموعه‌ی پوششی A (یعنی مجموعه‌ی تمام صفت‌هایی که با A قابل دسترسی‌اند) عبارت است از:

$$\{A\}^* = \{A, B, C, D\}$$

ملاحظه می‌شود که فقط صفت E در مجموعه‌ی فوق نیست پس A و E به همراه هم کلیدند.

به طور کلی کلید کاندید باید دو شرط زیر را داشته باشد:

۱- ابرکلید باشد (خاصیت کلیدی داشته باشد) یعنی همه خصیصه‌ها را تولید کند.

۲- عضو زائد نداشته باشد.

به طور کلی عضو کلید کاندید از روابط زیر به دست می‌آید:

قانون اول ارسطو

روش اول:

اجتماع تمام خصیصه‌های سمت راست وابستگی‌های غیر بدیهی- تمام خصیصه‌های جدول = عضو کلید کاندید

روش دوم:

$$\text{عضو کلید کاندید} = R - \bigcup_{i=1}^{n} [\text{(راست)}\ y_i - \text{(چپ)}\ x_i]$$

با توجه به وابستگی‌های مطرح شده برای رابطه $R(A, B, C, D, E)$ داریم:

$$A \rightarrow B$$
$$AB \rightarrow CD$$
$$D \rightarrow ABC$$
$$ABCDE - ABCD = E$$

بنابر رابطه فوق صفت E حتماً باید عضو کلید کاندید باشد. بستار صفت E به صورت زیر است:

$$\{E\}^+ = \{E\}$$

براساس بستار فوق، صفت E، فقط ستون E را تولید می‌کند، پس صفت E فقط عضو کلید کاندید می‌باشد و کلید کاندید نمی‌باشد.

قانون سوم ارسطو

هرگاه عضو کلید کاندید، حاصل از تفاضل قانون اول (روش اول یا دوم)، برخی از ستون‌ها را تولید کند، بدین معنی است که، جدول موردنظر، چندین کلید کاندید دارد، که این عضو کلید کاندید، در بین تمامی کلیدهای کاندید، به طور مشترک قرار دارد، بنابراین صفات دیگری نیز، باید عضو کلید کاندید را همراهی کنند تا کلید کاندید ایجاد گردد.

همچنین مطابق این قانون، صفاتی که توسط عضو کلید کاندید، قابل دسترسی هستند، در کلید کاندید جایگاهی نخواهند داشت.

براساس بستار فوق، صفت E به عنوان عضو کلید کاندید همه ستون‌ها را تولید نمی‌کند، بنابراین مطابق

قانون سوم ارسطو باید صفاتی در کنار صفت E قرار گیرد تا کلید کاندید تولید گردد. این صفات کناری از صفات باقی مانده به غیر از صفات E انتخاب می‌گردند، صفات باقی مانده عبارتنداز D و C و B و A البته از این مجموعه صفات C و B را هم کنار می‌گذاریم، زیرا کمکی در تولید صفات دیگر نمی‌کنند. چون در سمت چپ هیچ یک از وابستگی‌های تابعی به تنهایی نیامده‌اند. بنابراین با ترکیب صفات A یا D با صفت E کلیدهای کاندید تولید می‌گردند.

$$\{AE\}^+ = \{A, E, B, C, D\}$$

$$\{DE\}^+ = \{D, E, A, B, C\}$$

پس ترکیبات صفات AE و DE کلیدهای کاندید جدول R هستند.

به بیان دیگر با کمی دقت و بررسی، پُر واضح است که اگر صفت A در کنار عضو کلید کاندید E قرار بگیرد، همکاری صفات (E و A) می‌تواند، همه ستون‌ها را تولید کند، بنابراین صفات AE کلید کاندید جدول R خواهد بود.

بستار صفات AE به صورت زیر است:

$$\{AE\}^+ = \{A, E, B, C, D\}$$

براساس بستار فوق، صفات AE، همه ستون‌ها را تولید می‌کنند، پس صفات AE، کلید کاندید است.

توجه: دقت کنید که هیچ‌گاه، کلید کاندید نباید عضو زائد داشته باشد.

همچنین از آن‌جا که $D \rightarrow A$، پس می‌توان ترکیب دو خصیصه (D, E) را هم کلید کاندید دیگری برای این جدول تلقی کرد.

چون وقتی (A, E) کلید کاندید است و همه ستون‌ها را تولید می‌کند، پس (D, E) هم کلید کاندید است و همه ستون‌ها را تولید می‌کند، زیرا در نهایت طبق وابستگی $D \rightarrow A$، صفت D، صفت A را می‌دهد و (D, E) به (A, E) تبدیل می‌گردد.

بستار صفات (D, E) به صورت زیر است:

$$\{DE\}^+ = \{D, E, A, B, C\}$$

براساس بستار فوق، صفات (D, E)، همه ستون‌ها را تولید می‌کنند، پس صفات (D, E) کلید کاندید است.

توجه: دقت کنید که هیچ‌گاه، کلید کاندید نباید عضو زائد داشته باشد. بنابراین رابطه داده شده، در مجموع، دارای **دو کلید کاندید** است که مطابق قانون سوم ارسطو، عضو کلید کاندید E در بین هر دو کلید کاندید به طور مشترک قرار دارد.

گزینه اول نادرست است زیرا، عضو کلید کاندید E را ندارد.

گزینه دوم نادرست است. زیرا، عضو کلید کاندید E را ندارد.

گزینه سوم درست است. زیرا، صفات AE کلید کاندید جدول R است.

گزینه چهارم نادرست است. زیرا، کلیدهای کاندید فقط و فقط AE و DE هستند. به بیان دیگر بستار BE همه ستون‌ها رو تولید نمی‌کند: $\{B, E\}^+ = \{B, E\}$

۲- گزینه (۲) صحیح است.

به طور کلی کلید کاندید باید دو شرط زیر را داشته باشد:

۱- ابرکلید باشد (خاصیت کلیدی داشته باشد) یعنی همه خصیصه‌ها را تولید کند.

۲- عضو زائد نداشته باشد.

به طور کلی عضو کلید کاندید از روابط زیر به دست می‌آید:

قانون اول ارسطو

روش اول:

اجتماع تمام خصیصه‌های سمت راست وابستگی‌های غیر بدیهی- تمام خصیصه‌های جدول = عضو کلید کاندید

روش دوم:

$$\text{عضو کلید کاندید} = R - \bigcup_{i=1}^{n} [\text{(راست)}\ y_i - \text{(چپ)}\ x_i]$$

با توجه به وابستگی‌های مطرح شده برای رابطه $R(A,B,C,D,E)$ داریم:

$AB \rightarrow C$

$CD \rightarrow E$

$DE \rightarrow B$

$ABCDE - BCE = AD$

بنابر رابطه فوق صفات AD حتماً باید عضو کلید کاندید باشد. بستار صفات AD به صورت زیر است:

$$\{AD\}^+ = \{A,D\}$$

براساس بستار فوق، صفات AD، فقط ستون‌های A و D را تولید می‌کند، پس صفات AD فقط عضو کلید کاندید می‌باشد و کلید کاندید نمی‌باشد.

قانون سوم ارسطو

هرگاه عضو کلید کاندید، حاصل از تفاضل قانون اول (روش اول یا دوم)، برخی از ستون‌ها را تولید کند، بدین معنی است که، جدول موردنظر، چندین کلید کاندید دارد، که این عضو کلید کاندید، در بین تمامی کلیدهای کاندید، به طور مشترک قرار دارد، بنابراین صفات دیگری نیز، باید عضو کلید کاندید را همراهی کنند تا کلید کاندید ایجاد گردد.

همچنین مطابق این قانون، صفاتی که توسط عضو کلید کاندید، قابل دسترسی هستند، در کلید کاندید جایگاهی نخواهند داشت.

براساس بستار فوق، صفات AD به عنوان عضو کلید کاندید همه ستون‌ها را تولید نمی‌کند، بنابراین مطابق قانون سوم ارسطو باید صفاتی در کنار صفات AD قرار گیرد تا کلید کاندید تولید گردد. این صفات کناری از صفات باقی مانده به غیر از صفات AD انتخاب می‌گردند، صفات باقی مانده عبارتنداز E و C و B که از این مجموعه هیچ صفت دیگری را کنار نمی‌گذاریم، زیرا در تولید صفات دیگر کمک می‌کنند. بنابراین با ترکیب صفات E و C و B با صفات AD کلیدهای کاندید تولید می‌گردند.

$$\{ADE\}^+ = \{A,D,E,B,C\}$$

$$\{ADC\}^+ = \{A,D,C,E,B\}$$

$$\{ADB\}^+ = \{A,D,B,C,E\}$$

پس ترکیبات صفات ADE و ADC و ADB کلیدهای کاندید جدول R هستند.
به بیان دیگر با کمی دقت و بررسی، پُر واضح است که اگر صفت E در کنار عضو کلید کاندید AD قرار بگیرد، همکاری صفات (E و D و A) می‌تواند، همه ستون‌ها را تولید کند، بنابراین صفات ADE کلید کاندید جدول R خواهد بود.
بستار صفات ADE به صورت زیر است:

$$\{ADE\}^+ = \{A, D, E, B, C\}$$

براساس بستار فوق، صفات ADE، همه ستون‌ها را تولید می‌کنند، پس صفات ADE، کلید کاندید است.
توجه: دقت کنید که هیچ‌گاه، کلید کاندید نباید عضو زائد داشته باشد.
در ادامه به شکل بازگشتی جهت کشف مابقی کلیدهای کاندید داریم:
همچنین از آن‌جا که $CD \rightarrow E$، پس می‌توان ترکیب سه خصیصه (A, D, C) را هم کلید کاندید دیگری برای این جدول تلقی کرد.
چون وقتی (A, D, E) کلید کاندید است و همه ستون‌ها را تولید می‌کند، پس (A, D, C) هم کلید کاندید است و همه ستون‌ها را تولید می‌کند، زیرا در نهایت طبق وابستگی $CD \rightarrow E$، صفات CD، صفت E را می‌دهد و (A, D, C) به (A, D, E) تبدیل می‌گردد.
بستار صفات (A, D, C) به صورت زیر است:

$$\{ADC\}^+ = \{A, D, C, E, B\}$$

براساس بستار فوق، صفات (A, D, C)، همه ستون‌ها را تولید می‌کنند، پس صفات (A, D, C) کلید کاندید است.
توجه: دقت کنید که هیچ‌گاه، کلید کاندید نباید عضو زائد داشته باشد.
همچنین از آن‌جا که $AB \rightarrow C$، پس می‌توان ترکیب سه خصیصه (A, D, B) را هم کلید کاندید دیگری برای این جدول تلقی کرد.
چون وقتی (A, D, C) کلید کاندید است و همه ستون‌ها را تولید می‌کند، پس (A, D, B) هم کلید کاندید است و همه ستون‌ها را تولید می‌کند، زیرا در نهایت طبق وابستگی $AB \rightarrow C$، صفات AB، صفت C را می‌دهد و (A, D, B) به (A, D, C) تبدیل می‌گردد.
بستار صفات (A, D, B) به صورت زیر است:

$$\{ADB\}^+ = \{A, D, B, C, E\}$$

براساس بستار فوق، صفات (A, D, B)، همه ستون‌ها را تولید می‌کنند، پس صفات (A, D, B) کلید کاندید است.
توجه: دقت کنید که هیچ‌گاه، کلید کاندید نباید عضو زائد داشته باشد. بنابراین رابطه داده شده، در مجموع، دارای **سه کلید کاندید** است که مطابق قانون سوم ارسطو، عضو کلید کاندید AD در بین هر سه کلید کاندید به طور مشترک قرار دارد.

ترکیب اول: هیچ یا ترکیبی از صفات باقی مانده + کلید کاندید ADE = ابرکلید

ابرکلیدها	ترکیبات صفات باقی مانده	BC
ADE	تهی	00
ADEC	C	01
ADEB	B	10
ADEBC	BC	11

ADE +

توجه: واضح است که 4 ابرکلید، ایجاد می‌گردد. (2^2)

ترکیب دوم: هیچ یا ترکیبی از صفات باقیمانده + کلید کاندید ADC= ابرکلید

ابرکلیدها	ترکیبات صفات باقی مانده	BE
ADC	تهی	00
ADCE	D	01
ADCB	E	10
ADCBE	ED	11

ADC +

توجه: واضح است که 4 ابرکلید، ایجاد می‌گردد. (2^2)

ترکیب سوم: هیچ یا ترکیبی از صفات باقیمانده + کلید کاندید ADB= ابرکلید

ابرکلیدها	ترکیبات صفات باقی مانده	CE
ADB	تهی	00
ADBE	E	01
ADBC	C	10
ADBCE	CE	11

ADB +

توجه: واضح است که 4 ابرکلید، ایجاد می‌گردد. (2^2)

بنابراین حاصل جمع ابرکلیدها 12 عدد خواهد بود، که از این مجموعه حاصل، ابرکلیدهای ADCE و ADCBE و ADBE و ADBC و ADBCE در سه مجموعه تکراری هستند، بنابراین با کنار گذاشتن ابرکلیدهای تکراری، در نهایت 7 ابرکلید خواهیم داشت. به همین سادگی.

تست‌های فصل نهم: نرمال‌سازی

۱- رابطه و وابستگی‌های تابعی زیر را در نظر بگیرید.

$R(X, Y, Z)$

$1) Y \rightarrow Z \quad 2) XZ \rightarrow Y \quad 3) X \rightarrow Z$

با توجه به رابطه فوق کدام عبارت نادرست است؟ **(مهندسی کامپیوتر- دولتی ۹۸)**

۱) X کلید کاندید است.

۲) این رابطه در فرم نرمال BCNF نیست.

۳) وابستگی سوم، اضافه و قابل حذف است.

۴) X در وابستگی دوم، مشخصه اضافه و قابل حذف است.

پاسخ تست‌های فصل نهم: نرمال‌سازی

۲- گزینه (۳) صحیح است.

به طور کلی کلید کاندید باید دو شرط زیر را داشته باشد:

۱- ابرکلید باشد (خاصیت کلیدی داشته باشد) یعنی همه خصیصه‌ها را تولید کند.

۲- عضو زائد نداشته باشد.

به طور کلی عضو کلید کاندید از روابط زیر به دست می‌آید:

قانون اول ارسطو

روش اول:

اجتماع تمام خصیصه‌های سمت راست وابستگی‌های غیر بدیهی- تمام خصیصه‌های جدول = عضو کلید کاندید

روش دوم:

$$\text{عضو کلید کاندید} = R - \bigcup_{i=1}^{n} [\text{(راست) } y_i - \text{(چپ) } x_i]$$

با توجه به وابستگی‌های مطرح شده برای رابطه $R(X,Y,Z)$ داریم:

$Y \to Z$

$XZ \to Y$

$X \to Z$

$XYZ - YZ = X$

بنابر رابطه فوق صفت X حتماً باید عضو کلید کاندید باشد. بستار صفت X به صورت زیر است:

$$\{X\}^+ = \{X,Y,Z\}$$

براساس بستار فوق، صفت X، همه ستون‌ها را تولید می‌کند، پس صفت X، کلید کاندید می‌باشد.

قانون دوم ارسطو

هرگاه عضو کلید کاندید، حاصل از تفاضل قانون اول (روش اول یا دوم)، همه ستون‌ها را تولید کند، آن عضو کلید کاندید، تنها کلید کاندید جدول خواهد بود.

حال یک‌بار دیگر وابستگی‌های مطرح شده برای رابطه $R(X,Y,Z)$ را در نظر بگیرید:

وابستگی انتقالی $Y \longrightarrow Z$ غیرکلید غیرکلید

وابستگی تابعی $XZ \longrightarrow Y$ غیرکلید ابرکلید

وابستگی کامل $X \longrightarrow Z$ غیرکلید کلید کاندید

به طور کلی می‌توان شروط قرار داشتن یک جدول در نرمال فرم اول را به صورت زیر بیان کرد:

- دارای حداقل یک کلید کاندید باشد.
- همه خصیصه‌های آن غیرقابل تجزیه باشند (جدول باید فاقد خصیصه‌های مرکب باشد)
- همه خصیصه‌های آن تک مقداری باشند (جدول باید فاقد خصیصه‌های چند مقداری باشد)

واضح است که جدول مطرح شده در فرم اول نرمال قرار دارد.

به طور کلی می‌توان شروط قرار داشتن یک جدول در نرمال فرم دوم را به صورت زیر بیان کرد:

- جدول باید در نرمال فرم اول باشد.
- جدول باید فاقد وابستگی بخشی باشد.

وابستگی بخشی: وابستگی یک مولفه غیرکلیدی، به جزئی از کلید کاندید را وابستگی بخشی می‌نامند.
مولفه غیرکلید: هر صفتی که عضو هیچ کلید کاندیدی نباشد، به عنوان مؤلفه غیرکلیدی نامیده می‌شود.
مؤلفه جزء کلید کاندید: هر صفتی که عضو حداقل یک کلید کاندید باشد، به عنوان مؤلفه جزء کلید نامیده می‌شود.
در وابستگی‌های فوق، وابستگی بخشی وجود ندارد. بنابراین جدول مربوطه در نرمال فرم دوم هم قرار دارد.
به طور کلی می‌توان شروط قرار داشتن یک جدول در نرمال فرم سوم را به صورت زیر بیان کرد:

- جدول باید در نرمال فرم دوم باشد.
- جدول باید فاقد وابستگی انتقالی باشد.

وابستگی انتقالی: وابستگی یک مؤلفه غیرکلیدی به یک مؤلفه غیرکلیدی دیگر را وابستگی انتقالی می‌نامند.
در وابستگی‌های فوق، وابستگی انتقالی وجود دارد. بنابراین جدول مربوطه در نرمال فرم سوم قرار ندارد. و به تبع در نرمال فرم BCNF هم قرار ندارد.
بطور کلی می‌توان شروط قرار داشتن یک جدول در نرمال فرم BCNF را به صورت زیر بیان کرد:

- جدول باید در نرمال فرم سوم باشد.
- جدول باید فاقد وابستگی معکوس باشد.

وابستگی معکوس: وابستگی یک عضو کلید کاندید به عضو یک کلید کاندید دیگر یا مؤلفه غیرکلیدی را، وابستگی معکوس می‌نامند.

عضو کلید کاندید $\rightarrow$ عضو کلید کاندید
عضو کلید کاندید $\rightarrow$ غیرکلید

در وابستگی‌های فوق، وابستگی معکوس وجود ندارد. اما جدول فوق در نرمال فرم سوم قرار ندارد، بنابراین جدول مربوطه در نرمال فرم BCNF هم قرار ندارد.
در یک نگاه دیگر برای نرمال فرم BCNF می‌توان گفت، جدولی در نرمال فرم BCNF قرار دارد که همه‌ی شروع‌های وابستگی‌ها، ابرکلید باشد. به بیان دیگر هرگاه سمت چپ همه وابستگی‌ها، ابرکلید باشد، آن‌گاه آن جدول در BCNF قرار دارد که در وابستگی‌های فوق این چنین نیست. پس BCNF هم نیست.
در وابستگی‌های فوق، سمت چپ وابستگی‌های دوم و سوم ابرکلید است. اما سمت چپ وابستگی اول ابرکلید نیست.
بنابراین این جدول به دلیل نقض شرایط مربوطه، در نرمال فرم BCNF قرار ندارد. در نتیجه گزینه دوم گزاره درستی است.
گزینه اول گزاره درستی است، زیرا صفت X در سمت راست هیچ یک از وابستگی‌های تابعی مطرح شده در صورت سوال نیست، پس از روی صفات دیگر قابل دستیابی نیست و بنابراین حتماً باید در کلید کاندید حضور داشته باشد. بعلاوه، با داشتن X می‌توان تمام صفت‌های دیگر را به دست آورد، پس {X} تنها کلید کمینه است و بنابراین ستون X تنها کلید کاندید رابطه R است.
گزینه دوم نیز گزاره درستی است، زیرا با توجه به وابستگی $Y \rightarrow Z$ وابستگی انتقالی یعنی وابستگی

غیرکلید به غیرکلید وجود دارد، بنابراین رابطه R در سطح نرمال فرم سوم قرار ندارد و به تبع همین موضوع در سطح نرمال فرم BCNF هم قرار ندارد.

گزینه سوم گزاره نادرستی است، زیرا وابستگی سوم یعنی وابستگی $X \to Z$ اضافه و قابل حذف نیست، چون با داشتن X نمی‌توان از روی وابستگی‌های تابعی باقی‌مانده به Z رسید. در واقع وابستگی $X \to Z$ یک وابستگی اصلی از مجموعه وابستگی‌های بهینه است و یک وابستگی فرعی نیست.

گزینه چهارم نیز گزاره نادرستی است، زیرا ستون X در وابستگی دوم، مشخصه اضافه و قابل حذف نیست. چون ستون X توسط ستون دیگری تولید نمی‌شود یعنی در سمت راست هیچکدام از وابستگی‌های مطرح شده در صورت سوال قرار ندارد که بتوانیم ستون X را حذف کنیم. به عبارت دیگر اگر ستون X توسط ستون Z تولید می‌شد، آنگاه ستون X در وابستگی دوم، مشخصه اضافه و قابل حذف می‌بود که اینطور نیست.

توجه: البته اگر طراح محترم در گزینه چهارم به جای ستون X از ستون Z استفاده می‌کرد که ماهم فکر می‌کنیم همین بوده است اما در مرحله حروفچینی سوال، ستون Z به صورت X حروفچینی شده است. آنگاه گزینه چهارم هم گزاره درستی می‌بود و همان گزینه سوم در بین چهار گزینه فقط گزاره‌ای نادرست می‌شد و پاسخ سوال هم همان گزینه سوم و مد نظر طراح محترم می‌شد.

توجه: دقت کنید که با توجه به وابستگی $X \to Z$ با داشتن X می‌توان Z را به دست آورد. پس با حذف Z از وابستگی $XZ \to Y$ اطلاعاتی از بین نمی‌رود و حالت بهینه و کمینه وابستگی دوم به فرم $X \to Y$ می‌باشد. دقت کنید که عامل حذف ستون X در وابستگی دوم، وجود و حضور وابستگی اصلی سوم است. وابستگی اصلی سوم تحت هیچ شرایطی قابل حذف نیست.

توجه: سازمان سنجش آموزش کشور، در کلید اولیه و نهایی خود، گزینه سوم را به عنوان پاسخ اعلام کرده بود.