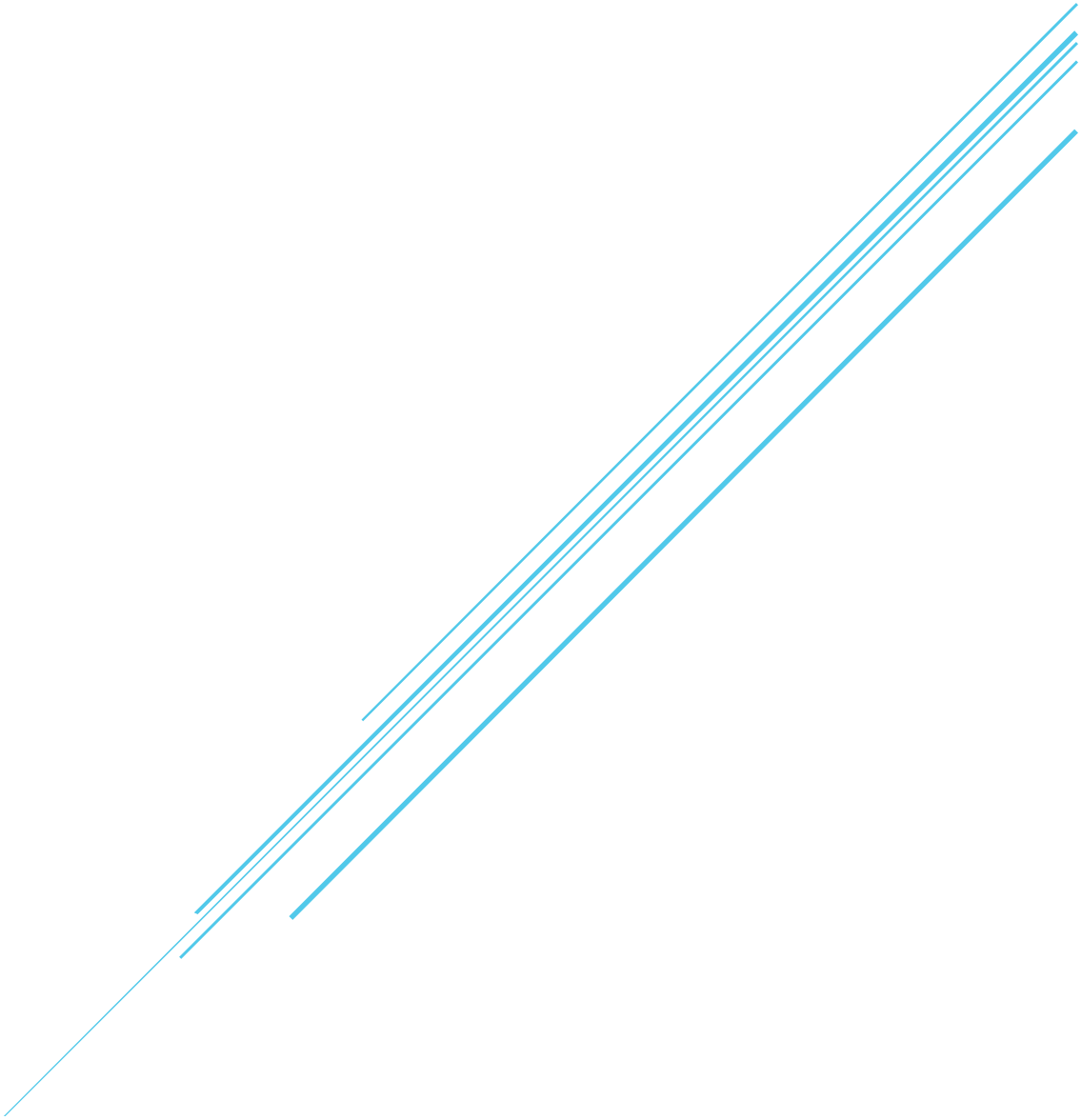
گزارش

اسم درس



- **نام و نام خانوادگی :**
- **نام استاد :**
- **گروه :**

فایل اول

برای طراحی ER از قوانین زیر استفاده شد. همچنین به دلیل کمبود جا و کوچک شدن بیش از حد ، رابطه ی چند به چند نشان داده نشد ولی از شکل واضح است.

۱. طراحی Library: کتابخانه از صفات خود تشکیل شده است که Lib_id کلید اصلی است.
۲. طراحی Section: موجودیت ضعیف نسبت به Library تعریف شده است ، پس کلید آن با نقطه چین نشان داده میشود و طبق قوانین زیر جدول آن طراحی میشود. (sec_id). همچنین این قسمت میتواند یکی از ۵ قسمت ذکر شده در متن سوال باشد ، پس رابطه ی ISA به صورت غیرپوشا و کامل است و هر کدام از موجودیت ها هم یک صفت اضافی دارند.
۳. طراحی Book: هر بخش کتابخانه ، کتاب های خود را دارد پس کتاب موجودیت ضعیف آن است ، و کلید با نقطه چین نمایش داده میشود و طبق قوانین زیر جدول آن طراحی میشود.
۴. طراحی Manager: هر بخش کتابخانه ، مسئول خود را دارد ، پس مسئول موجودیت ضعیف آن است و کلید با نقطه چین نمایش داده میشود و طبق قوانین زیر جدول آن طراحی میشود.
۵. طراحی Admin: ادمین موجودیت ضعیف نسبت به Manager است و کلید آن با نقطه چین نمایش داده میشود و طبق قوانین زیر جدول آن طراحی میشود.
۶. طراحی Member: عضو موجودیت ضعیف نسبت به کتابخانه است و کلید آن با نقطه چین نمایش داده میشود و طبق قوانین زیر جدول آن طراحی میشود.
۷. طراحی Reserve: یک مسئول میتواند یک کتاب را برای یک عضو رزرو کند. رابطه ی سه تایی را میتوان به رابطه ی دوتایی تبدیل کرد (صرفا یک موجودیت و یک جدول اضافه میشود که صرف نظر شد)
۸. طراحی Lend: یک مسئول میتواند یک کتاب را به یک عضو امانت بدهد. رابطه ی سه تایی را میتوان به رابطه ی دوتایی تبدیل کرد (صرفا یک موجودیت و یک جدول اضافه میشود که صرف نظر شد)

وابستگی وجودی

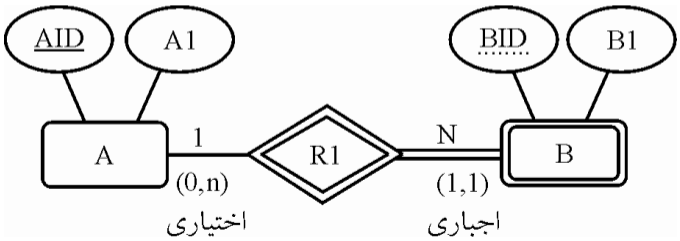
اگر در یک بانک اطلاعاتی، وجود یک موجودیت، وابسته به موجودیت دیگری باشد که در صورت حذف و تغییر موجودیت اصلی یعنی موجودیت قوی این موجودیت نیز تغییر کند، این نوع وابستگی را وابستگی وجودی گفته و به پدیده وابسته، **موجودیت ضعیف** گویند. همچنین **موجودیت ضعیف** کلید **موجودیت قوی** را در بر دارد تا هرگونه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

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

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

حالت اجباری و اختیاری

مدل تحلیل:



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

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

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

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

مدل طراحی:

AID	A1

جدول A

AID	BID	B1

جدول B

گزارش کار

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

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

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

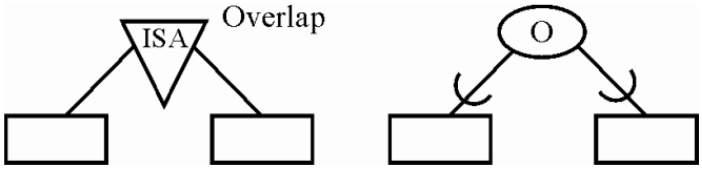
توجه: یک موجودیت ضعیف **همیشه** در ارتباطش با موجودیت قوی رابطه **اجباری** دارد.

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

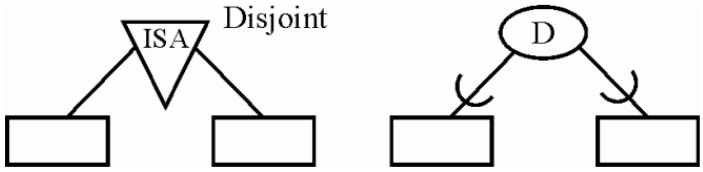
نگاشت رابطه ISA به مدل رابطه‌ای

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

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



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



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

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

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

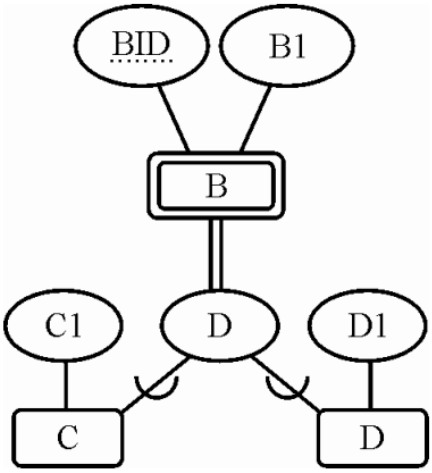
به عنوان مثالی دیگر هر نمونه از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد حتماً می‌بایست با یکی از نمونه موجودیت‌های نوع ماشین که دومحور (2WD) یا چهارمحور (4WD) است در ارتباط باشد. به عبارت دیگر نمی‌توان نمونه‌ای از موجودیت ماشین که شماره شاسی یکتا و منحصر به فرد خود را دارد داشت که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

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

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

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

مدل تحلیل (نمودار (ISA))



مدل طراحی (مدل رابطه‌ای)

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

AID	BID	B1	C1

جدول C

AID	BID	B1	D1

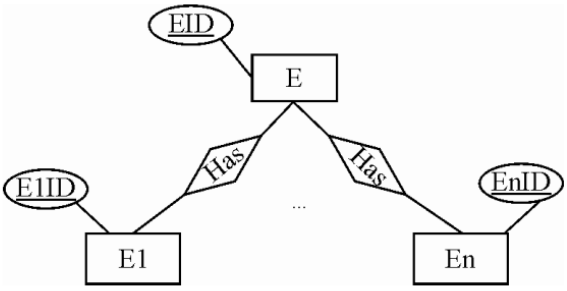
جدول D

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

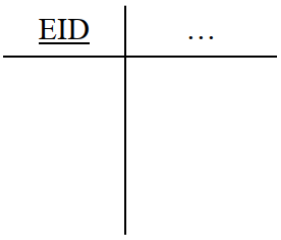
نگاشت رابطه IS-A-PART-OF به مدل رابطه‌ای

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

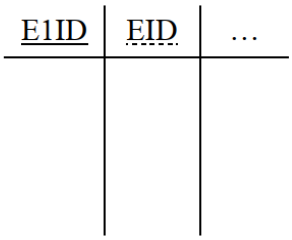
مدل تحلیل:



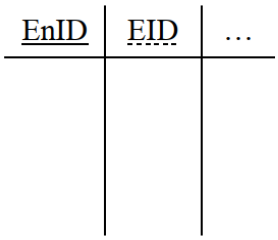
مدل طراحی:



جدول E



جدول E1



جدول En

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

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

فایل دوم

برای طراحی شمای متنی ، جدول ها طبق قوانین بالا طراحی میشوند.

طبق قوانین کلید اصلی هر موجودیت ضعیف به صورت :

Super PK + Weak PK

است ، به طور مثال Section ، موجودیت ضعیف Library است پس Primary Key آن ، به صورت

Lib_id + Sec_id

است.

اگر چند بار موجودیت ضعیف تکرار شود ، همه ی کلید های اصلی با هم جمع میشوند ، مثل Author.

در فایل دوم ، هر کلید اصلی که مربوط به موجودیت پایه باشد با زیر خط نمایش داده شده و هر کلید اصلی ضعیف با زیر خط نقطه چین نمایش داده میشود.

رنگ قرمز برای Foreign Key های موجود در موجودیت استفاده شده است.

فایل سوم

برای طراحی فایل SQL ، قسمتی که برای ساخت جدول هست ، طبق صفت هایی که در شمای متنی است طراحی شده است ،

و دیگر قوانین در پایین آمده است.

در قسمت Insert برای هر موجودیت ، چندین نمونه نوشته شده است.

در قسمت Procedure ، برای امانت دادن ، رزرو کردن ، برگرداندن کتاب و انتخاب ادمین از مسئولین طبق شروط ذکر شده در متن سوال ، کد هایی نوشته شده است.

و در قسمت EXEC هم ، Procedure ها اجرا شده اند.

Primary key

این دستور برای تعریف کلید اصلی مورد استفاده قرار می‌گیرد. مطابق قانون جامعیت درون رابطه‌ای، هر رابطه (جدول) باید حتماً حداقل دارای یک کلید کاندید باشد، همچنین مطابق قانون جامعیت موجودیت، هیچگاه نباید تمام یا بخشی از کلید کاندید مقدار NULL داشته باشد. بنابراین

اگر در تعریف یک جدول ستون (یا ستون‌هایی) به عنوان کلید اصلی تعریف شود، مطابق قانون جامعیت درون رابطه‌ای علاوه بر اینکه آن ستون نمی‌تواند مقادیر تکراری داشته باشد، مطابق قانون جامعیت موجودیت، آن ستون نمی‌تواند مقدار NULL هم داشته باشد و به طور پیش فرض NOT NULL در نظر گرفته می‌شود.

NOT NULL

هر ستونی می‌تواند حاوی مقدار NULL باشد، مگر اینکه در تعریف آن NOT NULL به کار رفته باشد. کلید اصلی به طور پیش فرض NOT NULL تعریف می‌شود.

Unique

این دستور برای تعریف کلید فرعی مورد استفاده قرار می‌گیرد. با قرار دادن نام یک ستون در جلوی دستور Unique می‌توان کلید فرعی تعریف کرد، که در این حالت آن ستون نمی‌تواند مقدار تکراری داشته باشد. کلید فرعی نوعی کلید کاندید است که مقدار یکتا دارد.

Foreign key

این دستور برای تعریف کلید خارجی مورد استفاده قرار می‌گیرد. کلید خارجی برای ارتباط میان جداول مورد استفاده قرار می گیرد، مطابق قانون جامعیت ارجاعی، هیچگاه نباید کلید خارجی، دچار ارجاع NULL گردد. برای این منظور باید تعریف ساختاری و محتوایی کلید خارجی برقرار باشد. کلید خارجی در دو دیگاه ساختاری و محتوایی مورد بررسی قرار می‌گیرد:

دیدگاه ساختاری کلید خارجی

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

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

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

دیدگاه محتوایی کلید خارجی

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

توجه: در جدول SP ستون S# به عنوان کلید خارجی تعریف شده است، و توسط دستور S(S#) References به جدول اصلی S و ستون S# ارجاع داده شده است، در این ارجاع نوع و درجه ستون S# در جدول SP به عنوان جدول فرعی با نوع و درجه ستون S# در جدول S به عنوان جدول اصلی باید یکسان و سازگار باشد.

توجه: در جدول SP ستون P# به عنوان کلید خارجی تعریف شده است، و توسط دستور References P(P#) به جدول اصلی P و ستون P# ارجاع داده شده است، در این ارجاع نوع و درجه ستون P# در جدول SP به عنوان جدول فرعی با نوع و درجه ستون P# در جدول P به عنوان جدول اصلی باید یکسان و سازگار باشد.

مثال:

S#	Sname	City
S1	Sn1	C1
S2	Sn2	C2
S3	Sn3	C3

جدول S

S#	P#	QTY
S1	P1	10
S1	P2	20
S2	P1	30
(S4)	P3	40

× ارجاع NULL

جدول SP

P#	Pname	Color
P1	Pn1	Red
P2	Pn2	Blue
P3	Pn3	Green

جدول P

درج رکورد (S4,P3,40) در جدول SP، غیرمجاز و غیرممکن است، زیرا سبب ارجاع NULL می‌گردد.

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

مثال: مانند S3 که در جدول S قرار دارد ولی لزومی ندارد در جدول SP هم باشد. اما باید همه مقادیر کلید خارجی در کلید کاندید باشد، از بالا به پایین می‌بارد، از پایین به بالا که نمی‌بارد!

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

مثال:

کلید کاندید — کلید خارجی

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

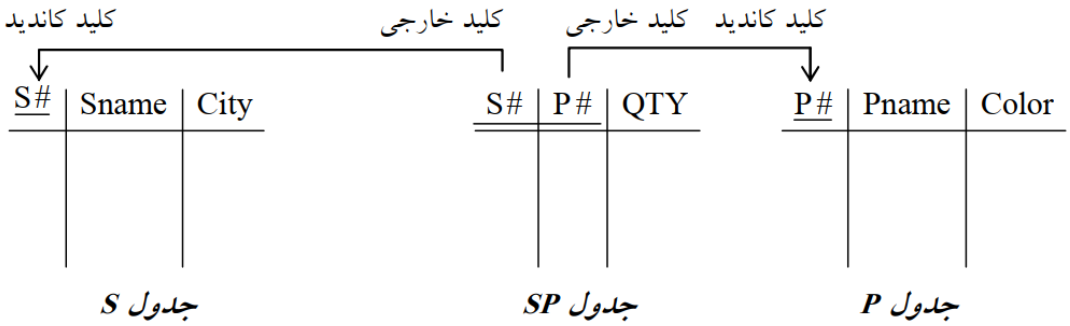
R_1

(a)	d
2	1
3	2
2	3
1	4

R_2

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

مثال:



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

توجه: برای رفتار ستون کلید خارجی در یک جدول فرعی، در قبال تغییرات کلید کاندید از یک جدول اصلی گزینه‌های زیر وجود دارد:

```
Create Table name
        (attr domain [NOT NULL],
         .
         .
         .
        attr …,
        Primary key (…),
        [Unique (…)],
        [Foreign key (…) References …(…)]
              [on delete option]
              [on update option],
        [Check (…)]
        )
```

فیلد option می‌تواند یکی از موارد زیر باشد:

(restrict) no action

گزینه پیش فرض است و هیچ عملی انجام نمی‌شود. اگر بر اثر عملیات **حذف** یا **بروزرسانی** در جدول اصلی، قانون جامعیت ارجاعی در جدول فرعی نقض گردد، آنگاه این اعمال انجام نمی‌گردد. در واقع زمانی عملیات حذف یا بروزرسانی در جدول اصلی انجام می‌گردد که قانون جامعیت ارجاعی در جدول فرعی نقض نگردد. به عبارت دیگر زمانی عملیات حذف یا بروزرسانی در سطری از جدول اصلی انجام می‌گردد که سطری از جدول فرعی به آن ارجاع نکرده باشد. برای مثال مقدار S3 در جدول S می‌تواند به S33 بروزرسانی شود و یا از جدول S حذف شود، زیرا هیچ ارجاعی روی مقدار S3 از جدول SP به جدول S وجود ندارد. اما مقدار S1 در جدول S مطابق رابطه no action نمی‌تواند به S11 بروزرسانی شود و یا از جدول S حذف شود، زیرا ارجاعی روی مقدار S1 از جدول SP به جدول S وجود دارد.

Cascade

اگر سطرهای جدول اصلی **حذف** یا **بروزرسانی** شود، آنگاه سطرهای جدول فرعی که توسط کلید خارجی به آن ارجاع کرده است نیز حذف یا بروزرسانی می‌شود. برای مثال اگر مقدار S1 در جدول S به S11 بروزرسانی شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطابق رابطه Cascade مقدار S1 در جدول SP به مقدار S11 نیز بروزرسانی می‌شود. برای مثالی دیگر اگر سطر S1 از جدول S حذف شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطابق رابطه Cascade سطر S1 از جدول SP نیز حذف می‌شود.

(NULLIFY) Set NULL

اگر سطرهای جدول اصلی **حذف** یا **بروزرسانی** شود، آنگاه ستون‌های جدول فرعی که توسط کلید خارجی به آن ارجاع کرده است با مقدار NULL پُر می‌شود. برای مثال اگر مقدار S1 در جدول S به S11 بروزرسانی شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطابق رابطه Set NULL مقدار S1 در جدول SP به مقدار NULL تغییر می‌کند. برای مثالی دیگر اگر مقدار S1 از جدول S حذف شود و این مقدار در جدول SP نیز موجود باشد، آنگاه مطابق رابطه Set NULL مقدار S1 در جدول SP به مقدار NULL تغییر می‌کند.

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

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

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

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

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

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

Check

این دستور برای بیان قوانین **جامعیت خارجی** مورد استفاده قرار می‌گیرد. این کنترل‌ها هنگام وارد کردن اطلاعات، اعمال می‌گردد.

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

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

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

توجه: می‌توان بخش Check را توسط دستور زیر و به طور جداگانه و خارج از تعریف جدول، تعریف کرد:

```
Create Assertion name
        Check (…)
```

توجه: دقت کنید که ASSERTIONها دقیقا همانطور که خوانده می‌شوند عمل می‌کنند، و دقیقا به همان شکل که خوانده می‌شوند اجازه ذخیره‌سازی داده‌ها را به جداول می‌دهند.