

به نام آنکه جان را فکرت آموخت



بخش چهارم: طراحی منطقی

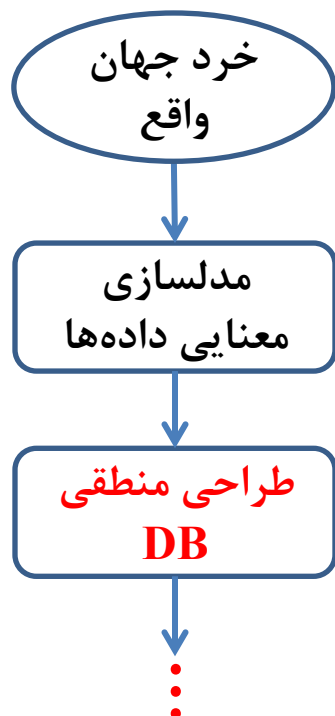
مرتضی امینی

نیمسال اول ۱۴۰۱-۱۴۰۰

(محتویات اسلایدها برگرفته از یادداشت‌های کلاسی استاد محمدتقی روحانی رانکوهی است.)



بخش چهارم: طراحی منطقی پایگاه داده‌ها



☐ مدل‌سازی داده‌ها می‌تواند در سطوح انتزاعی مختلفی صورت پذیرد.

☐ سطح پایین‌تر از سطح مدل‌سازی معنایی داده‌ها، سطح طراحی منطقی است.

☐ **سطح طراحی منطقی:** برای نمایش پایگاه داده‌ها در این سطح از مفاهیمی استفاده می‌شود که مستقل از مفاهیم محیط فایلینگ پایگاه داده‌ها است.



□ بحث مقدماتی: دیدگاه کاربردی [و نه تئوریک]

□ برای طراحی منطقی پایگاه داده‌ها (و همچنین عملیات در DB و کنترل DB) هم امکان خاصی لازم است: یک **مدل داده (DM)**، که شامل یک **ساختار داده (DS)** است.

□ مفاهیم مطرح در طراحی منطقی پایگاه داده‌ها

□ ساختار داده جدولی : TDS

□ پایگاه داده جدولی : TDB

□ زبان پایگاهی جدولی : TDBL



□ چرا ساختار داده (در معنای عام)؟

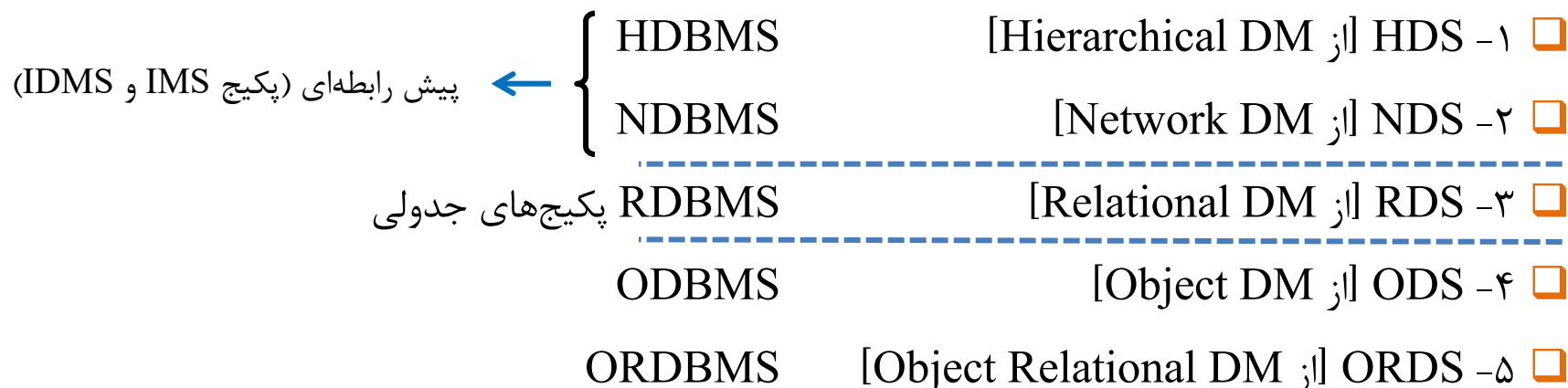
□ برای نمایش نوع موجودیت‌ها و ارتباط بین آنها در سطح منطقی

□ دلایل لزوم ساختار داده (DS) در حیطه پایگاهی:

- ۱- تامین کننده محیط فرافایلی (محیط انتزاعی)
- ۲- مبنا و چارچوب طراحی منطقی DB
- ۳- مبنا و چارچوب طراحی زبان پایگاه داده‌ها DBL
-
- ۴- مبنا و چارچوب طراحی خود DBMS
- ۵- ضابطه‌ای است برای مقایسه DBMS ها و ارزیابی آنها
- ۶- مبنایی است برای ایجاد و گسترش تکنیک‌های طراحی DB
- ۷- ...



□ DSها [در حیطه دانش و تکنولوژی DB]:



□ TDS - ساختار داده جدولی:

□ عنصر ساختاری اساسی در Relational Model (RM)، مفهوم **رابطه** است.

□ رابطه [Relation]: یک مفهوم ریاضی است ...


□ اما از دید کاربر [در عمل]: نمایش جدولی دارد.

▪ فعلا به جای ساختار داده رابطه‌ای (RDS) می‌گوییم ساختار داده جدولی (TDS).

(البته رابطه و جدول تفاوت‌های جدی با هم دارند که در مباحث بعدی درس بدان پرداخته می‌شود.)



□ TDS فقط یک عنصر ساختاری اساسی دارد : همان **نوع جدول**

عنصری است که به کمک آن نوع موجودیت، نوع ارتباط، و یا هردو آنها را نمایش می‌دهیم. 

□ اصطلاحات TDS:

□ **نوع جدول** ← { نام جدول
نام و نوع ستون‌ها } برای نمایش نوع { موجودیت و/یا
ارتباط }

□ **سطر** ← برای نمایش نمونه { موجودیت
ارتباط }

□ **ستون** ← برای نمایش صفت



TDB چیست؟ ☐

■ از دید کاربر

$$\left\{ \begin{array}{l} \text{طراح} \\ \text{پیاده ساز DB} \\ \text{برنامه ساز AP} \end{array} \right\}$$

■ از لحاظ نوع: مجموعه‌ای است از تعدادی **نوع جدول** (که آنها را طراحی می‌کنیم)

■ از لحاظ محتوای داده‌ای [در سطح نمونه]: مجموعه‌ای است از نمونه‌های متمایز یک [چند] **نوع سطر**

■ نوع سطر را همان نوع جدول مشخص می‌کند.



□ در طراحی پایگاه داده‌های جدولی (رابطه‌ای) باید موارد زیر را مشخص نمود:

□ مجموعه‌ای از جدول‌ها (رابطه‌ها)

□ کلید(های) هر جدول (در مدل رابطه‌ای کلیدهای کاندید رابطه - معرفی در جلسات بعدی درس)

□ کلید اصلی هر جدول (رابطه)

□ کلیدهای خارجی هر جدول (رابطه)، در صورت وجود

□ محدودیت‌های جامعیتی ناظر بر هر جدول (رابطه) - در جلسات بعدی درس در مورد انواع و نحوه

تعریف آن در پایگاه داده بحث خواهد شد.

طراحی با روش بالا به پایین (Top-Down) - معرفی در ادامه

□ روشهای طراحی TDB (RDB):

طراحی با روش سنتز [نرمال‌ترسازی رابطه‌ها] - معرفی در بخش‌های

پایانی درس



□ روش طراحی بالا به پایین

□ ابتدا مدلسازی داده‌ها را (با روش [E]ER یا UML) انجام می‌دهیم و سپس مدلسازی را به

مجموعه‌ای از جدول‌ها (رابطه‌ها) تبدیل می‌کنیم.

□ روش طراحی سنتز رابطه‌ای (نرمال ترسازی)

□ ابتدا مجموعه صفات خرد جهان واقع را مشخص می‌کنیم. سپس با تحلیل قواعد و محدودیت‌های ناظر

به صفات و تشخیص وابستگی‌های بین آنها، صفات را متناسباً با هم سنتز می‌کنیم (نوعی گروه‌بندی)

تا به مجموعه‌ای از جدول‌ها (رابطه‌های) نرمال دست یابیم.

□ **در عمل** روش ترکیبی استفاده می‌شود، یعنی ابتدا روش بالا به پایین، سپس نرمال ترسازی.



بخش چهارم: طراحی منطقی پایگاه داده‌ها

□ نمایش صحیح و واضح از خردجهان واقع باشد.

□ تمام داده‌های کاربران قابل نمایش باشد و همه محدودیت‌های (قواعد) جامعیتی منظور شده باشد.

□ کمترین افزونگی

□ کمترین هیچمقدار

□ کمترین مشکل در عملیات ذخیره‌سازی

□ بیشترین کارایی در بازیابی

نکته: تامین چهار ویژگی آخر به صورت همزمان، در عمل ناممکن است!



□ تبدیل نمودار [E]ER به مجموعه‌ای از جدول‌ها (رابطه‌های نرمال – معرفی در بخش‌های پایانی)، نهایتاً

طراح تصمیم می‌گیرد چند جدول (رابطه) داشته باشد.

□ در نمودار مدلسازی معنایی داده‌ها، حالات متعدد داریم، که در ادامه به آنها می‌پردازیم.

□ **فرض:** تا اطلاع ثانوی، همه صفات ساده‌اند و موجودیت‌ها ضعیف نیستند.



☐ صفت **شناسه** در نوع موجودیت‌ها، حکم **کلید** را در جدول دارد.

☐ مفهوم کلید در مدل داده جدولی تعریف نشده است و برگرفته از مفاهیم تعریف شده در مدل داده‌ای رابطه است.

☐ **تعریف** [از دیدگاه کاربردی] **کلید** امکان دسترسی به تک نمونه (از یک نوع موجودیت یا نوع ارتباط) را فراهم می‌نماید. لذا مقدار آن در سطرهای جدول مربوط به موجودیت یا ارتباط، **یکتا** است.

☐ **تعریف** [از دیدگاه کاربردی] یک یا چند صفت (ستون) تشکیل **کلید اصلی** را در یک جدول می‌دهند اگر مقادیر آن(ها) در سطرهای جدول **یکتا** و **معلوم** باشد.

☐ در مواقعی ممکن است بیش از یک کلید داشته باشیم. یکی از کلیدها که مقادیرش در همه سطرها معلوم است را کلید اصلی می‌گیریم (بقیه را با یکتا بودن مقادیر – با استفاده از UNIQUE در SQL – مشخص می‌نماییم).

☐ در معرفی مدل داده رابطه‌ای، با انواع کلید و تعاریف آنها آشنا می‌شوید.



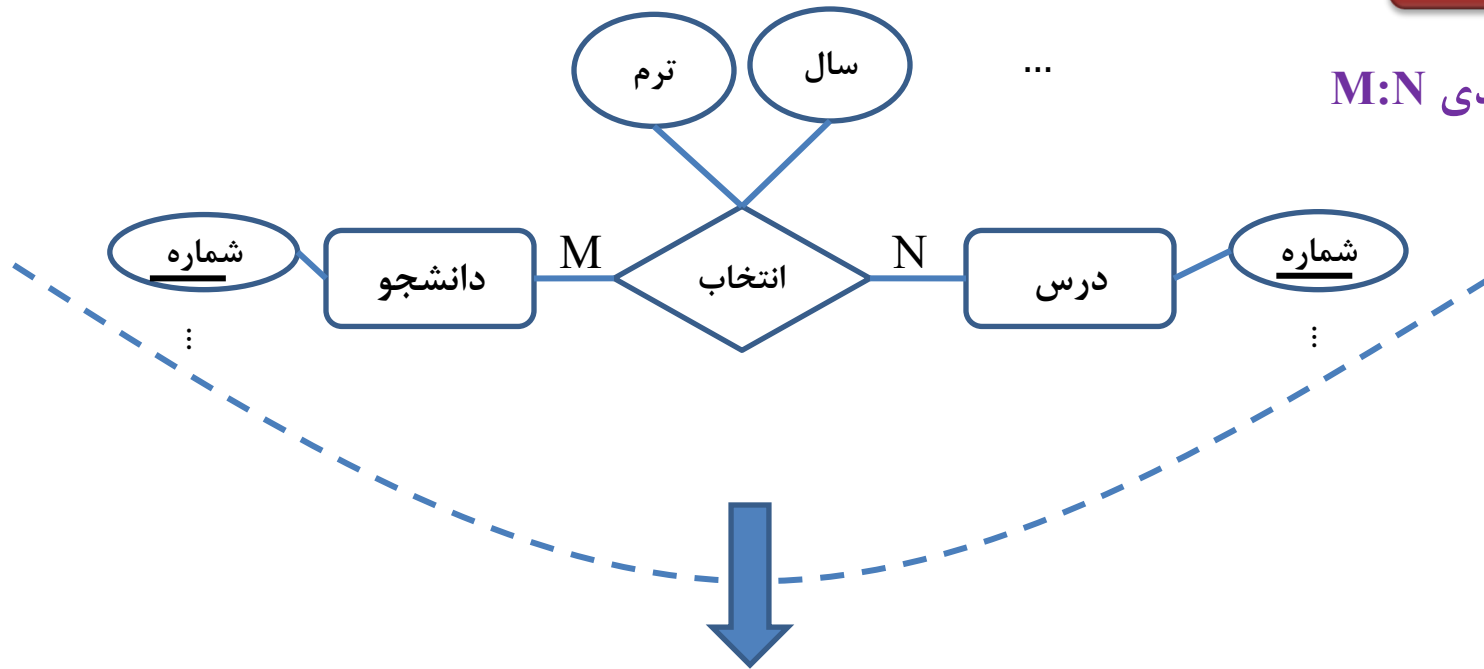
حالت ۱: طراحی ارتباط چند به چند با درجه ۲

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۱۳

حالت ۱

مثال چندی M:N



مساله: تبدیل به TDB [با TDS]

سه نوع جدول لازم داریم: ←
برای هر نوع موجودیت یک نوع جدول
برای نوع ارتباط M:N یک نوع جدول

طراحی در این حالت با کمتر از سه نوع جدول، افزونگی و هیچ‌مقداری زیادی پدید می‌آورد.



حالت ۱: طراحی ارتباط چند به چند با درجه ۲ (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۱۴

STT	<u>STID</u>	STNAME	STLEV	STMJR	STDEID
	p.k. 177	st7	bs	phys	d11
	888	st8	ms	math	d12
	444	st4	ms	phys	d11
		:	:	:	:

خط ممتد زیرین و
نماد p.k. نمایانگر
کلید اصلی

COT	<u>COID</u>	COTITLE	CREDIT	COTYPE	CEDEID
	p.k. :	:	:	:	:
	co3	programming	4	t (تئوری)	d13
	:	:	:	:	:



حالت ۱: طراحی ارتباط چند به چند با درجه ۲ (ادامه)

۱۵

بخش چهارم: طراحی منطقی پایگاه داده‌ها

طبق قواعد معنایی محیط ممکن است سال و ترم هم جزو کلید باشند.

(در واقع اگر جزیی از صفت **چند مقداری مرکب** برای رابطه باشند، جزو کلید محسوب می‌شوند).

STCOT

<u>STID</u>	<u>COID</u>	TR	YR
:	:	:	:
888	co2	1	87
888	co3	1	87
444	co2	1	87

p.k.

خط چین زیرین
نمایانگر کلید
خارجی

* ستون STID در جدول STCOT **کلید خارجی** است و با خط چین مشخص می‌شود.

کلید خارجی [کاربردی]: ستون c در جدول T2 کلید خارجی است هرگاه این ستون در جدول دیگری

مانند T1 کلید اصلی باشد.





حالت ۱: طراحی ارتباط چند به چند با درجه بزرگتر از ۲ (ادامه)

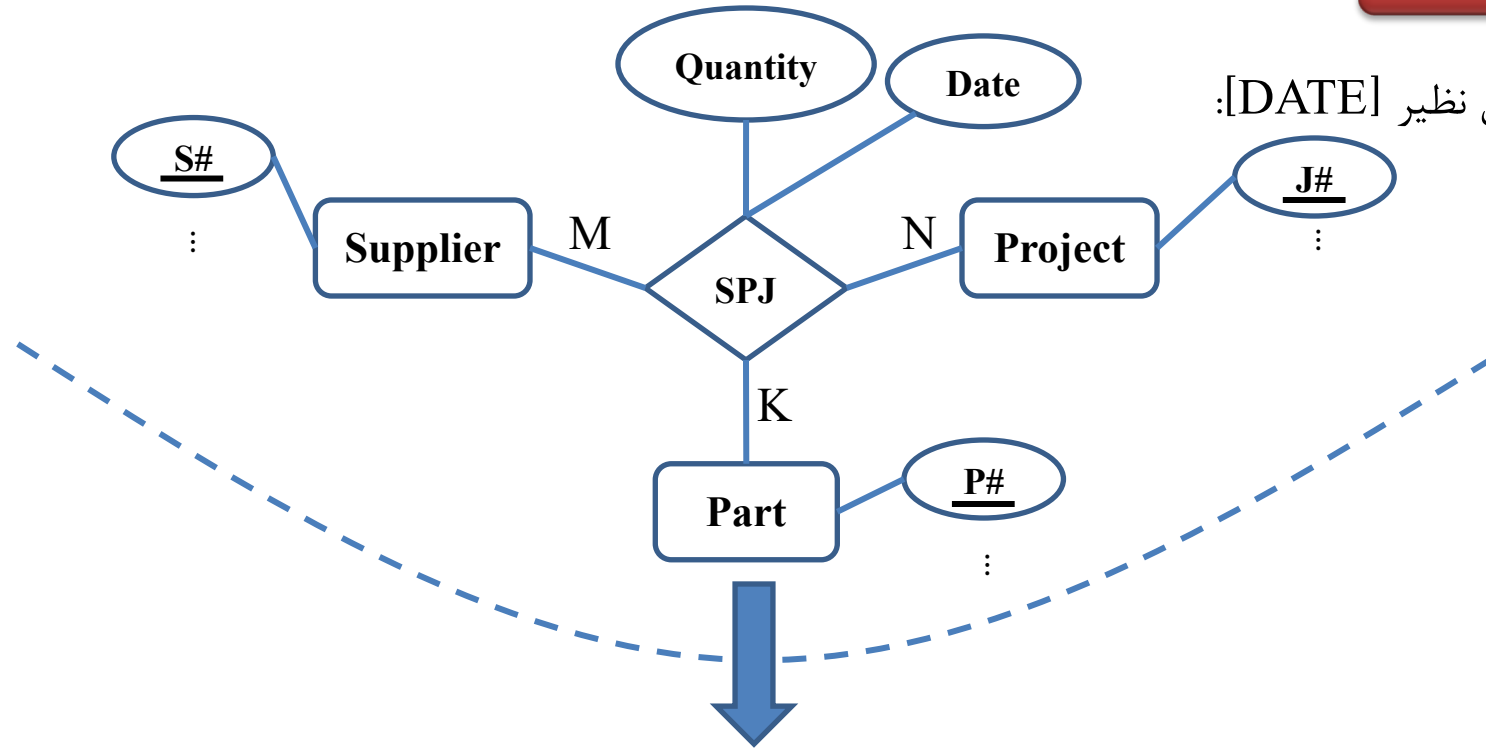
بخش چهارم: طراحی منطقی پایگاه داده‌ها

۱۶

تعمیم حالت ۱



مثال نظیر [DATE]:



مساله: تبدیل به TDB [با TDS]

چهار نوع جدول داریم: ←
} برای هر نوع موجودیت یک نوع جدول
} برای نوع ارتباط یک نوع جدول



حالت ۱: طراحی ارتباط چند به چند با درجه بزرگتر از ۲ (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۱۷

Supplier	<u>S#</u>	SNAME	CITY	...
	<small>p.k.</small> s1	...	c1	...
	s2	...	c1	...
	⋮	⋮	⋮	⋮

Part	<u>P#</u>	PNAME	CITY	...
	<small>p.k.</small> p1	...	c1	...
	p2	...	c2	...
	⋮	⋮	⋮	⋮

Project	<u>J#</u>	JNAME	CITY	...
	<small>p.k.</small> j1	...	c2	...
	j2	...	c1	...
	⋮	⋮	⋮	⋮

طبق قواعد معنایی محیط ممکن است تاریخ هم جزو کلید بشود.
(در واقع اگر صفت چند مقداری باشد، جزو کلید محسوب می‌شود.)

SPJ	<u>S#</u>	<u>P#</u>	<u>J#</u>	<u>Date</u>	QTY
	s1	<small>p.k.</small> p1	j1	d1	100
	s1	p1	j1	d2	50
	⋮	⋮	⋮	⋮	⋮



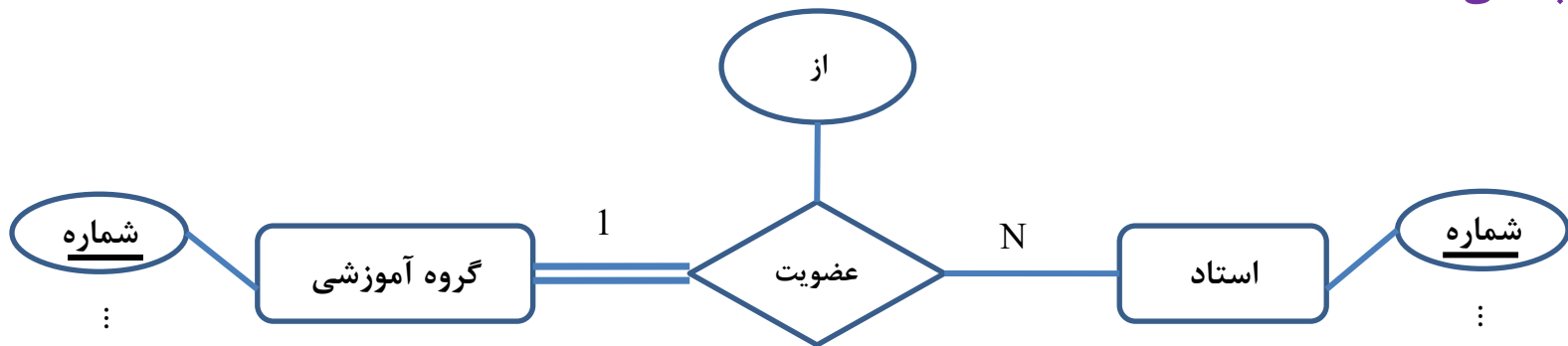
حالت ۲: طراحی ارتباط یک به چند

۱۸

بخش چهارم: طراحی منطقی پایگاه داده‌ها

حالت ۲

مثال چندی 1:N



□ دو نوع جدول داریم: } یکی برای نوع موجودیت سمت 1
یکی برای نوع موجودیت سمت N و نیز خود ارتباط



حالت ۲: طراحی ارتباط یک به چند (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۱۹

DEPT	<u>DEID</u>	DETITLE	...	DEPHONE
	^{p.k.} D11	Phys
	D12	Math
	⋮	⋮	⋮	⋮

PROF	<u>PRID</u>	PRNAME	RANK	...	FROM	<u>DEID</u>
	^{p.k.} Pr100	...	استاد	...	d1	D13
	Pr200	...	استادیار	...	d2	D11
	Pr300	...	دانشیار	...	?	?



حالت ۲: طراحی ارتباط یک به چند (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۰

□ در چه وضعی طراحی این حالت با سه جدول قابل توجیه است؟

- ۱- وقتی که مشارکت سمت N در ارتباط غیرالزامی باشد (درصد مشارکت کمتر از ۳۰ درصد) و تعداد استاد زیاد باشد، برای کاهش مقدار Null، جدول نمایشگر ارتباط را جدا می‌کنیم.
- ۲- فرکانس ارجاع به خودِ ارتباط بالا باشد و به صفات دیگر با فرکانس پایین‌تری احتیاج باشد.
- ۳- تعداد صفات خودِ ارتباط زیاد باشد و باعث زیاد شدن ستونهای جدول PROF شود.

□ اگر مشارکت سمت N الزامی باشد، باید این محدودیت معنایی را از طریق **هیچمقدار ناپذیر بودن** صفت

کلید خارجی (با استفاده از **NOT NULL**) در جدول نمایانگر نوع موجودیت سمت N ، اعلام کرد.



حالت ۳: طراحی ارتباط یک به یک

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۱

حالت ۳

مثال چندی 1:1



□ دو نوع جدول داریم: } یکی برای نوع موجودیت سمت 1 غیرالزامی
یکی برای نوع موجودیت سمت 1 الزامی و نیز خود ارتباط



حالت ۳: طراحی ارتباط یک به یک (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۲

یک طرز طراحی ممکن: ☐

DEPT	<u>DEID</u>	DETITLE	...	DEPHONE	<u>PRID</u>
	p.k.				
	D11	Phys
	D12	Math
	:	:	:	:	:

PROF	<u>PRID</u>	PRNAME	RANK	...
	p.k.			
	Pr100	...	استاد	...
	Pr200	...	استادیار	...
	Pr300	...	دانشیار	...
	:	:	:	:



حالت ۳: طراحی ارتباط یک به یک (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۳

- ☐ وقتی با **سه** جدول توجیه دارد که مشارکت طرفین غیرالزامی باشد، تعداد شرکت کنندگان (نمونه‌ها) در ارتباط زیاد باشد، درصد مشارکت در رابطه ضعیف (کمتر از ۳۰٪) باشد و نیز ملاحظات در مورد فرکانس ارجاع.
- ☐ وقتی با **یک** جدول توجیه دارد که تعداد صفات موجودیت‌ها کم باشد، مشارکت طرفین الزامی باشد و فرکانس ارجاع به ارتباط کم باشد.

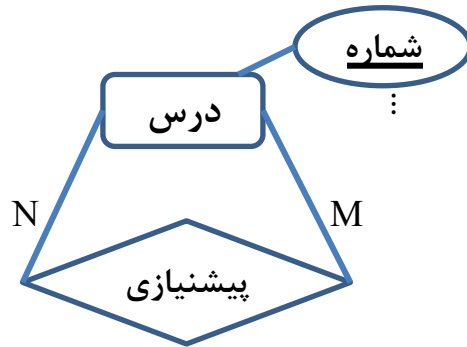


حالت ۴: طراحی ارتباط خود ارجاع چند به چند

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۴

حالت ۴



حالت خاص حالت اول

درجه ارتباط: $n=1$

چندى ارتباط: $M:N$

دو نوع جدول لازم است.

COUR

<u>COID</u>	COTITLE	CREDIT	COTYPE
p.k. :	:	:	:
co3	programming	4	t (تئوری)
:	:	:	:

COPRECO

<u>COID</u>	<u>PRECOID</u>
p.k. :	:
co3	co2
:	:



حالت ۵: طراحی ارتباط خود ارجاع یک به چند

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۵

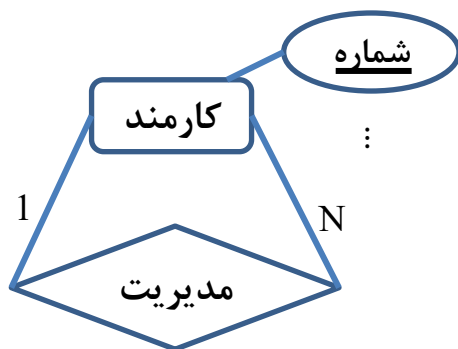
حالت ۵

□ حالت خاص حالت دوم

□ درجه ارتباط: $n=1$

□ چندی ارتباط: $1:N$

یک نوع جدول لازم است.



EMPL

<u>EMID</u>	ENAME	EPHONE	<u>EMGRID</u>
p.k. ⋮	⋮	⋮	⋮
e1	ahmadi	091276983	e10
⋮	⋮	⋮	⋮



حالت ۶: طراحی ارتباط خود ارجاع یک به یک

بخش چهارم: طراحی منطقی پایگاه داده‌ها

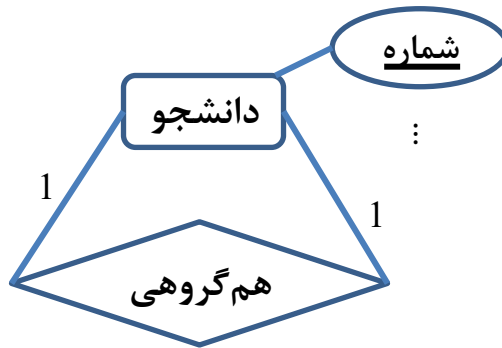
۲۶

حالت ۶

□ حالت خاص حالت سوم

□ درجه ارتباط: $n=1$

□ چندی ارتباط: $1:1$



با یک یا دو نوع جدول طراحی می‌کنیم.



حالت ۶: طراحی ارتباط خود ارجاع یک به یک (ادامه)

۲۷

بخش چهارم: طراحی منطقی پایگاه داده‌ها

□ اگر مشارکت در هم‌پروژگی زیاد نباشد، از مدل II استفاده می‌کنیم.

STPROJST

<u>STID</u> p.k.	STNAME	...	<u>JSTID</u>
:	:	:	:
st1	moradi	...	j15
:	:	:	:

صفت STID کلید اصلی و JSTID کلید است و یکتا بودن آن با Unique تعریف می‌شود.

□ در STJST هر یک از صفات می‌توانند کلید اصلی باشند.

STUD

<u>STID</u> p.k.	STNAME	...
:	:	:
st1	moradi	...
:	:	:

STJST

<u>STID</u> p.k.	<u>JSTID</u>
:	:
st1	moradi
:	:

□ آیا طرز دیگری هم برای طراحی وجود دارد؟



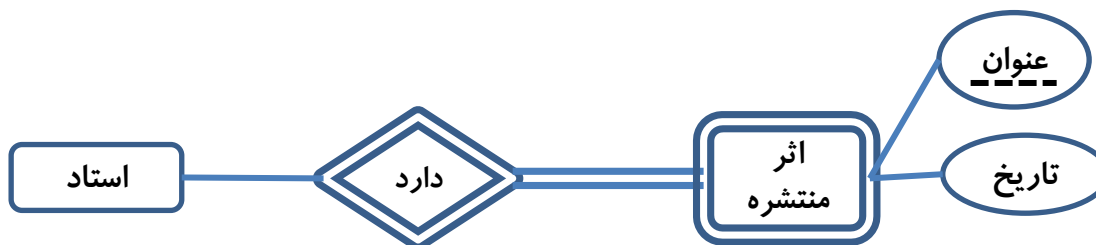
حالت ۷: طراحی موجودیت ضعیف و رابطه شناسا

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۸

حالت ۷

مثال موجودیت ضعیف و رابطه شناسا



□ دو نوع جدول داریم: ← یکی برای نوع موجودیت قوی } یکی برای نوع موجودیت ضعیف و رابطه (حاوی شناسه موجودیت قوی)



حالت ۷: طراحی موجودیت ضعیف و رابطه شناسا (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۲۹

PROF	<u>PRID</u>	PRNAME	RANK	...
	^{p.k.} Pr100	...	استاد	...
	Pr200	...	استادیار	...
	Pr300	...	دانشیار	...
	⋮	⋮	⋮	⋮

PUB	<u>PRID</u>	PTITLE	...	PDATE
	Pr100	^{p.k.} Data Encryption...
	Pr100	Semantic Analysis of
	⋮	⋮	⋮	⋮

* دو صفت PRID (کلید خارجی از جدول PROF) و PTITLE (صفت ممیزه)، کلید اصلی جدول انتشارات را تشکیل می‌دهند.

حذف و بروزرسانی در جدول PROF چه تاثیری بر PUB باید داشته باشد.



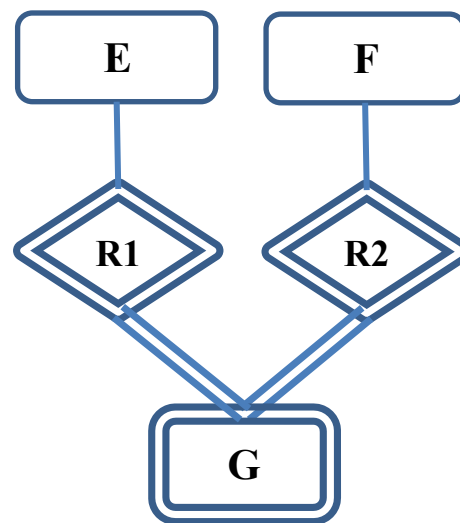
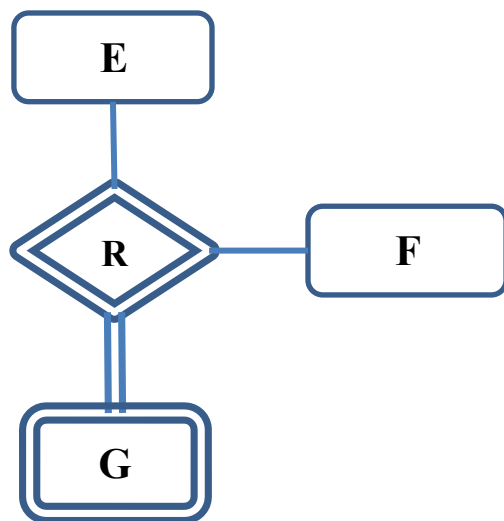


حالت ۷: طراحی موجودیت ضعیف و رابطه شناسا (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۳۰

تمرین: جداول لازم برای مدل‌های داده‌ای زیر طراحی شود. □



در این حالات، کلید جدول G از ترکیب کلید جدول‌های E و F (و در صورت وجود، صفت ممیزه G) حاصل می‌گردد.



حالت ۸: طراحی صفت چندمقداری

۳۱

بخش چهارم: طراحی منطقی پایگاه داده‌ها

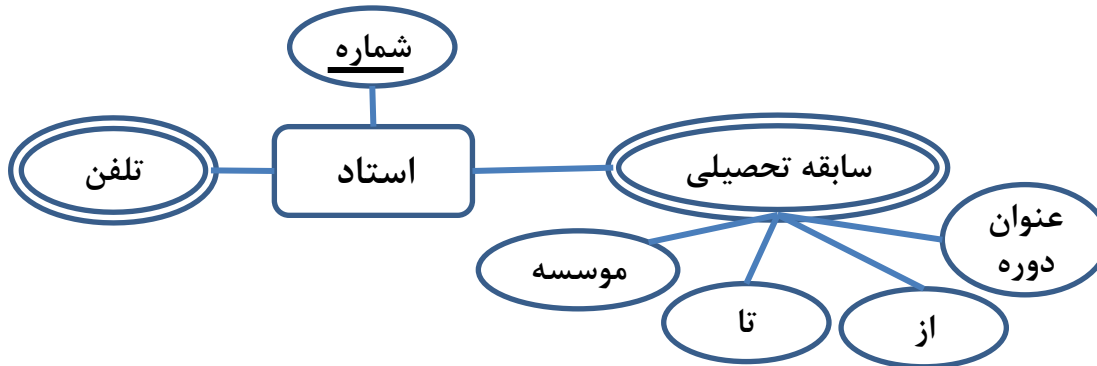
حالت ۸

□ وجود یک صفت چندمقداری برای یک نوع موجودیت.

□ دو تکنیک دارد:

[تکنیک عمومی] یک نوع جدول برای خود نوع موجودیت و یک نوع جدول برای هر صفت چندمقداری.

(بنابراین اگر نوع موجودیت E ، m صفت چندمقداری داشته باشد، $m+1$ جدول داریم.)



PROF (PRID, PRNAME,)

p.k.

PRTEL (PRID, PHONE)

p.k.

✓ جدول نمایشگر صفت چندمقداری از نوع

موجودیت اصلی FK می‌گیرد داخل کلید.



حالت ۸: طراحی صفت چندمقداری (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۳۲

□ در مدلسازی، موجودیت ضعیف به صفت چندمقداری ارجحیت دارد ولی تکنیک عمومی طراحی آنها مثل هم است.

PRHIS (PRID, TTL, FROM, TO, INSTNAME,)
p.k.

□ اشکال تکنیک عمومی: اگر برای نوع موجودیت اصلی اطلاعات کامل بخواهیم، باید اطلاعات دو جدول را با هم پیوند بزنیم که می‌تواند زمانگیر باشد.



حالت ۸: طراحی صفت چندمقداری (ادامه)

۳۳

بخش چهارم: طراحی منطقی پایگاه داده‌ها

[در شرایط خاص] طراحی با یک جدول (فرض: یک صفت چندمقداری): یک جدول برای خود نوع موجودیت و صفت چندمقداری.

□ با فرض مشخص بودن حداکثر تعداد مقداری که صفت چندمقداری می‌گیرد، به همان تعداد صفت در جدول در نظر می‌گیریم.

فرض: هر استاد حداکثر سه شماره تلفن دارد.



PRTELTEL (PRID, PRNAME, PRRANK, PHONE1, PHONE2, PHONE3)
p.k.

□ مزیت این تکنیک: نیازی به پیوند زدن اطلاعات چند جدول ندارد.

□ عیب این تکنیک: هیچمقدار (Null) در آن زیاد است، اگر تعداد کمی از استادان، سه شماره تلفن داشته باشند.



حالت ۹

□ وجود ارتباط IS-A بین دو نوع موجودیت.

□ چهار تکنیک دارد:

۱- فرض: نوع موجودیت E، n زیرنوع دارد.

n+1 نوع جدول طراحی می‌کنیم. یک نوع جدول برای زیرنوع و یک نوع جدول برای هر یک از زیرنوع‌ها.

E (EID, X, Y)

p.k.

E1 (EID, C, D)

p.k.

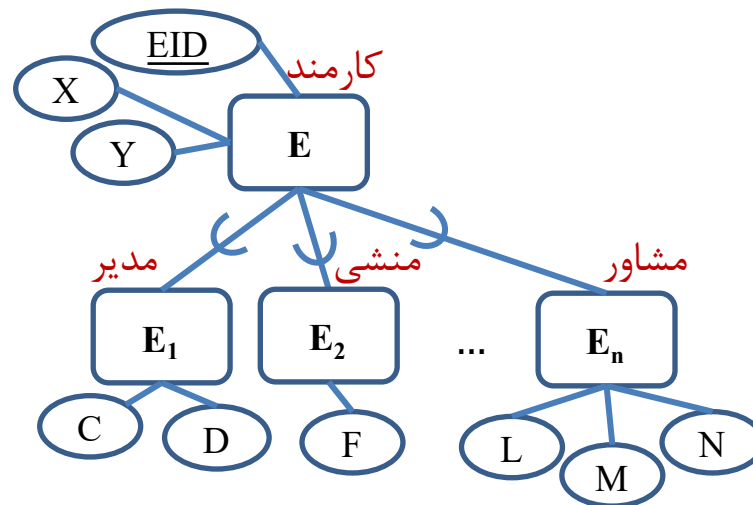
E2 (EID, F)

p.k.

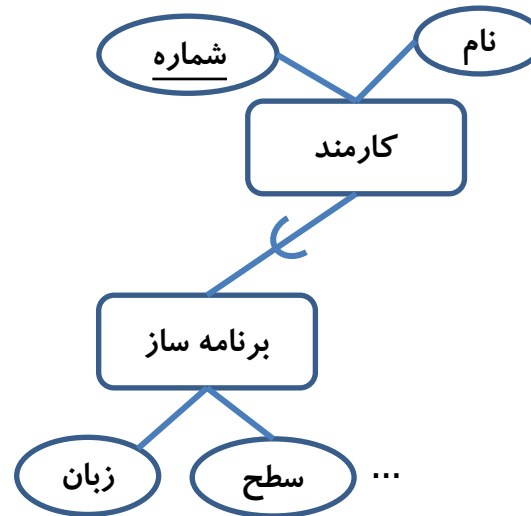
...

En (EID, L, M, N)

p.k.



مثال ارتباط IS-A



□ دو نوع جدول داریم: ← یکی برای زیرنوع موجودیت (حاوی صفات خاص زیرنوع و شناسه زیرنوع) یکی برای زبرنوع موجودیت (حاوی صفات عام یا مشترک)



حالت ۹: طراحی ارتباط IS-A (ادامه)

۳۶

بخش چهارم: طراحی منطقی پایگاه داده‌ها

EMP	<u>EID</u>	ENAME	EBDATE	...	EPHONE
	^{p.k.} E100
	E101
	E102
	⋮	⋮	⋮	⋮	⋮

PROG	<u>EID</u>	LANG	...	LEVEL
	^{p.k.} E100	C++
	E102	Java
	⋮	⋮	⋮	⋮

* EID (کلید خارجی از جدول EMP) کلید اصلی جدول PROG نیز هست.

حذف و بروزرسانی در جدول EMP چه تاثیری بر PROG باید داشته باشد (و بالعکس)?





□ مزیت این تکنیک: شرط خاصی از نظر نوع تخصیص ندارد (تکنیک‌های دیگری که مطرح می‌شود، همگی

برای شرایط خاص هستند).

□ عیب این تکنیک: اگر بخواهیم در مورد یک زیرنوع، اطلاعات کامل به دست آوریم، باید اطلاعات جدول

زیرنوع را با جدول زیرنوع پیوند بزنیم.



۲- طراحی با n جدول: برای زیرنوع، جدولی طراحی نمی‌کنیم. بنابراین صفات مشترک باید در جدول نمایشگر هر زیرنوع وجود داشته باشد.

□ شرط لازم: باید تخصیص کامل باشد. اگر نباشد، بخشی از داده‌های محیط قابل نمایش نیستند.

E1 (EID, X, Y, A, B)
p.k.

E2 (EID, X, Y, F)
p.k.

...

En (EID, X, Y, L, M, N)
p.k.

□ مزیت نسبت به تکنیک اول: برای به دست آوردن اطلاعات کامل زیرنوع‌ها نیازی به پیوند نیست.

□ نکته: در این تکنیک، لزوماً افزونگی پیش نمی‌آید. اگر تخصیص هم‌پوشا باشد میزانی افزونگی پیش می‌آید.



۳- طراحی فقط با یک جدول، با استفاده از صفت نمایشگر نوع زیرنوع‌ها

□ شرط استفاده از این تکنیک: تخصیص مجزا باشد؛ یعنی یک نمونه کارمند، جزء نمونه‌های حداکثر یک زیرنوع باشد.

E (EID, X, Y, A, B, F, L, M, N, TYPE)

<u>p.k.</u>									
100	x1	y1	a1	b1	?	?	?	?	مدیر
200	x2	y2	?	?	?	l2	m2	n2	مشاور

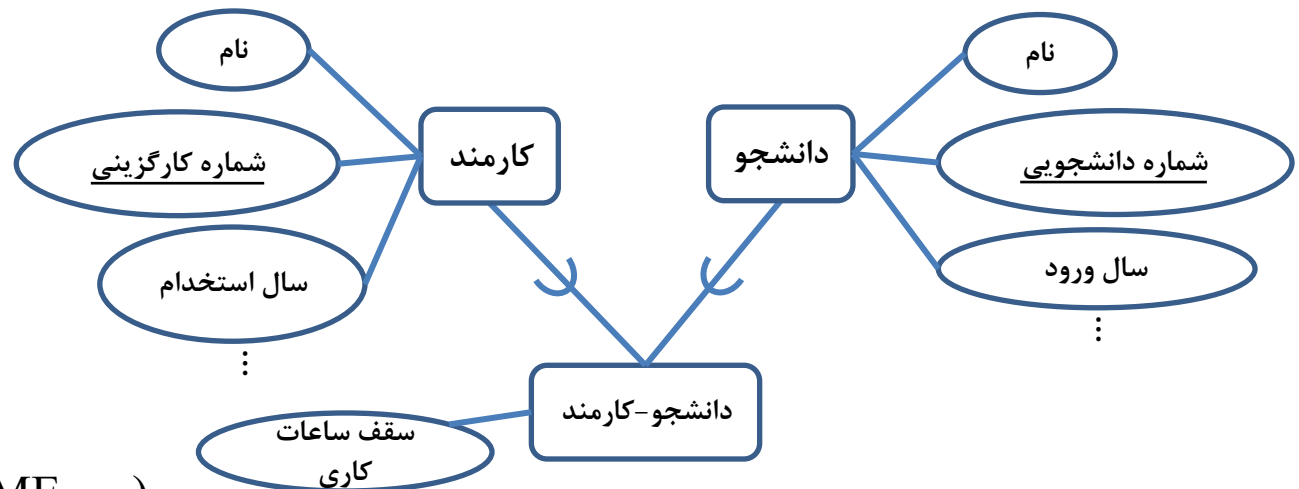
□ مزیت این تکنیک: برای به دست آوردن اطلاعات کامل زیرنوع‌ها نیازی به پیوند نیست.

□ عیب این تکنیک: هیچ مقدار (Null) زیاد دارد و تعداد ستون‌های جدول زیاد است.

حالت ۱۰

وجود ارث‌بری چندگانه بین یک زیرنوع با چندزیرنوع

□ اگر زیرنوع، n زیرنوع داشته باشد، جدول نمایشگر زیرنوع حداقل n کلید ممکن دارد. کلید با ارجاع بیشتر کلید اصلی انتخاب می‌شود.



STUD (STID, STNAME, ...)
p.k.

EMPL (EID, ENAME, ...)
p.k.

STEM (STID, EID, MAXW)
p.k.

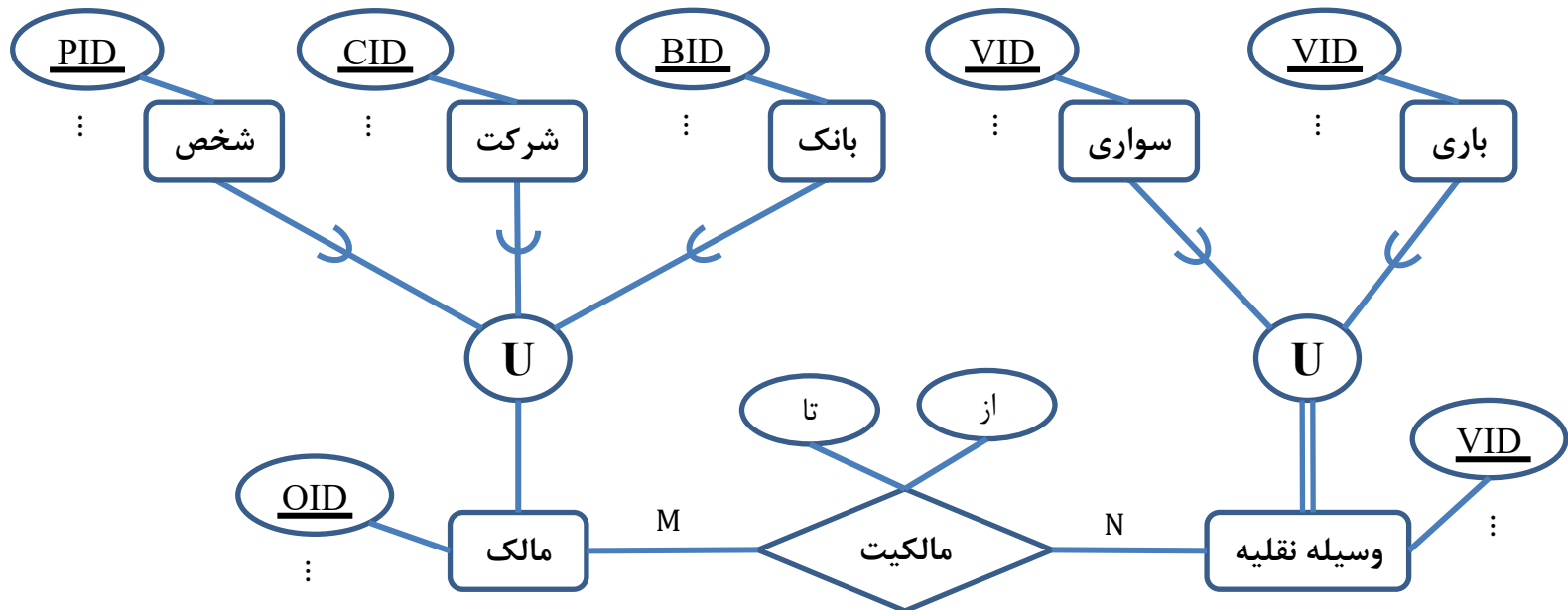
آیا ممکن است برای زیرنوع اصلاً جدول طراحی نکنیم؟



حالت ۱۱

□ نوع موجودیت E، زیرنوع U-Type (دسته یا Category) n زیرنوع است.

n+1 نوع جدول طراحی می‌کنیم.





حالت ۱۱: طراحی زیرنوع اجتماع (ادامه)

۴۳

بخش چهارم: طراحی منطقی پایگاه داده‌ها

□ $n+1$ نوع جدول

□ اگر شناسه زبرنوع‌ها از دامنه‌های متفاوت باشد، جدول نمایشگر زیرنوع، FK می‌دهد به جدول‌های نمایشگر زبرنوع‌ها، خارج از کلید.

□ اگر شناسه زبرنوع‌ها از یک دامنه باشد (و مقادیر شناسه در همه نمونه‌های زبرنوع‌ها یکتا باشد)، کلید

جدول نمایشگر زیرنوع، همان کلید جدول‌های نمایشگر زبرنوع‌ها است. **PERS** (PID, ..., OID)
p.k.

COMP (CID, ..., OID)
p.k.

BANK (BID, ..., OID)
p.k.

OWNER (OID, ...) ← چون دامنه کلیدهای زبرنوع‌ها یکسان نیست، خودمان کلید ساختگی می‌گذاریم.
p.k.

VEHIC (VID, ...) p.k.

OWNS (OID, VID, F, T, ...) p.k.

SAVARY (VID, N, ...) p.k.

BARY (VID, T, ...) p.k.

آیا طرز طراحی دیگری وجود دارد.

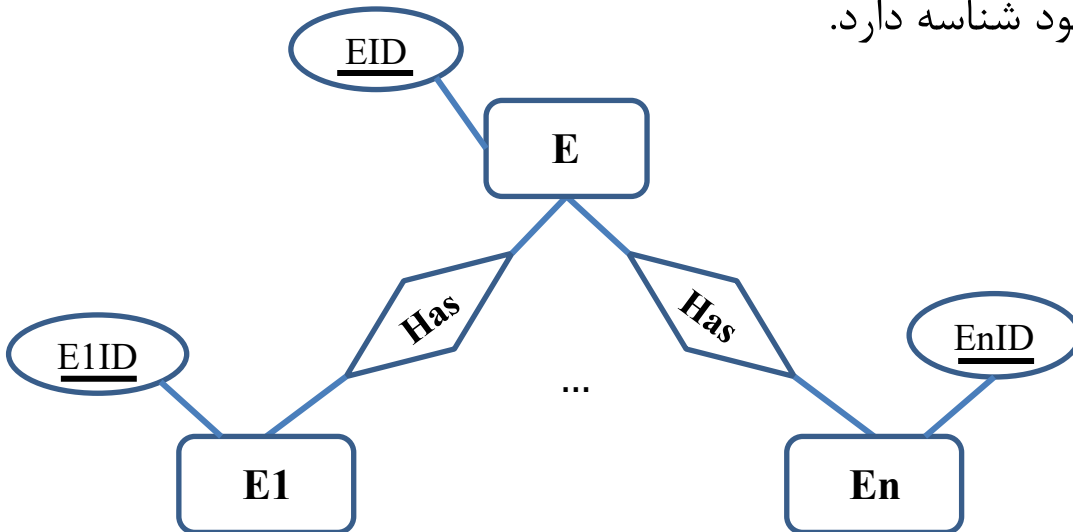


حالت ۱۲

وجود ارتباط IS-A-PART-OF

اگر نوع موجودیت کل، n نوع موجودیت جزء داشته باشد، تعداد $n+1$ نوع جدول طراحی می‌کنیم.

توجه داریم که نوع موجودیت جزء از خود شناسه دارد.



$E (EID, \dots)$

p.k.

$E1 (E1ID, EID, \dots)$

p.k.

....

$En (EnID, EID, \dots)$

p.k.

آیا طرز طراحی دیگری وجود دارد؟ در چه شرایطی؟





حالت ۱۳: طراحی تکنیک تجميع (Aggregation)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۴۵

حالت ۱۳

استفاده از تکنیک Aggregation در مدلسازی

ابتدا نوع موجودیت انتزاعی (بخش درون مستطیل خط‌چین) را طراحی می‌کنیم (با توجه به درجه و چندی ارتباط). سپس بخش بیرون آن را (باز هم با توجه به چندی ارتباط و درجه آن).

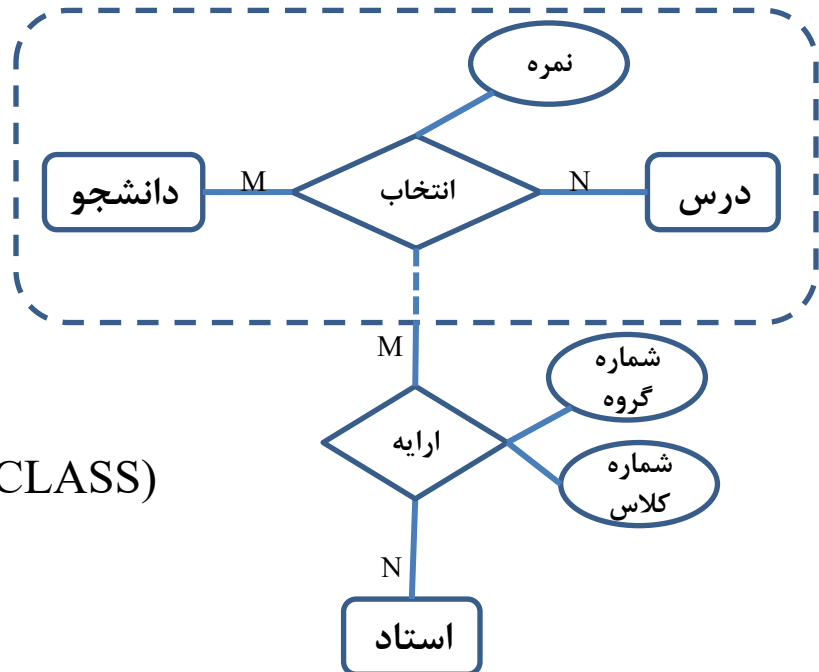
STUD (STID,)
p.k.

COUR (COID,)
p.k.

SCR (STID, COID, GR)
p.k.

PROF (PRID,)
p.k.

OFFERING (STID, COID, PROFID, GR#, CLASS)
p.k.





حالت ۱۳: طراحی تکنیک تجميع (ادامه)

بخش چهارم: طراحی منطقی پایگاه داده‌ها

۴۶

این تکنیک چگونه کارایی سیستم را افزایش می‌دهد (نسبت به طراحی با یک ارتباط سه-تایی)؟ ☐

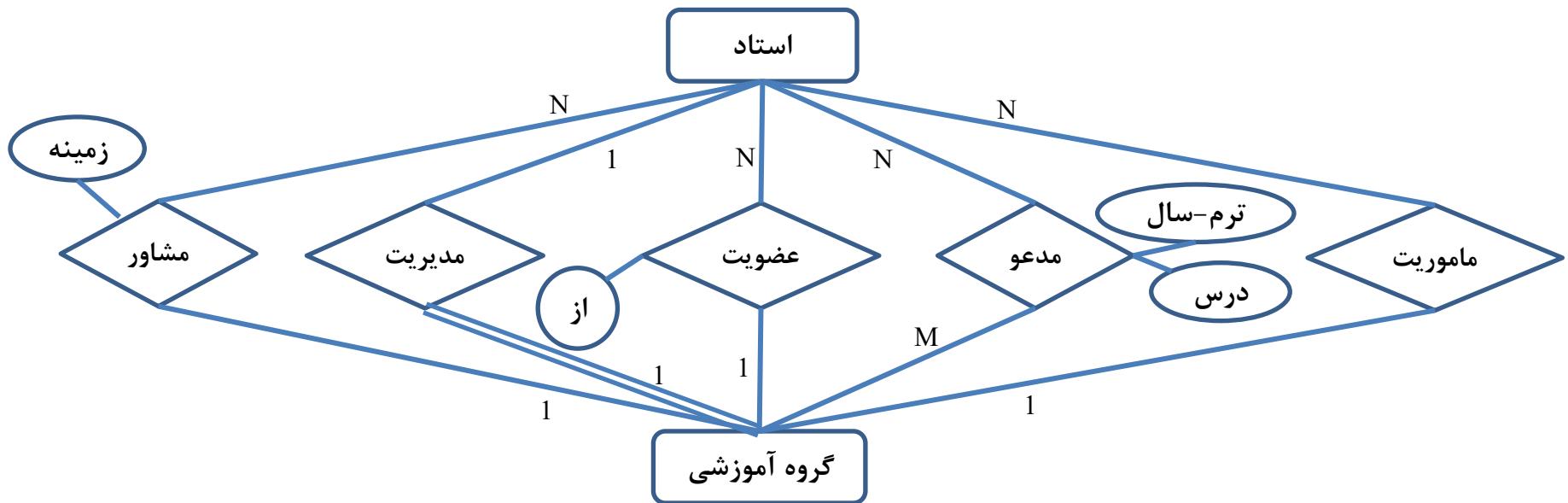
اگر مراجعه به ارتباط «انتخاب» بالا باشد و فرکانس ارجاع به ارتباط «ارائه» پایین باشد، سیستم با این ☐

طراحی کاراتر عمل می‌کند.

□ در صورتیکه چند ارتباط مثلاً بین دو نوع موجودیت برقرار باشد.

□ هر ارتباط را با توجه به وضع آن از نظر درجه و چندی ارتباط طراحی می‌کنیم. اما برای کاهش احتمال

اشتباه در طراحی توصیه می‌شود اول ارتباطهای $M:N$ ، سپس $1:N$ و در آخر $1:1$ را طراحی نماییم.





طراحی منطقی با وجود چند ارتباط (ادامه)

۴۸

بخش چهارم: طراحی منطقی پایگاه داده‌ها

DEPT (DEID,, DPHONE, PRID)
p.k. p.k.

PROF (PRID,, PRRANK, MDEID, SUB, MEMDEID, FROM, CDEID, INT)
p.k.

زمینه مشاور از عضویت موضوع ماموریت

سه کلید خارجی از یک دامنه

INVITED (DEID, PRID, YR, TR)
p.k.

همین سیستم حداکثر با هفت جدول نیز قابل طراحی است. □



پرسش و پاسخ ...

amini@sharif.edu