دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

# طراحی پایگاه داده‌ها

## ( فصل سوم : اصول طراحی پایگاه داده )


مهدی دادبخش

mahdi.dadbakhsh@sharif.edu


شماره درس : ۴۰۳۸۴

یکشنبه – سه‌شنبه ( ۱۶:۳۰ الی ۱۸:۰۰ )

۱۴۰۲ – ۱۴۰۱

طراحی منطقی پایگاه داده

ساختار داده

ساختار داده جدولی

پایگاه داده جدولی

ویژگی‌های طراحی خوب

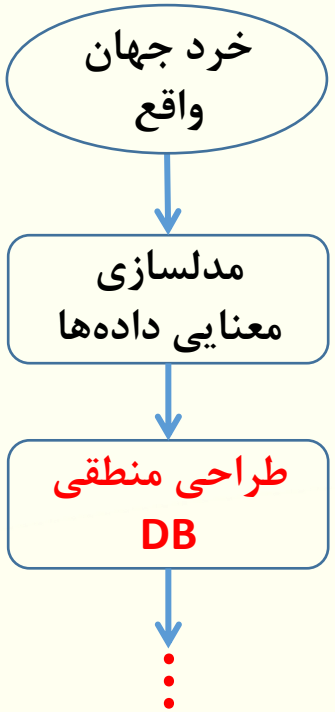
مفهوم کلید و انواع آن

روش طراحی بالا به پایین

پایان

# طراحی منطقی پایگاه داده ( Database Logical Design )

- مدلسازی داده‌ها می‌تواند در سطوح انتزاعی مختلفی صورت پذیرد.
- 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.
- 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 – Table Data Structure )
  - پایگاه داده جدولی ( TDB - Table Database )
  - زبان پایگاهی جدولی ( TDBL – Table Database Language )

# ساختار داده ( Data Structure )

برای نمایش موجودیت‌ها و ارتباط بین آنها در سطح منطقی به یک ساختار داده نیاز داریم.

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

- تامین کننده محیط انتزاعی
- مبنا و چارچوب طراحی منطقی پایگاه داده
- مبنا و چارچوب طراحی زبان پایگاه داده‌ها
- مبنا و چارچوب طراحی خود سیستم مدیریت پایگاه داده
- ضابطه‌ای است برای مقایسه سیستم‌های مدیریت پایگاه داده و ارزیابی آنها
- مبنایی است برای ایجاد و گسترش تکنیک‌های طراحی پایگاه داده
- . . .

**ساختارهای داده در حیطه دانش و تکنولوژی پایگاه داده :**

- ساختار داده سلسله مراتبی ( HDS - Hierarchical Data Structure )
- ساختار داده شبکه‌ای ( NDS – Network Data Structure )
- ساختار داده رابطه‌ای ( RDS – Relational Data Structure )
- ساختار داده شی ای ( ODS – Object Data Structure )
- ساختار داده شی-رابطه ای ( ORDS – Object Relational Data Structure )

# ساختار داده جدولی

- عنصر ساختاری اساسی در مدل رابطه‌ای Relational Model ، مفهوم رابطه است.
- رابطه ( Relation ) یک مفهوم ریاضی است ، اما از دید کاربر و در عمل، نمایش جدولی دارد.
- ساختار داده جدولی ( TDS ) فقط یک عنصر ساختاری اساسی دارد که همان جدول نامیده می‌شود.
  - جدول، عنصری است که به کمک آن موجودیت، ارتباط، و یا هردو آنها را نمایش می‌دهیم.

- **اصطلاحات مربوط به ساختار داده جدولی :**
  - **جدول :** در ساختار داده جدولی داده ها در قالب جدول ذخیره می‌شوند. هر جدول دارای تعدادی سطر و ستون می‌باشد. هر موجودیت به صورت جدول در پایگاه داده ذخیره می‌گردد. مثال : جدول "دانشجو" که اطلاعات مربوط به دانشجویان را در خود جای می‌دهد.
  - **ستون یا فیلد :** کوچکترین واحد ذخیره داده در پایگاه داده، فیلد نامیده می‌شود. هر صفت خاصه موجودیت در قالب یک فیلد در جدول موجودیت ذخیره می‌شود. مثال: نام و نام خانوادگی و ... فیلدهای جدول دانشجو هستند.
  - **سطر یا رکورد :** هر نمونه از یک موجودیت را یک رکورد از آن موجودیت می‌نامند. اطلاعات هر رکورد از موجودیت در قالب یک سطر در جدول آن موجودیت ذخیره می‌شود. مثال: هر سطر جدول دانشجو حاوی اطلاعات یک دانشجو می باشد.

# پایگاه داده جدولی

- **پایگاه داده جدولی :**
  - از نظر نوع، مجموعه‌ای از تعدادی جدول می‌باشد.
  - از نظر محتوای داده‌ای، مجموعه‌ای از نمونه‌های متمایز موجودیت یک یا چند سطر است.

- **طراحی پایگاه داده جدولی :**
  - برای طراحی پایگاه داده جدولی ( رابطه‌ای ) باید موارد زیر را مشخص کنیم :
    - ❑ مجموعه‌ای از جدول‌ها (رابطه‌ها)
    - ❑ کلید(های) هر جدول (در مدل رابطه‌ای کلیدهای کاندید رابطه )
    - ❑ کلید اصلی هر جدول (رابطه)
    - ❑ کلیدهای خارجی هر جدول (رابطه)، در صورت وجود
    - ❑ محدودیت‌های جامعیتی ناظر بر هر جدول (رابطه)

- **روش‌های طراحی پایگاه داده جدولی :**
  - طراحی به روش بالا به پایین ( Top Down ) :
    - ❑ ابتدا مدل سازی داده‌ها را با روشِ ER یا UML انجام می‌دهیم و سپس مدل سازی را به مجموعه‌ای از جداول (رابطه‌ها) تبدیل می‌کنیم.
  - طراحی به روش نرمال سازی رابطه‌ها :
    - ❑ درآینده شرح داده خواهد شد.

# ویژگی‌های طراحی خوب

طراحی خوب است که ویژگی‌های زیر را داشته باشد :

- نمایش صحیح و واضح از خرد جهان 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!

# مفهوم کلید و انواع آن

- صفت شناسه در 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.
- کلید امکان دسترسی به تک نمونه (از یک موجودیت یا ارتباط) را فراهم می‌نماید. لذا مقدار آن در سطرهای جدولِ مربوط به موجودیت یا ارتباط، یکتا است.
- 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 و با استفاده از UNIQUE در SQL مشخص می‌نماییم.

- **انواع کلید :**

۱ – ابر کلید: هر ترکیبی از صفات خاصه که خاصیت کلید داشته باشد ( منحصر به فرد باشد ) ابر کلید نام دارد.
مانند: شماره دانشجویی _ کد ملی _ شماره دانشجویی و کد ملی _کد ملی ، نام و نام خانوادگی.
۲ - کلید کاندید: ابر کلیدی است که از نظر تعداد فلید کمینه است.
یعنی اگر هر یک از فیلد های آن حذف شود دیگر منحصر به فرد نباشد. مانند: شماره دانشجویی _ کد ملی.
۳ - کلید اصلی: کلید کاندیدی است که توسط طراح بانک بر اساس دو معیار زیر انتخاب می‌شود:
الف) نقش و اهمیت آن نسبت به سایر کلید های کاندید ب) طول کمتر
مثال : در سیستم آموزش دانشگاه شماره دانشجویی نسبت به کد ملی اهمیت بیشتری دارد، پس کلید اصلی است.
۴ – کلید فرعی یا بدیل: هر کلید کاندید غیر از کلید اصلی را کلید فرعی می‌نامند مانند : کد ملی.
۵ – کلید خارجی: وسیله ای است برای ایجاد ارتباط بین جداول. مقدار کلید خارجی می تواند منحصر به فرد نباشد.

# روش طراحی بالا به پایین

- در تبدیل نمودار ER یا EER به مجموعه‌ای از جداول، نهایتاً طراح تصمیم می‌گیرد چند جدول (رابطه) داشته باشد.
- در نمودار مدلسازی معنایی داده‌ها، حالات متعدد داریم، که در طراحی باید به آنها توجه بپردازیم :

طراحی ارتباط یک به یک

طراحی ارتباط یک به چند

طراحی ارتباط چند به چند

طراحی ارتباط خود ارجاع یک به یک

طراحی ارتباط خود ارجاع یک به چند

طراحی ارتباط خود ارجاع چند به چند

طراحی موجودیت ضعیف و رابطه شناسا

طراحی صفت چندمقداری

طراحی ارتباط IS-A

طراحی ارث بری چندگانه

طراحی زیر نوع اجتماع ( U-type )

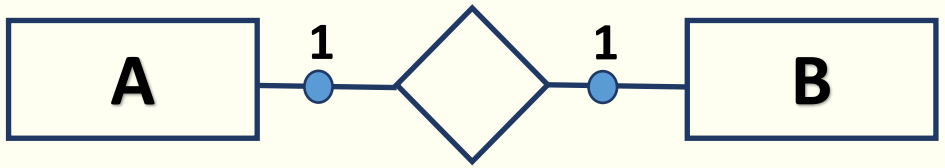
طراحی ارتباط IS-A-PART-OF

طراحی تکنیک تجمیع ( Aggregation )

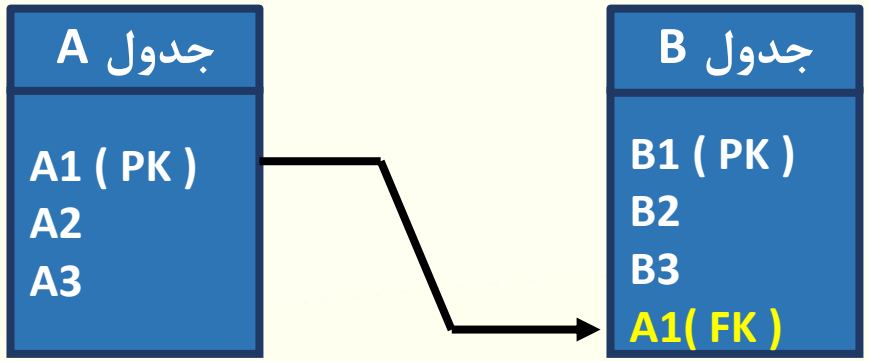
طراحی منطقی با وجود چند ارتباط

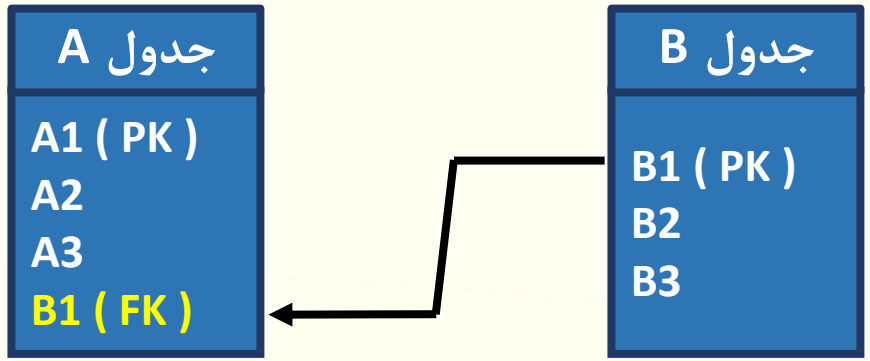
# طراحی ارتباط یک به یک

کلید اصلی ( PK - Primary Key ) یکی از جداول به عنوان کلید خارجی ( FK - Foreign Key ) در جدول دیگر قرار می گیرد.




فرض کنید جدول A ( با فیلدهای A1 ، A2 و A3 ) و جدول B ( با فیلدهای B1 ، B2 و B3 ) موجود است. ارتباط آنها به صورت زیر می باشد:




مثال

# طراحی ارتباط یک به یک – مثال


**DEPT**

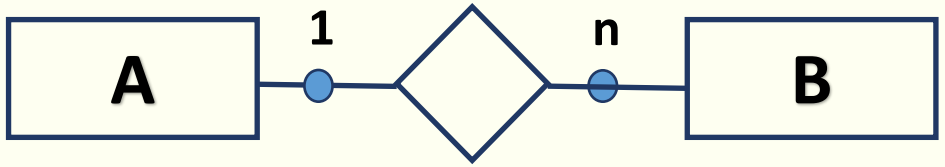
| DEID | DETITLE | ... | DEPHONE | PRID |
|---|---|---|---|---|
| D11 | Phys | ... | ... | ... |
| D12 | Math | ... | ... | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

**PROF**

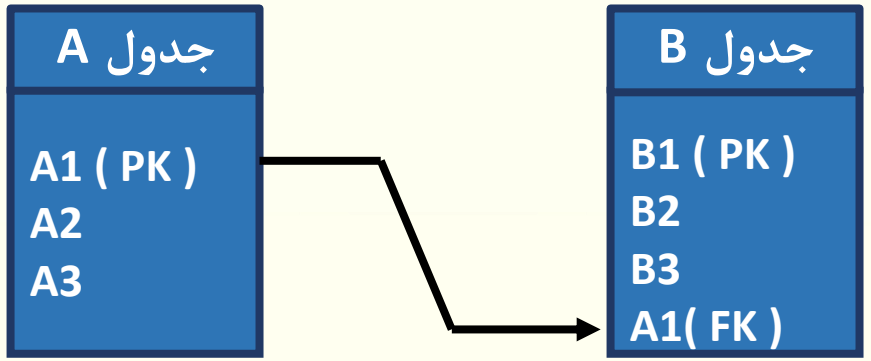
| PRID | PRNAME | RANK | ... |
|---|---|---|---|
| Pr100 | ... | استاد | ... |
| Pr200 | ... | استادیار | ... |
| Pr300 | ... | دانشیار | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

# طراحی ارتباط یک به چند

کلید اصلی ( PK - Primary Key ) جدول طرف یک رابطه به عنوان کلید خارجی ( FK - Foreign Key ) جدول دیگر قرار می گیرد.

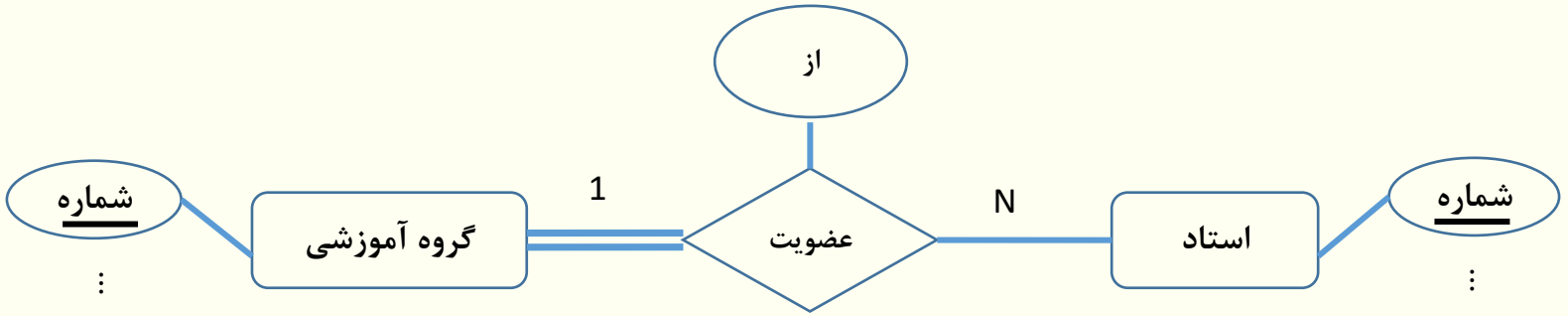



فرض کنید جدول A ( با فیلدهای A1 ، A2 و A3 ) و جدول B ( با فیلدهای B1 ، B2 و B3 ) موجود است. ارتباط آنها به صورت زیر می باشد:




مثال

# طراحی ارتباط یک به چند – مثال




**DEPT**

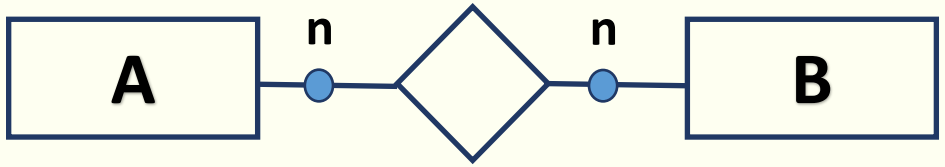
| DEID | DETITLE | ... | DEPHONE |
|---|---|---|---|
| D11 | Phys | ... | ... |
| D12 | Math | ... | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

**PROF**

| PRID | PRNAME | RANK | ... | FROM | DEID |
|---|---|---|---|---|---|
| Pr100 | ... | استاد | ... | d1 | D13 |
| Pr200 | ... | استادیار | ... | d2 | D11 |
| Pr300 | ... | دانشیار | ... | ? | ? |

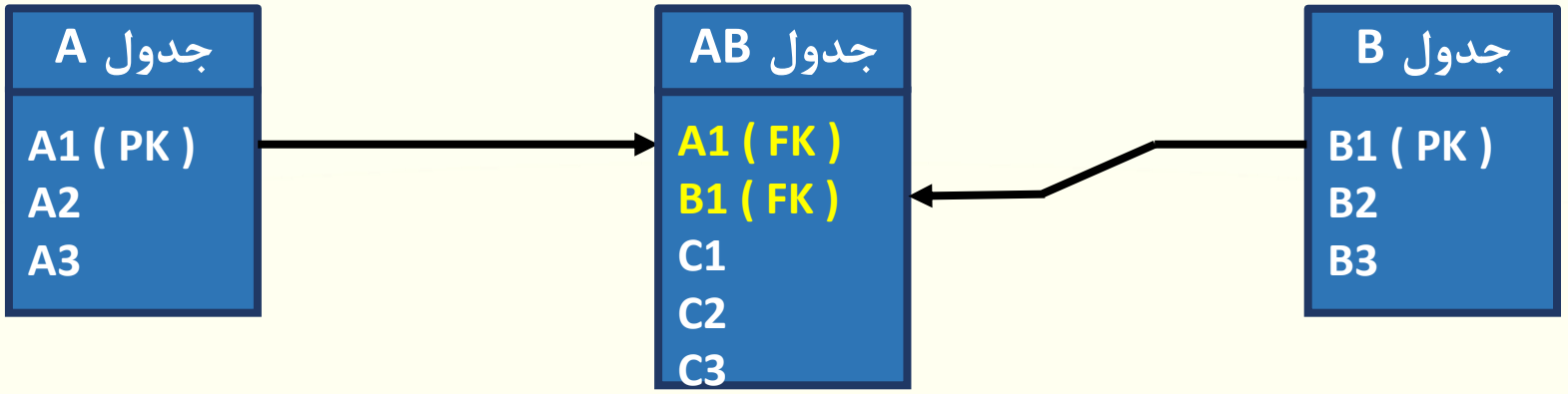
# طراحی ارتباط چند به چند

برای ایجاد ارتباط بین دو جدول با ید از یک جدول واسط استفاده کرد، به طوری که کلید اصلی هر جدول به عنوان کلید خارجی در جدول واسط قرار داده می‌شود. جدول واسط علاوه بر این کلیدهای خارجی می تواند فیلدهای دیگری نیز داشته باشد.



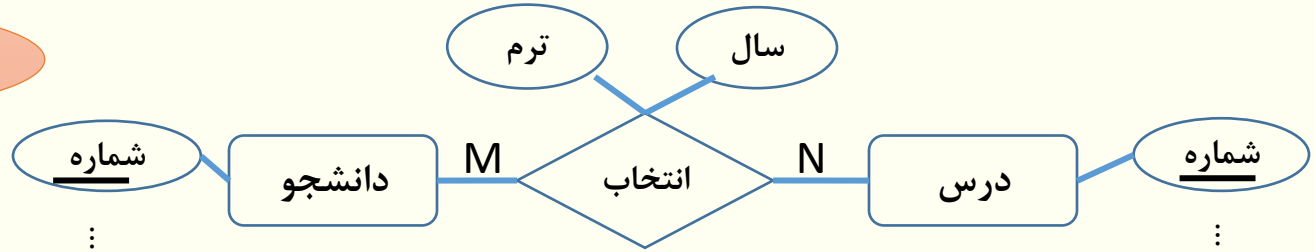

کلید اصلی جدول واسط می تواند یکی از شرایط زیر باشد:

۱ – ترکیب کلیدهای خارجی هر دو جدول ۲ – فیلدی غیر از کلیدهای خارجی ۳ - جدول واسط می تواند فاقد کلید اصلی باشد.




مثال

# طراحی ارتباط چند به چند – مثال




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

STT

| STID | STNAME | STLEV | STMJR | STDEID |
|---|---|---|---|---|
| 777 | st7 | bs | phys | d11 |
| 888 | st8 | ms | math | d12 |
| 444 | st4 | ms | phys | d11 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

COT

| COID | COTITLE | CREDIT | COTYPE | CEDEID |
|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| co3 | programming | 4 | t (تئوری) | d13 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

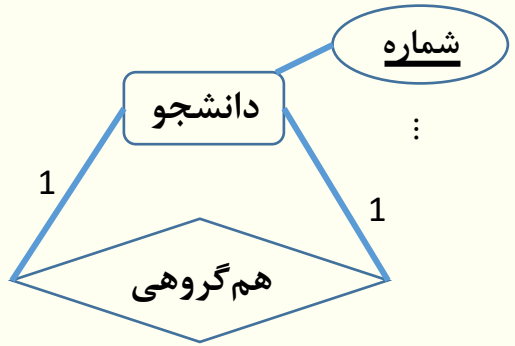
STCOT

| STID | COID | TR | YR |
|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ |
| 888 | co2 | 1 | 87 |
| 888 | co3 | 1 | 87 |
| 444 | co2 | 1 | 87 |

خط‌چین زیرین نمایانگر کلید خارجی

# طراحی ارتباط خود ارجاع یک به یک

- در این حالت به دوصورت می‌توان عمل کرد :
  - ۱ - استفاده از یک جدول
  - ۲ – استفاده از دو جدول




**STPROJST**

| STID | STNAME | ... | JSTID |
|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ |
| st1 | moradi | ... | j15 |
| ⋮ | ⋮ | ⋮ | ⋮ |

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

**STUD**

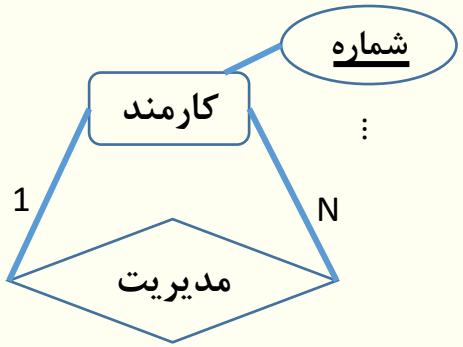
| STID | STNAME | ... |
|---|---|---|
| ⋮ | ⋮ | ⋮ |
| st1 | moradi | ... |
| ⋮ | ⋮ | ⋮ |

**STJST**

| STID | JSTID |
|---|---|
| ⋮ | ⋮ |
| st1 | moradi |
| ⋮ | ⋮ |

# طراحی ارتباط خود ارجاع یک به چند

- در این حالت به دو جدول نیاز داریم :

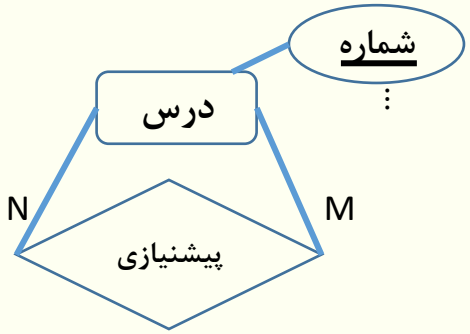



**EMPL**

| EMID | ENAME | EPHONE | EMGRID |
|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ |
| e1 | ahmadi | 091276983 | e10 |
| ⋮ | ⋮ | ⋮ | ⋮ |

# طراحی ارتباط خود ارجاع چند به چند

- در این حالت به دو جدول نیاز داریم :




**COUR**

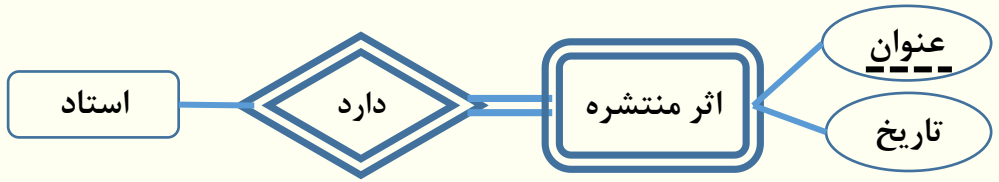
| COID | COTITLE | CREDIT | COTYPE |
|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ |
| co3 | programming | 4 | t (تئوری) |
| ⋮ | ⋮ | ⋮ | ⋮ |

**COPRECO**

| COID | PRECOID |
|---|---|
| ⋮ | ⋮ |
| co3 | co2 |
| ⋮ | ⋮ |

# طراحی موجودیت ضعیف و رابطه شناسا

- در این حالت دو جدول نیاز داریم :
  - یکی برای موجودیت قوی
  - یکی برای موجودیت ضعیف و رابطه ( حاوی شناسه موجودیت قوی )



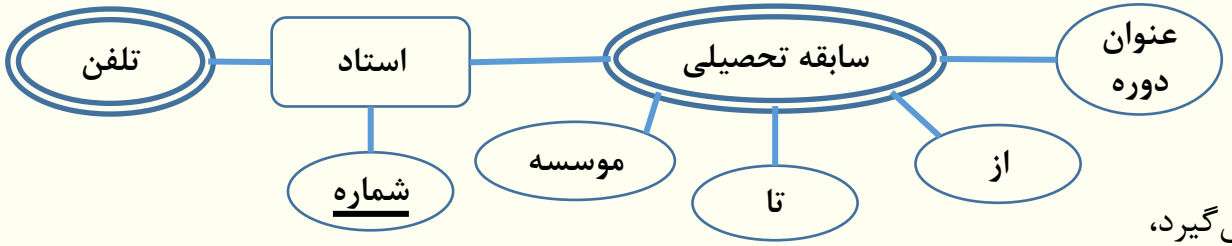

**PROF**

| PRID | PRNAME | RANK | ... |
|---|---|---|---|
| Pr100 | ... | استاد | ... |
| Pr200 | ... | استادیار | ... |
| Pr300 | ... | دانشیار | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

**PUB**

| PRID | PTITLE | ... | PDATE |
|---|---|---|---|
| Pr100 | Data Encryption... | ... | ... |
| Pr100 | Semantic Analysis of ... | ... | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

# طراحی صفت چندمقداری




- برای این منظور دو تکنیک متفاوت وجود دارد :
  - **تکنیک اول :**

یک جدول برای 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 کنید هر استاد حداکثر سه شماره تلفن دارد.

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

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

**PRTELTEL** (PRID, PRNAME, PRRANK, PHONE1, PHONE2, PHONE3)

- **تکنیک دوم :**

یک جدول برای موجودیت و یک جدول برای هر صفت چندمقداری در نظر گرفته می‌شود.

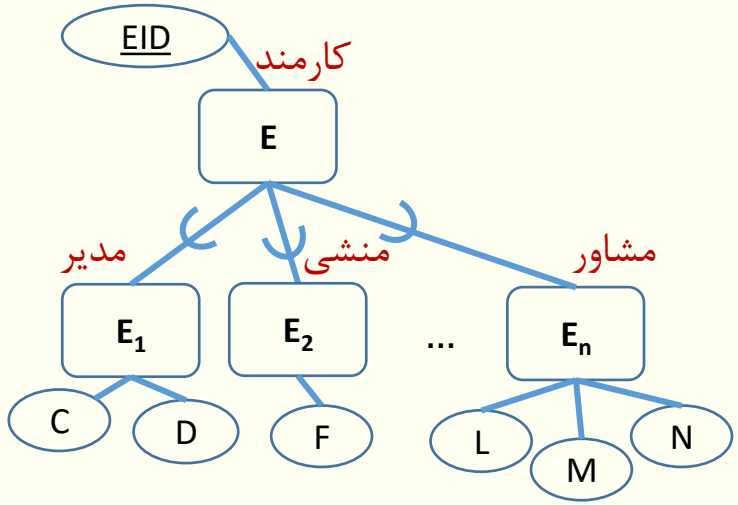
**عیب این تکنیک :** اگر برای نوع موجودیت اصلی اطلاعات کامل بخواهیم، باید اطلاعات دو جدول را با هم پیوند بزنیم که می‌تواند زمانگیر باشد.

**PROF** (PRID, PRNAME, ....)

**PRTEL** (PRID, PHONE)

# طراحی ارتباط IS-A

- فرض کنید در سیستم مورد نظر، موجودیتی مانند E ( کارمند ) داریم که تعداد n زیرموجودیت E1 ( مدیر ) ، E2 ( منشی ) ، ... و En ( مشاور ) دارد.
- برای طراحی ارتباط IS-A چهار تکنیک مختلف وجود دارد که به معرفی آنها می‌پردازیم.




تکنیک اول | تکنیک دوم | تکنیک سوم | تکنیک چهارم

# تکنیک اول طراحی ارتباط IS-A

**طراحی با n+1 جدول :**

- در این حالت، n+1 جدول تعریف می‌کنیم. یک جدول برای موجودیت اصلی و n جدول دیگر برای n زیرموجودیت آن.
- **مزیت این تکنیک :**

  شرط خاصی از نظر نوع تخصیص ندارد.
- **عیب این تکنیک :**

  اگر بخواهیم در مورد یک زیرموجودیت، اطلاعات کامل به دست آوریم، باید اطلاعات جدول موجودیت را با جدول زبرموجودیت پیوند بزنیم.

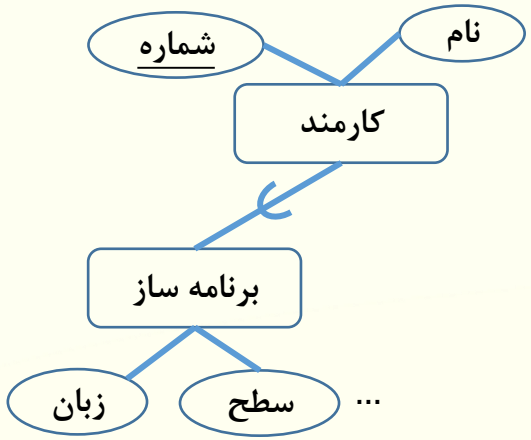
**E** (EID, X, Y)

**E1** (EID, C, D)

**E2** (EID, F)

...

**En** (EID, L, M, N)




**EMP**

| EID | ENAME | EBDATE | ... | EPHONE |
|---|---|---|---|---|
| E100 | ... | ... | ... | ... |
| E101 | ... | ... | ... | ... |
| E102 | ... | ... | ... | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

**PROG**

| EID | LANG | ... | LEVEL |
|---|---|---|---|
| E100 | C++ | ... | ... |
| E102 | Java | ... | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

# تکنیک دوم طراحی ارتباط IS-A

**طراحی با n جدول :**

- در این حالت، برای موجودیت اصلی جدول تعریف نمی‌کنیم و فقط n جدول دیگر برای n زیرموجودیت آن داریم.
- بنابراین صفات مشترک باید در جدول هر زیرموجودیت حضور داشته باشد.
- **شرط لازم:**

  باید تخصیص کامل باشد. اگر نباشد، بخشی از داده‌های محیط قابل نمایش نیستند.
- **مزیت این تکنیک:**

  برای به دست آوردن اطلاعات کامل زیرموجودیت نیازی به پیوند نیست.
- **نکته :**

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

**E1** (EID, X, Y, A, B)

**E2** (EID, X, Y, F)

...

**En** (EID, X, Y, L, M, N)

# تکنیک سوم طراحی ارتباط IS-A

**طراحی فقط با یک جدول :**

- در این حالت فقط از یک جدول با آرایه‌های بیتی استفاده می‌کنیم. هر بیت نشان دهنده یک زیرموجودیت است.
- در واقع برای نمایش هر نمونه موجودیت، بسته به اینکه در مجموعه نمونه‌های کدام زیرنوع باشد، بیت مربوطه‌اش را ۱ می‌کنیم.
- **شرط لازم :**

  باید تخصیص مجزا باشد، یعنی یک نمونه کارمند، جزئی از نمونه‌های حداکثر یک زیرموجودیت باشد.
- **مزیت این تکنیک :**

  برای به دست آوردن اطلاعات کامل زیرموجودیت نیازی به پیوند نیست.
- **عیب این تکنیک :**

  هیچ‌مقدار ( Null ) زیاد دارد و تعداد ستون‌های جدول زیاد است.

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

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 100 | x1 | y1 | a1 | b1 | ? | ? | ? | ? | مدیر |
| 200 | x2 | y2 | ? | ? | ? | l2 | m2 | n2 | مشاور |

# تکنیک چهارم طراحی ارتباط IS-A

**طراحی فقط با یک جدول :**

- در این حالت فقط از یک جدول با آرایه‌های بیتی استفاده می‌کنیم. هر بیت نشان دهنده یک زیرموجودیت است.
- در واقع برای نمایش هر نمونه موجودیت، بسته به اینکه در مجموعه نمونه‌های کدام زیرموجودیت باشد، بیت مربوطه‌اش را "یک" می‌کنیم.
- **شرط لازم :**

  وقتی تخصیص هم‌پوشا باشد ( سایر شرایط همانند تکنیک است ).

آرایه بیتی

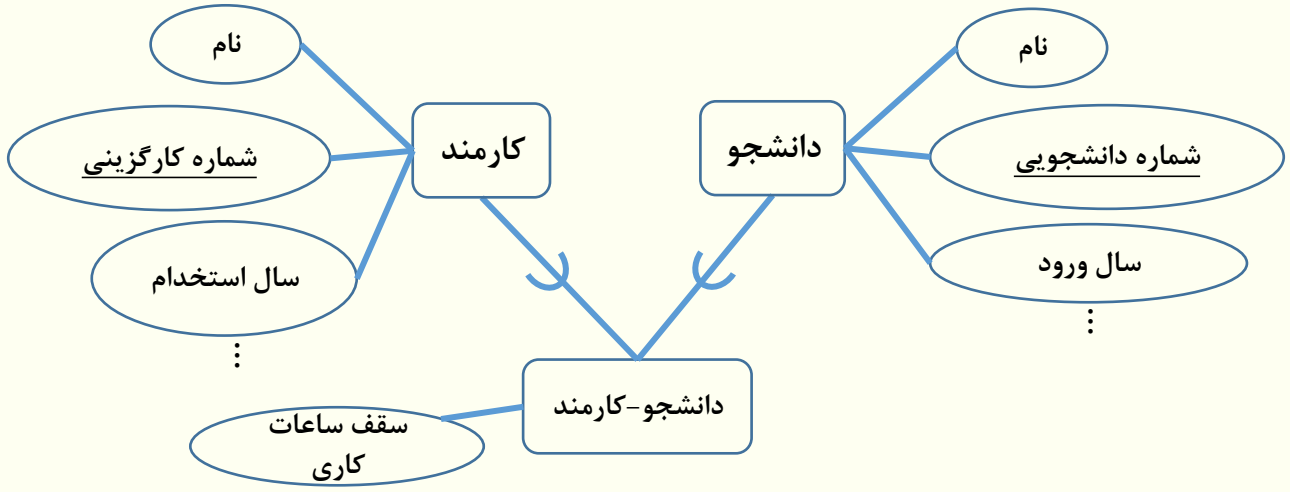
**E** (EID, X, Y, A, B, F, L, M, N, TB1, TB2, ...,TBn)

مدیر منشی مشاور

| | | | | | | |
|---|---|---|---|---|---|---|
| 100 | x1 | y1 | | 1 | 0 | 0 |
| 200 | x2 | y2 | | 0 | 1 | 0 |

# طراحی ارث بری چندکانه

- در این حالت، بین یک زیرموجودیت و چند موجودیت رابطه ارث بری وجود دارد .
- اگر زیر موجودیت مورد نظر از n موجودیت ارث بری داشته باشد، جدول نشان دهنده زیرموجودیت حداقل n کلید دارد. کلید با ارجاع بیشتر به عنوان کلید اصلی در نظر گرفته می‌شود.



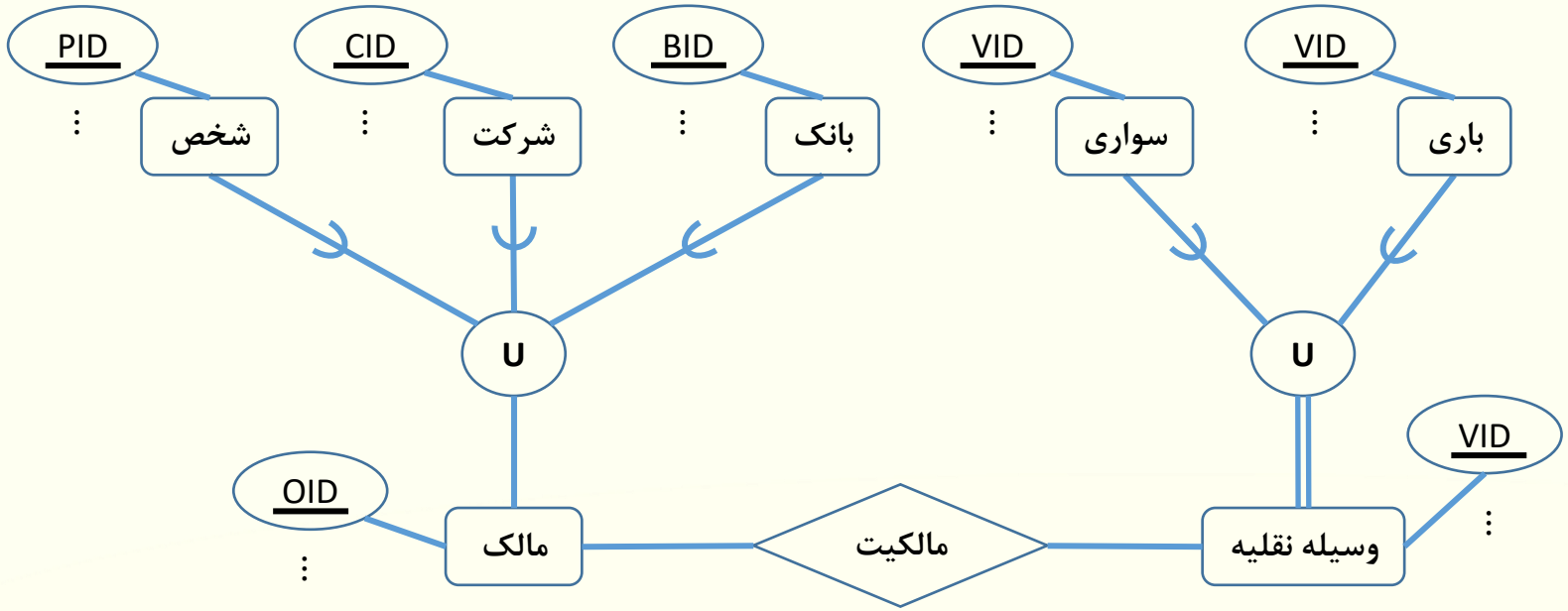

**STUD** (STID, STNAME, ...)

**EMPL** (EID, ENAME, ...)

**STEM** (STID, EID, MAXW)

# طراحی زیر نوع اجتماع ( U-type )

- فرض کنید موجودیت E، زیرموجودیت U-Type ( دسته یا Category ) n موجودیت است. در این حالت n+1 جدول طراحی می‌کنیم.
- اگر شناسه موجودیت‌ها از دامنه‌های متفاوت باشد، جدول زیرموجودیت، به جداول موجودیت‌های اصلی کلید خارجی می‌دهد که کلید اصلی آن جدول نیست.
- اگر شناسه موجودیت‌ها از یک دامنه باشد (و مقادیر شناسه در همه نمونه‌های زبرموجودیت‌ها یکتا باشد)، کلید جدول زیرموجودیت، همان کلید جداول موجودیت‌های اصلی است.




**PERS** (PID, …., OID)

**COMP** (CID, …., OID)

**BANK** (BID, …., OID)

**OWNER** (OID,….)

**VEHIC** (VID, ….)

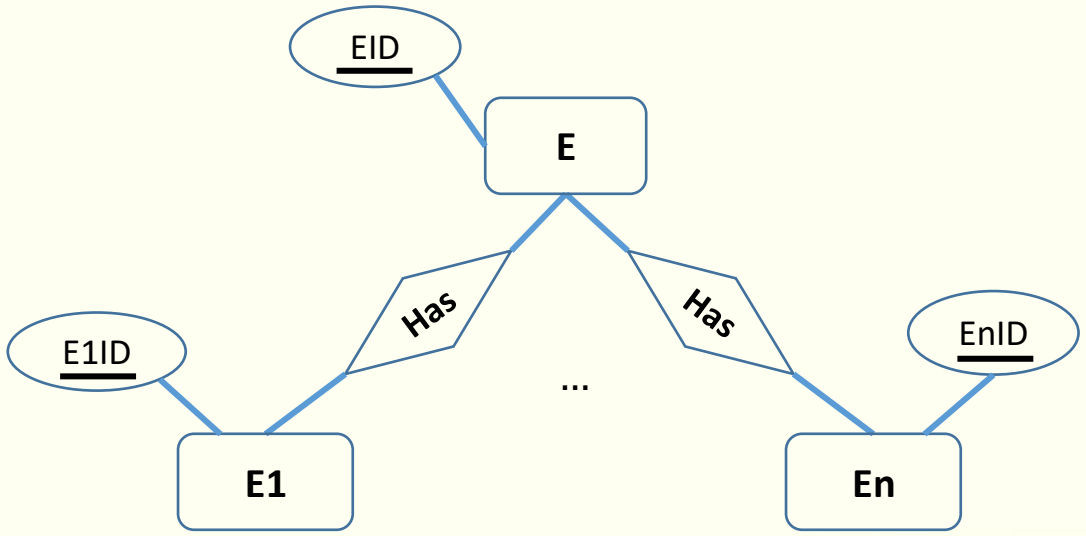
**OWNS** (OID, VID, F, T, ….)

**SAVARY** (VID, N, ….)

**BARY** (VID, T, ….)

# طراحی ارتباط IS-A-PART-OF

- اگر موجودیت کل، n موجودیت جزء داشته باشد، تعداد n+1 جدول طراحی می‌کنیم.
- توجه داشته باشید که موجودیت جزء از خود شناسه دارد.




**E** (EID, ....)

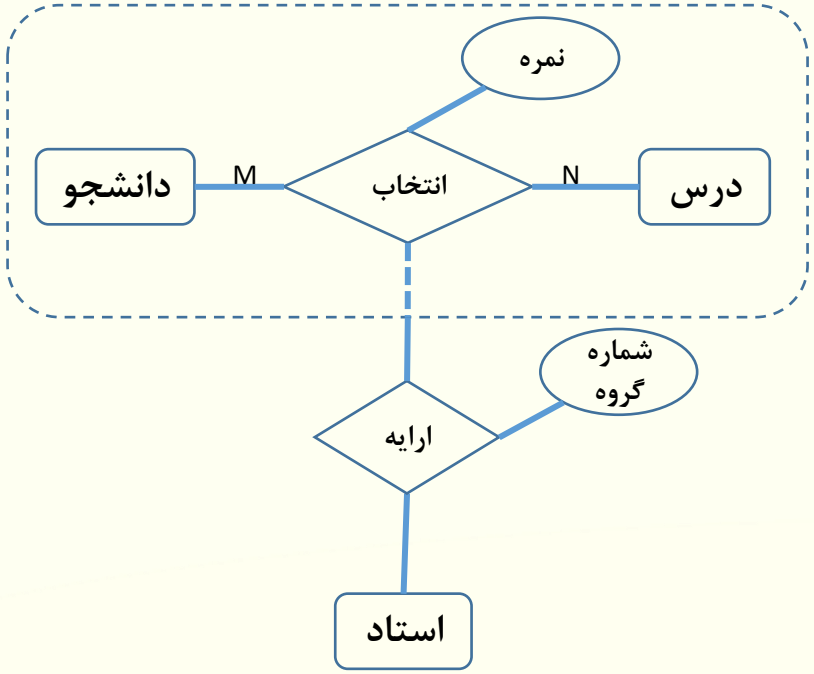
**E1** (E1ID, EID, ....)

....

**En** (EnID, EID, ....)

# طراحی تکنیک تجمیع ( Aggregation )

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




**STUD** (STID, ….)

**COUR** (COID, ….)

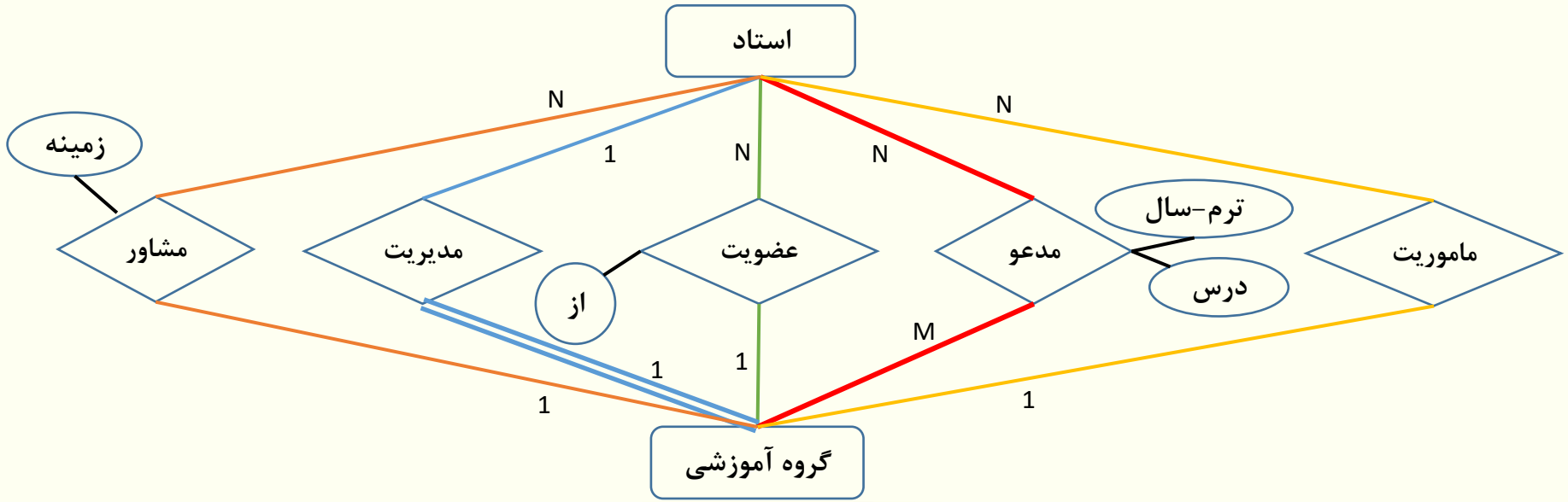
**SCR** (STID, COID, GR)

**PROF** (PRID, ….)

**OFFERING** (STID, COID, PROFID, GR#, CLASS)

# طراحی منطقی با وجود چند ارتباط

- در صورتی‌که چند ارتباط مثلاً بین دو 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 چند به چند، ، سپس یک به چند و در نهایت یک به یک را طراحی نماییم.




**DEPT** (DEID, …., DPHONE, PRID)

**PROF** (PRID, …., PRRANK, MDEID, SUB, MEMDEID, FROM, CDEID, INT)

**INVITED** (DEID, PRID, YR, TR)

# پایان فصل سوم


مهدی دادبخش

mahdi.dadbakhsh@sharif.edu


۱۴۰۲ – ۱۴۰۱