



# اصول طراحی پایگاه داده

By Dr. Taghinezhad

Mail:

[a0taghinezhad@gmail.com](mailto:a0taghinezhad@gmail.com)

SEVENTH EDITION

## Database System Concepts



Abraham Silberschatz  
Henry F. Korth  
S. Sudarshan

Mc  
Graw  
Hill  
Education



# Chapter 3: Introduction to SQL

**Database System Concepts, 7<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Outline

- مروری بر زبان پرس وجوی SQL
- تعریف داده در SQL (SQL Data Definition)
- ساختار پایه‌ای پرس وجوها در SQL
- عملیات پایه اضافی
- عملیات مجموعه‌ای
- مقادیر NULL
- توابع تجمعی (Aggregate Functions)
- پرس وجوهای تو در تو (Nested Subqueries)
- تغییر در پایگاه داده



## تاریخچه

- زبان IBM Sequel به عنوان بخشی از پروژه System R در آزمایشگاه تحقیقاتی IBM San Jose توسعه یافت.
- این زبان به Structured Query Language (SQL) تغییر نام داد.
- استاندارد ANSI و ISO برای SQL :
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL1999 : (نام زبان با Y2K سازگار شد.)
  - SQL:2003
- سیستم‌های تجاری بیشتر ویژگی‌های SQL-92 را ارائه می‌دهند، به علاوه مجموعه‌ای از ویژگی‌های متنوع از استانداردهای جدیدتر و قابلیت‌های اختصاصی خاص.
  - ممکن است همه مثال‌ها در سیستم شما کار نکنند.



# بخش‌های SQL

- **DML (Data Manipulation Language)**: توانایی پرس‌وجوی اطلاعات از پایگاه داده و درج، حذف، و اصلاح رکوردها در پایگاه داده را فراهم می‌کند.
- **یکپارچگی (integrity)**: زبان تعریف داده (DDL) شامل دستورات مشخص کردن محدودیت‌های یکپارچگی است.
- **تعریف نما (View Definition): DDL** شامل دستوراتی برای ایجاد و تعریف نماها (Views) می‌باشد.
- **کنترل تراکنش (Transaction Control)**: شامل دستوراتی برای مشخص کردن آغاز و پایان تراکنش‌ها است.
- **SQL توکار (Embedded SQL) و SQL پویا (Dynamic SQL)**: نحوه‌ی جاسازی دستورات SQL در زبان‌های برنامه‌نویسی عمومی را تعریف می‌کند.
- **احراز هویت و مجوزها (Authorization)**: شامل دستوراتی برای تعیین حقوق دسترسی کاربران به جداول و نماها می‌باشد.



# زبان تعریف داده (Data Definition Language)

زبان تعریف داده در (SQL Data-Definition Language - DDL) SQL امکان تعیین و مشخص سازی اطلاعات مربوط به روابط (Relations) را فراهم می سازد، از جمله:

- طرح (Schema) برای هر رابطه.
- نوع مقادیر (Data Type) مرتبط با هر ویژگی (Attribute).
- محدودیت های یکپارچگی (Integrity Constraints).
- مجموعه ای از شاخص ها (Indices) که باید برای هر رابطه نگهداری شوند.
- اطلاعات امنیتی و مجوزهای دسترسی (Security and Authorization Information) برای هر رابطه.
- ساختار فیزیکی ذخیره سازی (Physical Storage Structure) هر رابطه بر روی دیسک.





# انواع دامنه در SQL

- **char(n)**: رشته‌ی متنی با طول ثابت که طول آن توسط کاربر تعیین می‌شود (n).

- **varchar(n)**: رشته‌ی متنی با طول متغیر که دارای حداکثر طول n است (توسط کاربر مشخص می‌شود).

- **int**: عدد صحیح (Integer) زیرمجموعه‌ای محدود از اعداد صحیح که به سخت‌افزار (ماشین) وابسته است.

- **Smallint**: عدد صحیح کوچک (Small Integer) زیرمجموعه‌ای از نوع داده‌ی عدد صحیح که به ماشین وابسته است.

- **numeric(p,d)**: عدد اعشاری با دقت ثابت؛ دارای p رقم در کل و d رقم در سمت راست ممیز. (مثال: numeric(3,1) امکان ذخیره‌ی مقدار ۴۴.۵ را به صورت دقیق دارد، اما ۴۴۴.۵ یا ۰.۳۲ را نمی‌توان دقیق ذخیره کرد).

- **real, double precision**: اعداد با ممیز شناور (Floating Point) و ممیز شناور با دقت دوبرابر (Double Precision) که دقت آن‌ها به ماشین وابسته است.

- **float(n)**: عدد با ممیز شناور که دارای دقت حداقل n رقم است (n توسط کاربر تعیین می‌شود).

- انواع داده‌ی بیشتر در فصل ۴ بررسی می‌شوند.



# ساخت جدول

■ در SQL، یک رابطه (Relation) با استفاده از دستور **CREATE TABLE** تعریف می شود:

**create table *r***

$(A_1 D_1, A_2 D_2, \dots, A_n D_n,$   
 $(\text{integrity-constraint}_1),$   
 $\dots,$   
 $(\text{integrity-constraint}_k))$

- $r$  نام رابطه (جدول) است.
- هر  $A_i$  نام یک ویژگی در طرح رابطه  $r$  است.
- $D_i$  نوع داده (Data Type) مربوط به مقادیر موجود در دامنه ی ویژگی  $A_i$  است.

مثال: چگونه می توان جدول **Instructor** را در پایگاه داده ایجاد کرد؟

```
create table instructor(  
    ID           char(5),  
    name         varchar(20),  
    dept_name    varchar(20),  
    salary      numeric(8,2))
```





# محدودیت‌های یکپارچگی (Integrity Constraints) در دستور CREATE TABLE

■ انواع محدودیت‌های یکپارچگی:

• کلید اصلی:  $(A_1, \dots, A_n)$  primary key

• کلید خارجی:  $(A_m, \dots, A_n)$  references  $r$  foreign key

• غیر تهی

■ SQL از انجام هرگونه تغییر یا به‌روزرسانی که باعث نقض یک محدودیت یکپارچگی شود جلوگیری می‌کند.

■ مثال :

```
create table instructor (  
    ID          char(5),  
    name        varchar(20) not null,  
    dept_name    varchar(20),  
    salary       numeric(8,2),  
    primary key (ID),  
    foreign key (dept_name) references department);
```



## چند تعریف دیگر از ساخت جدول

- **create table** *student* (  
    *ID*                **varchar**(5),  
    *name*            **varchar**(20) not null,  
    *dept\_name*      **varchar**(20),  
    *tot\_cred*        **numeric**(3,0),  
    **primary key** (*ID*),  
    **foreign key** (*dept\_name*) **references** *department*);
- **create table** *takes* (  
    *ID*                **varchar**(5),  
    *course\_id*       **varchar**(8),  
    *sec\_id*           **varchar**(8),  
    *semester*        **varchar**(6),  
    *year*             **numeric**(4,0),  
    *grade*            **varchar**(2),  
    **primary key** (*ID*, *course\_id*, *sec\_id*, *semester*, *year*) ,  
    **foreign key** (*ID*) **references** *student*,  
    **foreign key** (*course\_id*, *sec\_id*, *semester*, *year*) **references**  
    *section*);



- **create table** *course* (  
    *course\_id*       **varchar**(8),  
    *title*           **varchar**(50),  
    *dept\_name*      **varchar**(20),  
    *credits*         **numeric**(2,0),  
    **primary key** (*course\_id*),  
    **foreign key** (*dept\_name*) **references** *department*);



# Updates to tables

## Insert (اضافه کردن)

- `insert into instructor values ('10211', 'Smith', 'Biology', 66000);`

## Delete (حذف)

- حذف تمامی رکوردها از رابطه student

- `delete from student`

## Drop Table (حذف جدول)

- `drop table r`

## Alter (تغییر دادن جدول)

- `alter table r add A D`

- که در آن A نام ویژگی جدید و D دامنه‌ی آن است.

- تمام تاپل‌های موجود در رابطه، مقدار null برای ویژگی جدید دریافت می‌کنند.

- `alter table r drop A`

- که در آن A نام ویژگی مورد نظر از رابطه‌ی r است.

- حذف ویژگی در بسیاری از پایگاه‌های داده پشتیبانی نمی‌شود.



## ساختار پایه‌ای پرس‌وجو

■ یک پرس‌وجوی معمولی در SQL به شکل زیر است:

```
select  $A_1, A_2, \dots, A_n$   
from  $r_1, r_2, \dots, r_m$   
where  $P$ 
```

- $A_i$  نماینده‌ی یک ویژگی (Attribute) است.
  - $r_i$  نماینده‌ی یک رابطه (Relation) است.
  - $P$  یک شرط منطقی (Predicate) می‌باشد.
- نتیجه‌ی یک پرس‌وجوی SQL همیشه یک رابطه است.



# عبارت select

- عبارت **SELECT** ویژگی‌هایی را که در نتیجه‌ی پرس‌وجو مورد نظر هستند مشخص می‌کند.
- این عبارت معادل عملیات نمایش در جبر رابطه‌ای است.
- مثال : یافتن نام تمام اساتید
- 

**select** *name*  
**from** *instructor*

- توجه: نام‌های SQL حساس به حروف کوچک و بزرگ نیستند، یعنی می‌توان از حروف بزرگ یا کوچک استفاده کرد.

*Name*  $\equiv$  **NAME**  $\equiv$  *name* •

- برخی افراد برای نمایش نام‌ها به سبک **Bold** از حروف بزرگ استفاده می‌کنند.



## عبارت **select** (ادامه)

■ SQL اجازه می‌دهد که مقادیر تکراری هم در روابط و هم در نتایج پرس‌وجو وجود داشته باشند.

■ برای حذف مقادیر تکراری، می‌توان از کلمه کلیدی **DISTINCT** پس از **SELECT** استفاده کرد.

■ مثال: یافتن نام دپارتمان تمام اساتید و حذف مقادیر تکراری

```
select distinct dept_name  
from instructor
```

■ کلمه کلیدی **ALL** مشخص می‌کند که مقادیر تکراری حذف نشوند:

```
select all dept_name  
from instructor
```

*dept\_name*

Comp. Sci.

Finance

Music

Physics

History

Physics

Comp. Sci.

History

Finance

Biology

Comp. Sci.

Elec. Eng.





## عبارت select (ادامه)

- علامت ستاره (\*) در عبارت SELECT به معنی «تمام ویژگی‌ها» است:

```
select *  
from instructor
```

- یک ویژگی می‌تواند یک مقدار ثابت (Literal) باشد و نیازی به FROM نداشته باشد:

```
select '437'
```

- نتیجه: یک جدول با یک ستون و یک سطر که مقدار آن "437" است.
- می‌توان نام ستون را با استفاده از AS مشخص کرد:

```
select '437' as FOO
```

- یک ویژگی می‌تواند یک مقدار ثابت باشد و همراه با FROM استفاده شود:

```
select 'A'  
from instructor
```

- نتیجه: یک جدول با یک ستون و N سطر (تعداد تاپل‌ها در جدول (instructors)، هر سطر دارای مقدار "A" است.



## عبارت **select** (ادامه)

- عبارت **SELECT** می تواند شامل عملیات حسابی مانند جمع (+)، تفریق (-)، ضرب (\*) و تقسیم (/) باشد و بر روی مقادیر ثابت یا ویژگی های تاپل ها اعمال شود.
- سوال: چگونه می توان نتیجه ای مشابه جدول **instructor** داشته باشیم، اما مقدار ویژگی **salary** تقسیم بر ۱۲ شود.



## عبارت **select** (ادامه)

• پرس و جو

```
select ID, name, salary/12  
from instructor
```

این پرس و جو جدولی مشابه جدول *instructor* باز می گرداند، با این تفاوت که مقدار ویژگی *salary* بر ۱۲ تقسیم شده است.

• می توان مقدار *salary / 12* را با استفاده از عبارت **AS** نام گذاری کرد:

```
select ID, name, salary/12 as monthly_salary
```



# عبارت where

- عبارت **WHERE** شرایطی را مشخص می کند که نتیجه ی پرس وجو باید آن ها را برآورده کند.
- این عبارت معادل گزاره ی انتخاب (Selection) در جبر رابطه ای است.
- برای یافتن تمام اساتید دپارتمان علوم کامپیوتر:

```
select name  
from instructor  
where dept_name = Comp. Sci.
```

- در SQL می توان از عملگرهای منطقی (and, or, not) استفاده کرد.
- عملوندهای این عملگرها می توانند شامل **عبارات مقایسه ای** باشند:  
 $<$  ،  $<=$  ،  $>$  ،  $>=$  ،  $=$  ،  $<>$

- این مقایسه ها را می توان بر روی نتیجه ی عبارات حسابی نیز اعمال کرد.
- برای یافتن تمام اساتید دپارتمان علوم کامپیوتر با حقوق بیش از ۷۰۰۰۰:

```
select name  
from instructor  
where dept_name = 'Comp. Sci.' and salary > 70000
```

name
Katz
Brandt



## عبارت from

- عبارت **FROM** فهرست رابطه‌هایی را مشخص می‌کند که در پرس‌وجو مورد استفاده قرار می‌گیرند.
- این عبارت معادل عمل ضرب دکارتی (Cartesian Product) در جبر رابطه‌ای است.
- یافتن ضرب دکارتی بین جدول‌های `instructor` و `teaches`:

```
select *  
from instructor, teaches
```

- این پرس‌وجو تمام جفت‌های ممکن `instructor-teaches` را تولید می‌کند و شامل تمام ویژگی‌ها از هر دو رابطه است.
- اگر ویژگی‌های مشترکی مانند `ID` وجود داشته باشند، در نتیجه‌ی پرس‌وجو نام آن‌ها به شکل `instructor.ID` و `teaches.ID` نمایش داده می‌شود.
- عمل ضرب دکارتی به‌تنهایی معمولاً کاربرد چندانی ندارد، اما در ترکیب با عبارت **WHERE** بسیار مفید است و معادل عمل انتخاب (Selection) در جبر رابطه‌ای می‌باشد.



## مثال‌ها

<i>name</i>	<i>course_id</i>
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

■ برای یافتن نام تمام اساتیدی که حداقل یک درس را تدریس کرده‌اند، همراه با شناسه‌ی آن درس:

○ ***select name, course\_id  
from instructor, teaches  
where instructor.ID = teaches.ID***

■ یافتن نام اساتید دپارتمان هنر که درسی را تدریس کرده‌اند همراه شناسه‌ی درس

○ ***select name, course\_id  
from instructor, teaches  
where instructor.ID = teaches.ID  
and instructor.dept\_name = 'Art'***



## عملیات تغییر نام

- در SQL می‌توان رابطه‌ها و ویژگی‌ها را با استفاده از عبارت **AS** تغییر نام داد:

- *old-name as new-name*

- یافتن نام تمام اساتیدی که حقوق بیشتری از برخی اساتید دپارتمان علوم کامپیوتر دارند:

- **select distinct** *T.name*  
**from** *instructor as T, instructor as S*  
**where** *T.salary > S.salary and S.dept\_name = 'Comp. Sci.'*

- کلمه‌ی کلیدی **AS** اختیاری است و می‌توان آن را حذف کرد:

- *instructor as T*  $\equiv$  *instructor T*





## مثال Self Join

■ رابطه‌ی emp-super

<i>person</i>	<i>supervisor</i>
Bob	Alice
Mary	Susan
Alice	David
David	Mary

■ یافتن سرپرست "Bob"

■ یافتن سرپرستِ سرپرستِ "Bob"

■ آیا می‌توان تمام سرپرستان مستقیم و غیرمستقیمِ "Bob" را یافت؟



# عملیات رشته‌ای (String)

- SQL شامل یک عملگر تطبیق رشته‌ای برای مقایسه‌ی رشته‌های متنی است. عملگر **like** از الگوهایی با دو کاراکتر ویژه استفاده می‌کند:
- درصد ( % ) : با هر زیررشته‌ای مطابقت دارد.
- ( \_ ) : با هر کاراکتری مطابقت دارد.
- یافتن نام تمام اساتیدی که زیررشته‌ی "dar" در نام آن‌ها وجود دارد:

```
select name  
from instructor  
where name like '%dar%'
```

تطبیق رشته‌ی "%۱۰۰"

```
like '100 \%' escape '\'
```

در مثال بالا، از نویسه‌ی \ به عنوان کاراکتر گریز استفاده شده است.



## عملیات رشته‌ای (String) (ادامه)

- الگوها در SQL به حروف بزرگ و کوچک حساس هستند.
- نمونه‌هایی از تطبیق الگوها:
  - 'Intro%': هر رشته‌ای که با Intro آغاز شود..
  - '%Comp%': هر رشته‌ای که شامل Comp به‌عنوان زیررشته باشد.
  - '\_\_\_': هر رشته‌ای با دقیقاً سه کاراکتر.
  - '\_\_\_%': هر رشته‌ای با حداقل سه کاراکتر.
- SQL از مجموعه‌ای از عملیات رشته‌ای پشتیبانی می‌کند، مانند:
  - اتصال رشته‌ها ("||")
  - تبدیل حروف بزرگ به کوچک و برعکس
  - یافتن طول رشته، استخراج زیررشته‌ها و سایر توابع متنی.



## مرتب‌سازی نمایش تاپل‌ها

- فهرست کردن نام تمام اساتید به ترتیب حروف الفبا:

```
select distinct name  
from   instructor  
order by name
```

- می‌توان برای هر ویژگی ترتیب نزولی (**desc**) یا ترتیب صعودی (**asc**) مشخص کرد؛ به صورت پیش فرض، ترتیب صعودی است.

○ مثال: **order by *name desc***

- می‌توان بر اساس چند ویژگی مرتب کرد:

○ مثال: **order by *dept\_name, name***



## عبارت WHERE و عملگرهای مقایسه‌ای

■ SQL شامل عملگر مقایسه‌ای **between** است.

■ مثال: یافتن نام تمام اساتیدی که حقوق آنها بین ۹۰,۰۰۰ تا ۱۰۰,۰۰۰ دلار است (یعنی  $90,000 \leq$  و  $100,000 \geq$ ):

- **select name**  
**from instructor**  
**where salary between 90000 and 100000**

■ مقایسه‌ی تاپل‌ها

- **select name, course\_id**  
**from instructor, teaches**  
**where (instructor.ID, dept\_name) = (teaches.ID,**  
**'Biology');**



## عملیات مجموعه‌ای

■ یافتن درس‌هایی که در پاییز ۲۰۱۷ یا بهار ۲۰۱۸ برگزار شده‌اند:

```
(select course_id from section where sem = 'Fall' and year = 2017)  
union  
(select course_id from section where sem = 'Spring' and year = 2018)
```

■ یافتن درس‌هایی که هم در پاییز ۲۰۱۷ و هم در بهار ۲۰۱۸ برگزار شده‌اند:

```
(select course_id from section where sem = 'Fall' and year = 2017)  
intersect  
(select course_id from section where sem = 'Spring' and year = 2018)
```

یافتن درس‌هایی که در پاییز ۲۰۱۷ برگزار شده‌اند اما در بهار ۲۰۱۸ برگزار نشده‌اند:

```
(select course_id from section where sem = 'Fall' and year = 2017)  
except  
(select course_id from section where sem = 'Spring' and year = 2018)
```



## عملیات مجموعه‌ای (ادامه)

- عملیات مجموعه‌ای: UNION، INTERSECT و EXCEPT
  - هر یک از عملیات بالا به صورت پیش فرض مقادیر تکراری را حذف می‌کنند.
- برای حفظ تمام مقادیر تکراری می‌توان از نسخه‌های ALL استفاده کرد:
  - **union all**
  - **intersect all**
  - **except all**





# مقادیر NULL

- ممکن است برخی تاپل‌ها دارای مقدار **null** برای برخی ویژگی‌هایشان باشند.
- **null** نشان‌دهنده‌ی مقدار نامعلوم یا عدم وجود مقدار است.
- نتیجه‌ی هر عبارت حسابی که شامل **null** باشد نیز **null** خواهد بود.
  - مثال : **5 + null returns null**
- برای بررسی مقادیر **null** می‌توان از **is null** استفاده کرد.
  - مثال: یافتن تمام اساتیدی که حقوقشان برابر **null** است.

```
select name  
from instructor  
where salary is null
```

- عبارت **is not null** زمانی موفق است که مقدار مورد بررسی **null** نباشد.



## مقادیر NULL (ادامه)

- SQL نتیجه‌ی هر مقایسه‌ای که شامل null باشد را به عنوان unknown در نظر می‌گیرد، به جز دو عبارت `is null` و `is not null`.

■ مثال: `5 < null` or `null <> null` or `null = null`

- در عبارت WHERE می‌توان از عملیات منطقی AND، OR، NOT استفاده کرد؛ بنابراین تعاریف این عملیات برای مقدار unknown به صورت زیر گسترش می‌یابد:

**and :**  $(true \text{ and } unknown) = unknown,$   
 $(false \text{ and } unknown) = false,$   
 $(unknown \text{ and } unknown) = unknown$   
**or:**  $(unknown \text{ or } true) = true,$   
 $(unknown \text{ or } false) = unknown$   
 $(unknown \text{ or } unknown) = unknown$

- اگر نتیجه‌ی پیش شرط WHERE برابر unknown شود، SQL آن را false در نظر می‌گیرد و ردیف انتخاب نمی‌شود.



# Aggregate Functions (توابع تجمیعی)

■ این توابع روی مجموعه‌ای از مقادیر یک ستون در یک رابطه عمل می‌کنند و یک مقدار بازمی‌گردانند.

**avg:** مقدار متوسط

**min:** حداقل مقدار

**max:** حداکثر مقدار

**sum:** مجموع مقادیر

**count:** تعداد مقادیر



## مثال توابع تجمیعی

- میانگین حقوق اساتید دپارتمان علوم کامپیوتر:

```
select avg (salary)  
from instructor where dept_name= 'Comp. Sci.';
```

- تعداد کل اساتیدی که در ترم بهار ۲۰۱۸ درسی را تدریس کرده‌اند:

```
select count (distinct ID)  
from teaches  
where semester = 'Spring' and year = 2018;
```

- تعداد تاپل‌ها در جدول **course**:

```
select count (*)  
from course;
```



# توابع تجمیعی – Group by

■ میانگین حقوق اساتید در هر دپارتمان

```
select dept_name, avg (salary) as avg_salary  
from instructor  
group by dept_name;
```

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

<i>dept_name</i>	<i>avg_salary</i>
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000



## توابع تجمیعی (ادامه)

- ویژگی‌هایی که در عبارت **SELECT** خارج از توابع تجمیعی استفاده می‌شوند، باید در فهرست **GROUP BY** قرار گیرند.

*/\* پرس‌وجوی نادرست \*/*

```
select dept_name, ID, avg (salary)
from instructor
group by dept_name;
```



# مثال

■ مشخص کنید چند تهیه کننده P2 را تهیه کرده اند؟

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200





## مثال

■ مشخص کنید چند تهیه کننده P2 را تهیه کرده اند؟

■ `Select count(*) from SP where p#='P2'`

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200



## مثال

- کل مقدار تهیه شده از هر قطعه را در جدول جواب بدهد همراه با شماره قطعه

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200



## مثال

- کل مقدار تهیه شده از هر قطعه را در جدول جواب بدهد همراه با شماره قطعه

■ **Select P#, SUM(Qty) from SP Group By P#**

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200



## مثال

- برای هر قطعه تهیه شده، شماره قطعه، کل تعداد و ماکزیمم تعداد تهیه شده از آن را بدون در نظر گرفتن S1 بدهد

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200



## مثال

■ برای هر قطعه تهیه شده، شماره قطعه، کل تعداد و ماکزیمم تعداد تهیه شده از آن را بدون در نظر گرفتن S1 بدهد

- Select P#, SUM(Qty),MAX(Qty)
- From SP
- Where S# != 'S1'
- Group By P#

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200



## توابع تجمیعی - عبارت Having

- یافتن نام و میانگین حقوق دپارتمان‌هایی که میانگین حقوق آن‌ها بیش از ۴۲۰۰۰ است:

```
select dept_name, avg (salary) as avg_salary  
from instructor  
group by dept_name  
having avg (salary) > 42000;
```

- نکته: شرط‌های موجود در **HAVING** پس از تشکیل گروه‌ها اعمال می‌شوند، در حالی که شرط‌های **WHERE** پیش از تشکیل گروه‌ها اعمال می‌گردند.



# زیرپرس وجوهای تو در تو (Nested Subqueries)

- SQL روشی برای تو در تو قرار دادن پرس وجوها (Subqueries) فراهم می کند. زیرپرس وجو عبارت **SELECT-FROM-WHERE** است که درون یک پرس وجوی دیگر قرار می گیرد.
- شکل کلی پرس وجوی اصلی:

```
select  $A_1, A_2, \dots, A_n$   
from  $r_1, r_2, \dots, r_m$   
where  $P$ 
```

پرس وجو را می توان در جاهای زیر استفاده کرد:

- عبارت **From**: هر  $r_i$  میتواند با یک زیرپرس وجوی معتبر جایگزین شود.
- عبارت **Where**: عبارت  $P$  می تواند به شکل زیر باشد:

$B <\text{operation}> (\text{subquery})$

که در آن  $B$  یک ویژگی و  $<\text{operation}>$  عملگری است که بعداً تعریف می شود.

- عبارت **Select**:

هر  $A_i$  می تواند یک زیرپرس وجو باشد که یک مقدار منفرد (single value) تولید می کند.



## عضویت در مجموعه‌ها

```
select id  
from  $r_1$   
where id in ( $r_2$ );
```





## عضویت در مجموعه‌ها

■ یافتن درس‌هایی که هم در ترم پاییز ۲۰۱۷ و هم در ترم بهار ۲۰۱۸ ارائه شده‌اند:

```
select distinct course_id
from section
where semester = 'Fall' and year = 2017 and
       course_id in (select course_id
                       from section
                       where semester = 'Spring' and year = 2018);
```

■ یافتن درس‌هایی که در ترم پاییز ۲۰۱۷ ارائه شده‌اند اما در بهار ۲۰۱۸ ارائه نشده‌اند:

```
select distinct course_id
from section
where semester = 'Fall' and year = 2017 and
       course_id not in (select course_id
                           from section
                           where semester = 'Spring' and year = 2018);
```



## عضویت در مجموعه‌ها (ادامه)

- یافتن نام تمام اساتیدی که نام آن‌ها نه "Mozart" است و نه "Einstein":

```
select distinct name from instructor  
where name not in ('Mozart', 'Einstein')
```

- یافتن تعداد کل (و متمایز) دانشجویانی که درس‌هایی را گذرانده‌اند که توسط استادی با شناسه ۱۰۱۰۱ تدریس شده‌اند:

```
select count (distinct ID)  
from takes  
where (course_id, sec_id, semester, year) in  
      (select course_id, sec_id, semester, year  
       from teaches  
       where teaches.ID= 10101);
```

- نکته: پرس‌وجوی بالا را می‌توان به شکل ساده‌تری نوشت؛ این نسخه صرفاً برای نشان دادن ویژگی‌های مختلف SQL ارائه شده است.



## مقایسه مجموعه‌ای



## مقایسه مجموعه‌ای - عبارت "some"

- یافتن نام اساتیدی که حقوق آنها از حقوق حداقل یکی از اساتید دپارتمان زیست‌شناسی بیشتر است:

```
select distinct T.name  
from instructor as T, instructor as S  
where T.salary > S.salary and S.dept name = 'Biology';
```

- عبارت معادل با استفاده از < some

```
select name from instructor  
where salary > some (select salary  
from instructor where dept name = 'Biology');
```



## معنی عبارت “some”

- $F <\text{comp}> \text{some } r \Leftrightarrow \exists t \in r \text{ such that } (F <\text{comp}> t)$

Where  $<\text{comp}>$  can be:  $<$ ,  $\leq$ ,  $>$ ,  $=$ ,  $\neq$

$$(5 < \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{true}$$

(read: 5 < some tuple in the relation)

$$(5 < \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{false}$$

$$(5 = \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true}$$

$$(5 \neq \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true (since } 0 \neq 5)$$

$(= \text{some}) \equiv \text{in}$

However,  $(\neq \text{some}) \not\equiv \text{not in}$



## مقایسه مجموعه‌ای - عبارت "all"

- یافتن نام تمام اساتیدی که حقوق آنها از حقوق تمام اساتید دپارتمان زیست‌شناسی بیشتر است:

```
select name  
from instructor  
where salary > all (select salary  
                        from instructor  
                        where dept name = 'Biology');
```



## معنی عبارت “all”

- $F <\text{comp}> \mathbf{all} \ r \Leftrightarrow \forall t \in r (F <\text{comp}> t)$

$$(5 < \mathbf{all} \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{false}$$

$$(5 < \mathbf{all} \begin{array}{|c|} \hline 6 \\ \hline 10 \\ \hline \end{array}) = \text{true}$$

$$(5 = \mathbf{all} \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline \end{array}) = \text{false}$$

$$(5 \neq \mathbf{all} \begin{array}{|c|} \hline 4 \\ \hline 6 \\ \hline \end{array}) = \text{true (since } 5 \neq 4 \text{ and } 5 \neq 6)$$

$(\neq \mathbf{all}) \equiv \mathbf{not\ in}$

However,  $(= \mathbf{all}) \not\equiv \mathbf{in}$



## آزمون تهی بودن روابط

■ عبارت **EXISTS** در SQL زمانی مقدار **TRUE** بازمی گرداند که زیرپرسوجوی درون آن تهی نباشد.

■  $\text{exists } r \Leftrightarrow r \neq \emptyset$

■  $\text{not exists } r \Leftrightarrow r = \emptyset$





## کاربرد عبارت “exists”

```
select *  
from مستعار as نام جدول  
where [ شرط and ]  
exists (Select Op);
```

- **Correlation name**: متغیری مانند **S** در پرس وجوی بیرونی است که برای ارجاع به تاپل های آن در زیرپرس وجو استفاده می شود.
- **Correlated subquery**: زیرپرس وجویی است که به مقادیر موجود در پرس وجوی بیرونی وابسته است و برای هر تاپل از پرس وجوی بیرونی اجرا می شود.



## کاربرد عبارت “exists”

- روش دیگر برای نوشتن پرس و جوی “یافتن تمام درس‌هایی که هم در ترم پاییز ۲۰۰۹ و هم در ترم بهار ۲۰۱۰ تدریس شده‌اند”:

Q?



## کاربرد عبارت “not exists”

- یافتن تمام دانشجویانی که تمام درس‌های ارائه‌شده در دپارتمان زیست‌شناسی را گذرانده‌اند:

Q?

- زیرپرس‌وجوی اول: همه‌ی درس‌های ارائه‌شده در دپارتمان Biology را فهرست می‌کند.
- زیرپرس‌وجوی دوم: درس‌هایی را که هر دانشجو گذرانده است فهرست می‌کند.

توجه:  $X - Y = \emptyset \Leftrightarrow X \subseteq Y$  □

این نوع پرس‌وجو را نمی‌توان با  $ALL =$  یا حالت‌های مشابه آن نوشت. □



# تست نبود تاپل‌های تکراری

- عبارت **UNIQUE** در SQL بررسی می‌کند که آیا زیرپرس‌وجو دارای هیچ تاپل تکراری نیست.
- اگر نتیجه‌ی زیرپرس‌وجو فاقد مقادیر تکراری باشد، مقدار آن **TRUE** می‌شود.

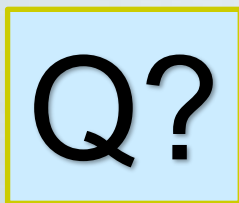
**select \***

**from** مستعار **as** نام جدول

**where** [ شرط **and** ]

**where unique (Select Operation)**

- یافتن تمام درس‌هایی که در سال ۲۰۰۹ حداکثر یک‌بار ارائه شده‌اند.





## زیر پرس وجوها در عبارت From



# زیرپرس وجوها در عبارت From

- SQL این امکان را می دهد که یک زیرپرس وجو در بخش FROM قرار گیرد و مانند یک جدول موقت از آن استفاده شود.

- یافتن میانگین حقوق اساتید آن دپارتمان هایی که میانگین حقوق شان بیش از ۴۲۰۰۰ دلار است:

```
select dept_name, avg_salary
from ( select dept_name, avg (salary) as avg_salary
      from instructor
      group by dept_name)
where avg_salary > 42000;
```

- در اینجا از HAVING استفاده نشده است، چون فیلتر میانگین ها مستقیماً روی نتیجه ی زیرپرس وجوی داخل FROM اعمال می شود.

- روش دیگر برای نوشتن همین پرس وجو

```
select dept_name, avg_salary
from ( select dept_name, avg (salary)
      from instructor
      group by dept_name)
      as dept_avg (dept_name, avg_salary)
where avg_salary > 42000;
```



## عبارت With

- عبارت WITH روشی برای تعریف یک رابطه‌ی موقت (temporary relation) است که تنها در همان پرس‌وجو قابل استفاده است.
- یافتن تمام دپارتمان‌هایی که دارای بیشترین بودجه هستند:

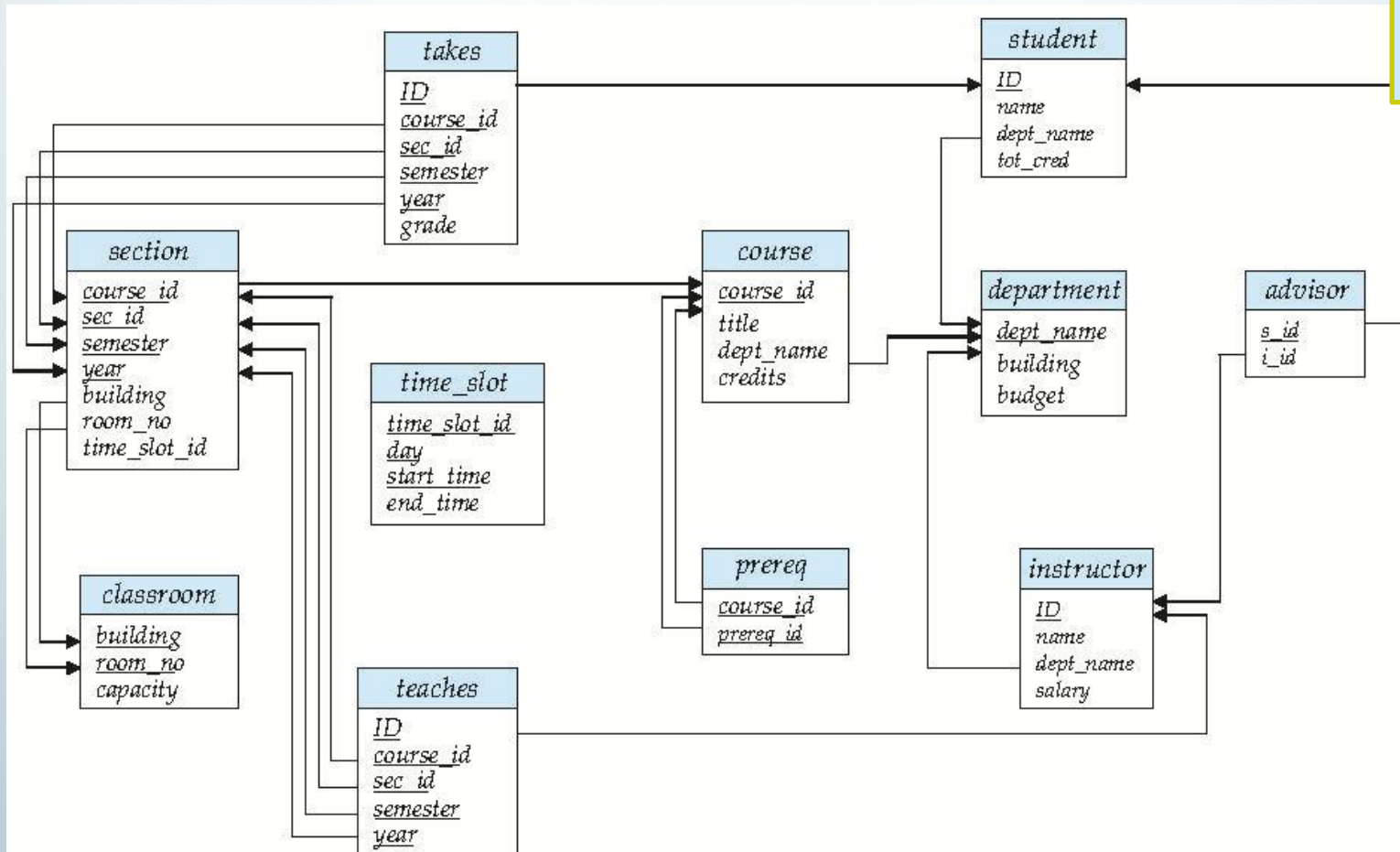
```
with max_budget (value) as  
    (select max(budget)  
     from department)  
select department.name  
from department, max_budget  
where department.budget =  
max_budget.value;
```



# پرس و جوهای پیچیده با استفاده از With

- یافتن تمام دپارتمان‌هایی که مجموع حقوق اساتید آن‌ها از میانگین مجموع حقوق همه‌ی دپارتمان‌ها بیشتر است.

Q?



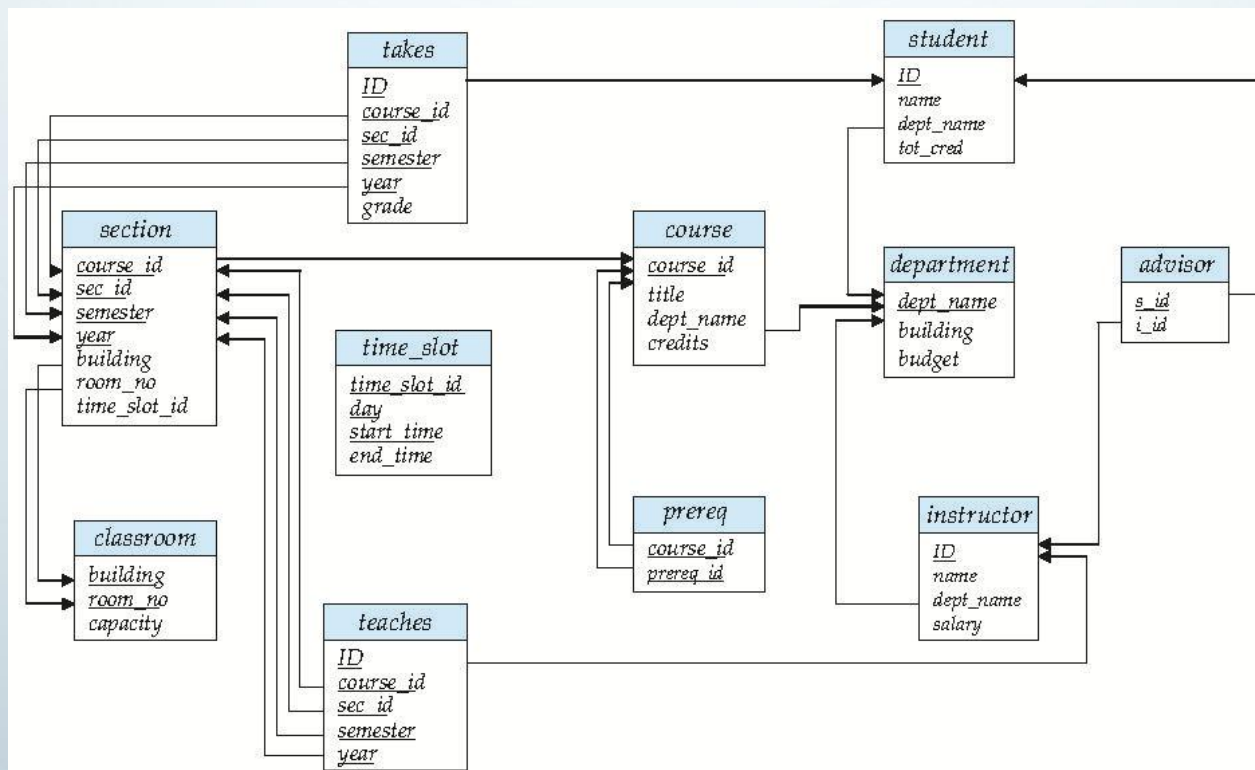




# زیر پرس وجوی عددی

■ زیر پرس وجوی عددی یا **Scalar Subquery** ، زیر پرس وجویی است که تنها یک مقدار (تک مقدار) را برمی گرداند.

■ یافتن تمام دپارتمان ها همراه با تعداد اساتید هر دپارتمان.



Q?



## تغییرات در پایگاه داده

- حذف رکوردها از یک رابطه‌ی مشخص.
- افزودن رکوردهای جدید به یک رابطه.
- تغییر مقادیر در رکوردهای موجود.



# حذف (Deletion)

■ حذف تمام اساتید:

```
delete from instructor
```

■ حذف تمام اساتید دپارتمان Finance:

```
delete from instructor  
where dept_name = 'Finance';
```

■ حذف تمام تاپل‌های جدول *instructor* مربوط به اساتیدی که در دپارتمانی واقع در ساختمان Watson هستند:

```
delete from instructor  
      where dept name in (select dept name  
                           from department  
                           where building = 'Watson');
```



## حذف (Deletion) (ادامه)

حذف اساتیدی با حقوق کمتر از میانگین:

```
delete from instructor
where salary < (select avg (salary)
                from instructor);
```

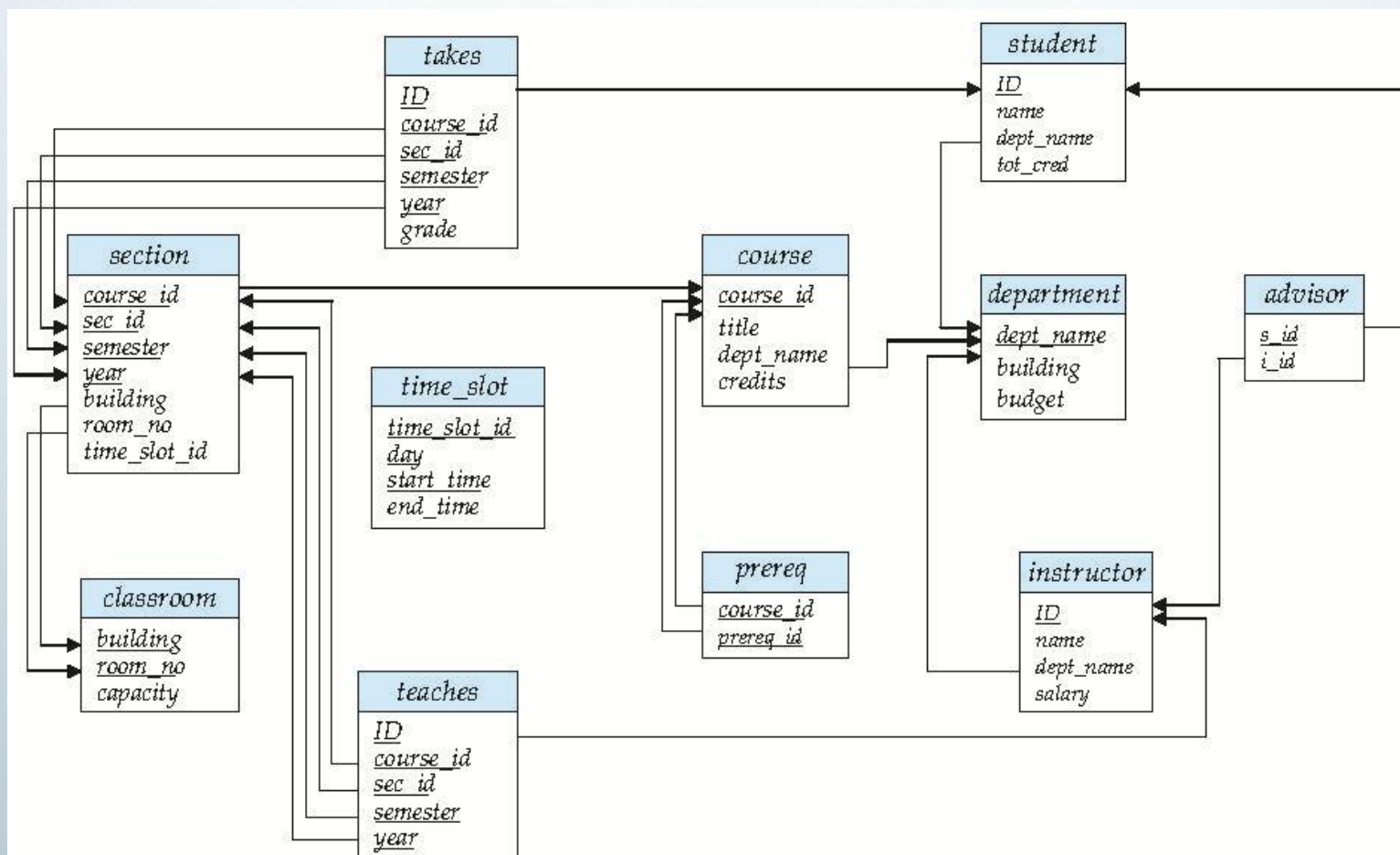
- مشکل: هنگام حذف رکوردها از جدول instructor، میانگین حقوق تغییر می کند.
  - راه حل های موجود در SQL:
1. ابتدا مقدار `avg(salary)` محاسبه شده و تمام تاپل هایی که باید حذف شوند مشخص می شوند.
  2. سپس همه ی تاپل های مشخص شده حذف می شوند، بدون آنکه میانگین دوباره محاسبه یا رکوردها مجدداً بررسی شوند.



# حذف (Deletion)

Q?

- حذف تمام تاپل‌های جدول instructor مربوط به اساتیدی که در دیپارتمانی واقع در ساختمان Watson هستند.





## درج (Insertion)

- افزودن یک تاپل جدید به جدول `course`:

```
insert into course  
values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- یا به صورت معادل با ذکر ستون‌ها:

```
insert into course (course_id, title, dept_name, credits)  
values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- افزودن یک تاپل جدید به جدول `student` با مقدار `tot_creds` برابر `null`:

```
insert into student  
values ('3003', 'Green', 'Finance', null);
```



## درج (Insertion) (ادامه)

- تبدیل هر دانشجوی دپارتمان Music که بیش از ۱۴۴ واحد گذرانده است به استاد در همان دپارتمان با حقوق ۱۸,۰۰۰ دلار:

**insert into *instructor***

```
select ID, name, dept_name, 18000  
from student  
where dept_name = 'Music' and total_cred > 144;
```

- عبارت **SELECT ... FROM ... WHERE** قبل از درج نتایج به طور کامل ارزیابی می شود. در غیر این صورت، پرس و جوهایمانند:

```
insert into table1 select * from table1
```

می توانند باعث مشکل شوند.



## به روز رسانی ها (Updates)

- افزایش ۵٪ حقوق تمام اساتید:

```
update instructor  
  set salary = salary * 1.05
```

- افزایش ۵٪ حقوق اساتیدی که کمتر از ۷۰۰۰۰ دلار حقوق می گیرند:

```
update instructor  
  set salary = salary * 1.05  
 where salary < 70000;
```

- افزایش ۵٪ حقوق اساتیدی که حقوقشان کمتر از میانگین است:

```
update instructor  
  set salary = salary * 1.05  
 where salary < (select avg (salary)  
                  from instructor);
```





## به روز رسانی ها (Updates) (ادامه)

- افزایش ۳٪ حقوق اساتیدی که بیش از ۱۰۰,۰۰۰ دلار حقوق می گیرند و افزایش ۵٪ درصدی بقیه اساتید:

- دو جمله **update** بنویسید:

**update instructor**

**set salary = salary \* 1.03**

**where salary > 100000;**

**update instructor**

**set salary = salary \* 1.05**

**where salary <= 100000;**

- نکته: ترتیب اجرای دو دستور مهم است.
- روش بهتر با استفاده از **CASE** در یک دستور واحد قابل انجام است.



# عبارت CASE برای به روزرسانی شرطی

■ همان پرس و جوی قبلی را می توان با استفاده از CASE نوشت تا در یک دستور واحد اجرا شود.

**update** *instructor*

**set** *salary* = **case**

**when** *salary* <= 100000 **then** *salary* \* 1.05

**else** *salary* \* 1.03

**end**



## به روزرسانی با زیر پرس وجوی عددی

■ بازمحاسبه و به روزرسانی مقدار tot\_creds برای همه دانشجویان:

- **update student S**  
**set tot\_cred = (select sum(credits)**  
**from takes, course**  
**where takes.course\_id = course.course\_id and**  
**S.ID= takes.ID.and**  
**takes.grade <> 'F' and**  
**takes.grade is not null);**

■ مقدار tot\_creds را برای دانشجویانی که هیچ درسی نگذرانند برابر null قرار می دهد.

- **update student S**  
**set tot\_cred = (SELECT CASE**  
**WHEN SUM(course.credits) IS NOT NULL**  
**THEN SUM(course.credits)**  
**ELSE 0 END**  
**from takes, course where takes.course\_id = course.course\_id and**  
**S.ID= takes.ID.and**  
**takes.grade <> 'F' and**  
**takes.grade is not null);**



## End of Chapter 3