مقدمه ای بر SQL

قمرناز تدین

- مقدمه
- تعریف داده
- ساختار اصلی پرس و جو
- عملگرهای مجموعه
- مقادیر Null
- توابع تجمیع (Aggregate)
- زیرپرس و جوهای تودرتو
- تغییرات پایگاه داده

تاریخچه

- دهه 70: زبان Sequel شرکت IBM به عنوان بخشی از پروژه System R در آزمایشگاه تحقیقاتی IBM در سن خوزه ارائه شد.
- تغییر نام به Structured Query Language (SQL)
- ANSI and ISO standard SQL:
  - SQL-86, SQL-89, SQL-92
  - SQL:1999, SQL:2003, SQL:2008 , SQL:2011
- در سیستمهای تجاری بیشتر از ویژگیهای SQL-92 به اضافه برخی خواص استانداردهای بعدی استفاده میشود.

Data Definition Language

data-definition language (DDL) SQL تعیین اطلاعات درباره روابط را امکان پذیر می سازد، که شامل موارد زیر است:

- شمای هر رابطه
- دامنه مقادیر مربوط به هر خصیصه
- محدودیتهای درستی (جامعیت)
- و همانطور که بعدا گفته خواهد شد اطلاعات بیشتری مثل:
  - مجموعه اندیسهایی که برای هر رابطه نگهداری می شود.
  - اطلاعات امنیتی و مجوزدهی برای هر رابطه
  - ساختار ذخیره سازی فیزیکی هر رابطه در دیسک

Domain Types in SQL

- **char(n)** : رشته های کاراکتری با طول ثابت n که توسط کاربر تعریف می شود.
- **varchar(n).** رشته های کاراکتری با طول متغیر حداکثر n که توسط کاربر تعریف می شود.
- **int.** عدد صحیح
- **smallint.** عدد صحیح کوتاه
- **numeric(n,p).**عدد ممیز ثابت، n رقم با دقت p رقم که توسط کابر تعریف میشود.
- **real, double precision.** اعداد ممیز شناور و اعداد با دقت مضاعف
- **float(n).** اعداد ممیز شناور با دقت حداقل n رقم که توسط کاربر تعریف میشود.

ساختار ایجاد جدول

- رابطه SQL با فرمان **create table** تعریف می شود.

create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

  - r نام رابطه است.
  - هر A_i نام یک خصیصه در شما یا رابطه r است.
  - r is the name of the relation
  - D_i نوع داده مقادیر در دامنه خصیصه A_i است.

- Example:

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null**,
dept_name **varchar**(20),
salary **numeric**(8,2))

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

محدودیت های درستی
Integrity Constraints in Create Table

- **not null**
- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*

Example: Declare *ID* as the primary key for *instructor*

.

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)*

تعریف کلید اصلی برای یک خصیصه به صورت خودکار باعث ایجاد محدودیت **not null** میشود.**I**

And a Few More Relation Definitions

- **create table** *student* (
  ID **char**(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 **char**(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*);

And more still

- **create table** *course* (
  course_id **varchar**(8) **primary key**,
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  foreign key *(dept_name*) **references** *department)*);
  - Primary key declaration can be combined with attribute declaration as shown above

Drop and Alter Table Constructs

- **drop table** *student*
  - جدول و محتویات آن را حذف می کند
- **delete from** *student*
  - همه محتویات جدول را حذف می کند ولی تعریف آن را حفظ میکند.
- **alter table**
  - **alter table** *r* **add** *A D*
    - خصیصه *A* با دامنه D به رابطه *r* اضافه می شود.
    - مقدار خصیصه جدید در همه رکوردها برابر با null می شود.
  - **alter table** *r* **drop** *A*
    - A نام خصیصه ای از رابطه r است که حذف می شود.

Basic Query Structure

- The SQL **data-manipulation language** (**DML**) زبان دستکاری داده ها در SQL امکان پرس و جوی اطلاعات و حذف، درج و بروزرسانی رکوردها را فراهم می سازد.
- A typical SQL query has the form:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

  - A_i خصیصه را نشان می دهد.
  - R_i رابطه را نشان می دهد.
  - P گزاره است.
- نتیجه پرس و جوی SQL یک رابطه است.

عبارت select

- لیست خصیصه های موردنیاز در نتیجه پرس و جو را لیست میکند.
  - مشابه عملگر پرتو(تصویر) در جبر رابطه ای است.
- Example: find the names of all instructors:

  select *name*

  from *instructor*
- توجه: نامها در SQL به حروف بزرگ و کوچک حساس نیستند.
  - E.g. *Name* ≡ *NAME* ≡ *name*

select *name* from *instructor*

name
Srinivasan
Wu
Mozart
Einstein
El Said
Gold
Katz
Califieri
Singh
Crick
Brandt
Kim

The select Clause (Cont.)

- در SQL رکوردهای تکراری مجاز است.
- برای حذف رکوردهای تکراری کلمه کلیدی **distinct** استفاده می شود.
- Find the names of all departments with instructor, and remove duplicates

```
select distinct dept_name
from instructor
```

- کلمه کلیدی **all** نشان میدهد که تکرارها نباید حذف شوند.

```
select all dept_name
from instructor
```

select all *dept_name* from *instructor*

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

The select Clause (Cont.)

- علامت ستاره در عبارت select به معنی "همه خصیصه ها" است.

```
select *
from instructor
```

- در عبارت می توان از عبارتهای ریاضی شامل عملگرهای + ، - ، * و / بر روی مقادیر ثابت یا خصیصه های رکوردها استفاده کرد.
- پرس و جوی زیر:

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

رابطه ای را برمیگرداند که مشابه رابطه instructor است به جز اینکه مقدار خصیصه salary تقسیم بر 12 شده است.

The where Clause

- عبارت **where** شرایطی را مشخص میکند که نتیجه پرس و جو باید داشته باشد.
  - مشابه عبارت انتخاب در جبر رابطه ای است.
- مثال: یافتن همه استادها در گروه کامپیوتر که حقوق آنها از 70000 بیشتر است:

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

- نتایج مقایسه ها را می توان با عملگرهای منطقی and , or , not ترکیب کرد.
- مقایه ها را می توان به نتایج عبارت های ریاضی هم اعمال کرد.

name
Katz
Brandt

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name= department.dept_name;
```

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

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

name	*dept_name*	*building*
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter
Mozart	Music	Packard
Einstein	Physics	Watson
El Said	History	Painter
Gold	Physics	Watson
Katz	Comp. Sci.	Taylor
Califieri	History	Painter
Singh	Finance	Painter
Crick	Biology	Watson
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor

The from Clause

- عبارت **from** همه رابطه هایی که در پرس و جو درگیر می شود را مشخص می کند.
  - هم ارز با ضرب کارتزین در جبر رابطه ای است.
- مثال: یافتن ضرب کارتزین *instructor X teaches*

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

  - همه جفت اطلاعات ممکن instructor – teaches را تولید میکند و شامل فیلدهای هر دو رابطه می باشد.
- ضرب کارتزین مستقیما خیلی استفاده نمی شود، اما زمانی که با عبارت where ترکیب میشود بسیار مورد استفاده است. (عملگر انتخاب در جبر رابطه ای)

Cartesian Product: *instructor X teaches*

instructor

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

teaches

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

inst.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

```
select name, course_id
        from instructor, teaches
        where   instructor.ID = teaches.ID
```

name	*course_id*
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

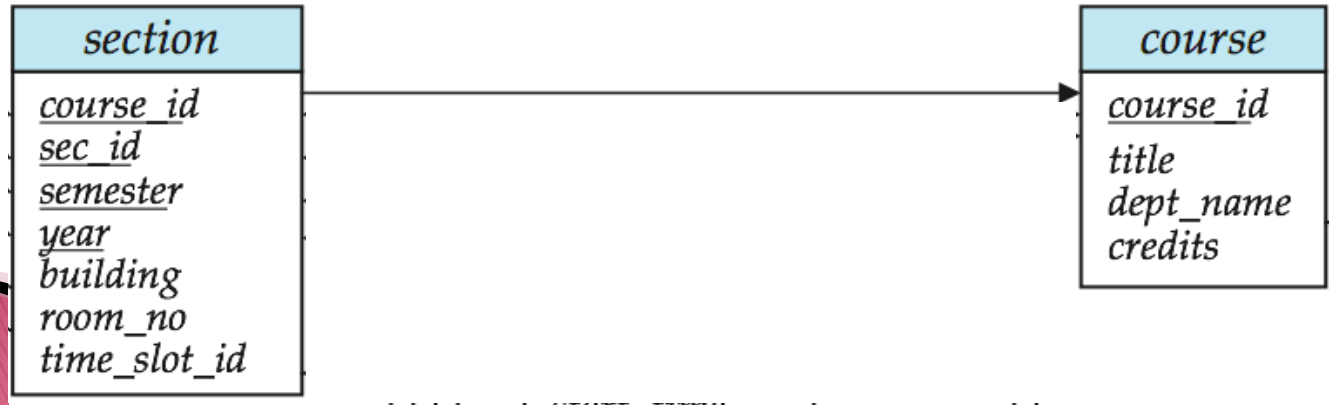
الحاق Joins

- مثال: برای همه استادهایی که درسی را ارائه کرده اند، نام استاد و کد درس آنها را پیدا کن

```
select name, course_id
  from instructor, teaches
  where   instructor.ID = teaches.ID
```

- مثال: کد درس، ترم، سال و عنوان هر درسی که گروه کامپیوتر ارائه کرده است را پیدا کن

```
select section.course_id, semester, year, title
  from section, course
  where   section.course_id = course.course_id  and
          dept_name = 'Comp. Sci.'
```



الحاق طبیعی Natural Join

- در الحاق طبیعی رکوردهایی که مقدار همه خصیصه های مشترکشان برابر است با هم تطبیق داده می شوند و فقط یکی کپی از هریک از ستونهای مشترک نگه داشته می شود.

- **select** *
  from *instructor* **natural join** *teaches*;

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Natural Join Example

- لیست نام همه استادها به همراه کد دروسی که تدریس می کنند.

- **select** *name*, *course_id*
  from *instructor, teaches*
  where *instructor.ID* = *teaches.ID*;

  - **select** *name*, *course_id*
    from *instructor* **natural join** *teaches*;

Natural Join (Cont.)

- توجه: در الحاق طبیعی باید درنظر داشت که بعضی از خصیصه های نامرتبط هستند که نام مشابهی دارند و به طور نادرست یا هم تطبیق داده می شوند.
- لیست نام استادها و عنوان دروسی که تدریس میکنند:
  - روش نادرست: (که در آن instructor.dept_name = course.dept_name)
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches* **natural join** *course*;
  - روش صحیح
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches*, *course*
      where *teaches*.*course_id* = *course*.*course_id*;
  - یک روش صحیح دیگر
    - **select** *name*, *title*
      from (*instructor* **natural join** *teaches*)
      join *course* **using**(*course_id*);

The Rename Operation عملگر تغییر نام

- در SQL میتوان با استفاده از عبارت as نام رابطه ها و خصیصه ها را تغییر داد:

  old-name **as** *new-name*

- مثال

  - **select** *ID, name, salary/12* **as** *monthly_salary*
    from *instructor*

- یافتن نام همه استادهایی که حقوق آنها از برخی استادهای گروه کامپیوتر بیشتر است.

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

- میتوان کلمه کلیدی as را استفاده نکرد:

  instructor **as** *T* ≡ *instructor T*

عملگرهای مجموعه

- SQL دارای یک عملگر تطبیق رشته برای مقایسه رشته های کاراکتری است. عملگر "like" از الگوهایی که با استفاده از دو کاراکتر خاص توصیف می شوند استفاده میکند:
  - علامت درصد (%) : با هر زیررشته ای منطبق می شود.
  - خط تیره (_) : با هر کاراکتری منطبق می شود.

- مثال: یافتن نام همه استادهایی که نام آنها شامل زیررشته "dar" است.

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

- برای تطبیق رشته "100 %"

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

عملگرهای رشته (ادامه)

- الگوها نسبت به حروف کوچک و بزرگ حساس هستند.
- مثال:
  - ’Intro%‘ با هر رشته ای که با ”Intro“ شروع شود منطبق می شود.
  - ’%Comp%‘ با هر رشته ای که شامل زیر رشته ”Comp“ باشد منطبق می شود.
  - ’_ _ _‘ با هر رشته سه کاراکتری منطبق می شود.
  - ’% _ _ _‘ با هر رشته ای که حداقل سه کاراکتری باشد منطبق می شود.

مرتب کردن نمایش رکوردها

- نمایش نام همه استادها به ترتیب الفبا

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

- کلمه **desc** برای ترتیب نزولی و **asc** برای ترتیب صعودی به کار می رود (که پیش فرض است).
  - مثال: **order by** *name* **desc**
- می توان اطلاعات را بر اساس رکیبی از فیلدها نیز مرتب کرد.
  - **order by** *dept_name, name*

گزاره های عبارت Where

- عملگر **between** هم در SQL برای مقایسه تعریف شده است.
- یافتن نام همه استادهایی که حقوق آنها بین $90,000 و $100,000 است (یعنی ≥ $90,000 and ≤ $100,000)

```
select name
  from instructor
  where salary between 90000 and 100000
```

- مقایسه رکوردها

```
select name, course_id
from instructor, teaches
where (instructor.ID, dept_name) = (teaches.ID, 'Biology');
```

عملگرهای مجموعه ای

- اجتماع: همه دروسی که در پاییز 2009 یا بهار 2010 ارائه شده اند.

(**select** *course_id* **from** *section* **where** *sem* = ‘Fall’ **and** *year* = 2009)
union
(**select** *course_id* **from** *section* **where** *sem* = ‘Spring’ **and** *year* = 2010)

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

- اشتراک: همه دروسی که در پاییز 2009 و بهار 2010 ارائه شده اند.

(**select** *course_id* **from** *section* **where** *sem* = ‘Fall’ **and** *year* = 2009)
intersect
(**select** *course_id* **from** *section* **where** *sem* = ‘Spring’ **and** *year* = 2010)

course_id
CS-101

- تفاضل: همه دروسی که در پاییز 2009 ارائه شده اند ولی در بهار 2010 ارائه نشده اند.

(**select** *course_id* **from** *section* **where** *sem* = ‘Fall’ **and** *year* = 2009)
except
(**select** *course_id* **from** *section* **where** *sem* = ‘Spring’ **and** *year* = 2010)

course_id
CS-347
PHY-101

عملگرهای مجموعه

- در عملیات مجموعه ای، تکرار ها حذف می شود.
- برای حفظ همه تکرارها از نسخه multiset استفاده میشود: **union all,** **intersect all** and **except all.**

فرض کنید یک رکورد mمرتبه در رابطه r و n مرتبه در s ظاهر شده باشد. در این صورت تعداد دفعات آن در رابطه نتیجه به صورت زیر است:

- $m + n$ times in *r* **union all** *s*
- min(*m,n)* times in *r* **intersect all** *s*
- max(0, $m - n$) times in *r* **except all** *s*

مقادیر تهی Null Values

- ممکن است برخی از خصیصه های یک رکورد مقدار تهی داشته باشند که با null نشان داده می شود.
- منظور از null یک مقدار نامعلوم یا مقداری است که وجود ندارد.
- نتیجه هر عبارت محاسباتی که در آن null باشد، null خواهد بود.
  - Example: 5 + *null* returns null
- عبارت is null را میتوان برای بررسی مقادیر null استفاده کرد.
  - مثال: یافتن همه استادهایی که حقوق آنها null است:

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

توابع تجمیع Aggregate Functions

- این توابع برروی مقادیر یک ستون در رابطه اعمال می شوند و یک مقدار برمیگردانند.

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

توابع تجمیع (ادامه)

- میانگین حقوق استادهای گروه کامپیوتر را تعیین کن

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

- تعداد استادهایی که درسی در بهار 2010 ارائه کرده اند را تعیین کن.

```
select count (distinct ID)
from teaches
where semester = 'Spring' and year = 2010
```

- تعداد رکوردها در رابطه course را تعیین کن.

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

توابع تجمیع- Group By

- تعیین میانگین حقوق استادهای هر گروه:
  - **select** *dept_name*, **avg** (*salary*)
    from *instructor*
    group by *dept_name*;

  - توجه: گروه هایی که استادی ندارند در نتیجه دیده نمی شوند.

ID	*name*	*dept_name*	*salary*
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

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

توابع تجمیع- Group By

- یافتن تعداد استادهای هر گروه که درسی را در بهار 2010 ارائه کرده اند

- **select** *dept name*, **count** (**distinct** *ID*) **as** *instr_count*
  from *instructor* **natural join** *teaches*
  where *semester* = 'Spring' **and** *year* = 2010
  group by *dept_name*;

dept_name	*count*
Comp. Sci.	3
Finance	1
History	1
Music	1

توابع تجمیع (ادامه)

- خصیصه ها در عبارت select خارج از توابع تجمیع باید در لیست group by وجود داشته باشند.
- /* پرس و جوی نادرست */
  select *dept_name*, *ID*, **avg** (*salary*)
  from *instructor*
  group by *dept_name*;

توابع تجمیع- عبارت Having

- یافتن نام و میانگین حقوق همه گروههایی که میانگین حقوق آنها از 42000 بیشتر است.

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

توجه: گزاره های عبارت having بعد از گروه بندی اعمال می شوند درصورتی که عبارت where قبل از گروه بندی اعمال می شود.

dept_name	*avg(salary)*
Physics	91000
Elec. Eng.	80000
Finance	85000
Comp. Sci.	77333
Biology	72000
History	61000

تجمیع و مقادیر null

- مجموع کل حقوقها

```
select sum (salary)
from instructor
```

  - در عبارت فوق، مقادیر null نادیده گرفته می شوند.
  - اگر هیچ مقدار غیرتهی وجود نداشته باشد نتیجه null را برمیگرداند.
- همه عملگرهای تجمیع به جز count رکوردهایی را که مقدار خصیصه تجمیع شده آنها null است را نادیده میگیرند.
- اگر همه رکوردها در خصیصه موردنظر مقدار null را داشته باشند:
  - count مقدار 0 را برمیگرداند
  - همه توابع دیگر مقدار null را برمیگردانند.

زیرپرس و جوهای تودرتو

- SQL مکانیسمی برای زیرپرس و جوهای تودرتو دارد.
- مظور از زیرپرس و جو، یک عبارت **select-from-where** است که در پرس و جوی دیگری قرار گرفته است.
- یکی از کاربردهای متداول زیرپرس و جوها، بررسی عضویت، مقایسه مجموعه ها و کاردینالیتی مجموعه است.

مثال

- تعیین همه دروسی که در پاییز 2009 و بهار 2010 ارائه شده اند.

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
        course_id in (select course_id
                      from section
                      where semester = 'Spring' and year= 2010);
```

- یافتن دروسی که در پاییز 2009 ارائه شده اند ولی در بهار 2010 ارائه نشده اند.

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
        course_id  not in (select course_id
                      from section
                      where semester = 'Spring' and year= 2010);
```

مثال

- تعیین مجموع تعداد دانشجویان (مجزا) که درسی با استاد شماره 10101 گرفته اند.

```
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);
```

■توجه: پرس وجوی بالا را به روش ساده تری هم می توان نوشت.

مقایسه مجموعه ها

- یافتن نام استادهایی که حقوق آنها بیشتر از حداقل یکی از استادهای گروه Biology است.

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

- همان پرس و جو با استفاده از عبارت **> some**

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

تعریف عبارت some

- F <comp> **some** *r* به این معنی است که جداقل یک رکورد t در r وجود داشته باشد که رابطه (*t* <comp> F) برای آن برقرار باشد. <comp>می تواند یکی از عملگرهای زیر باشد:
- <, ≤, >, =, ≠

(5 < **some** | 0 / 5 / 6 |) = true (read: 5 < some tuple in the relation)

(5 < **some** | 0 / 5 |) = false

(5 = **some** | 0 / 5 |) = true

(5 ≠ **some** | 0 / 5 |) = true (since 0 ≠ 5)

(= **some**) ≡ **in**

However, (≠ **some**) ≢ **not in**

مثال

- نام همه استادهایی را پیدا کن که حقوق آنها از حقوق همه استادهای گروه Biology بیشتر باشد.

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

تعریف عبارت all

- F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ (F <comp> *t)*

(5 < **all** | 0 / 5 / 6 |) = false

(5 < **all** | 6 / 10 |) = true

(5 = **all** | 4 / 5 |) = false

(5 ≠ **all** | 4 / 6 |) = true (since 5 ≠ 4 and 5 ≠ 6)

(≠ **all**) ≡ **not in**

However, (= **all**) ≠ **in**

بررسی رابطه های خالی

- عبارت exists در صورتی که نتیجه زیرپرس و جوی موردنظر خالی نباشد مقدار true را برمیگرداند..

- **exists** $r \Leftrightarrow r \neq \varnothing$
- **not exists** $r \Leftrightarrow r = \varnothing$

Correlation Variables متغیرهای همبسته

- راه دیگر برای پرس و جوی « یافتن همه دروسی که هم در پاییز 2009 و هم در بهار 2010 ارائه شده اند»

```
select course_id
from section as S
where semester = 'Fall' and year= 2009 and
        exists (select *
                from section as T
                where semester = 'Spring' and year= 2010
                        and S.course_id= T.course_id);
```

- Correlated subquery
- Correlation name or correlation variable

Not Exists

- یافتن نام همه دانشجویانی که همه درسهایی را که در گروه biology ارائه شده اند را گرفته اند.

```
select distinct S.ID, S.name
from student as S
where not exists ( (select course_id
                    from course
                    where dept_name = 'Biology')
                   except
                    (select T.course_id
                     from takes as T
                     where S.ID = T.ID));
```

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

بررسی عدم وجود رکوردهای تکراری

- عبارت unique بررسی میکند که آیا نتیجه یک زیرپرس و جو دارای رکوردهای تکراری هست یا خیر.
  - اگر مجموعه تهی باشد نتیجه true را برمیگرداند.
- تعیین همه دروسی که حداکثر یک بار در سال 2009 ارائه شده اند.

```
select T.course_id
from course as T
where unique (select R.course_id
              from section as R
              where T.course_id= R.course_id
                    and R.year = 2009);
```

زیرپرس و جو در عبارت from

- در SQL میتوان از عبارت زیرپرس و جو در عبارت from استفاده کرد.
- تعیین میانگین حقوق در گروههایی که میانگین حقوق در آنها بیشتر از 42000 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;
```

- توجه: دراینجا لازم نیست از عبارت having استفاده کنیم.
- روش دیگر برای نوشتن پرس و جوی بالا:

```
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

- با استفاده از عبارت with می توان یک دیدگاه موقتی تعریف کرد که تعریف آن فقط برای پرس و جویی در دسترس است که در آن عبارت with ورخ داده باشد.
- تعیین همه گروههایی کهبودجه آنها حداکثر است.

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

پرس و جوهای پیچیده با استفاده از گزاره with

- عبارت with برای نوشتن پرس و جوهای پیچیده کاربرد زیادی دارد.
- یافتن همه گروه هایی که در آنها مجموع حقوق از میانگین حقوق کل همه گروهها بیشتر است.

```
with dept_total (dept_name, value) as
        (select dept_name, sum(salary)
         from instructor
         group by dept_name),
dept_total_avg(value) as
       (select avg(value)
       from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value >= dept_total_avg.value;
```

زیرپرس و جوهای اسکالر

- پرس و جوهای اسکالر، پرس و چوهایی هستند که زمانی استفاده میشوند که یک مقدار تکی به دست می آید.
- مثال.
```
select dept_name,
        (select count(*)
            from instructor
            where department.dept_name =
instructor.dept_name)
        as num_instructors
    from department;
```
- مثال.
```
select name
    from instructor
    where  salary * 10 >
        (select budget  from department
            where department.dept_name =
instructor.dept_name)
```

تغییر در پایگاه داده

- حذف رکوردها از یک رابطه
- درج رکوردهای جدید در یک رابطه
- بروزرسانی برخی مقادیر رکوردهایی در یک رابطه

تغییر در پایگاه داده- حذف

- حذف همه استادها

```
delete from instructor
```

- حذف همه استادهای گروه Finance.

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

- حذف همه رکوردهای استادهایی که در ساختمان Watson هستند از رابطه *instructor*

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

حذف (ادامه)

- حذف همه استادهایی که حقوق آنها کمتر از میانگین حقوق استادها است.

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

- 1- میانگین حقوق محاسبه میشود و همه رکوردهایی که باید حذف شوند پیدا می شوند.
- 2- همه رکوردهایی که قبلا پیدا شده اند حذف می شوند (بدون محاسبه مجدد میانگین)

تغییر در پایگاه داده- درج

- اضافه کردن یک رکورد جدید به رابطه *course*

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

- یا معادل آن:

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

- اضافه کردن یک رکورد جدید به رابطه دانشجوکه در آن فیلد tot_creds تهی است.

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

درج (ادامه)

- اضافه کردن همه استادها به رابطه دانشچو و تعیین tot_creds برابر با صفر

```
insert into student
  select ID, name, dept_name, 0
    from   instructor
```

- عبارت **select from where** کاملا قبل از درج در رابطه محاسبه می شود.

تغییر در پایگاه داده - بروزرسانی

- افزایش حقوق استادهایی که حقوق آنها بیشتر از 100000 است به میزان 3% و افزایش حقوق سایر استادها به میزان 5%.
  - دو دستور بروزرسانی نوشته می شود (ترتیب دستورها مهم است)

```
update instructor
   set salary = salary * 1.03
   where salary > 100000;
update instructor
    set salary = salary * 1.05
    where salary <= 100000;
```

  - با استفاده از دستور case هم امکان پذیر است (اسلاید بعد)

عبارت case برای بروزرسانی های شرطی

- مشابه پرس و جوی قبل با استفاده از عبارت case

```
update instructor
    set salary = case
        when salary <= 100000 then salary * 1.05
        else salary * 1.03
    end
```

بروزرسانی با استفاده از زیرپرس وجوی اسکالر

- محاسبه مجدد و بروزرسانی مقدار tot_creds برای همه دانشجویان

```
update student S
    set tot_cred = ( select sum(credits)
    from takes natural join course
    where S.ID= takes.ID and
            takes.grade <> 'F' and
            takes.grade is not null);
```

- Sets *tot_creds* to null for مقدار tot_creds را برای دانشجویانی که هیچ درسی انتخاب نکرده اند صفر می کند.
- به جای **sum**(*credits*) از عبارت زیر استفاده می شود:

```
case
    when sum(credits) is not null then sum(credits)
    else 0
 end
```