

Chapter 3: SQL基础

主要内容

- 3.1 SQL语言概览
- 3.2 数据定义
- 3.3 SQL查询基本结构
- 3.4 附加的基本运算
- 3.5 集合运算
- 3.6 空值
- 3.7 聚集函数
- 3.8 嵌套子查询
- 3.9 数据库的修改

3.1 SQL 查询语言

□ 发展历史

□ 命名为结构化查询语言 Structured Query Language (SQL)

□ ANSI and ISO 标准 SQL:

■ SQL-86

■ SQL-89

■ SQL-92

■

■ SQL:2016

■ 不是这里所有的例子都可以在你的特定的系统上可以运行。
注意：教材上的例子都是根据SQL 标准设计的，和具体的DBMS软件中的语法有所区别。

标准是企业竞争的制高点

□ SQL 语言包括：

- Data-definition language 数据定义语言
- Data-manipulation language 数据操纵语言
- Integrity 完整性
- View definition 视图定义
- Transaction control 事务控制
- Embedded SQL/Dynamic SQL 嵌入/动态SQL
- Authorization 授权

3.2 数据定义语言

□ DDL 可以定义:

- 每个关系的模式
- 与每个属性相关的值域
- 完整性约束
- 为每个关系维护的索引的集合
- 每个关系的安全和授权信息
- 磁盘上每个关系物理存储结构

3.2.1 基本数据类型

char(n). 具有用户指定长度n的固定长字符串.

varchar(n). 具有用户指定的最大长度n的变长字符串.

int. 整数（一个与机器相关的整数的有限子集）.

smallint. 小整数（一个与机器相关的整数域类型的有限子集）.

numeric(p,d). 定点数, 具有用户指定的p位数字精度, 小数点右边具有d位数字.

real, double precision. 浮点和双精度浮点数, 精度与机器有关。

float(n). 浮点数, 具有用户指定的至少n位精度.

更多域类型件后续章节

□ 小组讨论:

- 如何为表中属性选择恰当的数据类型?

一般原则

- 判断数据的种类，需要存储的数据的大小和最大长度，对于数字数据类型还要考虑精度和小数位数；
- 在所用的DBMS找到可用的数据类型，够用即可；
- 要权衡存储空间和查询效率
 - 数据类型的长度越小，存储大量数据所需要的存储空间就越小，分配的页面就越少，一次读取到内存中的数据记录就会越多，操作数据的效率就会提高很多。

3.2.2 基本模式定义

SQL关系使用 **create table**命令定义:

```
create table  $r$  ( $A_1 D_1, A_2 D_2, \dots, A_n D_n,$   
                (integrity-constraint1),  
                ...,  
                (integrity-constraintk))
```

- R 是关系名
- 每个 A_i 是关系 r 的模式的属性名
- D_i 是属性 A_i 的域的值的数据类型

□ 如:

```
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);

完整性约束

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

□ 如:

□ **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*)

如:

```
❑ 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*);

❑ 主码也可以由多个属性的构成

例:

```
❑ 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) );
```

□ 小组讨论:

- 定义表结构时, 主要考虑哪些问题?

Drop and Alter Table 命令

□ **drop table** *student*

- 删除数据表和它的所有数据

□ **delete from** *student*

- 删除表的所有数据,但是保留表结构

□ **alter table**

■ **alter table r add A D :** *增加属性*

其中， A 是要添加到关系 r 的属性名， D 是 A 的域。

■ 关系中的所有元组对于新的属性被设置为null值。

■ **alter table r drop A :** *删除属性*

其中 A 是关系 r 的属性名

■ 许多数据库不支持属性的删除

University 数据库模式

- ❑ *classroom*(building, room number, capacity)
- ❑ *department*(dept name, building, budget)
- ❑ *course*(course id, title, dept_name, credits)
- ❑ *instructor*(ID, name, dept_name, salary)
- ❑ *section*(course id, sec id, semester, year, building, room_number, time_slot_id)
- ❑ *teaches*(ID, course id, sec id, semester, year)
- ❑ *student*(ID, name, dept_name, tot_cred)
- ❑ *takes*(ID, course id, sec id, semester, year, grade)
- ❑ *advisor*(s_ID, i_ID)
- ❑ *time_slot*(time slot id, day, start time, end_time)
- ❑ *prereq*(course id, prereq id)

3.3 基本的查询结构

□ 典型的关系查询具有如下形式:

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

- A_i 表示属性
- R_i 表示关系
- P 是谓词.

□ 注意: 查询结果是表

□ 小组讨论:

■ SQL query和Relational Algebra的关系?

3.3.1 单表查询

□ select子句：列出查询的结果集的属性

■ 对应于关系代数的投影操作

■ 如：查询所有教师的名字：

select *name*
from *instructor*

□ 注意：SQL命令大小写不敏感的

■ E.g. *Branch_Name* \equiv *BRANCH_NAME* \equiv *branch_name*

The select Clause 选择子句 投影

- ❑ SQL允许在查询结果中出现重复查询结果
- ❑ 要强制消除重复，可以在select后插入distinct.
- ❑ 如：找出教师表中中的所有系名，并消除重复

```
select distinct dept_name  
from instructor
```

- ❑ 关键词all指定不要去重，默认，可省略不写

```
select all dept_name  
from instructor
```

The select Clause (Cont.)

- 选择子句中的*代表所有的属性

```
select *  
from instructor
```

- select 子句可以包含涉及+, -, *, /和对元组属性和常量操作的算术表达式
- 查询:

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

查询返回结果中, 对属性*salary* 进行了计算, 除以了12, 求月平均工资.

□ 小结:

- SQL 允许重复 （关系和结果）
- 强调 去重复: **distinct**
- 默认保留重复记录，也可以用all

□ 小组讨论:

- 用select....from对instructor表设计一个查询

where 子句

- where子句:指定结果必须满足的条件

- 对应于关系代数的选择谓词

- 如: 查找工资大于70000的计算机系的教师

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

True, False,
逻辑操作

- 比较结果可以使用逻辑连接词 and, or和 not组合得到

□ 小组讨论:

- 用select....from.....where对instructor表设计一个查询.

3.3.2 多关系查询

□ 实际中，需要从多个表中查找需要的数据

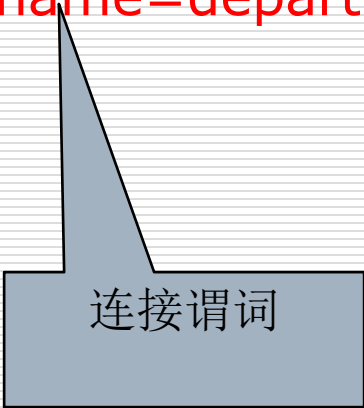
□ 如：

■ 查找所有教师的姓名，所属系及系所在的建筑物

Select name,instructor.dept_name,building

From instructor,department

Where instructor.dept_name=department.dept_name



连接谓词

□ SQL queries包括:

- Select clause
- From clause
- Where clause

From clause

- from子句列出查询所涉及的关系
 - 对应于关系代数的笛卡尔积运算
- 对 *instructor* 和 *teaches* 进行 笛卡积运算

select *
from *instructor, teaches*

- 笛卡积运算会产生错误的结果，但是它可以合并多个表的数据。

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

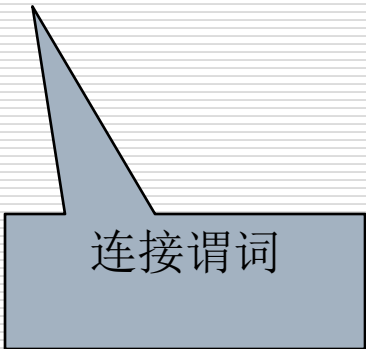
inst.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	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
...
...

郝老师

使用连接谓词来限制笛卡尔积运算结果

□ 如：查询教师及其所授课程。

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



连接谓词

❑ EX: 查找计算机系开设的课程，及这些课程开课的时间。

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

连接谓词

其它谓词



□ 小组讨论:

- 设计多表查询时，要注意哪些问题？
- 对instructor和teaches表设计一个多表查询。

3.4 附加的基本运算符

□ SQL中还支持几种附加的基本运算。

3.4.1 更名运算

- SQL允许使用as子句对关系和属性重新命名:

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

重命名表达式

- E. g.

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

- 查找教师的姓名，这些教师的工资比计算机系的教师工资高

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

重命名表

- 关键字 as 可以省略

instructor **as** *T* \equiv *instructor T*

3.4.2 字符串操作

- ❑ SQL使用一对单引号标识字符串
- ❑ SQL标准中，字符串大小写敏感
- ❑ SQL标准中，可以对字符串使用函数
- ❑ SQL包含一个用于字符串比较的字符串匹配运算符。运算符 “like” 使用由两个特殊字符描述的模式：

3.4.2 字符串操作

□ SQL包含一个用于字符串比较的字符串匹配运算符. 运算符 “like” 使用由两个特殊字符描述的模式:

- % 匹配任何子串.

- _ 匹配任何子串

□ 查找名字中包含 “dar” 字符串的教师的姓名

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

□ escape关键字定义转义字符

- 要匹配名字 “100%”

```
like '100 \%' escape '\转义符\' , 使 “%” 作被处理对象
```

□ 小组讨论:

- 用“like”对instructor表设计一个模糊查询.

3.4.3 Select 子句中的“*”

□ “*” 表示所有属性

Select instructor.*

From instructor,teaches

Where instructor.ID=teaches.ID

3.4.4 对元组排序显示

- 按字母顺序列出所有教师的姓名。

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

升序

- 可以指定升序 **desc** 或降序 **asc** ; 对每个属性, 升序是缺省情形.

- 如: **order by name desc**

- 可以按多个字段进行排序

- Example: **order by dept_name, name**

降序

3.4.5 where子句谓词

□ 区间查询: *between and*

- 找出贷款金额在90000和100000之间的所有贷款号码

```
select loan-number  
      from loan  
      where amount between 90000 and 100000
```



- 闭区间 [90000, 100, 000]

□ *not between and*

□ 小组讨论:

- 综合运用 `order by` 和 `between... and` 对 `takes` 表设计一个查询.

3.5 集合运算

□ Union : 并运算

- 查找在2017年秋季或2018年春季开设的课程

□ (**select** *course_id* **from** *section* **where** *sem* =
'Fall' **and** *year* = 2017)

union

(**select** *course_id* **from** *section* **where** *sem* =
'Spring' **and** *year* = 2018)

□ **Intersect:** 交运算

- 查找在2017年秋季和2018年春季都开设的课程

□ (**select** *course_id* **from** *section* **where** *sem*
= 'Fall' **and** *year* = 2017)

intersect

(**select** *course_id* **from** *section* **where** *sem*
= 'Spring' **and** *year* = 2018)

□ **Except** : 差运算

- 查找在2017年秋季开设的, 但是在2018年春季不开设的课程

□ (**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2017)

except

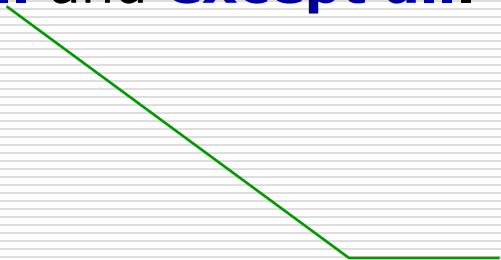
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2018)

Set Operations集合操作，自动删除重复

□ Set operations **union**, **intersect**, and **except**

- 这些运算自动去重复
- 集合运算保留重复记录，要用：

union all, intersect all and except all.



不删除重复
要专门说明

3.6 空值: Null Values

- ❑ 对元组的某些属性来书, 取空值null是可能的
- ❑ *null* 表示一个未知值或值不存在.
- ❑ 涉及到null的算术表达式返回null
 - 如: *5 + null* returns null
- ❑ 谓词 **is null**可以用于检查null值
 - Example: 查询每个教师的工资是否为空.

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



检查salary是否为空

□ 1 < null return

■ .T. ?

■ .F. ?

■ *unknown*



A third logical value

空值和三值逻辑 (Three Valued Logic)

- 与null的比较返回 *unknown*
 - Example: $5 < null$ or $null <> null$ or $null = null$
- 三值逻辑 (Three-valued logic) 使用真值 *unknown*:
 - OR: $(unknown \textbf{ or } true) = true$,
 $(unknown \textbf{ or } false) = unknown$
 $(unknown \textbf{ or } unknown) = unknown$
 - AND: $(true \textbf{ and } unknown) = unknown$,
 $(false \textbf{ and } unknown) = false$,
 $(unknown \textbf{ and } unknown) = unknown$
 - NOT: $(\textbf{not } unknown) = unknown$
 - 如果谓词P求值为unknown, 则 “***P is unknown***”求值为 true
- **where** 子句谓词的结果视为 *false*, 如果它的值求得为 *unknown*

3.7 聚集函数: Aggregate Functions

- 这些函数操作于一个关系的一列的值的多重集版本并返回一个值.

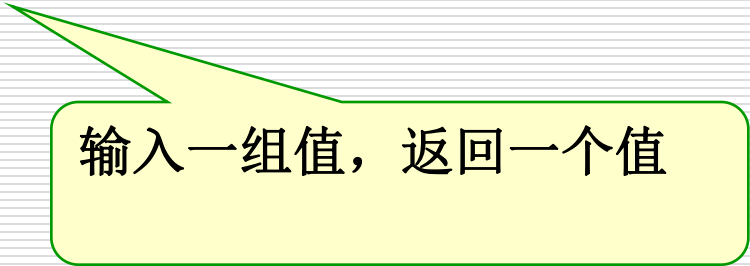
avg: 平均值 average value

min: 最小值 minimum value

max: 最大值 maximum value

sum: 值的和 sum of values

count: 值的数量 number of values



输入一组值, 返回一个值

Aggregate Functions (Cont.)

- 查询计算机系教师的平均工资

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

投影和聚集函数同时进行

- 查询在2018年春季教授一门课程的教师人数。

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

Aggregate Functions (Cont.)

□ 查询course 表的记录数。

■ **select count (*)**
from course;

3.7.2 分组聚集

□ 查询每个系的教师平均工资

■ **select** dept_name, **avg** (salary)
from instructor
group by dept_name;

聚集函数作用
在一组上

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

□ 小组讨论:

- 查询中使用分组和聚集函数时，哪些属性可以出现在**select** 子句中？

Aggregation (Cont.)

- ❑ 在select子句中而在聚集函数外的属性必须出现在group by列表中。

- `/* erroneous query */`
select *dept_name*, *ID*, **avg** (*salary*)
from *instructor*
group by *dept_name*;

错在哪里？

- ❑ 出现在select 中的属性：
 - 或被聚集函数使用
 - 或该属性用于分组

3.7.3 Having 子句

- ❑ 对分组限定条件
- ❑ 查找平均工资大于42000的系的名字和平均工资

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

此 **平均** 在对
什么数据操作？

此 **平均** 对什
么数据操作？

- ❑ Note:
 - ❑ having子句中的谓词必须应用在组形成之后；
 - ❑ 而where子句中的谓词必须应用在成组之前；

3.7.4 带有空值和逻辑值的聚集函数

□ 求所有教师工资总和：

聚集一般忽略空值

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

□ 上述语句忽略 null 的金额

□ 聚集函数简单忽略空值；

■ 除了 `count (*)`

惟一例外：
count(*) 计算 元
组个数，含空值的元
组也计数

□ 小组讨论:

- 综合运用group by和聚集函数对 takes表设计一个查询.

3.8 嵌套子查询

- SQL提供子查询的嵌套机制：
 - 子查询 `subquery` 是一个 `select-from-where`表达式，它被嵌套在另外的查询里面；
- 子查询可以对集合成员资格，集合比较和集合基数进行测试。

3.8.1 集合成员资格 in 和not in

- 查找在2017年秋季和2018年春季都开设的课程.

```
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);
```

- 查找在2017年秋季开设但是在2018年春季不开设的课程

```
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);
```

Example Query

- 查询选修了由ID为10101的教师开设的课程的学生人数（去重）。

```
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');
```



□ 枚举集合

- Select distinct name
- From instructor
- Where name not in ('Mozart','Einstein')

□ 讨论:

■ In能用“=”代替吗?

3.8.2 集合比较

- 查找工资至少比Biology 系某位教师的工资要高的所有教师的姓名.

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

- 使用 some 子句:

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

Example Query

- 查找工资比Biology 系所有教师的工资要高的所有教师的姓名。

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

3.8.3 空关系测试

- 测试一个子查询的结果中是否存在元组;
- **exists** 谓词: 子查询不为空就返回true, 否则为false。

-
- 小组讨论:
 - 分析以下查询的执行过程

Correlation Variables

- ❑ 查找在2017年秋季和2018年春季都开设的课程.

```
select course_id
from section as S
where semester = 'Fall' and year= 2017 and
      exists (select *
               from section as T
               where semester = 'Spring' and year= 2018
                  and S.course_id= T.course_id);
```

- ❑ **Correlated subquery:**相关子查询
- ❑ **Correlation name** or **correlation variable**

Not Exists

□ 查询选修了生物系开设的所有课程的所有学生。

```
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));
```

Not Exists

```
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));
```

查找**Biology**系开设的课程

查找当前学生所选
修的课程

3.8.4 重复元组存在性测试

- **unique** 子句测试查询结果里是否存在重复元组.
- 查询在 2017年最多开设一次的所有课程。

```
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 = 2017);
```

3.8.5 From 子句中的子查询

- ❑ 也可以在 from 子句中使用子查询
- ❑ 查询系平均工资超过 \$42,000的那些系的教师平均工资。

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

计算每个系的平均
工资

- ❑ 不需要使用 having 子句

3.8.5 From 子句中的子查询

□ 另一种实现方式:

```
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;
```

Subqueries in the From Clause (Cont.)

□ **lateral**子句:

```
select name, salary, avg_salary  
from instructor I1,  
      lateral (select avg(salary) as avg_salary  
              from instructor I2  
              where I2.dept_name= I1.dept_name);
```

3.8.6 With Clause

- **with** 子句可以定义临时关系;
- 如: 查询具有最大预算值的系:

临时关系名和属性

with *max_budget (value)* **as**

(select max(budget)

from department)

select budget

使用临时关系

from department, max_budget

where department.budget = max_budget.value;

Complex Queries using With Clause

□ 查询工资总额大于所有系平均工资的系：

```
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;
```

临时关系1

临时关系2

3.8.7 标量子查询

□ 标量子查询：返回一个包含单个属性的元组。

□ 如：查询所有的系及每个系的教师人数：

□ **select** *dept_name*,
 (**select count**(*)
 from *instructor*
 where *department.dept_name* = *instructor.dept_name*)
 as *num_instructors*
from *department*;

3.8.8 不帶 from子句的标量

□ 查找平均每个教师所讲授的课程段数:

■ 方法1: 在有的系统上会报错

```
(select count(*) from teaches ) /  
(select count(*) from instructor );
```

■ 方法2: 使用dual虚拟关系:

```
select(select count(*) from teaches ) /  
(select count(*) from instructor );  
from dual;
```


3.9 数据库的修改

- 数据库的内容可以使用下列操作修改：
 - 删除 Deletion
 - 插入 Insertion
 - 更新 Updating
- 所有这些操作都使用赋值运算符表达。

3.9.1 删除

- 删除所有教师:

```
delete from instructor
```

- 删除 Finance 系的所有教师

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

- 从instructor表中删除所有在Watson大楼工作的教师.

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

Deletion (Cont.)

- ❑ delete命令可以使用嵌套子查询:

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

3.9.2 插入

- ❑ 向course表中新增一套记录:

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

- ❑ 或

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

- ❑ *tot_creds* 为 空值 null

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

Insertion (Cont.)

- 将查询结果写入student表:

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

3.9.3 更新

- 更新所有教师的工资：

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

- 更新工资小于70000的教师工资：

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

-
- 给工资超过\$100,000的教师涨3%，其余教师涨5%

- Write two **update** statements:

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

- 这两条命令执行的顺序很重要
- 使用 **case** 语句更好

□ Case 命令:

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


Update 使用标量子查询

- 更新每个学生的学分总和:

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);

Any Question ?