CH2 关系数据库语言

一、More SQL

1. SELECT-WHERE-FROM

```
SELECT L
FROM R
WHERE C
```

等价于$\pi_L(\sigma_C(R))$

> L 属性集合
>
> C 条件
>
> R 关系

2. Pattern Matching

> % 匹配0或任意长度的字串　%三% -> 张三、三张、张三四
>
> _ 匹配单个字符
>
> [] 表示括号内所列字符的一个 [0-9]、[1, 2, 3]类似yacc
>
> [^] 表示不在括号所列字符之内的单个字符 [^123]除1,2,3都行
>
> ESCAPE 指示转意字符不转意

3. NULL Values

3.1 定义

> 未知的值
>
> 无法使用
>
> 值保留

3.2 运算

`+ * = >`等运行均得到NULL

NULL不是一个常数（无法将其视为操作数）

> x = NULL, x + 3 = NULL (permitted)
>
> NULL + 3 (Denied)

3.3 判断

IS NULL 、 IS NOT NULL

4. UNKNOWN

TRUE as 1, FALSE as 0, UNKNOWN as $\frac{1}{2}$

$\wedge$ →取最大下界

$\vee$ →取最小上界

$\neg$ → 取$1-v$

```
SELECT * FROM Movies WHERE length < 1 AND length >= 1;
```

不会筛选到NULL的元组

5. Ordering排序

`ORDER BY <list of attributes>`

> 默认升序ascending
>
> 使用DESC改为降序排序
>
> 属性表第一个为首要关键字，优先级依次递减

6. Products

```
SELECT name FROM Movies, MovieExec WHERE title='Star Wars' AND producerC# =
cert#;
```

会使用两重循环，在$m \times n$个元组中查询符合条件的，然后将投影结果输出。

7. Tuple Variables

```
SELECT s1.name, s2.name FROM MovieStar s1, MovieStar s2 WHERE s1.address =
s2.address AND s1.name < s2.name;    -- 查询地址相同的两个元组
```

8. Querys总结

> FROM 笛卡儿积(Cartesian Product)
>
> WHERE 选择(selection)
>
> SELECT 投影(projection)

二、Subquery子查询

1. 定义

> 可返回一个值 》》应用于*WHERE*
>
> 可返回一个元组》》应用于*FROM*，后跟元组变量（重命名）
>
> 可返回一个关系》》应用于*WHERE*

关系：

```
SELECT height FROM users WHERE name IN (SELECT DISTINCT name FROM scores);
```

*** unary relation 》》一元关系，一个元组

2. 条件

> 应用到子查询中
>
> 产生Boolean结果

```
EXISTS R;    -- R是否至少含有一条元组
s IN R;      -- R中存在s
s > ALL R;   -- s比R中所有值都大     (< = >= <= = <>[NOT IN])
s > ANY R:   -- s比R中某一个值大
```

3. 相关查询

```
SELECT title
FROM Movies OLD
WHERE yead < ANY
(SELECT year FROM Movies WHERE title=Old.title
);  -- 查询被使用过两次及以上的电影名字
```

三、Group and Aggregation分组与聚合

1. 消除重复元组

```
SELECT DISTINCT name FROM users
```

2. 分组

> 将关系中的元组划分为不同的groups
>
> 基于值

```
SELECT studioName, SUM(length)
FROM Movies GROUP BY studioName;
```

出现在SELECT后的列必须是GROUP BY后面的分组列或对其他列应用聚合函数

```
SELECT studioName FROM Movies GROUP BY studioName -- 等价于使用DISTINCT去重
```

HAVING语句可以让分组满足适当条件

```
SELECT * FROM users GROUP BY id HAVING id > 5
```

HAVING的规则与SELECT一致。

3. 聚合

> 聚合操作分开作用于每个group
>
> 使用聚合操作整合列
>
> 聚合操作忽略空值***

```
SUM     -- 求和
AVG     -- 平均值
MIN     -- 最小值
MAX     -- 最大值
COUNT   -- 数量
```

四、Modification更新

```
INSERT INTO R(A1, A2, ..., An) VALUES (v1, v2, .., Vn);     -- 插入（插入所有值时可省略属性表）
INSERT INTO Studio(name) SELECT DISTINCT studioName FROM Movies WHERE ...;  -- 插入表

DELETE FROM R WHERE ...;    -- 按条件删除
DELETE FROM R;              -- 全部删除

UPDATE scores SET sc=5 WHERE name='aa';
```