# 数据库系统概念笔记

## Part 1：关系语言

### 一·关系模型简介

I.概念定义

- 1.关系数据库由表的集合组成
- 2.关系，元组，属性：在关系模型中，每个表称为关系（relation），表中的每一行称为元组（tuple），表中的每一列称为属性（attribute）
- 3.关系实例：一个关系的特定实例
- 4.域：属性的取值集合
- 5.原子：属性值是不可再分的
- 6.空值：表示属性值未知或不存在
- 7.数据库模式：数据库的逻辑设计
- 8.数据库实例：数据库在某一时刻的内容
- 9.关系模式：关系的逻辑设计，由一个属性列表和各属性所对应的域组成（关系~ x，而关系模式（relation schema） ~ int，关系实例~3）

### 二·SQL 简介

I.概念定义

- 1.SQL：结构化查询语言（Structured Query Language），用于管理和操作关系数据库
- 2.SQL 语言的组分：
  - (1)数据定义语言（DDL，Data Definition Language）：用于定义数据库结构，如创建、修改和删除表
  - (2)数据操作语言（DML，Data Manipulation Language）：用于查询和修改数据库中的数据，如插入、更新、删除和查询操作
  - (3)完整性（Integrity）：用于定义和维护数据的完整性约束
  - (4)视图（View）：用于创建和管理视图，视图是基于一个或多个表的虚拟表
  - (5)事务控制（Transaction Control）：用于管理数据库事务，确保数据的一致性和完整性
  - (6)嵌入式 SQL（Embedded SQL）和动态 SQL：允许在其他编程语言中嵌入 SQL 代码
  - (7)授权（Authorization）：用于管理用户权限和访问控制
- 3.SQL 的数据定义
  - (1)基本类型：
    - `char(n)`：定长字符串，长度为 n
    - `varchar(n)`：变长字符串，最大长度为 n
    - `int`：整数
    - `smallint`：小整数
    - `numeric(p,s)`：定点数，p 为总位数，s 为小数位数
    - `real, double precision`：浮点数，双精度浮点数

  - float(n)：精度至少为 n 位数字的浮点数
- 每种类型都可能包含 NULL 值，表示未知或不存在
- char(5):abc 后面接两个空格，varchar(5):abc 后面不接空格
- 当比较两个 char 类型的值时，如果它们的长度不同，在比较之前会自动在短值后面附加额外的空格以使它们的长度一致
- 即便上述属性 A 和 B 中存放的是相同的值 abc，A=B 的比较也可能返回假，因此建议始终用 varchar

- 4. 基本模式定义
  - 我们用 CREATE TABLE 语句来定义 SQL 模式

```
CREATE TABLE <表名> (
  <属性1> <数据类型1> [完整性约束1],
  <属性2> <数据类型2> [完整性约束2],
  ...
  <属性n> <数据类型n> [完整性约束n],
  primary key(<属性1>)# create table 命令还指明了属性1是该表的主码
);
```

  - SQL 支持的几种完整性约束：
    - primary key $(A_{j1}, A_{j2}, ..., A_{jm})$：主码声明表示属性$A_{j1}, A_{j2}, ..., A_{jm}$构成关系的主码；主码必须是非空且唯一的
    - foreign key$(A_{j1}, A_{j2}, ..., A_{jm})$references s：外码声明表示关系中任意元组在属性$(A_{j1}, A_{j2}, ..., A_{jm})$上的值必须对关系 s 中某元组在主码属性$(B_{k1}, B_{k2}, ..., B_{km})$上的取值
    - not null：非空约束表示该属性不能取 NULL 值

```
create table department(
  dept_name varchar(20),
  building varchar(15),
  budget numeric(12,2),
  primary key(dept_name)
);
create table course(
  course_id varchar(7),
  title varchar(50),
  dept_name varchar(20),
  credits numeric(2,0),
  primary key(course_id),
  foreign key(dept_name) references department
  # 此外码声明表示对于每个课程元组来说，该元组中指定的系名必须存在于department关系的主码
dept_name中
);
create table instructor(
  ID varchar(5),
  name varchar(20) not null,# name属性不能为空
  dept_name varchar(20),
  salary numeric(8,2),
  primary key(ID),
  foreign key(dept_name) references department
);
```

  - SQL 禁止破坏完整性约束的任何数据库更新

- 去掉一个关系
  - `drop table r`：删除关系 r 及其所有数据
  - `delete from r`：删除关系 r 中的所有元组，但保留关系模式
- 增删属性
  - `alter table r add A D`：向关系 r 中添加新属性 A，其数据类型为 D，关系中所有现有元组在新属性 A 上的值均为 NULL
  - `alter table r drop A`：从关系 r 中删除属性 A

- 5. SQL 查询的基本结构
  - 单关系查询

```
select name from instrctor;
# 该查询返回instructor关系中所有元组的name属性构成的关系
select (all) dept_name from instrctor;
# 该查询返回instructor关系中所有元组的dept_name属性构成的关系，因为一个系可以有不止一位教师，所以在instructor关系中，一个系的名称可能不止一次出现
select distinct dept_name from instrctor;
# 该查询返回instructor关系中所有不同的dept_name属性值构成的关系
select name, salary*1.1 from instrctor;
# 该查询返回instructor关系中所有元组的name和salary*1.1属性构成的关系，这并不会改变instructor关系中的数据
select name from instrctor where salary>80000 and dept_name='Comp.Sci';
# 该查询返回instructor关系中所有满足salary>80000且dept_name
```

  - 多关系查询

```
select name,instructor.dept_name,building from instructor,department
where instructor.dept_name=department.dept_name;
# 该查询返回所有教师的姓名、系名和所在楼名（注意：department属性出现在两个关系中，因此用关系名作为前缀）
```

- 6. 附加的基本运算
  - 更名运算

```
select name as instructor_name,course_id from instructor,teaches where
instructor.ID=teaches.ID;
# 该查询返回所有教师的姓名和所授课程号，并将name属性更名为instructor_name
select distinct T.name from instructor as T,instructor as S where
T.salary>S.salary and S.dept_name='Comp.Sci';
# 找出满足下面条件的所有教师的姓名，他们比CS系教师的最低工资要高
```

在上述查询中，我们用 as 关键字将 instructor 关系更名为 T，将另一个 instructor 关系更名为 S，称为相关名称/表别名/相关变量/元组变量

  - 字符串运算
    - SQL 使用''括起字符串常量，若字符串常量中包含'，则用两个'表示一个'
    - 多种函数
      - `upper(s)`：将字符串 s 转换为大写形式
      - `lower(s)`：将字符串 s 转换为小写形式
      - `substr(s,n1,n2)`：返回字符串 s 中从位置 n1 开始的 n2 个字符组成的子串
      - `length(s)`：返回字符串 s 的长度
      - `trim(s)`：去掉字符串 s 两端的空格
      - `||`：字符串连接运算符

    - s1||s2：将字符串 s1 和 s2 连接成一个新字符串
    - select 'abc' || 'def'的结果是 abcdef
    - select 'abc' || ' ' || 'def'的结果是 abc def
    - select 'abc' || '' || 'def'的结果是 abcdef
    - select 'abc' || NULL || 'def'的结果是 NULL

  不同数据库系统可能支持不同的字符串函数
  - like 运算符
    - s1 like s2：如果字符串 s1 与模式 s2 匹配，则返回真，否则返回假
    - 模式 s2 可以包含两种通配符：
      - %：表示任意长度的任意字符串（包括空字符串）
      - _：表示任意单个字符
    - 例如，'abc' like 'a_c'返回真，'abc' like 'a%'返回真，'abc' like '%b%'返回真，'abc' like '_b_'返回真，'abc' like 'a_d'返回假
    - like 运算符通常区分大小写（即大小写字符不匹配），但有些数据库系统（例如 MySQL,PostgreSQL）提供了不区分大小写的 like 运算符，如 ilike

    ```
    select dept_name from department where building like '%Watson%';
    ```

    - escape 定义转义字符

    ```
    select name from instructor where name like 'Mc\_%' escape '\';
    ```

- * 运算符
  - SQL 使用 * 表示所有属性

  ```
  select * from instructor;
  # 该查询返回instructor关系中的所有属性
  select instructor.* from instructor,department where
  instructor.dept_name=department.dept_name;
  # 该查询返回instructor关系中的所有属性
  ```

- 排列元组的显示次序
  - order by 子句

  ```
  select * from instructor order by salary;
  # 该查询返回instructor关系中的所有属性，并按salary属性的值升序排列元组
  select * from instructor order by dept_name,salary desc;
  # 该查询返回instructor关系中的所有属性，并先按dept_name属性的值升序排列元组，在dept_name
  属性值相同的元组中，再按salary属性值降序排列元组
  ```

  - 缺省情况下，order by 子句按升序排列元组，可以用 desc 关键字指定按降序排列
  - SQL 标准允许在 order by 子句中使用属性的位置编号（从 1 开始）

  ```
  select name,dept_name,salary from instructor order by 2,3 desc;
  # 该查询与上一个查询等价
  ```

- where 子句谓词
  - between ... and ...：如果属性值在指定范围内，则返回真
  - 行构造器
    - (A1,A2,...,An)=(v1,v2,...,vn)：如果属性 A1,A2,...,An 的值分别等于 v1,v2,...,vn，则返回真
    - 例如，(dept_name,building)=('Comp.Sci','Watson')

- 集合运算
  - union：并运算，返回两个查询结果的并集，结果中不包含重复元组
  - union all：并运算，返回两个查询结果的并集，结果中包含重复元组
  - intersect：交运算，返回两个查询结果的交集，结果中不包含重复元组
  - intersect all：交运算，返回两个查询结果的交集，结果中包含重复元组（取$c_1, c_2$都出现的重复元组中较小的那一个）
  - except：差运算，返回第一个查询结果中有而第二个查询结果中没有的元组，在执行查运算之前分别对$c_1, c_2$自动去重
  - except all：差运算，返回第一个查询结果中有而第二个查询结果中没有的元组，在执行查运算之前不去重
  - 进行集合运算的两个查询必须是兼容的，即它们必须具有相同数量的属性，并且对应属性的数据类型必须兼容

```
select dept_name from instructor
union
select dept_name from department;
# 该查询返回所有不同的系名，这些系名要么出现在instructor关系中，要么出现在department关系中
select dept_name from instructor
intersect
select dept_name from department;
# 该查询返回所有不同的系名，这些系名既出现在instructor关系中，也出现在department关系中
select dept_name from instructor
except
select dept_name from department;
# 该查询返回所有不同的系名，这些系名出现在instructor关系中，但不出现在department关系中
```

- 空值
  - 若算术表达式中包含 null 值，则该表达式的值为 null
  - unknown：表示逻辑值既不是 true 也不是 false
  - 若 where 子句谓词对一个元组计算出为 unknown，则该元组不包含在查询结果中
  - is null：如果属性值为 null，则返回真
  - is not null：如果属性值不为 null，则返回真、
  - is unknown：如果属性值为 unknown，则返回真
  - is not unknown：如果属性值不为 unknown，则返回真

```
select name from instructor where dept_name is null;
# 该查询返回所有没有系名的教师的姓名
select name from instructor where dept_name is not null;
# 该查询返回所有有系名的教师的姓名
```

当一个查询使用 select distinct 时，结果中不包含重复元组；在遇到两个元组对应的属性值均为 null 时，它们被视为相同的值，因此结果中只包含一个这样的元组，上述对待空值的方式与谓词中的处理方式不同，在谓词中，null=null 返回 unknown，而不是 true

```
create table employees(
  id int,
  name varchar(10),
  department varchar(10),
  primary key(id)
);
```

```
insert into employees values(1,'Alice','HR');
insert into employees values(2,'Bob',NULL);
insert into employees values(3,'Charlie',NULL);
insert into employees values(4,'David','IT');

select distinct department from employees;
# 该查询返回三行：'HR', NULL, 'IT'
select * from employees where department=NULL;
# 该查询返回零行
select * from employees where department is null;
# 该查询返回两行：2,'Bob',NULL,3,'Charlie',NULL
```

如果元组在所有属性上取值相等，那么它们就被当作相同的元组，即使某些值为空。这种方式还被用于集合的并、交和差运算

- 聚集函数：以值集（集合或多重集合）为输入，返回单个值作为输出的函数
  - count：计数函数，返回查询结果中的元组数
    - count(*)：返回查询结果中的元组总数，包括重复元组
    - count(A)：返回查询结果中属性 A 的非空值的数量
  - sum(A)：返回查询结果中属性 A 的非空值的总和
  - avg(A)：返回查询结果中属性 A 的非空值的平均值
  - min(A)：返回查询结果中属性 A 的最小值
  - max(A)：返回查询结果中属性 A 的最大值
  - 若查询结果为空，则 sum(A)和 avg(A)返回 null，而 count(A)返回 0
  - sum 和 avg 输入必须是数集，但其它聚集函数可以作用于数值或非数值属性

```
select count(*), count(dept_name), avg(salary), min(salary), max(salary) from
instructor;
# 该查询返回instructor关系中的元组总数、dept_name属性的非空值数量、salary属性的平均值、最小值和最大值
select dept_name, count(*), avg(salary) from instructor group by dept_name;
# 该查询按系名分组，返回每个系名、该系的教师人数和该系教师的平均工资
select dept_name, count(*), avg(salary) from instructor group by dept_name
having avg(salary)>80000;
# 该查询按系名分组，返回每个系名、该系的教师人数和该系教师的平均工资，但只包括那些平均工资超过80000的系
```

  - group by 子句：将查询结果按一个或多个属性分组
  - having 子句：对分组后的结果进行筛选，仅保留满足指定条件的分组
  - 在使用聚集函数时，select 子句中的所有非聚集属性必须出现在 group by 子句中

## 三 • 中级 SQL

## 四 • 高级 SQL