

4 Intermediate SQL

- 连接表达式

- 自然连接：自然联接为所有公共属性匹配具有相同值的元组，并且只保留每个公共列的一个副本。对于相同的属性要求其全部相等

- from table1 **natural join** table2
- 可以多个表自然连接
- 小心相同的属性但是含义不同---使用**using**(A)指定A属性相等
- 如果属性名不同但又希望作为条件---**on**:from student join takes on student_ID = takes_ID

- 外连接：通过创建包含空值元组的方式保留自然连接可能会丢失的数据

- 左外连接：只保留出现在左外连接运算之前的关系中的元组
- 右外连接：保留出现在运算符之后的关系中的元组
- 全外连接：两个都保留

- 内连接

- 连接条件和连接类型

Join types
inner join
left outer join
right outer join
full outer join

Join conditions
natural
on <predicate>
using (A_1, A_2, \dots, A_n)

- 为了把常规连接和外连接区分开，常规连接成为内连接。join 等价 inner join

- 总结

- 连接关系

- 在From子句中的两个相邻关系之间，可以是一个逗号 —— 表示做笛卡尔积；
- 也可以是一个连接操作 —— 表示将它们连接成一个新的关系每个连接操作由一个连接类型和一个连接条件组成

- 连接条件

- natural :连接两边元组条件为同名属性相等（自然连接）
- using (属性1, 属性2, ...) :类似自然连接，但是只限于列出的的属性相等
- on 条件:按照指定的条件连接两边元组

- 连接类型

- (inner) join :内连接。结果不包含失配元组** 这里失配元组指的是因不满足连接条件，无法和其它元组相连接的元组
- left (outer) join :左外连接。结果包含左边关系的失配元组
- right (outer) join :右外连接。结果包含右边关系的失配元组

- full (outer) join : 全外连接。结果包含两边关系的失配元组
- 视图：视图提供了一种向某些用户的视图隐藏某些数据的机制。任何不属于概念模型的关系，但作为“虚拟关系”对用户可见的关系称为视图。
 - create view view-name as <SQL 查询语句>
 - 视图可以由视图创建，编译时展开
 - 物化视图 create materialized view：定义视图的实际关系改变，视图也跟着修改
 - 优点：简化用户操作、不同角度观察
 - 大多数 SQL 实现只允许对简单视图进行更新
 - from 子句只有一个数据库关系。
 - select 子句仅包含关系的属性名称，没有任何表达式、聚合或不同的规范。
 - 任何未在 select 子句中列出的属性都可以设置为 null
 - 该查询没有 group by 或 having 子句。
 - 修改视图受限
 - （查询的）Select 子句不包括主码
 - Select 子句中用聚集函数或算术表达式构造的新属性：视图一个元组“来自”关系里的多个元组时（count等计算）
 - Select 子句存在distinct关键字
 - From 子句中有多个关系
 - 有Group By子句
 - 同时满足才可修改
 - Select 子句中包含主码
 - Select 子句中只出现原有属性，而没有用聚集函数或算术表达式构造的新属性
 - Select 子句中无distinct关键字
 - From 子句中只有一个关系
 - 无Group By子句
- 事务
 - commit work
 - rollback work
- 完整性约束
 - 实体完整性
 - 元组在主码的每个属性上取唯一值，且不能为空。
 - 元组之间相互区分
 - 如果一个关系的主码由多个属性构成，那么每个属性都不能取空值
 - 单一关系上的约束
 - not null：name varchar(20) not null

- `unique(A_1, A_2,...,A_m)`: 属性 A_1, A_2, \dots, A_m 构成候选码, 候选码可以为空。唯一性约束用来限制不受主键约束的列上的数据的唯一性, 即表中任意两行在 指定列上都不允许有相同的值。
 - primary key
 - `alter table reader add constraint PK_Reader primary key(reader_id)`
- 参照完整性: 如果关系R的外部码对应关系S的主码, 则R每个元组在外部码上的取值必须满足: 要么等于空值要么等于某个对应的主码值 (S某个元组的主码值)
 - 外键:
 - **create table** statement **foreign key** (dept_name) **references** department
 - 默认引用被引用表的主键属性
 - 参照动作: 当某个主码值被删除/更新时, 如何处理对应的外部码值
 - **restrict**方式: 仅当没有任何对应的外码值时, 才可以删除/更新这个主码值, 否则系统拒绝执行此操作
 - **cascade**方式: 连带将所有对应的外码值一块删除/更新 (删除外码值, 实际上就是将所在的元组删除掉)
 - 级联: 在级联删除中, 删除父表中的记录时, 同时删除子表中外键引用此主键的记录。
 - ```
create table course (
 (...
 dept_name varchar(20),
 foreign key (dept_name) references department
 on delete cascade
 on update cascade,
 ...)
```
    - **set null**方式: 将所有对应的外码值设为空值
- 用户定义完整性: 用户根据具体的应用环境定义, 保证取值合理
  - `check(P)`, P是谓词确保元组属性满足条件
    - `check` 子句中的谓词可以包含查询: **check** (time\_slot\_id **in** (**select** time\_slot\_id **from** time\_slot))
    - `alter table table-name add constraint constraint-name check(xxx)`
- 授权
  - **grant** <privilege list> **on** <relation or view > **to** <user list>
    - <user list>: user-id, public
  - **revoke** <privilege list> **on** <relation or view> **from** <user list>
  - read, insert, update, delete, all privileges
  - 修改数据库模式的授权形式
    - 索引 - 允许创建和删除索引。
    - 资源 - 允许创建新关系。
    - 变更 - 允许添加或删除关系中的属性。

- Drop - 允许删除关系。
- 创建外键的引用权限：grant reference (dept\_name) on department to Mariano;
- role
  - **create role** <name>
  - **grant** <role> to <users>
  - 角色是一种区分不同用户的方法，这些用户可以在数据库中访问/更新什么。
  - 可以将权限授予角色
  - 角色可以授予用户，也可以授予其他角色
  - 角色链
- 断言
  - **create assertion** <assertion-name> **check** (<predicate>);
  - 断言就是一个谓词（条件），施加在较大范围，比如可以是整个数据库的所有数据
    - 主码约束、检查约束等作用范围较小（一个元组或属性内部），可以看作是“小断言”
    - 如果作用范围比较大，比如约束几个关系之间必须满足什么条件，就要“大断言”
- 索引
  - 关系属性的索引是一种数据结构，它允许数据库系统有效地查找关系中具有该属性指定值的那些元组，而无需扫描关系的所有元组。
  - **create index** <name> **on** <relation-name> (attribute);
  - 索引是关系型数据库的一个基本概念。
  - 数据库的索引类似图书的索引，能够使数据库程序不用浏览整个表，就可以找到表中的数据。
  - 索引是一个表中所包含的值的列表，它说明了表中包含各个值的行所在的存储位置。
  - 用户可以利用索引快速访问数据库表中的特定信息。
  - 但使用索引存储地址将占用磁盘空间，同时在数据维护时，也将花费一定的时间。因此要合理设计索引。
- SQL 数据类型和模式
  - **date**: 日期，包含（4 位）年、月和日期
    - Example: **date** '2005-7-27'
  - **time**: 一天中的时间，以小时、分钟和秒为单位。
    - Example: **time** '09:00:30'      **time** '09:00:30.75'
  - **timestamp**（时间戳）: date plus time of day
    - Example: **timestamp** '2005-7-27 09:00:30.75'
  - **interval**（间隔）: period of time
    - Example: **interval** '1' day
    - Subtracting a date/time/timestamp value from another gives an interval value
    - 间隔值可以添加到日期/时间/时间戳值

- Large objects (照片、视频、CAD 文件等) are stored as a *large object*:
  - **blob**: 二进制大对象——对象是未解释的二进制数据的大集合（其解释留给数据库系统之外的应用程序）
  - **clob**: 字符大对象——对象是字符数据的大集合
- 当查询返回大对象时，返回的是指针而不是大对象本身。
- **create type** construct in SQL creates 创建用户定义的类型

```
create type Dollars as numeric (12,2) final
```
- **create domain** 构造创建用户定义的域类型

```
create domain person_name char(20) not null
```
- 类型和域是相似的。域可以在其上指定约束，例如不为空。
- Example:

```
create domain degree_level varchar(10)
constraint degree_level_test
check (value in ('Bachelors', 'Masters', 'Doctorate'));
```

以上内容整理于 [幕布文档](#)