9. SQL

SQL (Structured Query Language) 包含了三个子语言：

- 数据定义语言；
- 数据操作语言；
- 数据控制语言。

SQL 查询：SQL 查询是不改变数据库的，具有以下形式：

```
SELECT desired attribute list L FROM table names R, S, ... WHERE conditions C
and theta;
```

在 RA 模型中，即为 $\pi_L(\sigma_C(R \bowtie_\theta S))$。特殊情况下，单关系查询：

```
SELECT atrributes/expression-list L FROM R WHERE condition-C;
```

在 RA 模型中，即为 $\pi_L(\sigma_C(R))$。操作语义：R 看成元组变量，每次可以赋值为表中一个元组且遍历所有元组；对于表中的每个元组，检查是否满足 `WHERE` 后的式子，若满足则将 `SELECT` 后的属性输出。

- 如果 `SELECT` 后接 `*`，则表示所有属性；

  ```
  SELECT * FROM S;
  ```

- 可以在 `SELECT` 后接 `AS` （可选）来对属性名字重命名；

  ```
  SELECT sno as StuNumber FROM S;
  ```

- 可以用表达式来创建新的属性（最好在其后接 `AS` 进行重命名）；

  ```
  SELECT name, 2020 - age AS BirthYear FROM S;
  ```

- 也可以使用常量表达式来创建新的属性（可能在报表中有用）。

  ```
  SELECT name, 2020 - age AS BirthYear, 'A.D.' AS AD FROM S;
  ```

条件表达式：在 SQL 语言中，除了普通的条件表达式，也可以使用 `BETWEEN ... AND ...`，`IN`，`AND`，`OR`，`NOT` 等表达式来表达。如

```
age BETWEEN 20 AND 30
dept IN ('CS', 'EE', 'MA')
```

字符串运算：SQL 的字符串被单引号描述 `'str'`，字符串中的单引号用两个单引号 `''` 描述，如 `'I''m OK'`。

- 字符串的比较是字典许比较；
- 字符串拼接用 `||` 来操作，如 `str1 || str2`；
- **字符串匹配**：找到匹配规律 `p` 的字符串，用 `LIKE` 语句。
  - 规律 `p`：一个可包含 `%` 与 `_` 的字符串；其中，`%` 可匹配任意字符串；`_` 可匹配任意单个字符；其余字符匹配自身。
  - `s LIKE p` 为真当且仅当 `s` 能被 `p` 匹配；
  - `s NOT LIKE p` 为真当且仅当 `s` 不能被 `p` 匹配；
  - 使用 `ESCAPE` 语句来进行转义（需要用到 `%` 或 `_` 原意时），如

    ```
    s LIKE '#%%#%' ESCAPE '#'
    ```

日期和时间：SQL 支持 `DATE`，`TIME`，`TIMESTAMP` 类型，他们为常量，如

```
DATE '1931-09-18'
TIME '22:20:01.23'
TIMESTAMP '1931-09-18 22:20:01'
TIME WITH TIME ZONE '12:00:00+8:00'
```

可以用 `<, >, =` 等来比较时间的大小（早晚），用 `-` 表示时间的差。

空值：特殊的值为空值 `NULL`，可以用于任何一个类型，可能在插入、外连接等情况下创建。

- 空值出现在算术表达式中，结果一定是空值；空值出现在比较表达式中，结果为 `UNKNOWN`；
- 空值可能代表一些逻辑意义上恒等表达式不成立，如 `0 * x = 0`；
- NULL 不是一个常数，因此其不能被显示用于操作数，如不能使用 `NULL + 3` 等等。
- 判断某个数 `x` 是否为 `NULL` 不能使用 `x = NULL`，需要使用 `x IS [NOT] NULL`。

三值逻辑：具有三个值的逻辑，分别是 `TRUE`、`FALSE`、`UNKNOWN`。

- 逻辑运算规则：令 `TRUE = 1`，`FALSE = 0`，`UNKNOWN = 1/2`；于是 `AND` 为 $\min$，`OR` 为 $\max$，`NOT` 为 $1-x$。
- 在 `SELECT ... FROM ... WHERE` 中，一个元组被输出当且仅当 `WHERE` 后表达式为 `TRUE`（注意空值的问题），如假设 `zhao` 的年龄为空，那么如下操作将不会出现 `zhao`：

  ```
  SELECT name FROM S WHERE age < 18 OR age >= 18;
  ```

在结果上进行排序：在 `SELECT ... FROM ... WHERE` 操作中运用 `ORDER BY` 语句进行排序，一般在 `ORDER` 中用行序号进行表示（特别用于表达式计算得到的列）。对每个属性，可用 `ASC` 和 `DESC` 分别表示升序、降序，其中 `ASC` 可省略。故通用形式如下：

```
SELECT ... FROM ... WHERE ...
ORDER BY A1[[ASC] | DESC], A2[[ASC] | DESC], ...;
```

一些典型例子如下：

```
SELECT sno, grade FROM SC
WHERE cno = 'CS145'
ORDER BY grade DESC, sno;
SELECT name, dept, 2020 - age FROM S
ORDER BY 3, 1;
SELECT name FROM S
ORDER BY 2020-age;
```

多关系查询

- **多关系查询中的重名现象**：使用关系名 + 点 + 属性名来区分不同关系的属性。
- **引用两个或多个同一关系**：可以使用 `FROM ... AS ...` 来重命名（`AS` 可省略），如

  ```
  SELECT firstS.sno, secondS.sno
  FROM S firstS, S secondS
  WHERE firstS.name = secondS.name AND firstS.sno < secondS.sno;
  ```

- 多关系查询时，实际上有三种理解方式：循环嵌套枚举、并行赋值以及 RA 模型中的笛卡尔积。因此，可能带来一些背离直觉的结果。如：

  ```
  SELECT R.A FROM R, S, T
  WHERE R.A = S.A OR R.A = T.A;
  ```

  当 `T` 为空时，结果为空，并不是 $R \cap S$。原因是：

  - 循环嵌套枚举时，循环 T 迭代了 0 次；
  - 并行赋值时，T 不能被赋值；
  - RA 模型中，笛卡尔积为空。

集合论操作：可以采用 `UNION`，`INTERSECT` 以及 `EXCEPT` 分别表示并集、交集、对称差。

子查询：一个是其他查询的一部分的 `SELECT-FROM-WHERE` 查询，子查询同样可以有自身的子查询。一个简单的子查询例子为用集合论操作连接的两个子查询。子查询可以用在另一个查询的 `WHERE` 表达式中，并且其可以返回一个原子值、单个元组或一个关系；子查询同样可以用在 `FROM` 表达式中，一般返回的是关系并且需要换名。

- 标量子查询：如果一个查询产生了单个值，则可以被看成标量常量，即如下表达式成立：`v = (SELECT ... FROM ... WHERE)`；一般用括号来包围子查询。例如：

  ```
  SELECT name
  FROM S
  WHERE age = (SELECT age FROM s WHERE sno = 'S!');
  ```

  - **作用域规则**：每个属性指代的关系的是最近的包含该属性的关系；
  - 必须用括号将子查询包围住。

- 元组子查询：如果一个子查询仅产生了一个元组，则可以被看成单个元组，即如下表达式成立：`(v1, v2) = (SELECT ... FROM ... WHERE)`。
- 关系子查询：一般来说，子查询产生一个关系 `R`，即可以使用如下表达式：
  - `t IN R`，`t NOT IN R`，`NOT (t IN R)`：判断 `t` 是否在 `R` 中。
  - `EXISTS(R)`，`NOT EXISTS(R)`：判断 `R` 是否为空。
  - `t theta ALL R`，`t theta ANY R`（`t theta SOME R`）：量词 $\forall, \exists$，满足条件 `theta`。
    - 如果 $R = \varnothing$，则 `t theta ALL R` 为真，`t theta SOME R` 为假。

包含元组的查询：可以使用括号+逗号表示一个标量元组，如 `(123, 'foo') IN R`。

- 注意：成员属性必须匹配！

连接表达式：SQL 中提供了许多连接运算子来构造连接，如

```
R CROSS JOIN S
R JOIN S ON theta
R NATURAL JOIN S
R OUTER JOIN S
```

这些 `JOIN` 表达式可以被用在任何需要连接而成的关系的地方，同时本身也可以作为查询。

- 外连接可以在 `OUTER JOIN` 前加入可选的 `NATURAL` 选项表示自然外连接；外连接可以在 `OUTER JOIN` 后加入外连接的条件；外连接可以在 `OUTER JOIN` 前加入 `LEFT`、`RIGHT` 或 `FULL` 表示左外连接、右外连接以及完全外连接（默认为完全外连接）。