关系数据库语言 SQL

3.1 SQL 概述

3.1.1 SQL 的特点

1.综合统一
1）集多种语言功能为一体，
2）可以独立完成数据库生命周期中的全部活动。
3）用户在数据库系统投入运行后可以根据需要随时逐步地修改模式
4）数据操作符统一
2.高度非过程化
3.面向集合的操作方式
4.以同一种语法结构提供两种使用方式
5.语言简洁，易学易用

SQL 即是自含式语言（交互式），又是嵌入式语言

3.1.2 SQL 基本概念

1.基本表：基本表是独立存在于数据库中的表，**是“实表”**。
2.视图：视图是从一个或几个基本表（或视图）导出的表，**是“虚表”**，它本身不独立存在于数据库中，数据库**只存放视图的定义而不存放视图的数据**。
3.存储文件：数据库是逻辑的，存储文件是物理的。**一个基本表可以用一个或多个春初文件存储。**
4.索引：表中数据如按其输入的时间排放，这种顺序记为记录的物理顺序；如按某个或某些属性进行排序，这种顺序称为逻辑顺序。索引即是根据索引表达式进行逻辑排序的一组指针。索引是关系数据库的内部实现技术，属于内模式，被存放在存储文件里。
5.模式

3.1.3 SQL 语言的组成

三个方面：
1.数据定义语言 DDL
2.数据操纵语言 DML
3.数据控制语言 DCL

3.1.4 SQL 语句分类

按功能可分为 4 类：
1.数据定义

2.数据查询
3.数据操纵
4.数据控制

3.2 SQL 语言的数据类型

数值型：bit 只存 0 和 1
字符型：char(n):长度不足 n 会以空格填充；varchar(n)：有多长放多长。（使用 ASCII 字符集）长度限制为 8000。
Unicode 字符型：Unicode 是统一字符编码标准。有nchar 和 nvarchar
文本型：当需要存储大量字符数据时。有 text 和 ntext。
日期时间类型：
datetime：1752.1.1 到 9999.12.31
smalldatetime：1900.1.1 到 2079.6.6
二进制型：二进制数据类型表示位数据流。有 binary（不足用 0 填充）、varbinary，image
货币类型：money 对应 bigint，smallmoney 对应 int

3.3 数据定义

3.3.1 模式定义

模式定义即定义了一个存储空间。一个 SQL 模式由模式名、用户名或账号来确定

1.定义数据库：

```
CREATE DATABASE <数据库名>
```

2.使用数据库

```
USE <数据库名>
```

3.修改数据库

```
ALTER DATABASE <数据库名>
```

4.删除数据库

```
DROP DATABASE <数据库名>
```

3.2.2 基本表定义

1.定义基本表

```
CREATE TABLE <基本表名>
(
    <列名> <数据类型> <列级完整性约束>,
    ......,
    <列名> <数据类型> <列级完整性约束>,
    <表级完整性约束>,
    ......
    <表级完整性约束>
)
```

约束：
1.NOT NULL 非空约束，限制列取值不能为空
2.DEFAULT 默认值约束，指定列的默认值
3.UNIQUE 唯一性约束，限制列的取值不能重复
4.CHECK 检查约束，限制列的取值范围
5.PRIMARY KEY 主码约束，指定本列为主码
6.FOREIGN KEY 外码约束，指定本列为引用其他表的外码。格式为：FOREIGN KEY (<外码列名>) REFERENCE <外表名> (<外表列名>)
单行注释符：--
多行注释符：/* */

2.修改基本表

```
ALTER TABLE <基本表名>
ALTER COLUMN <列名> <新数据类型> --修改已有列定义
| ADD <列名> <数据类型> <约束> --增加新列
| DROP COLUMN <列名> --删除列
| ADD CONATRAINT <约束名> <约束定义> --添加约束
| DROP CONSTRANT <约束名> --删除约束
```

3.删除基本表

```
DROP TABLE <基本表名>
```

3.3.3 索引定义

建立索引是为了提高数据查询速度

索引分类：
1.聚簇索引：对表中的物理数据页的数据按索引关键字进行排序，然后重新存储到磁盘上，即**聚簇索引与数据是一体的，一个数据表只能建立一个聚簇索引**。

2.非聚簇索引：**具有完全独立于数据的索引结构**。它不将物理数据也中的数据按索引关键词排序，非聚簇索引不改变数据的物理存储位置，一张表上可建立多个非聚簇索引。

当索引关键字能保证其所包含的各列值不重复时，该索引是唯一索引。聚簇索引和非聚簇索引都可以是唯一索引。

使用频率高，更新频率低的列适合建立索引
建立索引：

```
CREATE [UNIQUE] [CLUSTERED] [NONCLUSTERED] INDEX <索引名> ON <基本表名> (<列名> [ASC | DESC], <列
```

删除索引

```
DROP INDEX <基本表名> <索引名>
```

视图定义

视图数从一个或多个基本表（或视图）导出的表，是根据用户观点所定义的数据结构。

优点：
1.为用户集中数据，简化用户的数据查询和处理
2.屏蔽数据库的复杂性
3.简化用户权限管理
4.便于数据共享
5.可以重新组织数据以便输出到其他应用程序中

注意事项：
1.只有在当前数据库中才能创建视图。
2.视图的命名必须遵循标识符命名规则，不能与表同名，且对每个用户视图名必须是唯一的，即对不同用户，即使是定义相同的视图，也必须使用不同的名字。
3.不能在视图上建立任何索引。

定义视图：

```
CREATE VIEW <视图名> [列名] AS <SELECT 查询语句>
```

视图不仅可以建立在一个或多个基本表上，也可以建立在一个或多个已有视图上

修改视图：

```
ALTER VIEW <视图名> [列名] AS <SELECT 查询语句>
```

删除视图：

```
DROP VIEW <视图名>
```

删除视图不会影响基本表的数据，但如果被删视图还导出其他视图，则对由其导出的视图执行操作会发生错误

3.4 数据查询

3.4.1 SELECT 语句结构

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

顺序：FROM　　WHERE　　GROUP BY　　聚合函数　　HAVING　　SELECT　　ORDER BY

3.4.2 单表查询

1.选择列
1）选择表中指定的列：加列名
2）选择表中全部列：*
3）查询经过计算的值
4）更改结果列标题：用 AS 时列名必须在等号的右边
5）替换查询结果中的数据：

```
CASE
    WHEN 条件 THEN 表达式
END
```

、6）去除重复列：DISTINCT，省略时默认值为 ALL

2.选择行
1）查询满足条件的元组
比较运算：均不为空值时，返回 TRUE 或 FALSE；有一个为空值或都为空值时，返回 UNKNOWN

指定范围：

```
<表达式> [NOT] BETWEEN <表达式1> AND <表达式2>
```

确定集合：IN

字符匹配：
%：代表任意长度的字符串
_：代表任意一个字符
LIKE 语句使用通配符的查询也成为模糊查询，没有通配符时与 = 等效。

空值比较：IS [NOT] NULL

逻辑运算：AND OR

3.对查询结果进行排序

```
ORDER BY <列名1> [ASC | DESC] [, <列名2> [ASC | DESC]]
```

4.聚合函数（不可用于 WHERE 后）

聚合函数是指对集合操作但只返回单个值的函数，规则：
1）带有一个聚合函数的 SELECT 语句仅产生一行作为结果
2）聚合函数不允许嵌套
3）聚合函数（除 COUNT(*) 外）处理单个列中全部所选的值以生成结果值
4）SUM、AVG、COUNT、MAX、MIN 忽略空值，而 COUNT(*) 不忽略

```
SUM | AVG ([ALL | DISTINCT] <表达式>)
MAX | MIN ([ALL | DISTINCT] <表达式>)
```

SUM、AVG、MAX、MIN 都适用以下规则：
1）如果给定行中的一列仅包含 NULL 值， 则函数的值等于 NULL
2）如果一列某些值为 NULL， 跳过不计入
3）对于必须计算的 SUM 和 AVG 函数，如果中间结果为空，该函数的值为 NULL

```
COUNT (\{[ALL | DISTINCT] <表达式>\} | *)
```

COUNT(*) 时统计总行数，COUNT（） 用于计算非 NULL值数量

5.对查询结果分组

GROUP BY 用于将查询结果表按某一列或多列值进行分组，值相等的为一组。
特别注意：使用 GROUP BY 子句后，SELECT 字句列表中只能包含 GROUP BY 中知指出的列或在聚合函数中指出的列。

6.使用 HAVING 子句进行筛选

在使用 GROUP BY 子句分组后，还需要按条件进一步对这些组进行筛选，最终只输出满足指定条件的组。

```
HAVING <查询条件>
```

3.4.3 连接查询

若一个查询同时涉及两个或两个以上的表，则称为连接查询。连接查询是二元运算，类似于关系代数中的连接操作。

两种方式：
1.采用连接谓词
2.采用关键字 JOIN

采用连接谓词：
当 SELECT 语句中的 WHERE 子句中查询条件使用了比较为此或指定范围谓词，所涉及的列来源于两个或两个以上的表时，则改 SELECT 查询将设计多个表，即为连接查询。

当谓词为 = 时，就是等值连接；若在目标列中去除相同的字段名，则为自然连接。

以 JOIN 关键字指定的连接：
在 FROM 子句的扩展定义中 INNER JOIN 表示内连接， OUTER JOIN 表示外连接。

内连接：内连接按 ON 所指定的连接条件合并两个表，返回满足条件的行。
内连接是系统默认的。

多个 JOIN 连接时 ON 条件写的顺序应为从右向左，即各表出现的顺序应与 JOIN 部分各表出现的顺序相反。

外连接： LEFT OUTER JOIN, RIGHT OUTER JOIN

3.4.4 嵌套查询

上层查询块称为外层查询或父查询，下层查询块称为内查询或子查询。
子查询的 SELECT 语句不能包含 ORDER BY，ORDER BY 只能对最终查询结果进行排序。

带 IN 谓词的子查询：
在嵌套查询中，子查询的结果往往是一个集合，所以 IN 是嵌套查询中最常使用的谓词。
有些嵌套查询可以用连接查询替代。

子查询的查询条件不依赖于父查询，称为不相关子查询；反之称为相关子查询。

带比较运算符的子查询：
比较子查询是指父查询与子查询之间用比较运算符进行关联。如果能够确切的知道子查询返回的是单个值，就可以使用比较子查询。

处理不相关子查询时，可以先将子查询一次处理完成，然后再处理父查询，而处理相关子查询时，由于子查询的条件与父查询有关，因此必须反复求值。

带 ALL(SOME) 或 ANY 谓词的子查询：
当子查询返回多个值时，若父查询需要与子查询的返回结果进行比较，则不能直接使用比较运算符，而必须在比较运算符之后加上 ALL(SOME) 或 ANY 进行限制。

```
<表达式> <比较运算符> {ALL | ANY | SOME} (子查询)
```

ANY 在子查询返回 NULL 时，返回 FALSE；ALL 则返回 TRUE。

通常使用聚合函数实现子查询比直接用 ANY 或 ALL 查询效率高。

4.带 EXISTS 谓词的子查询：
EXISTS 谓词用于策实子查询的结果是否为空表。

```
[NOT] EXISTS (子查询)
```

3.4.5 集合查询

多个 SELECT 语句的结果集可以进行集合操作。
并（UNIOB）、交（INTERSECT）、差（EXCEPT）

3.4.6 视图查询

定义视图后，就可以如同查询基本表那样对视图进行查询
对视图查询时，首先进行有效性查询，检查查询的表、视图是否存在。如果存在，那么从系统表中取出对视图的定义，然后执行转换以后的查询。

3.5 数据更新

包括增（INSERT）、删（DELETE）、改（UPDATE）

3.5.1 数据插入

1.插入元组

```
INSERT INTO <表名> [(<列名1>[, <列名2>, ···])]
VALUES (<常量1> [, <常量2>, ···])
```

如果某些列在 INTO 子句中没有出现，则新元组在这些列上的值将取空值 NULL，但如果在定义表时说明了属性列不能取空值，则必须指定一个值。如果 INTO 子句后没有指明任何列，则新插入的元组必须为表的每个列赋值，列赋值的顺序与创建表时列的默认顺序相同。

2.插入子查询结果
子查询可以用在 INSERT 语句中，将生成的结果集插入到指定的表中。

```
INSERT INTO <表名> [(<列名1>[, <列名2>, ···])]
<子查询>
```

INTO 组剧中的列数要和 SELECT 子句中的表达式个数已知，数据类型也要一致。
DBMS 在执行插入语句时检查所插元组是否破坏表上已定义的完整性规则。

3.5.2 数据修改

```
UPDATE <表名> [[AS] <别名>]
    SET <列名> = <表达式> [, <列名> = <表达式>]
    [WHERE <条件表达式>]
```

修改指定表中满足 WHERE 子句指定条件的元组。其中 SET 子句给出需要修改的列及其新值。

DBMS 在执行修改语句时检查所插元组是否破坏表上已定义的完整性规则：1.实体完整性。2.主码不允许修改。3.用户定义的完整性。

3.5.3 数据删除

```
DELETE [FROM] <表名>
    [WHERE <条件表达式>]
```

从指定表中删除满足条件的元组，若省略 WHERE 子句，表示删除表中所有行，但不删去表的结构、定义。DROP 删去表的内容和定义。

3.5.4 视图更新

由于视图是不实际存储数据的虚表，因此对视图的更新最终要转换为对基本表的操作。
为了防止用户通过视图对数据进行增加、删除和修改时对不属于视图范围内的基本表数据进行操作，可以在定义视图时加上 WITH、CHECK、OPTION 子句。

3.5.5 更新操作与数据完整性