

数据库原理和应用

Kuladmir Present

V 1.0.1

2022-12-31

数据库原理和应用

目录/Contents

1 数据库相关概念-----	1
2 关系数据库模型-----	4
3 使用 SQL 数据库和表-----	5
4 存储过程与函数-----	22
5 Mysql 触发器-----	24
6 事务与并发控制-----	26
7 数据备份和还原-----	27
8 数据库设计方法-----	28

1 数据库相关概念

1.1 数据和数据库

数据是描述事物符号的记录。包含数值、文字、图像、图形、语音等。

数据库是以一定格式进行组织数据的集合长期存贮在计算机内的，可共享、大量数据的集合，具有共享性、独立性、完整性、数据冗余少的特点。

1.2 其他概念

数据库管理系统 (DBMS)：系统软件，可以进行数据定义、数据操作、数据库的运行和管理、数据库的建立和维护。包含：数据库文件集合(用于存储数据)，数据库服务器（负责对数据文件和文件中的数据进行管理），数据库客户端（负责和服务端通信，向服务器传输数据和获取数据）

数据库系统 (DBS)：是由数据库及其管理软件组成的系统。数据库系统是为适应数据处理的需要而发展起来的一种较为理想的数据处理系统，也是一个为实际可运行的存储、维护和应用系统提供数据的软件系统，是存储介质、处理对象和管理系统的集合体。

1.3 SQL 语句和数据库举例

SQL 语句是结构化查询语句，RDBMS 是用来管理关系型数据库的系统。
Oracle 数据库：有较好的开放性、高性能、安全性，但价格较贵。MS SQL Server 数据库：在微软的项目中使用，跨平台性差。MySQL 数据库。SQLite 数据库：轻量级，主要在移动端使用。

1.4 数据管理技术发展阶段

人工管理阶段 -> 文件系统管理阶段 -> 数据库系统阶段

数据库阶段特点：数据结构化（数据库的数据描述数据时，不仅要描述数据本身，也要描述数据之间的关系），数据共享性高、冗余度低、易扩展，数据独立性高（物理独立性、逻辑独立性），由 DBMS 统一管理和控制。

1.5 数据库系统的组成

- 1) 硬件平台与数据库：需要足够大的内存、硬盘、通道能力；
- 2) 软件：DBMS 为核心的应用开发工具；
- 3) 人员：DBA、SA、数据库设计员、程序员、用户等；

DBA 的职责：决定数据库中的信息内容和结构、数据库的存储结构和存取策略、定义数据的安全性要求和完整性约束条件、监控数据库的使用和维护。

1.6 数据模型相关概念

1.6.1 数据模型

数据模型：用来抽象表示和处理现实世界中的数据和信息，以便采用数据库技术对数据进行集中的管理和应用，是对客观事物及其联系的数据描述。具有三

类完整性：**实体完整性、参照完整性、用户自定义完整性**。数据模型分为：层次、网状、关系、面向对象模型。

1.6.2 模型层次

分为：概念模型、数据模型。二者间的关系可简单表达为：现实世界 -（认识抽象）> 概念模型 -> DBMS 支持的数据模型

概念模型：也称为信息模型，是根据用户的观点对数据和信息进行建模。

数据模型：按照计算机的观点将数据模型化，是计算机对数据间的结构、关系和操作的描述。

1.6.2.1 信息世界

实体：客观存在并可相互区别的事物或抽象的概念或联系等；

具有属性：实体具有的某特征，可由多个属性描述实体；

码：唯一标识实体的最小属性集，也称为关键字；

域：属性的取值范围；

实体型：有相同属性的实体必然具有共同的特征和性质，用实体名和其属性名集合来抽象描述同类实体；

实体集：同型实体的集合；

联系：现实世界中，实体内部和实体之间的联系；信息世界中，实体型内部的联系和实体型之间的联系。包含：一对一关系（省、省份）、一对多关系（医生、患者）、多对多（课程、商品）；

1.6.2.2 概念模型

用以信息世界的建模，和 DBMS 无关。需要较强的语义表达能力，能够方便、直接表达应用的语义知识，简单、清晰、易理解。

概念模型表示方法：

1) 实体-联系方法，绘制 E-R 图表述现实世界。E-R 图提供了表示的相关方法：

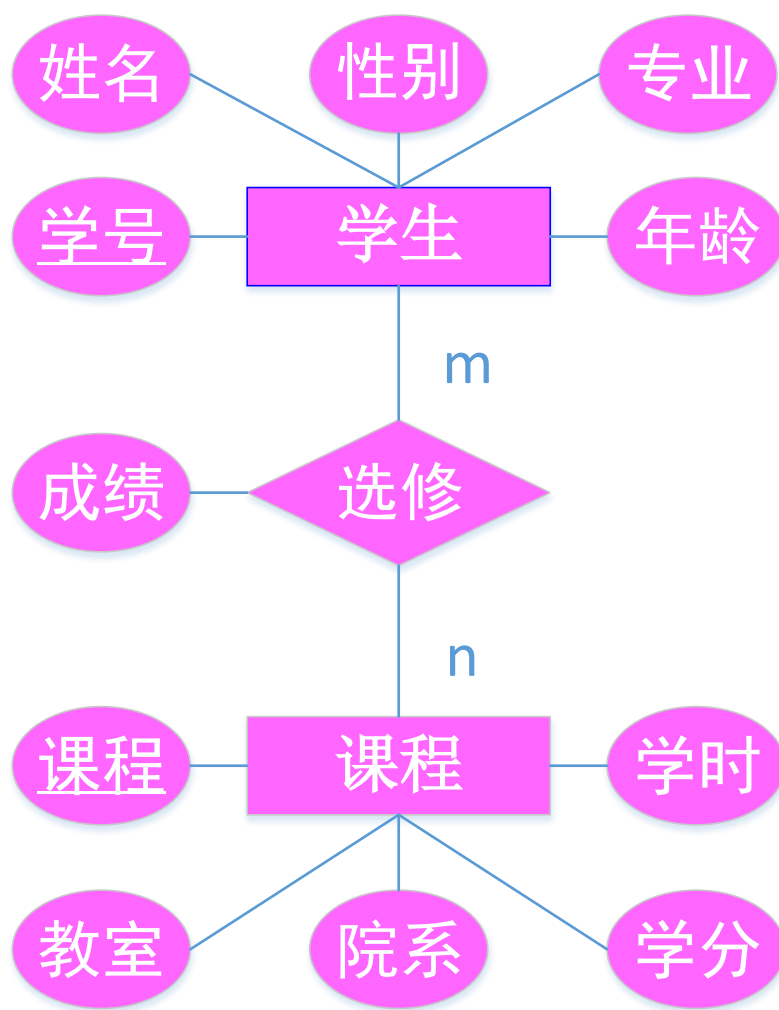
1.实体性：用矩形表示，需注明实体名；

2.属性：用椭圆表示，用不带箭头的线与其对应实体连接；

3.联系：用菱形表示，需注明联系名，并用不带箭头的线与其对应实体连接，在该线边上标注联系类型；

E-R 图设计原则：属性存在且只能存在某个地方，以避免冗余；实体是个单独的个体，不能存在于另一个实体中称为属性，避免“表中表”；同一个实体在同一个 E-R 图只能存在一次。

E-R 设计步骤：划分确定实体、划分确定联系、确定属性、画 E-R 模型、优化 E-R 模型。



1.6.3 三级模式

外模式是逻辑模式的子集，是对所有用户都能看见和使用的局部数据的逻辑结构和特征的描述。一个数据库可以有多个外模式，外模式保证了数据库的安全性。

内模式也称为存储模式，一个数据只有一个，是对数据的物理结构和存储方式的描述。

逻辑模式又称为模式，由数据库设计者综合所有用户的数据，按照统一观点构造全局逻辑模式，是数据库中全体数据的逻辑结构和特征的描述。一个数据库只有一个模式，不涉及具体硬件和存储细节，与具体应用无关。

1.6.4 三种独立性

数据独立性：逻辑独立性和物理独立性；

逻辑独立性：外模式和模式之间的映射；

物理独立性：内模式和模式之间的映射。

数据库常见运行和应用结构：

B/S(Browser/Server) 浏览器/服务器 结构。

C/S(Client/Server) 客户/服务器 结构。

1.6.5 二级映射

外模式/模式映射：模式描述了数据的全局逻辑结构，外模式描述了数据的局部逻辑结构。

内模式/模式映射：唯一，定义了数据库全局逻辑结构与存储结构之间的对应关系。

2 关系数据库模型

2.1 关系数据模型

基本术语：

1.关系：关系是一个属性数目相同的元组的集合，是一个由行、列组成的二维表；

2.元组和属性：表中的一行为元组，表中的每列为属性，给每个属性起名，即为属性名，也可称为字段名、列名；

3.码：能够唯一确定表中一个元组的属性，是整个关系的性质，而不是单个元组的形式。包含：超码（一个或多个属性的集合，关系中能够唯一标识元组）、候选码（最小的码，其任意真子集都不能称为超码，若一个关系中存在多个候选码时，通常选定一个码为主码）、主码、全码（若候选码包括全部属性，则这个候选码为全码）。

4.域：属性的取值范围；

5.分量：元组中的一个属性值为分量；

6.非主属性、主属性：关系中包含任意一个候选码中的属性为主属性，不包含再任意一个候选码的属性为非主属性。

关系模型的理论基础：集合论和数理逻辑。关系模型的结构：用户角度看，关系模型的数据结构是二维表，由行、列组成。

7.关系模型的型和值：

型：对某一类型数据的结构和属性的说明，值是型的具体复制，模式反映了数据结构和联系，实例反映了数据库某个时刻的状态。

8.关系数据库：是相互关联的表或关系的集合。

9.关系模型操作：

数据查询：数据检索、统计、排序、分组；

数据维护：数据添加、删除、修改；

数据控制：为保证数据安全性和完整性采用的数据存取控制。（授权/回收）

10.关系操作语言：包含关系代数语言、关系演算语言、具有关系代数和关系演算之间的语言。

11.关系模型的数据完整性：完整性规则是为保证关系中的数据正确、一致和有效。

12.完整性规则包含：实体完整性规则（要求关系中元组和组成主码的属性不能为空，如果为空，主码就不能保证唯一标识元组），参照完整性规则，取值规则（若属性和属性组 F 是基本关系 R 的外码，它与另一基本关系 S 的主码 K 对应，则对于关系 R 中每个元组在 F 上的取值必须为：空值、等于 S 中某个元组的主码值）

13.外码：设 F 是基本关系 R 的一个（组）属性，但不是 R 的码，如果 F 与基本关系 S 的主码 K 相对应，则 F 是 R 的外码，并称 R 为参照关系，S 为被参照关系。

示例参考：课程（课程号、学分、课程名） 选修（学号、课程号、成绩）

2.2 关系相关概念

关系的性质：同一属性的数据有同质性，属性名具有不能重复性、列的位置有顺序无关性，关系具有元组无重复性，元组的位置具有顺序无关性、关系中每个分量必须取原子值。

关系模式：一般表示为：关系名（属性 1，属性 2，属性 n），如果某个属性名或属性组为主码，需要下划线标明。

3 使用 SQL 数据库和表

3.1 SQL 基本知识

SQL 语言集数据查询、数据操纵、数据定义、数据控制功能融为一体。

特点：综合统一、高度非过程化、同一种语法结构（提供两种方式：自含式和嵌入式）、语言简洁、易学易用。

3.2 数据库基本操作

创建数据库

语法结构：

CREATE DATABASE/SCHEMA [IF NOT EXISTS] 数据库名称 [DEFAULT CHARACTER SET 字符集][DEFAULT COLLATE 校验规则];

注意：数据库名称必须符合操作系统文件名的命名规则。[]中内容为选加。

示例：CREATE DATABASE IF NOT EXISTS db_shop;

查看数据库

语法结构：SHOW DATABASES;

示例：SHOW DATABASES;

使用数据库

语法结构：USE 数据库名；

示例：USE db_shop；

修改数据

语法结构：

ALTER DATABASE/SCHEMA 数据库名称 [DEFAULT CHARACTER SET 字符集][DEFAULT COLLATE 校验规则]；

示例：将 studentinfo 数据库的字符集和校验规则从：gb2312、gb2312_chinese_ci 改为 utf8、utf8_general_ci。

ALTER DATABASE studentinfo DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci；

数据库删除//会直接全部清除

语法结构：DROP DATABASE [IF EXISTS] 数据库名；

示例：DROP DATABASE IF EXISTS dp_shop；

3.3 数据库的表定义和查询

DBMS 中常见的数据类型：数值型（整数 INT、浮点型 FLOAT、DOUBLE、定点数类型 DECIMAL）。日期时间类型（日期 DATE[YYYY-MM-DD]、时间 TIME[HH::MM::SS]、）。字符串类型（普通文本字符串类型 CHAR[固定长度]、VARCHAR、可变类型 TEXT[适合存储长文本]、BLOB[适合存储二进制数据]、特殊类型 SET[一个集合中取多个值]、ENUM[一个集合中取一个值]）

数字类型选择原则：<127 建议 TINYINT；全是整数，建议 INT；金额货币等，建议 DECIMAL。字符串类型选择原则：在满足应用的前提下，尽量使用短的数据类型，数据类型越简单越好；采用精度小数类型；对于时间日期，建议使用时间日期类型；尽量避免 NULL。

3.3.1 使用 SQL 定义数据表（原则：一个表只围绕一个主题，并使其容易维护。）

创建表

语法结构：CREATA TABLE [IF NOT EXISTS]表名(字段名 1 数据类型, 字段名 2 数据类型, 字段名 3 数据类型, 字段名 n 数据类型)；

示例：CREATE TABLE `speciality` (`Zno` char(4), `Zname` varchar(50)) ENGINE = InnoDB //存储引擎

查看表

语法结构：SHOW TABLES；

查看表的基本结构: DESC | DESCRIBE 表名;

查看表的详细信息: SHOW CREATE TABLE 表名;

示例: **SHOW TABLES; DESC student; SHOW CREATE TABLE student;**

修改表

【修改表名】

语法结构: ALTER TABLE 表名 RENAME TO 新表名;

RENAME TABLE 表名 to 新表名;

示例: **RENAME TABLE student to students;**

【修改字段数据类型】 可以同时修改多个字段

ALTER TABLE 表名 MODIFY 字段名 修改后的新类型 DEFAULT;

示例: **ALTER TABLE student MODIFY Ssex enum('男','女') DEFAULT '男';**

【修改字段名称】 可以同时修改字段和字段数据类型

语法结构: ALTER TABLE 表名 CHANGE 字段名 新字段名;

示例: **ALTER TABLE student CHANGE Ssex New_sex;**

【增加字段】

语法结构: ALTER TABLE 表名 ADD 新字段名;

示例: **ALTER TABLE student ADD create;**

【删除字段】

语法结构: ALTER TABLE 表名 DROP 字段名;

示例: **ALTER TABLE student DROP create;**

【修改外键】

语法结构: ALTER TABLE 表名 DROP FOREIGN key 字段名;

示例: **ALTER TABLE student DROP FOREIGN key Zno;**

【复制表】

语法结构: CREATA TABLE 表名 LIKE 表名;

示例: **CREATA TABLE students LIKE student;**

删除表//会直接全部清除

语法结构: DROP TABLE 表名;

示例: **DROP TABLE student;**

3.3.2 单表查询

单表查询: 从一张表中查询需要的数据。

【查询结构】

语法结构: SELECT 列表 FROM 表或视图 [WHERE 条件表达式];

关键字补充： **SELECT** 字句，用以指定查询列的名称，不同列之间需要,隔开，也可以给列起别名。**ALL** 关键字表示显示所有的行，包括重复的行。

[DISTINCT]表示显示结果要消除重复的行，放在 **SELECT** 后。**FROM** 指定要查询的表，可以指定多个表，不同表之间需要,隔开，也可以给列起别名。

[WHERE 条件表达式]：指定查询条件，如果没有，则查询所有的行。**[GROUP**

BY 列名]：用以对查询结果进行分组。**[HAVING 条件表达式]：**指定分组的条件，常放在**[GROUP BY]**之后。

[ORDER BY 列名]：用以对结果进行排序，有

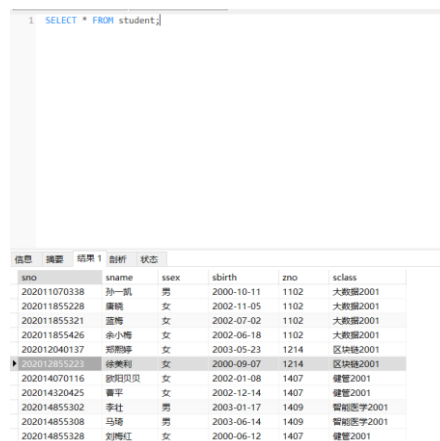
ASC（升序）、**DESC**（降序）两个参数指定。**[LIMIT 子句]：**限制查询的输出

结果数量。**[AS 新字段名]：**用以给字段起别名，放在字段后使用，可以省略。

示例 1：全列字段查询

SELECT sno,sname,ssex,sbirth,zno,sclass FROM student;

SELECT * FROM student;



The screenshot shows a SQL query editor with the command `1 SELECT * FROM student;` and its results. The results are displayed in a table with columns: sno, sname, ssex, sbirth, zno, and sclass. The data includes 12 rows of student information.

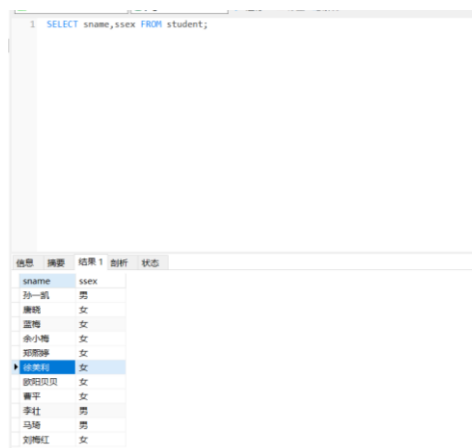
sno	sname	ssex	sbirth	zno	sclass
202011070338	孙一凯	男	2000-10-11	1102	大数据2001
202011852228	康晓	女	2002-11-05	1102	大数据2001
202011855321	蓝梅	女	2002-07-02	1102	大数据2001
202011855426	余小梅	女	2002-06-18	1102	大数据2001
202012040137	郑晓琳	女	2003-05-23	1214	区块链2001
202014070116	修美利	女	2000-09-07	1214	区块链2001
202014070116	修美利	女	2000-09-07	1214	区块链2001
202014070116	修美利	女	2000-09-07	1214	区块链2001
202014320425	曹平	女	2002-12-14	1407	键智2001
202014855302	李杜	男	2003-01-17	1409	智能数学2001
202014855308	马瑞	男	2003-06-14	1409	智能数学2001
202014855328	刘梅红	女	2000-06-12	1407	键智2001

示例 2：查询指定字段

SELECT sno,sname FROM student; //查询学号、姓名

SELECT sclass FROM student; SELECT DISTINCT sclass FROM student;

//后者可以消除重复的情况

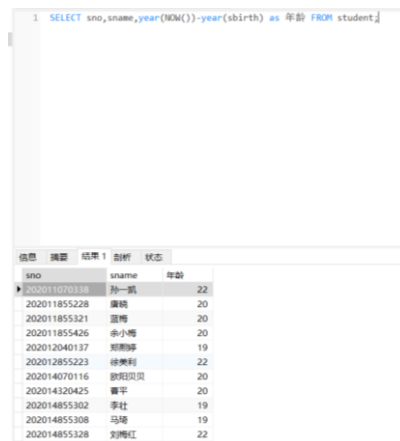


The screenshot shows a SQL query editor with the command `1 SELECT sname,ssex FROM student;` and its results. The results are displayed in a table with columns: sname, ssex. The data includes 12 rows of student information, showing only the name and sex columns.

sname	ssex
孙一凯	男
康晓	女
蓝梅	女
余小梅	女
郑晓琳	女
修美利	女
修美利	女
修美利	女
曹平	女
李杜	男
马瑞	男
刘梅红	女

示例 3：给表和字段取别名

SELECT sno,sname,year(NOW())-year(sbirth) AS 年龄 FROM student;

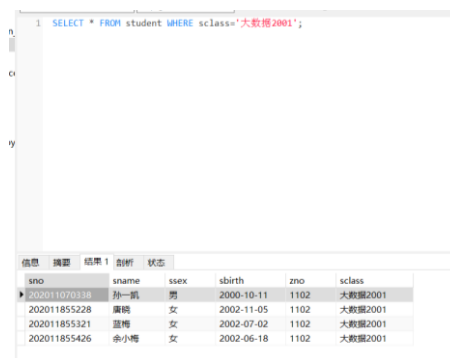


The screenshot shows a SQL query editor with the following query: `1. SELECT sno,sname,year(NOW())-year(sbirth) as 年龄 FROM student;` Below the query, a table of results is displayed with columns: sno, sname, and 年龄. The results list 12 students with their IDs, names, and calculated ages.

sno	sname	年龄
202011070338	孙一帆	22
202011855228	康晓	20
202011855321	蓝梅	20
202011855426	余小梅	20
202012040137	张雨婷	19
202012855223	徐美利	22
202014070116	张阳贝贝	20
202014320425	霍平	20
202014855302	李壮	19
202014855308	马晓	19
202014855328	刘梅红	22

示例 4：条件查询

SELECT * FROM student WHERE sclass='大数据 2001';



The screenshot shows a SQL query editor with the following query: `1. SELECT * FROM student WHERE sclass='大数据2001';` Below the query, a table of results is displayed with columns: sno, sname, ssex, sbirth, zno, and sclass. The results list 4 students from the '大数据2001' class.

sno	sname	ssex	sbirth	zno	sclass
202011070338	孙一帆	男	2000-10-11	1102	大数据2001
202011855228	康晓	女	2002-11-05	1102	大数据2001
202011855321	蓝梅	女	2002-07-02	1102	大数据2001
202011855426	余小梅	女	2002-06-18	1102	大数据2001

SELECT * FROM student WHERE sclass='智能感知 2001'AND ssex='女';



The screenshot shows a SQL query editor with the following query: `1. SELECT * FROM student WHERE sclass='智能感知2001' AND ssex='女';` Below the query, a table of results is displayed with columns: sno, sname, ssex, sbirth, zno, and sclass. The results list 1 student from the '智能感知2001' class who is female.

sno	sname	ssex	sbirth	zno	sclass
202011855232	王慧霞	女	2002-07-20	1805	智能感知2001

SELECT sname,ssex,sclass FROM student WHERE sclass IN("智能感知 2001","智能医学 2001");

```
1 SELECT sname,ssex,sclass FROM student WHERE sclass IN("智能感知2001","智能医学2001");
```

信息	摘要	结果 1	剖析	状态
sname	ssex	sclass		
李江	男	智能医学2001		
马峰	男	智能医学2001		
王松	男	智能医学2001		
李老旭	男	智能感知2001		
王翠雷	女	智能感知2001		

SELECT sno,grade FROM sc WHERE grade BETWEEN 70 AND 85;

```
1 SELECT sno,grade FROM sc WHERE grade BETWEEN 70 AND 85;
```

信息	摘要	结果 1	剖析	状态
sno	grade			
202014855328	85.0			
202014855406	75.0			
202012855223	77.0			
202014855406	84.0			

SELECT * FROM specialty WHERE zname IS NOT NULL;

```
1 SELECT * FROM specialty WHERE zname IS NOT NULL;
```

信息	摘要	结果 1	剖析	状态
zno	zname			
1102	数据科学与			
1103	人工智能			
1201	网络与新媒体			
1214	区块链工程			
1407	健康服务与			
1409	智能医学工			
1601	供应链管理			
1805	智能感知工			
1807	智能装备与			

示例 5：相似查询

SELECT * FROM student WHERE sname LIKE '李_';

SELECT * FROM student WHERE sname LIKE '李%';

```
1 SELECT * FROM student WHERE sname LIKE '李%';
```

信息	摘要	结果 1	分析	状态	
sno	sname	ssex	sbirth	zno	sclass
202014855302	李杜	男	2003-01-17	1409	智能医学2001
202018855212	李冬旭	男	2003-06-08	1805	智能感知2001

SELECT * FROM course WHERE cname LIKE '%数据%';

1	SELECT * FROM course WHERE cname LIKE '%数据%';
---	---

信息	摘要	结果 1	分析	状态
cno	cname	ccredit	cdept	
11110140	大数据管理	3	人工智能学院	
11110470	数据分析与可视化	3	人工智能学院	
18111850	数据库原理	3	大数据学院	
18132220	数据库技术及应用	2	大数据学院	
18132600	数据库原理与应用A	3	大数据学院	
58130540	大数据技术及应用	3	大数据学院	

补充：*可用以替代所有的列名，查询结果会按照你指定的顺序产生。条件查询中的符号：**= < > <= >= !=**，匹配字符：**LIKE**（在查询时，**_**表示任意一个字符，**%**表示任意个字符）、**NOT LIKE**，指定范围（连续）：

BETWEEN.....AND.....（包含端点）、**NOT BETWEEN.....AND.....**，指定范围（离散）**IN**，是否为空：**IS NULL**、**IS NOT NULL**。逻辑运算符：**AND**、**OR**、**!**。

【排序和限量】

排序语法结构：**ORDER BY 列名 ASC|DESC**（ASC 升序，DESC 降序）

示例：**SELECT * FROM student ORDER BY sbirth DESC;**

限量语法结构：**LIMIT [offset] row_count**

示例：**SELECT * FROM student ORDER BY sbirth ASC LIMIT 0,4;**

说明：**最前面的数字表示开始位置，后面的数字为显示的条数。**

```
1 SELECT * FROM student ORDER BY sbirth ASC LIMIT 8,4;
```

sno	sname	ssex	sbirth	zno	sclass
202014451222	刘海江	女	2000-06-12	1407	建筑2001
202012853223	孙奕利	女	2000-09-07	1214	区块链2001
202011070338	孙一帆	男	2000-10-11	1102	大数据2001
202016855313	郭奕	女	2001-02-14	1601	供应链2001

3.3.3 聚合函数和分组查询

3.3.3.1 聚合函数

COUNT() 统计非 NULL 值的数量。**SUM()** 求和。**AVG()**求平均数。**MAX()**、**MIN()**求最值。

示例: **SELECT COUNT(sno),SUM(grade),MAX(grade),MIN(grade) FROM sc;**

```
1 SELECT COUNT(sno),SUM(grade),MAX(grade),MIN(grade) FROM sc;
```

COUNT(sno)	SUM(grade)	MAX(grade)	MIN(grade)
14	1151.0	96.0	60.0

3.3.3.2 分组查询

语法结构: **GROUP BY** 字段名 **HAVING** 条件表达式 [with rollup]

示例:

SELECT sclass,count(sno) FROM student GROUP BY sclass;

SELECT sclass,count(sno) FROM student GROUP BY sclass HAVING count(sno)>=3;

SELECT sclass,ssex,count(sno) FROM student GROUP BY sclass,ssex WITH ROLLUP;

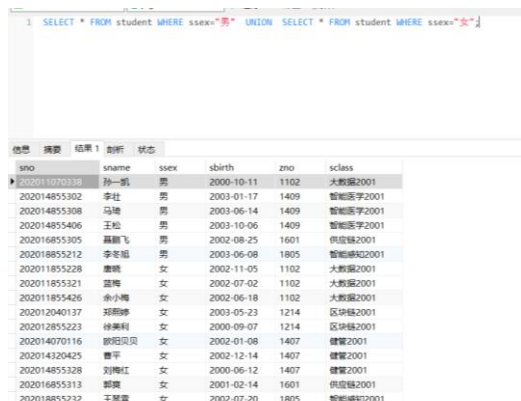
```
1 SELECT sclass,ssex,count(sno) FROM student GROUP BY sclass,ssex WITH ROLLUP;
```

sclass	ssex	count(sno)
供应链2001	男	1
供应链2001	女	1
供应链2001	(Null)	2
建筑2001	女	3
建筑2001	(Null)	3
区块链2001	女	2
区块链2001	(Null)	2
大数据2001	男	1
大数据2001	女	3
大数据2001	(Null)	4
智能医学2001	男	3
智能医学2001	(Null)	3
智能感知2001	男	1
智能感知2001	女	1
智能感知2001	(Null)	2
(Null)	(Null)	16

3.3.3.3 合并数据查询结果

语法结构：查询语句 1 union 查询语句 2;

示例：**SELECT * FROM student WHERE ssex="男" UNION SELECT * FROM student WHERE ssex="女";**



The screenshot shows a SQL query window with the following query: `1 SELECT * FROM student WHERE ssex="男" UNION SELECT * FROM student WHERE ssex="女";`. Below the query, a table displays the results. The table has columns: sno, sname, ssex, sbirth, zno, and sclass. The results are a union of two sets of data, with the first set having ssex='男' and the second set having ssex='女'.

sno	sname	ssex	sbirth	zno	sclass
202011070338	孙一凯	男	2000-10-11	1102	大数据2001
20201485302	李杜	男	2003-01-17	1409	智能医学2001
20201485308	马建	男	2003-06-14	1409	智能医学2001
20201485306	王松	男	2003-10-06	1409	智能医学2001
20201685305	蔡鹏飞	男	2002-08-25	1601	供应链2001
20201885212	李冬旭	男	2003-06-08	1805	智能感知2001
202011855228	廖晓	女	2002-11-05	1102	大数据2001
202011855321	田梅	女	2002-07-02	1102	大数据2001
202011855426	余小梅	女	2002-06-18	1102	大数据2001
202012040137	郑丽娟	女	2003-05-23	1214	区块链2001
202012855223	徐顺利	女	2000-09-07	1214	区块链2001
202014070116	郭阳贝贝	女	2002-01-08	1407	健康2001
202014320425	曹平	女	2002-12-14	1407	健康2001
202014855328	刘梅红	女	2000-06-12	1407	健康2001
202016855313	郭爽	女	2001-02-14	1601	供应链2001
202018855232	王琴露	女	2002-07-20	1805	智能感知2001

3.3.4 多表查询

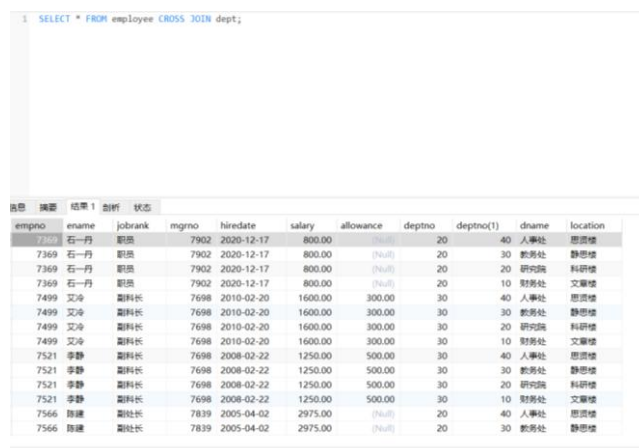
1) 交叉表查询 (CROSS JOIN): 对两个或多个表中的数据进行笛卡儿积操作。

语法结构：SELECT * FROM 表 1 CROSS JOIN 表 2;

示例：

SELECT * FROM employee CROSS JOIN dept;

SELECT * FROM employee,dept;



The screenshot shows a SQL query window with the following query: `1 SELECT * FROM employee CROSS JOIN dept;`. Below the query, a table displays the results. The table has columns: empno, ename, jobrank, mgrno, hiredate, salary, allowance, deptno, deptno(t), dname, and location. The results are a cross join of the employee and dept tables, showing all possible combinations of rows from both tables.

empno	ename	jobrank	mgrno	hiredate	salary	allowance	deptno	deptno(t)	dname	location
7369	石一丹	职员	7902	2020-12-17	800.00	[null]	20	40	人事处	唐溪楼
7369	石一丹	职员	7902	2020-12-17	800.00	[null]	20	30	教务处	静思楼
7369	石一丹	职员	7902	2020-12-17	800.00	[null]	20	20	研训处	科研楼
7369	石一丹	职员	7902	2020-12-17	800.00	[null]	20	10	财务处	文豪楼
7499	艾冲	副科长	7698	2010-02-20	1600.00	300.00	30	40	人事处	唐溪楼
7499	艾冲	副科长	7698	2010-02-20	1600.00	300.00	30	30	教务处	静思楼
7499	艾冲	副科长	7698	2010-02-20	1600.00	300.00	30	20	研训处	科研楼
7499	艾冲	副科长	7698	2010-02-20	1600.00	300.00	30	10	财务处	文豪楼
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30	40	人事处	唐溪楼
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30	30	教务处	静思楼
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30	20	研训处	科研楼
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30	10	财务处	文豪楼
7566	陈建	副处长	7839	2005-04-02	2975.00	[null]	20	40	人事处	唐溪楼
7566	陈建	副处长	7839	2005-04-02	2975.00	[null]	20	30	教务处	静思楼

2) 自然连接 (NATURAL JOIN): 是根据两张表相同字段 (字段名和字段类型) 自动进行匹配, 并会保留一个进行匹配的字段, 可以和 WHERE 联用。

语法结构：SELECT * FROM 表 1 NATURAL JOIN 表 2;

示例：**SELECT * FROM employee NATURAL JOIN dept;**

1 SELECT * FROM employee NATURAL JOIN dept;

deptno	empno	ename	jobrank	mgrno	hiredate	salary	allowance	dname	location
30	7369	右一丹	职员	7902	2020-12-17	800.00		研究院	科研楼
30	7499	艾冷	副科长	7698	2010-02-20	1600.00	300.00	财务处	静思楼
30	7521	李静	副科长	7698	2008-02-22	1250.00	500.00	财务处	静思楼
20	7566	陈建	副科长	7839	2005-04-02	2975.00	(Null)	研究院	科研楼
30	7654	马丁	副科长	7698	2010-09-28	1250.00	1400.00	财务处	静思楼
30	7698	徐德辉	副科长	7839	2009-05-01	2850.00	(Null)	财务处	静思楼
10	7782	杜峰	副科长	7839	2000-06-09	2450.00	(Null)	财务处	文萃楼
20	7788	王美英	科长	7566	2007-04-19	3000.00	(Null)	研究院	科研楼
10	7839	孙一崧	处长		2000-11-17	5000.00	(Null)	财务处	文萃楼
30	7844	天才瑞	副科长	7698	2000-09-08	1500.00	0.00	财务处	静思楼
20	7876	张明耀	职员	7788	1999-05-23	1100.00	(Null)	研究院	科研楼

3) 内连接 (INNER JOIN): 在两张表或多张表生成的笛卡儿积记录中筛选出与连接条件相匹配的数据记录, 并过滤不匹配的记录。

语法结构: SELECT 字段 1, 字段 2 FROM 表名[AS] [INNER] JOIN 表 2 ON 连接条件 [WHERE 条件语句];

SELECT 字段 1, 字段 2 FROM 表 1, 表 2 WHERE 条件;

示例: SELECT

e.deptno, empno, ename, jobrank, mgrno, hiredate, salary, allowance, dname, location
FROM employee AS e [INNER] JOIN dept d ON e.deptno = d.deptno;

1 SELECT e.deptno, empno, ename, jobrank, mgrno, hiredate, salary, allowance, dname, location
2 FROM employee AS e [INNER] JOIN dept d ON e.deptno = d.deptno;

deptno	empno	ename	jobrank	mgrno	hiredate	salary	allowance	dname	location
20	7369	右一丹	职员	7902	2020-12-17	800.00	(Null)	研究院	科研楼
30	7499	艾冷	副科长	7698	2010-02-20	1600.00	300.00	财务处	静思楼
30	7521	李静	副科长	7698	2008-02-22	1250.00	500.00	财务处	静思楼
20	7566	陈建	副科长	7839	2005-04-02	2975.00	(Null)	研究院	科研楼
30	7654	马丁	副科长	7698	2010-09-28	1250.00	1400.00	财务处	静思楼
30	7698	徐德辉	副科长	7839	2009-05-01	2850.00	(Null)	财务处	静思楼
10	7782	杜峰	副科长	7839	2000-06-09	2450.00	(Null)	财务处	文萃楼
20	7788	王美英	科长	7566	2007-04-19	3000.00	(Null)	研究院	科研楼

说明: JOIN 连接表, ON 描述连接条件; 如果两个表有相同的字段名, 必须用表名来区别。

内连接根据条件不同可以分为等值连接、非等值连接:

等值连接: 在 ON 子句中的连接条件中使用 = 指定两表中要进行匹配的字段, 从而在两张表生成的笛卡尔积记录中筛选出满足条件的记录。自然连接中不能使用 ON, 且会去掉重复的字段。

4) 自连接: 表自己和自己连接。

示例：SELECT e1.ename 姓名,e2.ename 领导名 FROM employee e1 INNER JOIN employee e2 ON e1.mgrno = e2.empno;

姓名	领导名
徐敏婷	孙一鸣
杜巍	孙一鸣
王美美	陈建
天才猫	徐敏婷
张明耀	王美美
李长江	徐敏婷
方立人	陈建
王萌萌	杜巍

5) 外连接查询 (OUTER JOIN): 不仅可以在两张表生成的笛卡儿积记录中筛选出与连接条件匹配的数据记录, 还能根据用户指定而保留部分不匹配的记录。根据不匹配记录来源的不同, 可以将外连接分为左外连接和右外连接。

语法结构: SELECT 字段 1, 字段 2 FROM 表名[AS] LEFT | RIGHT [OUTER] JOIN 表 2 ON 连接条件 [WHERE 条件语句];

1. 左外连接: 把原来不匹配的内容同时显示出。

示例：SELECT e1.ename 姓名,e2.ename 领导名 FROM employee e1 LEFT OUTER JOIN employee e2 ON e1.mgrno = e2.empno;

姓名	领导名
石一丹	方立人
文冲	徐敏婷
李静	徐敏婷
陈建	孙一鸣
马丁	徐敏婷
徐敏婷	孙一鸣
杜巍	孙一鸣
王美美	陈建
孙一鸣	(Null)
天才猫	徐敏婷
张明耀	王美美
李长江	徐敏婷
方立人	陈建
王萌萌	杜巍

2. 右外连接:

示例：SELECT ename,e.deptno,dname FROM employee e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno;

ename	deptno	dname
石一丹	20	研发部
李长江	30	销售部
天才瑞	30	销售部
徐依婷	30	销售部
马丁	30	销售部
李静	30	销售部
刘冲	30	销售部

如果数据表连接的字段相同，则连接时的匹配条件可使用 USING 代替 ON。

语法结构：SELECT 字段 FROM 表 1 [CROSS|INNER|RIGHT|LEFT] JOIN 表 2 USING(同名的连接字段列表);

示例：SELECT ename,e.deptno,dname FROM employee e RIGHT JOIN dept t USING(deptno);

ename	deptno	dname
王瑞瑞	10	销售部
孙一鸣	10	销售部
杜程	10	销售部
方立人	20	研发部
张知福	20	研发部
王来美	20	研发部
海建	20	研发部
石一丹	20	研发部
李长江	30	销售部
天才瑞	30	销售部
徐依婷	30	销售部
马丁	30	销售部
李静	30	销售部
刘冲	30	销售部

6) 子查询：在一个子查询语句中嵌入一个查询语句为执行条件或查询的数据源（代替 FROM 后的数据表）。子查询是一个完整的 SELECT 语句，能够独立执行，在含有子查询的语句中，子查询必须写在()内，并且 SQL 会限制性子查询中语句，将其结果作为外层语句的过滤条件。

1.标量子查询

示例：SELECT ename,salary FROM employee WHERE salary>(SELECT salary FROM employee WHERE ename="徐依婷");

```
1 SELECT ename,salary FROM employee WHERE salary>(SELECT salary FROM employee WHERE ename="徐依婷");
```

信息	摘要	结果 1	解析	状态
ename	salary			
陈建	2975.00			
王美英	3000.00			
徐依婷	5000.00			
方立人	3000.00			

2.行子查询（子查询返回的结果为一行）

示例：**SELECT * FROM employee WHERE (jobrank,deptno)=(SELECT jobrank,deptno FROM employee WHERE ename="艾冷");**

```
1 SELECT * FROM employee WHERE (jobrank,deptno)=(SELECT jobrank,deptno FROM employee WHERE ename="艾冷");
```

empno	ename	jobrank	mgrno	hiredate	salary	allowance	deptno
7690	艾冷	副科长	7698	2010-02-20	1600.00	300.00	30
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30
7654	马丁	副科长	7698	2010-09-28	1250.00	1400.00	30
7844	天才瑞	副科长	7698	2000-09-08	1500.00	0.00	30

3.列子查询（子查询（内层查询）返回的结果为一列，结果来自某个字段的出啊寻）

示例：**SELECT * FROM employee WHERE deptno in (SELECT deptno FROM dept WHERE location in("科研楼","文章楼"));**

```
1 SELECT * FROM employee WHERE deptno in (SELECT deptno FROM dept WHERE location in("科研楼","文章楼"));
```

empno	ename	jobrank	mgrno	hiredate	salary	allowance	deptno
7369	石一丹	职员	7902	2020-12-17	800.00	(Null)	20
7566	陈建	副处长	7839	2005-04-02	2975.00	(Null)	20
7782	杜麟	副处长	7839	2000-06-09	2450.00	(Null)	10
7788	王美英	科长	7566	2007-04-19	3000.00	(Null)	20
7839	陈一瑞	处长	7566	2000-11-17	5000.00	(Null)	10
7876	张明霞	职员	7788	1999-05-23	1100.00	(Null)	20
7902	方立人	科长	7566	2001-12-03	3000.00	(Null)	20
7934	王瑞南	职员	7782	2002-01-23	1300.00	(Null)	10

也可以使用 SOME ANY ALL 子查询：

ANY：对于查询返回的结果集中任意一个数据，如果比较结果为 TRUE，返回 TRUE。

示例：**SELECT * FROM employee WHERE salary < ANY(SELECT salary FROM employee WHERE jobrank="职员");**

```
1 SELECT * FROM employee WHERE salary < ANY(SELECT salary FROM employee WHERE jobrank="职员");
```

信息	摘要	结果 1	剖析	状态			
empno	ename	jobrank	mgrno	hiredate	salary	allowance	deptno
7369	石一丹	职员	7902	2020-12-17	800.00	(Null)	20
7521	李静	副科长	7698	2008-02-22	1250.00	500.00	30
7654	马丁	副科长	7698	2010-09-28	1250.00	1400.00	30
7876	张明耀	职员	7788	1999-05-23	1100.00	(Null)	20
7900	李长江	职员	7698	2000-12-03	950.00	(Null)	30

3.3.5 函数使用

1.CONCAT 函数：连接字符串。

```
1 SELECT CONCAT(ename,"-",jobrank) AS "姓名-职位" FROM employee;
```

信息	摘要	结果 1	剖析	状态
姓名-职位				
右一丹-职员				
艾冷-副科长				
李静-副科长				
陈建-副科长				
马丁-副科长				
徐晓婷-副科长				
杜静-副科长				
王美美-科长				
孙一鸣-处长				
天才熊-副科长				
张明耀-职员				
李长江-职员				
方立人-科长				
王瑞霞-职员				

2.LENGTH 函数：计算长度。

```
1 SELECT ename,LENGTH(ename) FROM employee;
```

信息	摘要	结果 1	剖析	状态
ename	LENGTH(ename)			
右一丹		9		
艾冷		6		
李静		6		
张静		6		
马丁		6		
徐晓婷		9		
杜静		6		
王美美		9		
孙一鸣		9		
天才熊		9		
张明耀		9		
李长江		9		
方立人		9		
王瑞霞		9		

3.LOWER、UPPER 函数：改为小写、大写。

4.REPLACE 函数：替代。

```
1 SELECT ename,jobrank,REPLACE(jobrank,"副科长","副科长-正科级") FROM employee WHERE jobrank="副科长";
```

信息	摘要	结果 1	剖析	状态
ename	jobrank	REPLACE(jobrank,"副科长","副科长-正科级")		
李静	副科长	副科长-正科级		
张静	副科长	副科长-正科级		
马丁	副科长	副科长-正科级		
天才熊	副科长	副科长-正科级		

5.SUBSTRING 函数：用以分离字符串。

```
1 SELECT ename, SUBSTRING(ename,2,2) AS 名, jobrank FROM employee WHERE jobrank="职员";
```

信息	摘要	结果 1	类型	状态
ename	名	jobrank		
石一丹	一丹	职员		
张明耀	明耀	职员		
李长江	长江	职员		
王瑞茜	瑞茜	职员		

6.ABS(num)函数，CELL(num)函数：返回大于 num 的最小整数，FLOOR(num)：返回小于 num 的最大整数，MOD(num1,num2)函数：返回余数，ROUND(num,n)函数：返回 num 四舍五入后的值，保留 n 位小数，TRUNCATE(num,n)函数：舍去小数点后 n 位的值。

7.IF()函数：

```
1 SELECT ename,salary,IF(salary>=3000,"高工资","低工资")AS 工资等级 FROM employee;
```

信息	摘要	结果 1	类型	状态
ename	salary	工资等级		
石一丹	800.00	低工资		
艾冷	1600.00	低工资		
李静	1250.00	低工资		
陈建	2975.00	低工资		
马丁	1250.00	低工资		
徐依婷	2850.00	低工资		
杜峰	2450.00	低工资		
王美美	3000.00	高工资		
孙一鸣	5000.00	高工资		
无才瑞	1500.00	低工资		
张明耀	1100.00	低工资		
李长江	950.00	低工资		
方立人	3000.00	高工资		
王瑞茜	1300.00	低工资		

8.CASE WHEN TRUE THEN ELSE END [AS]

```
1 SELECT ename,salary,CASE salary >3000
2 WHEN TRUE THEN
3 "高"
4 ELSE
5 "低"
6 END
7 AS 工资等级
8 FROM employee;
```

信息	摘要	结果 1	类型	状态
ename	salary	工资等级		
石一丹	800.00	低		
艾冷	1600.00	低		
李静	1250.00	低		
陈建	2975.00	低		
马丁	1250.00	低		
徐依婷	2850.00	低		
杜峰	2450.00	低		
王美美	3000.00	低		
孙一鸣	5000.00	高		
无才瑞	1500.00	低		
张明耀	1100.00	低		
李长江	950.00	低		
方立人	3000.00	低		
王瑞茜	1300.00	低		

3.3.6 数据插入

1.语法结构：INSERT INTO 表名[字段] VALUES(对应值);

示例：INSERT INTO specialty(zno,zname) VALUES ('2201','元宇宙');

```

1 -- INSERT INTO specialty(zno,zname) VALUES ('2201','元宇宙');
2 SELECT * FROM specialty;

```

信息	摘要	结果 1	剖析	状态
zno	zname			
1102	数据科学与			
1103	人工智能			
1201	网络与新媒体			
1214	区块链工程			
1407	健康服务与			
1409	智能医学工			
1601	供应链管理			
1805	智能纺织工			
1807	智能装备与			
2201	元宇宙			

注意字段和数据相符合。

2.同时插入多条语句:

INSERT INTO 表名 VALUES(值 1),(值 2);

示例: **INSERT INTO specialty VALUES('1121','计算机工程'),('3601','公寓管理');**

```

1 INSERT INTO specialty VALUES('1121','计算机工程'),('3601','公寓管理');
2 SELECT * FROM specialty;

```

信息	摘要	结果 1	剖析	状态
zno	zname			
1102	数据科学与			
1103	人工智能			
1121	计算机工程			
1201	网络与新媒体			
1214	区块链工程			
1407	健康服务与			
1409	智能医学工			
1601	供应链管理			
1805	智能纺织工			
1807	智能装备与			
2201	元宇宙			
3601	公寓管理			

3.复制表结构进行数据插入:

语法: CREATA TABLE 新表名 SELECT 字段名 FROM 参考表;

示例: **CREATE TABLE student_copy SELECT sno,sname,ssex,sclass FROM student;**

```

1 CREATE TABLE student_copy AS SELECT sno,sname,ssex,sclass FROM student;

```

信息	摘要	剖析	状态
查询	CREATE TABLE student_copy AS SELECT sno,sname,ssex,sclass FROM student		信息
			OK

此方法会将参考表中的对应字段的所有数据复制放入表。

4.数值替换:

语法结构: INSERT INTO 表名(字段名) VALUES(数值) ON DUPLICATE KEY UPDATE 条件;

REPLACE INTO 表名 VALUES(数值);

示例: **INSERT INTO specialty(zno,zname) VALUES('2205','Food Chemistry')
ON DUPLICATE KEY UPDATE zname="Food Chemistry";**

5.数据更新:

语法结构: UPDATE 表名 SET 字段名 1=值 1;

示例: **UPDATE sc SET grade = 90 WHERE cno="58130540";**

建议先查询后修改。

3.3.7 数据删除

1.DELECT 删除:

语法结构: DELECT FROM 表名 WHERE 条件;

示例: **DELECT FROM sc WHERE grade = 90;**

如果不加条件, 则会将表内数据全部删除。

2.TRUNCATE 删除:

语法结构: TRUNCATE [TABLE] 表名 WHERE 条件;

示例: **TRUNCATE sc;**

TRUNCATE 本质上先进行表的删除。

3.4 视图

3.4.1 视图创建

语法结构: CREATE VIEW 视图名;

示例: **CREATE VIEW vsg AS SELECT sname,cname,grade FROM student
s,course c,sc WHERE s.sno=sc.sno AND c.cno =sc.cno;**

可以基于视图创建视图:

示例: **CREATE VIEW v_info_max AS SELECT sno,sname,ssex,sclass FROM
v_info WHERE sclass="大数据 2001";**

3.4.2 视图操作

1.视图结构查看:

查看基本结构 (字段属性): DESCRIBE 视图名;

查看详细结构 (字段名): SHOW TABLE STATUS LIKE “字段名”;

2.视图删除: DROP VIEW [IF EXIST] 视图名;

3.视图更新: UPDATE 视图名 SET 更新内容 WHERE 条件;

不建议从视图对基本表进行操作。

3.5 索引

3.5.1 索引创建

1.为已有的表创建:

语法结构: CREATA INDEX 索引名 ON 表名(属性名[长度][ASC|DESC]);

2.创建唯一索引:

语法结构: CREATE UNIQUE INDEX 索引名 ON 表名(属性名[长度][ASC|DESC]);

3.查看索引:

语法结构: SHOW INDEX FROM 表名 [FROM 数据库名];

4.删除索引:

语法结构:

DROP INDEX 索引名 ON 表名;

ALTER TABLE 表名 DROP INDEX 索引名;

4 存储过程与函数

4.1 存储过程概述

特点: 一组 SQL 语句的集合, 存放在服务器端, 可以减少客户端和服务端之间数据传输。

存储过程: 一组为了完成特定功能的 SQL 语句集, 经编译后存储在数据库中, 用户通过指定存储过程的名字并给定参数(如果该存储过程带有参数)来调用和执行它。存储过程是由 SQL 语句和一些特殊的控制结构组成。

优点: 增强了 SQL 的功能和灵活性, 允许标准组件式编程, 较快的执行速度(原因是预编译), 能够减少网络流量, 安全机制充分利用;

缺点: SQL 更复杂, 编写时需要具备数据库对象的权限。

与函数的区别:

存储过程功能要复杂, 函数实现功能针对性比较强

存储过程可以返回参数, 比如记录, 函数只能返回值或者表对象, 函数只能返回一个变量。存储过程可以返回多个。存储过程的参数有三类: IN、OUT、INOUT, 函数只有 IN 类型。

存储过程声明时不需要返回类型, 而函数声明时需要描述返回类型的。

存储过程一般作为一个独立的部分来执行(CALL 存储过程名), 而函数可以作为查询语句的一部分来调用(SELECT 调用)

4.2 存储过程的创建与删除

查看当前用户是否具有创建存储过程的权限:

```
SELECT create_routine_priv FROM mysql.user WHERE user='root';
```

创建存储过程

语法格式:

```
CREATE PROCEDURE 存储过程名([IN|OUT|INOUT]参数名 1 数据类型 ,  
([IN|OUT|INOUT]参数名 2 数据类型,...))
```


BEGIN

过程体

END;

用来修改 SQL 语句的结束符，可以将结束符分号 (;) 修改为其他特殊符号。

原因：在存储过程（函数）的内容里可以有多条 SQL 语句，是用分号 (;) 隔开的，代表该 SQL 语句结束，修改后的结束符在函数的末尾添加，代表定义函数完毕，所以要修改结束符。

过程名：使用 PROCEDURE 来标识存储过程，存储过程的名称应该是**合法的标识符**，不能与已有的关键字或者存储过程名称冲突。一个存储过程是属于某个数据库的，可以使用 db_name.procedure_name 的形式执行存储过程所属的数据库。
参数类型：

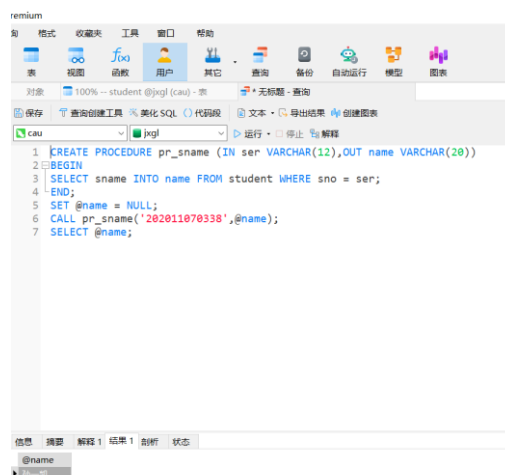
IN 参数的值必须在调用存储过程时指定，在存储过程中修改该参数的值，不能被返回；**OUT** 参数的值可以在存储过程的内部被改变，并可返回；**INOUT** 参数在调用时指定，并且可被改变和返回。**BEGIN...END**：用来标识过程体的开始和结束，当有多条 SQL 语句时，可以使用 BEGIN...END 来标识。**DECLERE**：用来声明变量，包括变量名和类型

用户变量：

一般使用“@”开头，如输入参数变量：@p_sex，输出参数变量：@p_count。

注意：滥用用户变量会导致程序难以理解及管理。

SET：用来设置变量的值；CALL：用来调用存储过程。



4.3 查看存储过程

1.方式一：

SHOW PROCEDURE STATUS LIKE 'getUserCountBySex';

查看自定义存储过程的状态：所属的数据库、类型、存储过程名称、修改时间等状态信息；

2.方式二:

```
SHOW CREATE PROCEDURE getUserCountBySex;
```

查看存储过程的信息，包括存储过程内容等。

3.、调用存储过程

```
CALL 存储过程名;
```

在调用自定义存储过程的时候，如果有参数，则在存储过程名称的括号里添加参数值。

存储过程变量分为局部变量和用户变量。

局部变量在过程体里的 BEGIN...END 内使用，使用 DECLARE 关键字定义局部变量，局部变量只在 BEGIN...END 内有效；而用户变量属于用户客户端变量，与当前用户客户端有关，不同 MySQL 客户端的用户变量不同，通常使用“@”符号作为变量的前缀。

4.修改存储过程

```
ALTER PROCEDURE 存储过程名
```

修改存储过程时，**只能修改存储过程的一些特性**（特性可以在创建存储过程时，通过 characteristic 来指定），**不能修改过程体**，如果要修改过程体，就需要先删除存储过程，再重新创建。

5.删除存储过程

```
DROP PROCEDURE [IF EXISTS] 存储过程名
```

在删除存储过程的时候，在 PROCEDURE 后面是存储过程名，不要括号，也可以通过 IF EXISTS 判断存储过程是否存在，然后删除。

调用存储过程:

第一种：调用存储过程时，输入参数可以直接在括号里赋值。

```
CALL getStudentInfoBySex('男')
```

第二种：IN 类型参数，可以通过定义用户变量“@p_sex”来传递参数。

```
SET @p_sex = '女';
```

```
CALL getStudentInfoBySex(@p_sex)
```

5 Mysql 触发器

5.1 触发器定义

Mysql 触发器是用户定义在数据表中的一类由事件驱动的特殊过程。当有指定事件时，会调用触发器对象，执行触发器操作。

5.2 触发器功能

触发器功能：使多个用户能够在保持数据完整性和一致性条件下进行修改。具有以下功能：

- 1.确保安全性;
- 2.审计;
- 3.实现复杂得到数据完整性规则;

5.3 触发器创建和使用

5.3.1 触发器创建

语法结构:

```
CREATA TRIGGER 触发器名字 Trigger_time Trigger_event ON 数据表 FROM  
EACH ROW SET Trigger_body
```

示例:

```
DELIMITER $$
```

```
CREATE TRIGGER update_specialty_num AFTER INSERT ON specialty_num  
FOR EACH ROW
```

```
BEGIN UPDATE specialty_num SET num = num + 1 WHERE id = 1;
```

```
END$$
```

```
DELIMITER;
```

说明: Trigger_time 表示触发器触发的时机, 有两个值: AFTER、BEFORE;
Trigger_event 表示触发器出发的时间, 有三个值: INSERT、UPDATE、DELETE;
FOR EACH ROW 表示对每行都生效; Trigger_body 表示触发器程序体, 可以使用 SQL 语句和 END 等。

使用时需注意: 一个表不能同时有两个相同类型的触发器。

补充: NEW 和 OLD。对于 **INSERT**, NEW 表示 BEFORE、AFTER 插入的数据,
对 **UPDATE**, OLD 表示 BEFORE、AFTER 被修改的原数据, NEW 表示 BEFORE、
AFTER 修改后的新数据。

示例:

```
DELIMITER $$
```

```
CREATE TRIGGER insert_grade_record AFTER INSERT ON sc FOR EACH  
ROW
```

```
BEGIN IF(NEW.grade < 60)THEN INSERT INTO sc_record  
VALUES(NEW.sno,NEW.cno,NEW.grade);
```

```
END if;
```

```
END $$
```

```
DELIMITER ;
```

5.3.1 触发器操作

显示触发器: **SHOW TRIGGERS;**

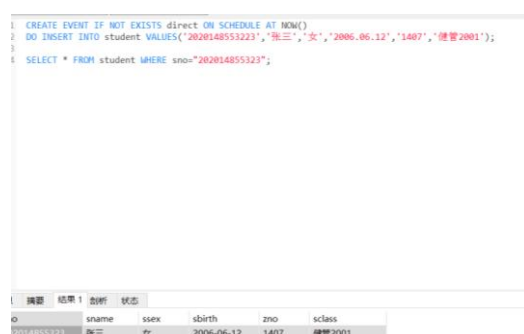
删除触发器： **DELETE TRIGGER** 触发器名称;

5.4 事件调度器

用作定时执行某些任务，以取代原先只能由系统的计划任务来执行的工作。它是基于特定事件周期触发来执行某些任务，触发器是基于某个表产生的、创建事件的基本语法：

```
CREATE EVENT[IF NOT exist]event_name ON SCHEDULE schedule DO  
sql_statement;
```

示例：



修改事件：ALTER EVENT event_name;

删除事件：DROP EVENT event_name;

6 事务与并发控制

6.1 事务和锁

事务和锁是实现数据一致性和并发性的基石。事务是一个不可分割的工作单位，其具有 ACID 特性：

A 原子性：每个事务是不可分割的单元；

C 一致性：由日志机制处理，记录了数据库所有的变化，为事务的回复提供跟踪记录；

I 隔离性：保证了任意时刻只能由一个用户访问数据库（每个事务再它自己的空间发生，和其他发生在系统中的事务隔离，且事务的结果只在被完全执行时才能看到）；

D 持久性：一个提交的事务始终保持最终的状态（通过保存了一个记录事务过程中系统变化的二级制事务日志文件来是持久性）。

6.2 控制语句

6.2.1 事务操作

使用 BEGIN 开始事务，使用 COMMIT 前可以使用 ROLLBACK 回滚事务。

回滚：指撤销指定 SQL 语句的过程，提交：指未储存的 SQL 语句，结果写入数据库表。

语法结构: SET COMMIT|BEGIN[WORK] COMMIT [WORK] ROLLBACK
SET AUTOCOMMIT=[0|1]

示例:

模拟银行转账

从某账号 A 向账号 B 转账 6000 元, 若出错, 则进行事务回滚。

- (1) 创建数据库 bank
- (2) 在 bank 中创建存放账号的表

DROP TABLE IF EXISTS `account`;

CREATE TABLE `account` (

`id` int(11) NOT NULL AUTO_INCREMENT,

`username` varchar(50) DEFAULT NULL,

`balance` decimal(10,0) unsigned DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

- (3) 向 account 表中插入两条记录。

INSERT INTO account(username,balance) VALUES('A',10000),('B',0);

- (4) 存储过程, 并向该存储过程创建事务, 实现从某 A 向 B 进行转账。

DELIMITER //

CREATE PROCEDURE transfer(IN id_from INT ,IN id_to INT ,IN money INT)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;

START TRANSACTION;

UPDATE account SET balance=balance + money WHERE id=id_to;

UPDATE account SET balance=balance - money WHERE id=id_from;

COMMIT;

END//

DELIMITER;

- (5) 调用存储过程

CALL transfer(1,2,6000);

6.2.2 并发操作

可能出现的问题: 丢失更新、脏读数据、不可重复读取、幻读。锁定可以实现数据库并发控制。

锁的类型：共享锁（读锁）、排他锁（写锁）。锁粒度：锁的作用范围。锁策略：表级锁、行级锁。死锁：当多个处于不同序列的用户计划同时更新相同的数据库时，因相互等待对方释放权限而导致双方一直处于等待状态。

MySQL 的隔离级别：

(1)SERIALIZABLE：串行化方式；(2)REPEATABLE READ：可重复读；

(3)READ COMMITTED：提交读；(4)READ UNCOMMITTED：未提交读。

7 数据备份和还原

7.1 常用的备份策略

策略：数据库需要定期做备份，定期做恢复测试，是否考虑异地备份，是否采用增量备份。

根据是否需要数据库离线，可分为：冷备（需关闭 MySQL 服务，在读取请求均不允许状态下进行），温备（服务在线，仅支持读请求，不允许写请求），热备（备份同时，业务不受影响）。

根据要备份的数据集合的范围可分为：完全备份、增量备份、差异备份。

7.2 数据库备份命令

使用命令：mysqldump

语法格式：

```
mysqldump -u username -p dbname table1,table2... >backupTable.sql
```

可以备份多个数据库：--p --databases dbname

备份所有数据库：--p --all --databases > 存放地点

7.3 数据库恢复

是通过将数据库从某一种错误状态等恢复到某一已知的正确状态。

8 数据库设计方法

8.1 数据库设计

数据库设计指利用现有的数据库管理系统，针对具体的应用对象构建合适的数据库模式，建立数据库及应用系统，使之有效收集、存储、操作和管理数据，满足用户需求。

数据库设计的目标：满足用户需求，得到某 DBMS 产品支持，效率高，易于维护扩充。

8.2 数据库设计步骤

1.需求分析：了解用户需求；

2.概念结构设计：对用户需求进行综合、归纳与抽象，形成 ER 图；

面向用户的应用需求

3.逻辑结构设计：将 ER 图转化为某 DBMS 支持的数据模型，并优化；

4.物理结构设计：为逻辑结构选取一个适合的物理结构；

面向数据库管理系统

5.实施阶段：创建数据库，调试数据库，数据存入；

6.数据库维护阶段：维护。

实现与运行阶段

8.3 系统需求分析

需求分析是设计的起点。通过调查实现现实世界的对象，充分了解原系统的工作状况，明确用户的需求，确定新系统的功能。

调查后去用户的数据库要求：信息要求、处理要求、系统要求。

需求分析方法：SA（结构化分析）、自上而下、逐层分解。

数据流图：数据流、加工、文件、外部实体。主要表达数据和处理之间的关系。

数据字典：是对数据流图的注释和补充。包括数据项、数据结构、数据流、数据存储和处理过程（判定树、判定表）。

8.4 概念结构设计

将需求分析的得到的用户需求抽象为信息结构。需要：语义表达能力丰富、易于理解和交流、易于修改和扩充、易于向各种数据模型转换。概念结构设计方法：自上而下、自下而上、逐步扩张、混合粗略。

8.5 逻辑结构设计

ER 图向关系模型的转换：解决如何将实体和实体的联系转换为关系，并确定这些关系的属性和码。

转换规则：一个实体转换为一个关系，实体的属性和码就是关系的属性和码。
关系模式规范化：

范式：1NF、2NF、3NF、BCNF、4NF、5NF。规范化：将低一级范式的关系模式分解为若干高一级模式。

8.6 数据库实施

建立实际数据库结构、数据导入数据库、应用程序编码和调试、数据库运行。