

# 01\_数据库概述（了解）

---

## 1.1\_引入

---

问题 :为什么要使用数据库不用 XML 或者文件呢?

举个例子: 有100W 条数据,我们可以将他们保存在一个 txt 或者 xml 文档中.但如果想从这些数据中找到某一条数据,只能通过线性搜索,从头到尾去找,效率很低.那我们能不能使用一种快捷高效的方式来组织这些数据,提高查询效率呢? 使用 **数据库**,就能大大的提高数据的操作效率.

## 1.2\_相关概述

---

### 1.2.1\_数据库(DataBase:DB)

数据库是**按照数据结构来组织、存储和管理数据的仓库**。----> 存储和管理数据的仓库.(说白了,数据库还是一些文件,只是这些文件使用了对应的数据结构来组织数据的).

### 1.2.2\_数据库管理系统 DBMS

**DBMS** ( Database Management System ) : 专门用于管理数据库的计算机系统**软件** 数据库管理系统能够为数据库提供数据的定义、建立、维护、查询和统计等操作功能, 并完成对数据完整性、安全性进行控制的功能。我们一般说的数据库,就是指的DBMS ( 数据库服务器: 管理多个数据库文件的软件 )



### 1.2.3\_数据库应用系统

DataBase Application System : 使用数据库技术的系统; 基本上所有的信息管理系统都是数据库应用系统。因为基本上需要用到数据库。

## 02\_数据库技术发展历程(了解)

### 2.1\_数据库发展流程

1.层次数据库和网状数据库技术阶段；使用指针来表示数据之间的联系。

2.关系数据库技术阶段(表,使用表这种关系将数据组织起来的)；经典的里程碑阶段。代表 DBMS: Oracle、DB2、MySQL、SQL Server等。

3.非关系数据库技术阶段

随着大数据的不断发展，关系型数据库在数据模型，性能，拓展伸缩性上存在一个的缺点，所以非关系型的数据库现在成了一个极其热门的新领域，非关系数据库产品的发展非常迅速。

非关系型数据库分类：

- ORDBMS：面向对象数据库技术。
- NoSQL not only sql：结构化数据库技术。

### 2.2\_出色的 NoSQL 数据库

- 键值存储数据库：Oracle BDB,Redis,BeansDB
- 列式储数数据库：HBase,Cassandra,Riak
- 文档型数据库：MongoDB,CouchDB
- 图形数据库：Neo4j,InfoGrid,Infinite Graph

### 2.3\_常见的关系数据库

数据库系统	所属公司
Oracle	Oracle
DB2	IBM
SQL Server	MS
MySQL	AB-->SUN-->Oracle

Oracle：运行稳定，可移植性高，功能齐全，性能超群！适用于大型企业领域，但是价格昂贵。DB2：速度快、可靠性好，适于海量数据，恢复性极强。适用于大中型企业领域，但是价格昂贵。SQL Server：全面，效率高，界面友好，操作容易，但是不跨平台。适用于中小型企业领域。MySQL：开源，体积小，速度快。适用于中小型企业领域。

## 03\_SQL 介绍（掌握）

### 3.1\_SQL 概述

人和人交流需要语言，人和数据库交流也需要语言，而这个专门特定为程序员和数据库打交道的语言就是 SQL 语言。

**SQL**：结构化查询语言(Structured Query Language)。是关系型数据库**标准语言**。特点：简单，灵活，功能强大。

## 3.2\_SQL包含的 6 个部分

**一：数据查询语言（DQL）**：其语句，也称为“数据检索语句”，用以从表中获得数据，确定数据怎样在应用程序给出。保留字 **SELECT** 是DQL（也是所有SQL）用得最多的动词，其他DQL常用的保留字有WHERE，ORDER BY，GROUP BY和HAVING。这些 DQL 保留字常与其他类型的SQL语句一起使用。

**二：数据操作语言（DML）**：其语句包括动词 INSERT，UPDATE和DELETE。它们分别用于添加，修改和删除表中的行。也称为动作语言。

**三：事务处理语言（TPL）**：它的语句能确保被DML语句影响的表的所有行及时得以更新。TPL语句包括BEGIN TRANSACTION，COMMIT和ROLLBACK。

**四：数据控制语言（DCL）**：它的语句通过GRANT或REVOKE获得许可，确定单个用户和用户组对数据库对象的访问。某些RDBMS可用GRANT或REVOKE控制对表单个列的访问。

**五：数据定义语言（DDL）**：其语句包括动词 CREATE 和 DROP。在数据库中创建新表或删除表（CREATE TABLE 或 DROP TABLE）；为表加入索引等。DDL包括许多与人数据库目录中获得数据有关的保留字。它也是动作查询的一部分。

**六：指针控制语言（CCL）**：它的语句，像DECLARE CURSOR，FETCH INTO和UPDATE WHERE CURRENT用于对一个或多个表单独行的操作。

## 3.3\_书写规则

1.数据库中，SQL 语句大小写不敏感. 如: select SELECT. SeleCt,为了提高可读性，一般**关键字大写，其他小写**. 2.SQL 语句可单行或多行书写,用分号来分辨是否结束; select 3.合理利用空格和缩进使程序易读

# 04\_表的概述（理解）

## 4.1\_什么是表(table)

我们说 MySQL 是一种关系型数据库。关系数据库最重要的概念就是表。

表具有固定的列数和任意的行数，在数学上称为“关系”。

**注意**: 表定义完,表的结构就固定.

学生表(Student)：

id	name	age
10	小狼	5
20	小马	5

二维表是 **同类实体** 的各种 **属性的集合**，每个实体对应于表中的一行，在关系中称为元组，相当于通常的一条记录；表中的列表示属性，称为Field，相当于通常记录中的一个数据项，也叫列、字段。**行**: 表示一个实体,一条记录 **列**: 字段,数据项

## 4.2\_表和对象的关系(ORM)

定义表就确定好结构能存那些属性的值好比定义了类就确定了有哪些属性

编号	姓名	年龄	性别

没有数据的表,表的结构(能存的数据项)已经固定

编号	姓名	年龄	性别
1	小明	18	女
2	小红	17	男

带有两个学生对象的表,两行数据

```
@Getter @Setter
class Student{
    private Long id;
    private String name;
    private Integer age;
    private String gender;
}
```

ORM 对象关系映射

表(关系)          类(对象)          描述

行                  对象                  表示一条记录  
列                  属性                  表示一个数据项

在开发中,我们需要将表中的数据查询出来保存到内存中,或者把内存中的数据保存到数据库中,此时就需要将数据表的数据和Java中的对象进行映射关联起来。

这种映射关联就称为 ORM 思想。

ORM : Oject Reraltional Mapping : 对象表的映射

面向对象概念	面向关系概念
类	表
对象	表的行(记录)
属性	表的列(字段)

## 05\_MySQL 安装

MySQL 安装过程中注意

- MySQL 的默认端口是:3306
- 数据库默认字符集就是 utf8
- 设置密码 : admin

配置停在某个步骤,卡死掉->强制关闭 ( ztrl+shift+esc ) ->重新执行配置 ( MySQL安装路径下/bin/MySQLInstanceConfig.exe )

数据库操作流程(重点):

1. 保证服务器是开启状态(如果是作为一个系统服务,不需要管).
2. 身份验证:通过账号,密码,数据库地址,端口等等连接数据库
3. 在数据库中选择一个数据库文件.
4. 选择某一张表.
5. 操作表中的数据.

验证安装成功 -> win -> 输入mysql -> MySQL 自带的客户端-> 输入密码 ->看到 mysql> 说明安装成功

## 06 MySQL服务&连接MySQL

---

如果 MySQL 服务没有启动,则不能访问 MySQL

### 6.1\_MySQL服务（了解）

---

打开数据库连接之前:一定要保证 MySQL 服务已经开启了.

net start 命令名字:开启一个服务, 如: net start MySQL

net stop 命令名字:关闭一个服务器, 如: net stop MySQL

### 6.2\_连接MySQL（掌握）

---

方式1: 进入 MySQL 自带的客户端, 在命令行中输入密码; 方式2: 在运行 (win + r || cmd 中) 中输入命令

格式: mysql -u账户 -p密码 -h数据库服务器安装的主机 -P数据库端口

```
mysql -uroot -padmin -h127.0.0.1 -P3306
```

若连接的数据库服务器在本机上, 并且端口是 3306。则可以简写: mysql -uroot -padmin

### 6.3\_Navicat for MySQL

---

Navicat for MySQL[1] 是一款强大的 MySQL 数据库管理和开发工具, 它为专业开发者提供了一套强大的足够尖端的工具, 但对于新用户仍然易于学习。Navicat for MySQL 基于Windows平台, 为 MySQL 量身订作, 提供类似于 MySQL 的用管理界面工具。此解决方案的出现, 将解放 PHP、J2EE 等程序员以及数据库设计者、管理者的大脑, 降低开发成本, 为用户带来更高的开发效率。

## 07\_数据库操作和存储引擎

---

### 7.1\_名词介绍

---

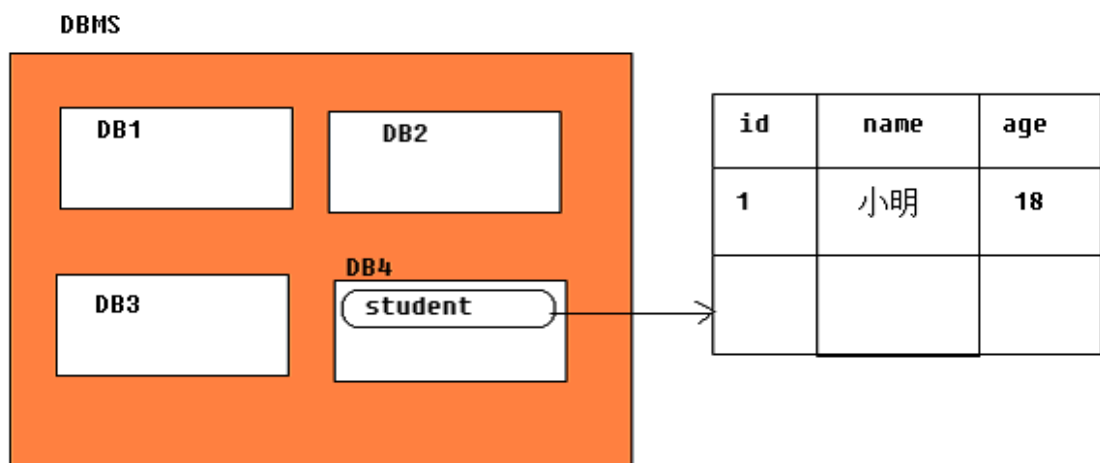
**数据库对象**: 存储, 管理和使用数据的不同结构形式, 如: 表、视图、存储过程、函数、触发器、事件等。**数据库**: 存储数据库对象的容器。

### 7.2\_数据库分类

---

1. 系统数据库（系统自带的数据库）：不能修改 information\_schema:存储数据库对象信息，如：用户表信息，列信息，权限，字符，分区等信息。 performance\_schema:存储数据库服务器性能参数信息。 mysql:存储数据库用户权限信息。 test:任何用户都可以使用的测试数据库。
2. 用户数据库（用户自定义的数据库）：一般的，一个项目一个用户数据库。

数据库对象，数据库，数据库服务器关系



## 7.3\_数据库操作(掌握)

- 1 查看数据库服务器存在哪些数据库. SHOW DATABASES;
- 2 使用指定的数据库. USE database\_name;
- 3 查看指定的数据库中有那些数据表: SHOW TABLES;
- 4 创建指定名称的数据库. CREATE DATABASE database\_name
- 5 删除数据库. DROP DATABASE database\_name

## 7.4\_存储引擎

MySQL 中的数据用各种不同的技术存储在文件（或者内存）中。这些技术中的每一种技术都**使用不同的存储机制、索引技巧、锁定水平并且最终提供不同的功能和能力**。通过选择不同的技术，你能够获得额外的速度或者功能，从而改善你的应用的整体功能。

### MySQL 常用存储引擎

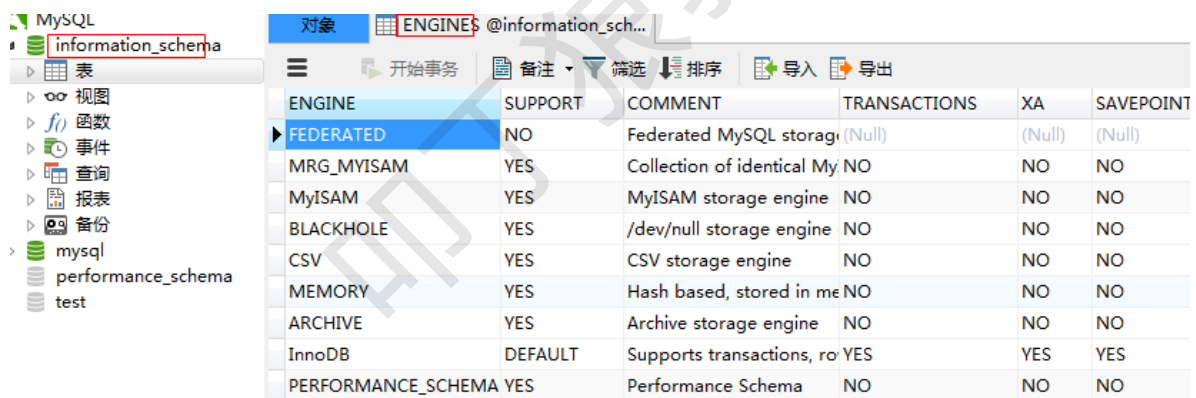
- MyISAM：拥有较高的插入，查询速度，但不支持事务，不支持外键。
- InnoDB：支持事务，支持外键，支持行级锁定，性能较低。

InnoDB 存储引擎提供了具有**提交、回滚和崩溃恢复能力的事务安全**。但对比MyISAM，处理效率差，且会占用更多的磁盘空间以保留数据和索引。

**注意：**一个系统，特别是金融系统，没有事务是很恐怖的事情，一般都要选择 InnDB.

特点	Myisam	BDB	Memory	InnoDB	Archive
存储限制	没有	没有	有	64TB	没有
事务安全		支持		支持	
锁机制	表锁	页锁	表锁	行锁	行锁
B 树索引	支持	支持	支持	支持	
哈希索引			支持	支持	
全文索引	支持				
集群索引				支持	
数据缓存			支持	支持	
索引缓存	支持		支持	支持	
数据可压缩	支持				支持
空间使用	低	低	N/A	高	非常低
内存使用	低	低	中等	高	低
批量插入的速度	高	高	高	低	非常高
支持外键				支持	

系统数据库information\_schema 数据库存储的 ENGINES



ENGINE	SUPPORT	COMMENT	TRANSACTIONS	XA	SAVEPOINT
FEDERATED	NO	Federated MySQL storage	(Null)	(Null)	(Null)
MRG_MYISAM	YES	Collection of identical My	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in me	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, ro	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

## 08 MySQL 列的常用类型

列的类型,主要是用来约束这一列数据能够存储何种类型的数据.无约束即可随便存则会乱套

### 8.1\_最常用类型（掌握）

以下是以后定义列的最常用类型，需要做数据转存，所以得有对应 Java 中的类型，



MySQL	Java
INT	int
BIGINT	long
DECIMAL	BigDecimal
DATE/DATETIME	java.util.Date
VARCHAR	String

## 8.2\_整数类型（了解）

整数类型有宽度指示器，作用是指定位宽。

例：某字段类型为 **INT(3)**,保证少于3个值，从数据库检索出来时能够自动地用 0 填充，需设置填充，默认不填充。

宽度指示器不影响列存值得范围。**一般不指定位宽。**

类型	大小	范围（有符号）	范围（无符号）	用途
TINYINT	1 字节	(-128, 127)	(0, 255)	小整数值
SMALLINT	2 字节	(-32 768, 32 767)	(0, 65 535)	大整数值
MEDIUMINT	3 字节	(-8 388 608, 8 388 607)	(0, 16 777 215)	大整数值
INT或 INTEGER	4 字节	(-2 147 483 648, 2 147 483 647)	(0, 4 294 967 295)	大整数值
BIGINT	8 字节	(-9 223 372 036 854 775 808, 9 223 372 036 854 775 807)	(0, 18 446 744 073 709 551 615)	极大整数值

## 8.3\_小数类型（了解）

FLOAT[(s,p)] 或 DOUBLE[(s,p)]：小数类型，可存放实型和整型，**精度 (p) 和范围 (s)**

money **double(5,2)**: 整数和小数**一共占 5 位**,其中小数占 2 位,**都不够精确。**

**DECIMAL**：高精度类型，金额货币优先选择。

类型	大小	范围（有符号）	范围（无符号）	用途
FLOAT	4 字节	(-3.402 823 466 E+38, 1.175 494 351 E-38), 0, (1.175 494 351 E-38, 3.402 823 466 351 E+38)	0, (1.175 494 351 E-38, 3.402 823 466 E+38)	单精度 浮点数值
DOUBLE	8 字节	(1.797 693 134 862 315 7 E+308, 2.225 073 858 507 201 4 E-308), 0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308)	0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308)	双精度 浮点数值
DECIMAL	对DECIMAL (M,D)，如果 M>D，为M+2否则为D+2	依赖于M和D的值	依赖于M和D的值	小数值



## 8.4\_字符类型（了解）

char(size)：定长字符，0 - 255字节，size 指 N 个字符数，若插入字符数超过设定长度，会被截取并警告。varchar(size)：变长字符，0 - 255字节，从 MySQL5 开始支持 65535 个字节，若插入字符数超过设定长度，在非严格模式下会被截取并警告。

CHAR(5)与VARCHAR(5)的对比

插入值	CHAR(5)	占用字节数	VARCHAR(5)	占用字节数
' '	'_____'	5 个字节	' '	1个字节
'1'	'1_____'	5 个字节	'1'	2个字节
'123'	'123____'	5 个字节	'123'	4个字节
'12345'	'12345'	5个字节	'12345'	6个字节
'123456'	'12345'	5个字节	'12345'	6个字节

CHAR(5)固定占用5个字节,不够使用空格补全  
VARCHAR(5)所占用的字节为实际长度加1,结束标识符占1个字节.  
**注意:最后一行只在非严格模式下才会被截取,严格模式下报错**

一般存储大量的字符串，比如文章的纯文本，可以选用 TEXT 系列类型，这个系列都是变长的。

注意: 在 MySQL 中，字符类型必须指定长度，值要使用单引号引起来。相当于Java中字符串 (String,StringBuilder/StringBuffer);

字符类型

类 型	允许的 长度	存 储 空 间
CHAR(M)	M字节	M为0~255之间的整数
VARCHAR(M)	M字节	M为0~65535之间的整数
TINYTEXT	0~255 字节	值的长度+2 个字节
TEXT	0~65535 字节	值的长度+2 个字节
MEDIUMTEXT	0~167772150 字节	值的长度+3 个字节
LONGTEXT	0~4294967295 字节	值的长度+4 个字节

## 8.5\_日期类型（了解）

常用日期和时间类型：DATE、DATETIME

注意: 在 MySQL 中，日期时间值使用单引号引起来。相当于 Java中 Date，Calender。

MySQL的日期与时间类型

整 数 类 型	字节数	取 值 范 围	零 值
YEAR	1	1901~2155	0000
DATE	4	1000-01-01~9999-12-31	0000:00:00
TIME	3	-838:59:59~838:59:59	00:00:00
DATETIME	8	1000-01-01 00:00:00~9999-12-31 23:59:59	0000-00-00 00:00:00
TIMESTAMP	4	19700101080001~20380119111407	0000000000000000

## 8.6\_二进制类型（了解）

二进制类型主要用于存放图形、声音和影像，二进制对象，0-4GB。

开发中，我们一般存储二进制文件保存路径,所以以上的类型非特殊需求不会使用。

BIT，一般存储 0 或 1，存储是 Java 中的 boolean/Boolean 类型的值（需要使用）。

类 型	取 值 范 围
BINARY(M)	字节数为 M，允许长度为 0~M 的定长二进制字符串
VARBINARY(M)	允许长度为 0~M 的变长二进制字符串，字节数为值的长度加 1
BIT(M)	M 位二进制数据，M 最大值为 64
TINYBLOB	可变长二进制数据，最多 255 个字节
BLOB	可变长二进制数据，最多 $(2^{16}-1)$ 个字节
MEDIUMBLOB	可变长二进制数据，最多 $(2^{24}-1)$ 个字节
LONGBLOB	可变长二进制数据，最多 $(2^{32}-1)$ 个字节

## 09\_表的操作（DDL掌握）

表的操作主要是使用 DDL 来创建表和删除表等操作

### 9.1\_创建表（掌握）

步骤：

1. 进入要建表的数据库
2. 输入建表命令

建表语法：

```
CREATE TABLE 表名 (  
    列名1    列的类型    [约束],  
    列名2    列的类型    [约束],  
    ....  
    列名N    列的类型    约束  
);  
-- 注意:最后一行没有逗号
```

例子：创建一张学生表(t\_student) 有id name email age

```
CREATE TABLE t_student (  
    id    BIGINT,  
    name  VARCHAR(15),  
    email VARCHAR(25),  
    age   INT  
);
```

注意：创建表时,不能使用 MySQL 的关键字,保留字.

问题：定义一个订单表 order,order 是数据库中排序的关键字，无法直接创建成功.

```
-- 反例
CREATE TABLE order (
  id    BIGINT,
  name  VARCHAR(15),
  email VARCHAR(25),
  age   INT
);
```

解决：

1. 尽量避免使用关键字,可以使用其他的单词或单词组合来代替
2. 一般情况下,创建表的时候习惯使用 t\_ 做表名的开头
3. 使用反引号 `` 将表名括起来就 ok `order`

```
CREATE TABLE `order`(
  id    BIGINT,
  name  VARCHAR(25),
  email VARCHAR(25),
  age   INT
);
```

## 9.2\_删除表（掌握）

语法：

```
DROP TABLE 表名;
```

实战

```
-- 删除订单表
DROP TABLE `order`;
```

注意：如果表名是数据库的关键字或保留字需要加上反引号 `

## 9.2 表的约束（理解）

表的约束(针对于某一列):

1. 非空约束：NOT NULL，不允许某列的内容为空。
2. 设置列的默认值：DEFAULT。
3. 唯一约束：UNIQUE，在该表中，该列的内容必须唯一
4. **主键约束：PRIMARY KEY**，非空且唯一。
5. **主键自增长：AUTO\_INCREMENT**，从 1 开始，步长为 1。
6. 外键约束：FOREIGN KEY，A表中的外键列. A表中的外键列的值必须参照于B表中的某一列(B表主键)

主键设计：

1. 单字段主键，单列作为主键，**建议使用**。
2. 复合主键，使用多列充当主键，不建议。

主键分为两种：

1. 自然主键:使用有业务含义的列作为主键 (不推荐使用);
2. 代理主键:使用没有业务含义的列作为主键 (**推荐使用**);

**结论：** 使用单字段的代理主键

例子：创建学生表，id为主键自增，name唯一，email不为空，age默认18

```
-- 移除存在的表
DROP TABLE IF EXISTS `t_student`;
CREATE TABLE t_student(
  id      BIGINT      PRIMARY KEY AUTO_INCREMENT,
  name    VARCHAR(25) UNIQUE,
  email   VARCHAR(25) NOT NULL,
  age     INT         DEFAULT 18
);
```

## 10 DML 操作（掌握）

DML 数据操作语句，用户对表的数据多增删改操作。

所有的 DML 操作有一个受影响行，表示 SQL 执行，操作了多少行数据。

### 10.1\_插入操纵

语法：

```
INSERT INTO 表名 (列1,列2,column3...) VALUES(值1,值2,value3...);
```

实战：

```
-- 1.插入完整数据记录
INSERT INTO t_student(name,email,age) VALUES('xiaoming','xiao@',18);
-- 2.插入数据记录一部分
INSERT INTO t_student(name,age) VALUES('xiaodong',19);
-- 3.插入多条数据记录（MySQL特有）
INSERT INTO t_student(name,email,age) VALUES('xiaohong','hong@',17),
('xiaohong2','hong2@',17),('xiaohong3','hong@3',17)
-- 4.插入查询结果
INSERT INTO t_student(name,email,age) SELECT name,email,age FROM t_student
```

**注意事项：** 一次插入操作只插入一行，插入多条数据为 MySQL 特有语法

### 10.2\_修改操纵

语法：

```
UPDATE 表名
SET 列1 = 值1, 列2 = 值2, column3 = value3...
WHERE [条件]
```

实战：

```
-- 将张三改为西门吹水
UPDATE t_student SET name='西门吹水' WHERE name='张三';
-- 将 id 为3 的 name 改为叶孤城, email 改为ye@, age 改为100
UPDATE t_student SET name='叶孤城' WHERE id=3;
```

注意事项：

1. 如果省略了 where 条件子句，则全表的数据都会被修改。
2. 没有 FROM 关键字

## 10.3\_删除操纵

语法：

```
DELETE FROM 表名 WHERE [条件]
```

实战：

```
-- 删除 id 为 2 的学生信息
DELETE FROM t_student WHERE id=2;
-- 删除叶孤城的所有信息
DELETE FROM t_student WHERE name='叶孤城'
```

注意事项：

1. FROM 不能写成 FORM
2. 如果省略了 WHERE 子句，则全表的数据都会被删除

# 11\_DQL 查询操作（掌握）

## 11.1\_语法说明

语法

```
SELECT 列1,列2,列3... FROM 表名 [WHERE];

-- SELECT 选择要查询的列
-- FROM 提供数据源（表、视图或其他的数据源）
```

如果查询列表写 \* 则结果列的显示顺序和创建表时的顺序一致。

```
SELECT * FROM 表名 [WHERE]
```

可以自己调整顺序，在 select 后边加上要查询的列名。

## 11.2\_导入 product.sql

直接把product.sql 拖到 navicat 的 表的显示位置 ->开始

id	productName	dir_id	salePrice	supplier	brand	cutoff	costPrice
1	罗技M90	3	90	罗技	罗技	0.5	35
2	罗技M100	3	49	罗技	罗技	0.9	33
3	罗技M115	3	99	罗技	罗技	0.6	38
4	罗技M125	3	80	罗技	罗技	0.9	39
5	罗技木星轨迹球	3	182	罗技	罗技	0.8	80
6	罗技火星轨迹球	3	349	罗技	罗技	0.87	290

id	:	唯一标识主键
productName	:	货品名
dir_id	:	分类的id
salePrice	:	零售价
supplier	:	供应商
brand	:	品牌名
cutoff	:	折扣
costPrice	:	进货价

### 实战

```
-- 查询所有货品信息
SELECT * FROM product;
-- 查询所有货品的 id,productName,salePrice
SELECT id,productName,salePrice FROM product;
```

## 11.3\_普通查询

### 11.3.1\_消除结果重复数据

#### 语法

```
SELECT DISTINCT 列名, ... FROM 表名;
```

#### 实战

```
-- 查询商品的分类编号
```

### 11.3.2\_算术运算符

对 number 型数据可以使用算数操作符创建表达式 ( + - \* / ) 对 date 型数据可以使用部分算数操作符创建表达式 ( + - )

**运算符优先级：** 1、乘法和除法的优先级高于加法和减法 2、同级运算的顺序是从左到右 3、表达式中使用"括号"可强行改变优先级的运算顺序

实战：

```
-- 查询所有货品的id, 名称和批发价(批发价=卖价*折扣)
SELECT id,productName,salePrice * cutoff FROM product
-- 查询所有货品的id, 名称, 和各进50个的成本价(成本=costPrice)
SELECT id,productName,costPrice * 50 FROM product
-- 查询所有货品的id, 名称, 各进50个, 并且每个运费1元的成本
SELECT id,productName,(costPrice + 1) * 50 FROM product
```

### 11.3.3 设置别名

语法：

```
SELECT 列名 AS 别名 FROM 表名 [WHERE];
SELECT 列名 别名 FROM 表名 [WHERE]
```

作用：

1. 改变列的标题头
2. 作为计算结果的含义
3. 作为列的别名
4. 如果别名中使用特殊字符, 或是强制大小写或有空格时都需要加单引号。

实战

```
-- 查询所有货品的id, 名称, 各进50个, 并且每个运费1元的成本(使用别名)
SELECT id,productName,(costPrice + 1) * 50 AS allPrice FROM product
SELECT id,productName,(costPrice + 1) * 50 allPrice FROM product
```

### 11.3.4 按格式输出

为方便用户浏览查询结果数据, 有时需要设置查询结果的显示格式, 可以使用 CONCAT 函数来连接字符串。

语法：

```
CONCAT(字符串1, 字符串2, ...)
```

实战

```
-- 查询商品的名字和零售价。格式: xxx 商品的零售价为: 000
SELECT CONCAT(productName, '商品的零售价为: ', salePrice) FROM product
```



## 11.4 过滤查询

使用 WHERE 子句限定返回的记录

语法：

```
SELECT <selectList>
FROM 表名
WHERE 条件；
```

注意：WHERE子句在 FROM 子句后

### 11.4.1 比较运算符

=, >, >=, <, <=, !=

不等于：<> 等价 !=

实战:

```
-- 查询商品名为 罗技G9X 的货品信息
SELECT * FROM product WHERE productName='罗技G9X';
-- 查询零售价小于等于 200 的所有货品信息
SELECT * FROM product WHERE salePrice <= 200
-- 查询批发价大于 350 的货品信息
SELECT *,salePrice * cutoff allPrice FROM product WHERE salePrice * cutoff > 350
```

思考：where 后使用别名不行，总结：SELECT 和 WHERE 的执行顺序

**查询语句的字句的执行顺序** 1 FROM 子句: 从哪张表中去查询数据 2 WHERE 子句: 筛选需要哪些行的数据 3 SELECT 子句: 筛选要显示的列

### 11.4.2 逻辑运算符

运算符	含义
AND (&&)	如果组合的条件都是TRUE，返回TRUE
OR (   )	如果组合的条件之一是TRUE，返回TRUE
NOT (!)	如果下面的条件是FALSE，返回 TRUE，如果是 TRUE ,返回 FALSE

实战：

```
-- 查询零售价在300-400（包括300和400）之间的货品信息
SELECT * FROM product WHERE salePrice >= 300 AND salePrice <= 400
-- 查询分类编号为2，4 的所有货品信息
SELECT * FROM product WHERE dir_id = 2 OR dir_id=4
-- 查询编号不为2的所有商品信息
SELECT * FROM product WHERE NOT dir_id = 2
-- 查询货品零售价大于等于250或者成本大于等于200 的所有货品信息
SELECT * FROM product WHERE salePrice >= 250 OR costPrice >= 200
```

## 运算符优先级：

优先级 运算符

1 所有比较运算符 2 NOT 3 AND 4 OR

**注意：括号将跨越所有优先级规则**

-- 分析SQL:

```
SELECT * FROM product WHERE (NOT productName LIKE '%M%' AND salePrice > 100) OR (dir_id = 2)
```



```
WHERE (NOT productName LIKE '%M%' AND salePrice > 100) OR (dir_id = 2)
```

## 11.4.3 范围和集合

**范围匹配：** BETWEEN AND 运算符,一般使用在数字类型的范围上。但对于字符数据和日期类型同样可用。需要两个数据

语法：

```
WHERE 列名 BETWEEN minVaLue AND maxVaLue; -- 闭区间
```

实战：

-- 查询零售价在300-400 之间的货品信息

```
SELECT * FROM product WHERE salePrice BETWEEN 300 AND 400
```

-- 查询零售价不在300-400之间的货品信息

```
SELECT * FROM product WHERE NOT salePrice BETWEEN 300 AND 400
```

**集合查询：** 使用 IN 运算符，判断列的值是否在指定的集合中。

语法：

```
WHERE 列名 IN (值1, 值2....);
```

实战：

-- 查询分类编号为2,4的所有货品的id，货品名称，

```
SELECT id,productName FROM product WHERE dir_id IN (2,4)
```

-- 查询分类编号不为2,4的所有货品的id，货品名称，

```
SELECT id,dir_id,productName FROM product WHERE dir_id NOT IN (2,4)
```

## 11.4.4 判空

**IS NULL:** 判断列的值是否为空值，非空字符串，空字符串使用==判断

语法：

```
WHERE 列名 IS NULL;
```

实战：

```
-- 查询商品名为NULL的所有商品信息。  
SELECT * FROM product WHERE productName IS NULL;  
SELECT * FROM product WHERE supplier = ''
```

结论：使用=来判断只能判断空字符串,不能判断null 的，而使用 IS NULL 只能判断null值，不能判断空字符串

## 11.4.5\_模糊匹配查询

模糊查询数据使用 LIKE 运算符执行通配查询

通配符：

%：表示可有零或多个任意字符

\_：表示需要一个任意字符

语法：

```
WHERE 列名 Like '%M_'
```

实战：

```
-- 查询货品名称匹配'%罗技M9_'的所有货品信息  
SELECT * FROM product WHERE productName LIKE '%罗技M9_'
```

## 11.5 结果排序

使用 ORDER BY 子句将查询结果进行排序 ASC：升序，缺省。 DESC: 降序。 ORDER BY 子句出现在 SELECT 语句的最后。

**排序分类：**

单列的排序

```
-- 选择id, 货品名称, 分类编号, 零售价并且按零售价降序排序  
SELECT id,productName,dir_id,salePrice FROM product ORDER BY salePrice DESC
```

多列的排序

```
-- 选择id, 货品名称, 分类编号, 零售价先按分类编号降序排序, 再按零售价升序排序
SELECT * FROM product ORDER BY dir_id DESC, salePrice ASC
```

注意: 谁在前面谁先排序。

注意: 如果列的别名使用 " 则按此别名进行的排序无效。

```
-- 反例
SELECT id, salePrice 'sp' FROM product ORDER BY 'sp'
```

### DQL 字句的执行顺序

1. FROM 字句: 从哪张表中去查数据
2. WHERE 字句: 筛选需要的行数据
3. SELECT 字句: 筛选需要显示的列的数据
4. ORDER BY 字句: 排序操作

## 11.6 MySQL 的分页查询

当前页: currentPage 每页显示的最大记录数: pageSize = 3

语法:

```
SELECT <selectList>
FROM 表名
[WHERE] LIMIT ?,?
-- 第一个?: 开始行的索引数 beginIndex
-- 第二个?: 每页显示的最大记录数 pageSize
```

第一页: SELECT \* FROM product LIMIT 0, 3  
第二页: SELECT \* FROM product LIMIT 3, 3  
第三页: SELECT \* FROM product LIMIT 6, 3  
第四页: SELECT \* FROM product LIMIT 9, 3

$beginIndex = (currentPage - 1) * pageSize$

实战:

```
-- 每页显示 3条数据
-- 第一页: SELECT * FROM product LIMIT 0,3
-- 第三页: SELECT * FROM product LIMIT 6,3
-- 第八页: SELECT * FROM product LIMIT 21,3
```

### 思考题

带有排序操作的分页查询,是先分页还是先排序 先排序再分页(银行卡操作流水)

```
-- 按照零售价升序排序, 设置每页显示5条数据, 查询第一页
SELECT * FROM product ORDER BY salePrice LIMIT 0,5
```

## 11.7 统计函数

- COUNT(\*) : 统计表中有多少条记录
- SUM(列) : 汇总列的总和
- MAX(列) : 获取某一列的最大值
- MIN(列) : 获取某一列的最小值
- AVG(列) : 获取列的平均值

实战：

```
-- 查询货品表中有多少数据
SELECT COUNT(*) FROM product
-- 计算所有货品的总的进货价
SELECT SUM(costPrice) FROM product
```

## 12\_数据备份和恢复

**忠告:** 在企业修改数据库之前先备份。

MySQL自身的数据库维护：

**通过 cmd 命令进入dos窗口：** 1.导出：mysqldump -u账户 -p密码 数据库名称 > 脚本文件存储地址

```
mysqldump -uroot -padmin jdbcdemo > D:/jdbcdemo_bak.sql
```

2.导入：mysql -u账户 -p密码 数据库名称 < 脚本文件存储地址

```
mysql -uroot -padmin jdbcdemo < D:/jdbcdemo_bak.sql
```

2.导入：mysql -u账户 -p密码 数据库名称 < 脚本文件存储地址 mysql -uroot -padmin jdbcdemo < D:/jdbcdemo\_bak.sql

Navicat工具的导入和导出：Navicat工具的备份和还原：需要工具支持的格式才可以恢复

## 小结

1 MySQL 安装成功

2 掌握 SQL 的分类以及书写规则

3 理解关系型数据库表的概念，理解 ORM 思想

4 掌握连接 MySQL 服务器的方法

5 掌握数据库的操作 SQL 语句

6 记住数据列的常用类型，已经和Java的类映射

7 掌握 DDL 语句，创建表以及表中的主键设置和自动增长设置，删除表的SQL

8 掌握 DML操作

9 掌握 DQL 操作

重点掌握：算术运算符，设置别名，按格式输出

整个过滤查询，结果排序，MySQL 分页

10 了解统计函数，掌握COUNT 函数的使用

11 掌握一种数据备份和恢复的方式

明丁狼教育