MySQL相关笔记

目 录

[第一章 数据库服务器概述 4](#_Toc16689)

[1. 如何保存数据 4](#_Toc14816)

[2. 数据库概述 4](#_Toc21932)

[3. 关系型数据库服务器 4](#_Toc26730)

[第二章 MySQL数据库 4](#_Toc22203)

[1. 数据库发展 4](#_Toc19798)

[2. 关系型数据库的结构 4](#_Toc17742)

[3. 常见的数据库 4](#_Toc10430)

[4. MySQL软件介绍 4](#_Toc16470)

[4.1. xampp软件 4](#_Toc2056)

[4.2. 安装使用MySQL数据库系统 5](#_Toc18674)

[第三章 mysql常用管理命令 5](#_Toc13281)

[1. SQL语句两种执行方式 5](#_Toc18689)

[1.1. 交互模式 5](#_Toc11012)

[1.2. 脚本模式 5](#_Toc30530)

[2. MySQL常用管理命令 5](#_Toc30818)

[第四章 MySQL中的SQL语句——重点 6](#_Toc19812)

[1. SQL语法规范 6](#_Toc26470)

[2. 标准SQL语句的分类 6](#_Toc10171)

[2.1. DDL：Data Define Language定义数据语言 6](#_Toc28513)

[2.2. DML:Data Manipulate Language操作数据语言 6](#_Toc9071)

[2.3. DQL:Data Query Language查询数据语言 7](#_Toc14190)

[2.4. DCL:Data Control Lanuage控制数据语言 7](#_Toc14185)

[3. 日常开发中常用的SQL语句 7](#_Toc24133)

[3.1. 丢弃指定的数据库，如果存在的话 7](#_Toc9535)

[3.2. 创建新的数据库 7](#_Toc19926)

[3.3. 进入创建的数据库 7](#_Toc2801)

[3.4. 创建数据表 7](#_Toc24749)

[3.5. 插入数据 7](#_Toc25409)

[3.6. 查询数据 7](#_Toc3938)

[3.7. 添加数据 8](#_Toc29141)

[3.8. 删除数据 8](#_Toc20136)

[3.9. 更改数据 8](#_Toc16793)

[3.10. 查询数据 8](#_Toc19128)

[3.11. 练习 8](#_Toc20212)

[4. 数据库中的中文乱码 10](#_Toc21288)

[4.1. 计算机如何存储英文字符 10](#_Toc1221)

[4.2. 计算机中如何存储中文存储 11](#_Toc8818)

[4.3. 练习 11](#_Toc10832)

[5. 列类型 13](#_Toc18100)

[5.1. 数值类型——数据可以不加引号 13](#_Toc4477)

[5.2. 日期时间类型——必须加引号 13](#_Toc23922)

[5.3. 字符串类型——必须加引号 13](#_Toc4116)

[5.4. 练习1. 14](#_Toc8798)

[5.5. 练习2.使用恰当的列类型 15](#_Toc25566)

[6. 列约束 18](#_Toc4469)

[6.1. 主键约束——PRIMARY KEY 18](#_Toc24790)

[6.2. 唯一约束——UNIQUE 18](#_Toc7925)

[6.3. 扩展知识：NULL 18](#_Toc456)

[6.3.1. 练习 18](#_Toc23531)

[6.4. 非空约束——NOT NULL 22](#_Toc30489)

[6.5. 默认值约束——DEFAULT 22](#_Toc21627)

[6.6. 检查约束——CHECK 22](#_Toc27781)

[6.7. 外键约束 22](#_Toc7431)

[6.8. 扩展知识：MySQL中的自增列 23](#_Toc17942)

[6.8.1. 练习 23](#_Toc28121)

[7. 简单查询 27](#_Toc21931)

[7.1. 查询特定的列 27](#_Toc26898)

[7.2. 查询所有的列 27](#_Toc21948)

[7.3. 给列取别名 27](#_Toc17618)

[7.4. 只显示不同的记录(合并相同的记录) 27](#_Toc6979)

[7.5. 在查询时执行计算 28](#_Toc9220)

[7.6. 查询结果集的排序 28](#_Toc16373)

[7.7. 条件查询 29](#_Toc31286)

[7.8. 模糊条件查询 32](#_Toc20266)

[7.9. 分页查询 33](#_Toc31573)

[7.9.1. 练习1 33](#_Toc4907)

[7.9.2. 练习2 33](#_Toc24152)

[7.9.3. 练习3 33](#_Toc19896)

[7.9.4. 练习4 34](#_Toc12817)

[8. 复杂查询 34](#_Toc15883)

[8.1. 聚合查询 34](#_Toc15809)

[8.2. 分组查询 35](#_Toc10144)

[8.3. 子查询 36](#_Toc29483)

[8.3.1. 练习 36](#_Toc26760)

[8.3.2. 练习 37](#_Toc26150)

[8.4. 多表查询 38](#_Toc25367)

[8.4.1. 内连接 INNER JOIN...ON...——和SQL-92结果一致 38](#_Toc7699)

[8.4.2. 左外连接 LEFT OUTER JOIN...ON... 39](#_Toc24099)

[8.4.3. 右外链接 RIGHT OUTER JOIN...ON... 39](#_Toc30616)

[8.4.4. 全连接 FULL JOIN 39](#_Toc22585)

[9. 综合的语法 40](#_Toc16778)

* 1. 数据库服务器概述
     1. 如何保存数据

内存，速度快，空间有限；

自定义文件，速度慢；

Access/excel；

数据库存储。

* + 1. 数据库概述

Database:按照特定的数据结构存储数据的软件。

* + 1. 关系型数据库服务器

SERVER->Database->Table->Row->Column

* 1. MySQL数据库

Oracle分支https://www.mysql.com/

Martin分支https://mariadb.org/

* + 1. 数据库发展

网状数据库->层次型数据库->关系型数据库->非关系型数据库

* + 1. 关系型数据库的结构

RDBMS中的数据在逻辑结构上通常呈现下述层级关系：

DATABASE->TABLE->ROW->COLUMN

* + 1. 常见的数据库

SQlite、MySQL、PostgreSQL、SQLServer、Oracle、DB2

LAMP组合：Linux+Apache+MySQL+PHP

（适合中小型站点，需要快速部署web应用）

* + 1. MySQL软件介绍
       1. xampp软件

xampp 服务器套装；包含有多个开源服务器：Apache，MySQL，PHP

mysqld.exe 服务器启动文件

d->deamon 精灵，守护者，服务器软件

mysql.exe 连接数据库服务器

data 保存数据库服务器中的数据

* + - 1. 安装使用MySQL数据库系统

1. **服务器端**：下载并安装mysql服务器软件
2. **服务器端**：启动mysql服务器

保证端口 3306 被mysql占用

1. **客户端**：使用mysql.exe连接数据库
2. **语法**

mysql.exe [-h服务器地址] [-p服务器监听端口] [-u登录用户名] [-p登录密码] [数据库名]

mysql.exe -h127.0.0.1 -P3306 -uroot -p

简写形式： mysql –uroot

* 1. mysql常用管理命令
     1. SQL语句两种执行方式
        1. 交互模式

在客户端输入一行，点击回车，服务器执行一行。适用于临时性的查看数据。

mysql -uroot 回车

* + - 1. 脚本模式

客户端把要执行的命令写在一个文本文件中，一次性提交服务器执行。适用于批量的增删改查。

mysql -uroot < c:/xampp/htdocs…../2.sql 回车

* + 1. MySQL常用管理命令

前提：必须要连接成功mysql数据库

show databases; 查看有哪些数据库

quit; 退出mysql

use数据库名称; 使用数据库(进入数据库)

show tables; 显示当前数据库中所有的数据表

desc数据表名称; 描述表中都有哪些列(表头);

* 1. MySQL中的SQL语句——重点

SQL：Structrued Query Language,结构化的查询语言，用于操作关系型数据服务器中的数据——**增删改查**。

SQL语言最早是ANSI提出的，后提交给ISO最终成为了数据库行业标准的语言。分为SQL-87、SQL-92、SQL-99，被绝大多数关系型数据库所支持。

* + 1. SQL语法规范

1. 每条SQL语句以英文的分号结尾，一条语句可以跨越多行(use 除外)，见到分号认为语句结束。
2. 假如第N条语句语法错误，则此语句之后的所有语句都不会再执行。
3. SQL语句**不区分大小写**，习惯上关键字用大写，非关键字用小写。
4. SQL语句分为单行注册(# )和多行注释(/\* … \*/ )，注释的语句不被服务器执行。
   * 1. 标准SQL语句的分类
        1. DDL：Data Define Language定义数据语言

CREATE（新建）/ DROP（丢弃）/ ALTER(修改)/ TRUNCATE（删除）

|  |
| --- |
| **注：**  当你不再需要该表时，用drop；  当你仍要保留该表，但要删除所有记录时，用truncate；  当你要删除部分记录时（always with a WHERE clause),用delete. |

* + - 1. DML:Data Manipulate Language操作数据语言

INSERT（插入）/ DELETE（删除）/ UPDATE（修改）

* + - 1. DQL:Data Query Language查询数据语言

SELECT（选择）

* + - 1. DCL:Data Control Lanuage控制数据语言

GRANT(授权，管理数据库)/ REVOKE(收权)

* + 1. 日常开发中常用的SQL语句
       1. 丢弃指定的数据库，如果存在的话

DROP DATABASE IF EXISTS jd;

* + - 1. 创建新的数据库

CREATE DATABASE jd;

* + - 1. 进入创建的数据库

USE jd;

* + - 1. 创建数据表

CREATE TABLE emp

(

id INT,

name VARCHAR(16),

addr VARCHAR(16),

phone VARCHAR(11)

);

* + - 1. 插入数据

INSERT INTO emp

VALUES('1', 'tom', 'bj', '13111111111');

* + - 1. 查询数据

SELECT \*

FROM emp;

* + - 1. 添加数据

INSERT INTO user

VALUES('1', 'dingding'

…),

('2','dangdang')…;

* + - 1. 删除数据

DELETE FROM user WHERE uid='1';

* + - 1. 更改数据

UPDATE user SET upwd='654321',sex='m'WHERE uid='2';

* + - 1. 查询数据

SELECT\*

FROM user;

* + - 1. 练习

创建一个脚本文件 test.sql；丢弃数据库test，如果存在；创建数据库test，进入该数据库，创建数据表user，包含列uid，uname，upwd，email，phone，sex，userName(真实姓名)，registerTime(注册时间 2018-08-20)； 插入5条记录，查询最终的数据

#设置客户端连接服务器端使用的编码为utf8

SET NAMES UTF8;

#丢弃数据库test，如果存在

DROP DATABASE IF EXISTS test;

#创建数据库test，使用utf8编码

CREATE DATABASE test CHARSET=UTF8;

#进入该数据库

USE test;

#创建数据表 user

CREATE TABLE user

(

uid INT,

uname VARCHAR(20),

upwd VARCHAR(16),

email VARCHAR(32),

phone VARCHAR(11),

sex VARCHAR(1),

userName VARCHAR(5),

registerTime VARCHAR(10)

);

#插入数据

INSERT INTO user

VALUES('1', 'dingding', '123456', 'dingding@qq.com', '13112345678', 'm', '张三', '2018-9-4');

INSERT INTO user

VALUES

('2', 'dangdang', '123456', 'dangdang@jd.com', '18812345678', 'w', 'KATE', '2018-8-31'),

('3', 'baidu', '123456', 'baidu@163.com', '19912345678', 'm', 'JERRY', '2018-7-1');

#更改数据

UPDATE user SET upwd='654321',email='baidu@sina.com' WHERE uid='3';

#删除数据

DELETE FROM user WHERE uid>'1';

#查询数据

SELECT \*

FROM user;

* + 1. 数据库中的中文乱码
       1. 计算机如何存储英文字符

**ASCII：**总共有128个，对所有的英文字符以及符号进行的编码。AB 6566

http://ascii.911cha.com/

**Latin-1：**总共有256个，兼容ASCII码，同时对欧洲的符号进行了编码。

|  |
| --- |
| **注：**  MySQL默认使用Latin-1编码，而Latin-1不支持中文字符，直接插入中文会出现乱码 |

* + - 1. 计算机中如何存储中文存储

**GB 2312**对六千多个中文字符进行编码，主要是简体中文，兼容ASCII码。

**GBK**对两万多个中文字符进行编码，简繁体，兼容ASCII。

**Unicode**对世界上主要国家的语言进行了编码，兼容ASCII码；有UTF-8，UTF-16，UTF-32三种存储方案。

|  |  |
| --- | --- |
| **解决方案：**   1. sql脚本文件存储使用UTF-8 2. 客户端连接服务器使用UTF-8 3. 服务器存储数据使用UTF-8 |  |

* + - 1. 练习

创建一个脚本文件01\_dangdang.sql，先删除再创建数据库dangdang,使用utf8编码；进入数据库，创建数据表book，包含bid，title，price，pubDate(出版时间)，isJia(是/否)是否加价购；插入4条数据；更改第2条数据，删除第3条数据。

#设置客户端连接服务器使用utf8

SET NAMES UTF8;

#丢弃数据dangdang，如果存在

DROP DATABASE IF EXISTS dangdang;

#创建数据库dangdang，使用utf8编码存储数据

CREATE DATABASE dangdang CHARSET=UTF8;

#进入数据库

USE dangdang;

#创建数据表 book

CREATE TABLE user

(

bid INT,

title VARCHAR(40),

price DECIMAL(5,2), #99999.99

pubDate VARCHAR(10),

isJia BOOL(1)

);

#插入数据

INTSERT INTO book VALUES

('1','三国演义','105','2018-6-6','是'),

('2','水浒传','88','2001-1-1','否'),

('3','西游记','22','1998-3-3','是')，

('4','红楼梦','33','2010-1-2','是');

#更改数据

UPDATE book SET title='孟子1',pubDate='2001-2-2'

WHERE bid='2';

#删除数据

DELETE

FROM book

WHERE bid='3';

* + 1. 列类型

创建数据表的时候，指定的列所保存的数据类型

create table book(bid列类型);

* + - 1. 数值类型——数据可以不加引号

TINYINT 微整型，占用1个字节，存储范围-128~127

SMALLINT 小整型，占用2个字节，存储范围-32768~32767

INT 整型，占用4个字节，存储范围-2147483648~2147483647

BIGINT 大整型，占用8个字节，存储范围很大

FLOAT 单精度浮点型，占4个字节，取出几位作为指数使用,实际存储的范围要INT大的多。

DOUBLE 双精度浮点型，占8个字节，实际存储范围要比BIGINT大的多1.79E308

DECIMAL(M,D) 定点小数，不会产生误差,M总共有效位数(不包含小数点)，D表示小数点后的有效位数。

BOOL 布尔型只有两个值TRUE/1和FALSE/0，最终的存储方式使用数字1和0

* + - 1. 日期时间类型——必须加引号

DATE 日期类型例如：2018-09-04

TIME 时间类型例如：14:50:30

DATETIME 日期时间类型例如：2018-09-04 14:50:30

* + - 1. 字符串类型——必须加引号

VARCHAR(M) 变长字符串，查询速度相对比较慢，不存在空间浪费。M最多存储65535

CHAR(M) 定长字符串，例如手机号，身份证号…，查询速度比较快，可能存在空间浪费。M最多存储255

TEXT(M) 大型变长字符串，M最多存储2G

|  |  |  |
| --- | --- | --- |
|  | VARCHAR(5) | CHAR(5) |
| a | a\0 | a\0\0\0\0 |
| ab | ab\0 | ab\0\0\0 |
| abc | abc\0 | abc\0\0 |
| 一二 | 一二\0 | 一二\0\0\0 |

存储时间，存储的是距离1970年1月1日的毫秒数。

1000毫秒=1秒

* + - 1. 练习1.

创建一个脚本文件02\_test.sql，设置编码为utf8，丢弃数据库，再创建数据库test，进入该数据库，创建保存笔记本信息的表laptop，包含列lid，title，price，stockCount(库存量)，shelfTime(上架时间)，isIndex(是否显示在首页)；插入4条记录

#设置客户端连接服务器端使用的编码为utf8

SET NAMES UTF8;

#丢弃数据库test，如果存在

DROP DATABASE IF EXISTS test;

#创建数据库test，使用utf8编码

CREATE DATABASE test CHARSET=UTF8;

#进入该数据库

USE test;

#创建数据表 laptop

CREATE TABLE laptop

(

lid INT,

title VARCHAR(20),

price DECIMAL(8,2),

stockCount SMALLINT,

sheltTime DATE,

isIndex BOOL

);

#插入数据

INSERT INTO laptop

VALUES

(1, 'a', 4999, 100, '2018-1-5', 1),

(2, 'b', 5999, 400, '2017-6-6', 0),

(3, 'c', 5299, 500, '2018-5-15', 0),

(4, 'd', 6199, 600, '2016-7-7', 1);

* + - 1. 练习2.使用恰当的列类型

创建脚本文件03\_test.sql，设置存储的编码为utf8；丢弃数据库test，如果存在；创建数据库test，进入该数据库；

创建保存部门信息的表dept，包含did，dname，empCount员工数量；

10 研发部 3

20 市场部 2

30 运营部 2

创建保存员工信息的表emp，包含eid，ename，sex，phone，birthday，salary工资，deptId部门的编号

1 tom 1 18711111111 1989-1-1 8000 20

#设置连接服务器使用的编码UTF-8

SET NAMES UTF8

#丢弃数据库test，如果存在

DROP DATABASE IF EXISTS test;

#创建数据库test，设置UTF-8

CREATE DATEBASE test CHARSET=UTF8;

#进入数据库

USE test;

#创建数据表dept

CREATE TABLE dept

(

did SMALLINT,

dname VARCHAR(10),

empCount SMALLINT

);

#插入数据

INSET INTO dept VALUES

(10,'研发部',3),

(20,'市场部',2),

(30,'运营部',2);

#创建员工表

CREATE TABLE emp

(

eid INT,

dname VARCHAR(6),

sex BOOL,

phone CHAR(11),

birthday DATE,

salary DECIMAL(7,2), #99999 .99

deptId SMALLINT

);

#插入数据

INSET INTO emp VALUES

(1,'Tom',1,'18711111111','1989-1-1',80000,20),

(2,'Jay',1,'13112345678','1988-2-2',70000,10),

(3,'Gay',0,'15122222222','1991-3-3',60000,30);

* + 1. 列约束

MySQL可以对插入表中的数据进行特定的验证，只有满足条件的数据才允许插入成功！

例如：一个人的年龄范围是在0~120之间，商品的价格只能是正数…

* + - 1. 主键约束——PRIMARY KEY

声明了主键约束的列，不允许插入相同的数据。不应插入NULL；一个表中只能使用一个主键，使用主键约束会加快查找的速度。

* + - 1. 唯一约束——UNIQUE

声明了唯一约束的列上不能插入重复的值，允许插入一个或者多个NULL；一个表中可以使用多个唯一约束。

* + - 1. 扩展知识：NULL

NULL：空的，空缺的。

通常用于某个列，还不能确定具体要保存的数据。

例如：年终奖，部门经理暂时还不能确定。

**注意：NULL和任何值都不相等，甚至自身。**

* + - * 1. 练习

创建脚本文件xuezi.sql，设置存储的编码；删除数据库xuezi，如果存在；创建数据库xuezi，进入该数据库。

创建笔记本型号表 laptop\_family ( fid，fname，laptopCount每个型号的笔记本数量 )

10 戴尔燃700 3

20 小米Air 2

30 联想小新系列 2

创建笔记本信息表laptop(lid，title，price，shelfTime，stockCount库存数量，isOnsale 是否在售，fid隶属的系列)

插入7条记录；修改第3条记录，删除第5条记录。

#设置客户端连接使用的编码

SET NAMES UTF8;

#丢弃数据库xuezi，如果存在

DROP DATABASE IF EXISTS xuezi;

#创建数据库，存储的编码

CREATE DATABASE xuezi CHARSET=UTF8;

#进入数据库xuezi

USE xuezi;

#创建数据表laptop

CREATE TABLE laptop\_family

(

fid SMALLINT PRIMARY KEY,

fname VARCHAR(16) UNIQUE,

laptopCount SMALLINT NOT NULL

);

#插入数据

INSERT INTO laptop\_family

VALUES

(10, '戴尔燃700', 3),

(20, '小米AIR', 2),

(30, '联想小新系列', 2),

(40, NULL, NULL);

#创建笔记本信息表

CREATE TABLE laptop(

lid INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(128) UNIQUE,

price DECIMAL(8,2) DEFAULT 3000,

shelfTime DATE,

ctockCount SMALLINT DEFAULT 9999,

isOnsale BOOL,

fid SMALLINT,

FOREIGN KEY(fid) REFERENCES laptop\_family (fid)

);

#插入数据

INSERT INTO laptop

VALUES

(NULL, '联想1', DEFAULT, '2018-5-1', DEFAULT, TRUE, 30),

(NULL, '小米1', 5100, '2018-7-1', 1000, FALSE, 20),

(NULL, '戴尔1', 6199, '2018-6-1', 5001, TRUE, 10),

(NULL, '联想2', 4699, '2018-5-1', 8000, TRUE, 30),

(NULL, '小米2', 5100, '2018-7-1', 1000, FALSE, 20),

(6, '戴尔2', 6199, '2018-6-1', 500, 1, 10),

(NULL, '戴尔3', 6199, '2018-6-1', 500, 1, 10);

INSERT INTO laptop

(lid,title)

VALUES(NULL, '戴尔4000');

#删除的笔记本是小米，小米系列也应该减1

DELETE FROM laptop WHERE lid=5;

#在原来的基础上减1

UPDATE laptop\_family SET laptopCount=laptopCount-1 WHERE fid=20;

#删除一款戴尔笔记本，戴尔系列也应该减1

DELETE FROM laptop WHERE lid=7;

#在原来的基础上减1

UPDATE laptop\_family SET laptopCount=laptopCount-1 WHERE fid=10;

* + - 1. 非空约束——NOT NULL

禁止插入NULL，如果插入的值是NULL，不会报错；字符串保存的值变成空的，数值型保存的值变成了0。

* + - 1. 默认值约束——DEFAULT

可以使用DEFAULT关键字为列声明默认值。

有两种方法可以使用默认值

INSERT INTO user

VALUES(1, ’tom’, DEFAULT);

INSERT INTO user

(uid,uname)

VALUES(1, ’tom’);

* + - 1. 检查约束——CHECK

检查约束对要插入的数据进行检验

CREATE TABLE student

(

score TINYINT CHECK(score>=0 AND score<=100)

);

**MySQL不支持检查约束，会降低数据的插入速度。**

* + - 1. 外键约束

声明的外键约束的列，取值必须在另一个表的主键列上出现过。

CREATE TABLE laptop(

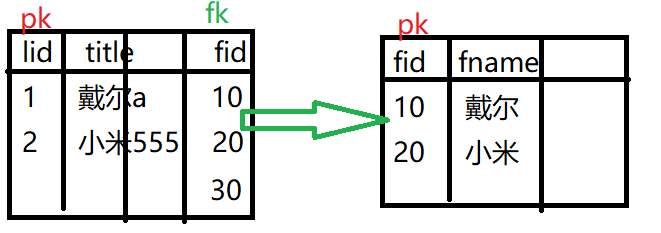
…

FOREIGN KEY

(列名称) REFERENCES laptop\_family

(列名称)

);



* + - 1. 扩展知识：MySQL中的自增列

AUTO\_INCREMENT，只能用于列类型为整型的主键列上。

作用：在原来最大的值基础上加1

允许手动赋值，如果设置的是NULL，就会自增。

* + - * 1. 练习

创建03\_test.sql， 设置存储的编码为utf8，丢弃数据库test，如果存在，创建数据库，进入该数据库；

创建部门表dept，包含列(did，dname)

10 研发部 development

20 测试部

30 运营部

40 市场部

创建员工表emp，包含(eid，ename，sex，phone，birthday，salary，deptId)；

插入15条记录，有一个员工部门为NULL，其中市场部没有员工

#设置客户端连接使用的编码UTF8

SET NAMES UTF8;

#丢弃数据库test，如果存在

DROP DATABASE IF EXISTS test;

#创建数据库，存储的编码UTF8

CREATE DATABASE test CHARSET=UTF8;

#进入数据库test

USE test;

#创建数据表部门表dept

CREATE TABLE dept(

did SMALLINT PRIMARY KEY AUTO\_INCREMENT,

dname VARCHAR(16) UNIQUE

);

#插入数据

INSERT INTO dept

VALUES

(10, '研发部'),

(20, '测试部'),

(30, '运营部'),

(40, '市场部');

#创建员工表emp

CREATE TABLE emp(

eid INT PRIMARY KEY AUTO\_INCREMENT,

ename VARCHAR(12),

sex BOOL,

phone CHAR(11),

birthday DATE,

salary DECIMAL(7,2),

deptId SMALLINT,

FOREIGN KEY

(deptId) REFERENCES dept

(did)

);

#插入数据

INSERT INTO emp

VALUES

(NULL, 'A', TRUE, '13112345678', '1991-1-1', 5000, 10),

(NULL, 'B', FALSE, '13212345678', '1992-1-1', 6000, 20),

(NULL, 'C', TRUE, '13132345678', '1993-1-1', 7000, 30),

(NULL, 'D', FALSE, '13412345678', '1994-1-1', 50000, 10),

(NULL, 'E', TRUE, '13152345678', '1991-5-1', 7000, 20),

(NULL, 'F', FALSE, '13612345678', '1996-1-1', 5000, 30),

(NULL, 'G', TRUE, '13172345678', '1991-7-1', 6000, 10),

(NULL, 'H', FALSE, '13812345678', '1991-8-1', 5000, 20),

(NULL, 'I', TRUE, '13192345678', '1999-1-1', 8000, 30),

(NULL, 'G', TRUE, '13102345678', '1991-9-1', 9000, 30),

(NULL, 'H', FALSE, '13132345678', '1991-1-1', 5000, 20),

(NULL, 'I', 1, '13114345678', '1991-1-8', 9000, 20),

(NULL, 'J', TRUE, '13115345678', '1995-5-1', 5300, 10),

(NULL, 'K', 0, '13117345678', '1996-6-1', 5200, 20),

(NULL, 'L', TRUE, '13116345678', '1993-2-4', 5000, NULL);

* + 1. 简单查询
       1. 查询特定的列

例：查询所有员工的编号和姓名。

SELECT eid, ename FROM emp;

练习：查询员工的姓名，性别，工资，生日。

SELECT ename, sex, salary, birthday FROM emp;

* + - 1. 查询所有的列

SELECT \* FROM emp;

* + - 1. 给列取别名

例：查询员工的姓名，工资，列名称使用汉字。

SELECT ename AS 姓名, salary 工资 FROM emp;

SELECT ename 姓名, salary 工资 FROM emp;

AS**可省略**，但要留空格，注意逗号需为**英文符号**

练习：查询所有员工的编号，姓名，性别，生日，全部使用中文别名。

SELECT eid AS 编号, ename AS 姓名, sex AS 性别, birthday AS 生日 FROM emp;

* + - 1. 只显示不同的记录(合并相同的记录)

例：查询哪些部门下有员工(员工都在哪些部门)?

SELECT DISTINCT deptId FROM emp;

练习：查询公司都有哪些性别的员工？

SELECT DISTINCT sex FROM emp;

* + - 1. 在查询时执行计算

例：计算3\*4/5+2

SELECT 3\*4/5+2 AS 结果;

例：查询出所有员工的姓名和年薪是多少

SELECT ename,salary\*12 AS 年薪 FROM emp;

练习：假设每个员工工资增加500，年终奖是5000，查询所有员工的姓名和年薪是多少

SELECT ename,(salary+500)\*12+5000 AS 年薪 FROM emp;

* + - 1. 查询结果集的排序

默认按照升序排列(asc)

排序可以按照数值，日期，字符串排列。

例：查询出所有部门的信息，结果按照部门编号升序排序

SELECT \* FROM dept ORDER BY did ASC; #ASC—ascendant

例：查询出所有部门的信息，结果按照部门编号降序排序

SELECT \* FROM dept ORDER BY did DESC; #DESC—descendant

练习：查询所有的员工，结果按照工资由低到高排序；

SELECT \* FROM emp ORDER BY salary ASC;

练习：查询所有的员工，结果按照年龄由高到低排序；

SELECT \* FROM emp ORDER BY birthday ASC;

练习：查询所有的员工，结果按照姓名升序排序

SELECT \* FROM emp ORDER BY ename ASC;

练习：查询所有的员工，结果按照工资升序排序，如果工资相同，按照姓名升序排序。

SELECT \* FROM emp ORDER BY salary ASC,ename ASC;

练习：查询所有的员工，结果按照生日升序排序，如果生日相同，按照工资降序排序。

SELECT \* FROM emp ORDER BY birthday ASC,salary DESC;

* + - 1. 条件查询

例：查询出编号为5的员工所有信息

SELECT \* FROM emp WHERE eid=5;

练习：查询出姓名为KING的员工的编号、工资、生日

SELECT eid,salary,birthday FROM emp WHERE ename='king';

练习：查询出20号部门下所有员工的信息

SELECT \* FROM emp WHERE deptId=20;

练习：查询出所有的男员工信息

SELECT \* FROM emp WHERE sex=1;

例：查询出工资大于等于5000的员工所有信息

SELECT \* FROM emp WHERE salary>=5000;

|  |
| --- |
| **SQL语句中比较运算符：> >= = != < <=** |

练习：查询出1991-1-1之后出生的员工所有信息

SELECT \* FROM emp WHERE birthday>'1991-1-1';

练习：查询出不在10号部门的所有员工信息

SELECT \* FROM emp WHERE deptId!=10;

练习：查询出暂时没有部门的员工信息(部门编号为NULL)

SELECT \* FROM emp WHERE deptId IS NULL;

练习：查询出有明确部门的员工信息

SELECT \* FROM emp WHERE deptId IS NOT NULL;

练习：查询出工资大于6000的女员工所有信息

SELECT \* FROM emp WHERE salary>6000 AND sex=0;

练习：查询出工资在5000~5999之间的员工信息

SELECT \* FROM emp WHERE salary>=5000 AND salary<=5999;

SELECT \* FROM emp WHERE salary BETWEEN 5000 AND 5999;

练习：查询出在1991年出生的员工信息

SELECT \* FROM emp WHERE birthday>='1991-1-1' AND birthday<='1991-12-31';

SELECT \* FROM emp WHERE birthday BETWEEN '1991-1-1' AND '1991-12-31';

练习：查询出工资小于4000和大于8000的员工信息

SELECT \* FROM emp WHERE salary NOT BETWEEN 4000 AND 8000;

练习：查询出在1990年之前和1993之后出生的员工信息

SELECT \* FROM emp WHERE birthday<'1990-1-1' OR birthday>'1993-12-31';

SELECT \* FROM emp WHERE birthday NOT BETWEEN '1990-1-1' AND '1993-12-31';

|  |
| --- |
| **BETWEEN ... AND ... 在两者之间**  **NOT BETWEEN...AND...不在两者之间** |

练习：查询出在10和30号部门的员工信息

SELECT \* FROM emp WHERE deptId=10 OR deptId=30;

SELECT \* FROM emp WHERE deptId IN(10,30);

练习：查询出在10、30、50、80号部门的员工信息

SELECT \* FROM emp WHERE deptId=10 OR deptId=30 OR deptId=50 OR deptId=80;

SELECT \* FROM emp WHERE deptId IN(10,30,50,80);

练习：查询出不在10和30号部门的员工信息

SELECT \* FROM emp WHERE deptId!=10 AND deptId!=30;

SELECT \* FROM emp WHERE deptId NOT IN(10,30);

|  |
| --- |
| **IN(10,30) 满足其中一个条件**  **NOT IN(10,30) 不满足任意一个条件** |

练习：删除工资在5000~10000之间的员工信息

DELETE FROM emp WHERE salary BETWEEN 5000 AND 10000;

* + - 1. 模糊条件查询

例：查询出姓名中包含字母E的员工所有信息

SELECT \* FROM emp WHERE ename LIKE '%e%';

练习：查询出姓名以E结尾的员工所有信息

SELECT \* FROM emp WHERE ename LIKE '%e';

练习：查询出姓名中倒数第二个字符是E的员工所有信息

SELECT \* FROM emp WHERE ename LIKE '%e\_';

|  |
| --- |
| % 任意多个字符 >=0  \_ 任意一个字符 =1  **注意：%和\_必须要结合着LIKE关键字使用** |

* + - 1. 分页查询

分页显示：查询结果太多的记录行，一次显示不完，可以分页显示。不同的数据库有不同的查询方式。MySQL的分页查询最简单。

**语法：**

SELECT \* FROM emp LIMIT start,count;

**start：**是一个数字，表示从结果集中的哪一行开始读取

**count：**是一个数字，表示读取的最多行数

**start=(页码-1)\*count**

* + - * 1. 练习1

假设每一页对多显示4条记录

第1页 SELECT\*FROM emp LIMIT 0,4;

第2页 SELECT\*FROM emp LIMIT 4,4;

第3页 SELECT\*FROM emp LIMIT 8,4;

第4页 SELECT\*FROM emp LIMIT 12,4;

第5页 SELECT\*FROM emp LIMIT 16,4;

* + - * 1. 练习2

假设每一页对多显示5条记录

第1页 SELECT\*FROM emp LIMIT 0,5;

第2页 SELECT\*FROM emp LIMIT 5,5;

第3页 SELECT\*FROM emp LIMIT 10,5;

* + - * 1. 练习3

查询工资最低的5个员工。

SELECT\*FROM emp ORDER BY salary ASC LIMIT 0,5;

* + - * 1. 练习4

查询男员工中工资最高的5个人

SELECT\*FROM emp WHERE sex=1 ORDER BY salary DESC LIMIT 0,5;

* + 1. 复杂查询
       1. 聚合查询

**聚合函数**：COUNT(...)/SUM(...)/AVG(...)/MAX(...)/MIN(...)

**函数**：功能体，接收若干个数据，返回特定的计算结果。

例：查询出所有员工的数量

SELECT COUNT(eid) FROM emp;

SELECT COUNT(ename) FROM emp;

SELECT COUNT(\*) FROM emp;

SELECT COUNT(deptId) FROM emp; #无法识别NULL

练习：查询男员工的数量

SELECT COUNT(sex) FROM emp WHERE sex=1;

例：查询出所有的员工的工资总和

SELECT SUM(salary) FROM emp;

练习：产出平均工资是多少

SELECT SUM(sarlary)/SUM(\*) FROM emp;

SELECT AVG(salary) FROM emp;

例：查询出工资最高的员工

SELECT MAX(salary) FROM emp;

SELECT MIN(salary) FROM emp;

练习：查询年龄最大的员工

SELECT MIN(birthday) FROM emp;

例：查询每个部门的员工数量是多少

SELECT deptId,COUNT(\*),SUM(salary) FROM emp GROUP BY deptId;

* + - 1. 分组查询

**分组查询：**只能查询分组的条件和聚合函数。

练习：查询出男女员工的平均工资

SELECT sex,ACG(salary) FROM emp GROUP BY sex;

练习：查询出男女员工的平均工资、最多、最少

SELECT sex,ACG(salary),MAX(salary),MIN(salary) FROM emp GROUP BY sex;

|  |
| --- |
| **year()**获取日期中的年份；  **month()**获取日期中的月份；  **day()**获取日期中的日子； |

例：查询1991年出生的员工

SELECT \* FROM emp WHERE year(birthday)=1991;

例：查询3月份出生的员工

SELECT \* FROM emp WHERE month(birthday)=3;

例：查询5号份出生的员工

SELECT \* FROM emp WHERE day(birthday)=5;

* + - 1. 子查询

本质上就是一个SQL语句的查询**结果是另**一SQL语句的查询**条件**

例：查询出研发部所有员工的信息

步骤1：查询出研发部的部门编号——10

SELECT did FROM dept WHERE dname='研发部';

步骤2：查询出员工表中部门编号为10的员工

SELECT \* FROM emp WHERE deptId=10;

综合：

SELECT \* FROM emp WHERE deptId=(SELECT did FROM dept WHERE dname='研发部');

* + - * 1. 练习

查出工资比tom高的所有员工的信息

步骤1：查询出tom的工资——6000

SELECT salary FROM emp WHERE ename='tom';

步骤2：查询出工资大于6000的员工

SELECT \* FROM emp WHERE salary>6000;

综合：

SELECT \* FROM emp WHERE salary>( SELECT salary FROM emp WHERE ename='tom');

* + - * 1. 练习

查询出与tom同一年出生的员工信息

法一、

步骤1：查询出tom的生日——1990-5-5

SELECT birthday FROM emp WHERE ename='tom';

步骤2：查询出1990年出生的员工信息

SELECT\*FROM emp WHERE year(birthday)=year('1990-5-5');

综合：

SELECT \* FROM emp WHERE year(birthday)=year((SELECT birthday FROM emp WHERE ename='tom'));

法二、

步骤1：查询出tom的生日——1990

SELECT year(birthday) FROM emp WHERE ename='tom';

步骤2：查询出1990年出生的员工信息

SELECT \* FROM emp WHERE year(birthday)=1990;

综合：

SELECT \* FROM emp WHERE year(birthday)=(SELECT year(birthday) FROM emp WHERE ename='tom');

* + - 1. 多表查询

例：查询出所有员工的姓名及其部门名称

SELECT ename,dname FROM emp,dept;

**错误：**笛卡尔积现象。

多表查询如何避免“笛卡尔积”，为查询添加条件。

SELECT ename,dname FROM emp,dept WHERE deptId=did;

|  |
| --- |
| 笛卡尔积  现象：  在进行多表连接时，表一有5条记录，表二有3条记录，那么对于第一张表而言有5种选择，而对于第二张表来说有3种选择。  所以结果就是5\*3种选择。  解决：  多表连接时如果要避免返回笛卡尔积，必须确保至少一方表是唯一的，否则在能查到记录的情况下，不论哪一种连接方式一定会返回笛卡尔积记录集的。 |

|  |
| --- |
| 上述的语法是SQL-92中的多表查询语法，无法查询出没有部门的员工，也不能查询出没有员工的部门。  SQL-99中提出了新的多表查询方法。 |

* + - * 1. **内连接 INNER JOIN...ON...**——和SQL-92结果一致

**语法：**

SELECT 列名列 FROM 表名1 [INNER] JOIN 表名2 ON 表名1.列名=表名2.列名

**[INNER]**：可选关键字，指定连接类型为内连接，为默认连接类型；

**表名1 JOIN 表名2**：等值连接对象的连接体；

**ON 表名1.列名=表名2.列名**：等值连接的连接条件，通常条件下为“ON主键=外键”；

|  |
| --- |
| 注意：  由于SQL-92和SQL-99的区别  WHERE——为过滤条件；  ON——为关联条件；  以此来避免SQL-92中过滤条件和关联条件混淆不清的问题 |

SELECT ename,dname FROM emp INNER JOIN dept ON deptId=did;

JOIN 后跟被连接的表；ON 后接条件

* + - * 1. 左外连接 LEFT OUTER JOIN...ON...

SELECT ename,dname FROM emp LEFT OUTER JOIN dept ON deptId=did;

查询结果是左侧所有记录都显示；OUTER可以省略。

即：两个表在连接过程中除返回满足连接条件的行以外，还返回左表中不满足条件的行。

* + - * 1. 右外链接 RIGHT OUTER JOIN...ON...

SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptId=did;

查询结果是**右侧所有记录都显示**；**OUTER可以省略**。

即：两个表在连接过程中除返回满足连接条件的行以外，还返回右表中不满足条件的行。

* + - * 1. 全连接 FULL JOIN

显示左侧和右侧所有的记录——**MySQL不支持**

即：两个表在连接过程中除返回满足连接条件的行以外，还返回两个表中不满足条件的所有行。

MySQL的全连接：**UNION**; **UNION ALL**

|  |
| --- |
| 两个查询结果：  集中合并记录 **UNION**  不合并相同记录 **UNION ALL** |

(SELECT ename, dname

FROM emp LEFT OUTER JOIN dept ON deptId=did)

UNION

(SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept ON deptId=did);

(SELECT ename, dname

FROM emp LEFT OUTER JOIN dept ON deptId=did)

UNION ALL

(SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept ON deptId=did);

* + 1. 综合的语法

SELECT > FROM > WHERE > ORER BY > LIMIT

**注意：**LIMIT后的两个值不能加引号。