1、课程名称：MySQL

1.1、MySQL数据库学习准备

(1)什么是数据库

数据库，顾名思义，是存入数据的仓库。只不过这个仓库是在计算机存储设备上的，而且数据是按一定格式存放的。指长期储存在计算机内的、有组织的、可共享的数据集合。其组织方式可支持对数据的有效存取。

当人们收集了大量的数据后，应该把它们保存起来进入近一步的处理，进一步的抽取有用的信息。当年人们把数据存放在文件柜中，可现在随着社会的发展，数据量急剧增长，现在人们就借助计算机和数据库技术科学的保存大量的数据，以便能更好的利用这些数据资源。

(2)数据库的类型

数据库包含关系数据库、面向对象数据库及新兴的XML数据库等多种，目前应用最广泛的是关系数据库，若在关系数据库基础上提供部分面向对象数据库功能的对象关系数据库。在数据库技术的早期还曾经流行过层次数据库与网状数据库，但这两类数据库目前已经极少使用。

(3)数据库管理

数据库管理(Database Administration)是有关建立、存储、修改和存取数据库中信息的技术，是指为保证数据库系统的正常运行和服务质量，有关人员须进行的技术管理工作。负责这些技术管理工作的个人或集体称为数据库管理员(DBA)。数据库管理的主要内容有：数据库的建立、数据库的调整、数据库的重组、数据库的重构、数据库的安全控制、数据的完整性控制和对用户提供技术支持。

数据库的建立：数据库的设计只是提供了数据的类型、逻辑结构、联系、约束和存储结构等有关数据的描述。这些描述称为数据模式。要建立可运行的数据库，还需进行下列工作：

1. 选定数据库的各种参数，例如最大的数据存储空间、缓冲决的数量、并发度等。这些参数可以由用户设置，也可以由系统按默认值设置。
2. 定义数据库，利用数据库管理系统(DBMS)所提供的数据定义语言和命令，定义数据库名、数据模式、索引等。
3. 准备和装入数据，定义数据库仅仅建立了数据库的框架，要建成数据库还必须装入大量的数据，这是一项浩繁的工作。在数据的准备和录入过程中，必须在技术和制度上采取措施，保证装入数据的正确性。计算机系统中原已积累的数据，要充分利用，尽可能转换成数据库的数据。

(4)数据库产品有

大型数据库有：Oracle、Sybase、DB2、SQL server

小型数据库有：MySQL、Access等。

(5)关系型数据库基本概念

关系型数据库是由多个表(table)和表之间的关联关系组成的数据的集合，表是一个由若干行、若干列组成的二维的关系结构。

表的列称为字段(field)

表的行成为记录(record)

1.2、MySQL数据库简介

MySQL是一个真正的多用户、多线程SQL数据库服务器。SQL（结构化查询语言）是世界上最流行的和标准化的数据库语言。MySQL是以一个客户机/服务器结构的实现，它由一个服务器守护程序mysqld和很多不同的客户程序和库组成。

SQL是一种标准化的语言，它使得存储、更新和存取信息更容易。例如，你能用SQL语言为一个网站检索产品信息及存储顾客信息，同时MySQL也足够快和灵活以允许你存储记录文件和图像。

MySQL 主要目标是快速、健壮和易用。最初是因为我们需要这样一个SQL服务器，它能处理与任何可不昂贵硬件平台上提供数据库的厂家在一个数量级上的大型数据库，但速度更快，MySQL就开发出来。自1996年以来，我们一直都在使用MySQL，其环境有超过 40 个数据库，包含 10,000个表，其中500多个表超过7百万行，这大约有100 个吉字节(GB)的关键应用数据。

MySQL建立的基础是业已用在高要求的生产环境多年的一套实用例程。尽管MySQL仍在开发中，但它已经提供一个丰富和极其有用的功能集。

MySQL官网：http://www.mysql.com

1.3、数据类型

MySQL支持多种列类型：数值类型、日期/时间类型和字符串(字符)类型。

长度以字节为单位

名称	长度	用法
TINYINT(M) BIT,BOOL,BOOLEAN	1	如果为无符号数，可以存储从0到255的数; 否则可以存储从-128到127的数。
SMALLINT(M)	2	如果为无符号数，可以存储从0到65535的数; 否则可以存储从-32768到32767的数。
MEDIUMINT(M)	3	如果为无符号数，可以存储从0到16777215的数;否则可以存储从-8388608到8388607的数
INT(M) INTEGER(M)	4	如果为无符号数，可以存储从0到4294967295的数，否则可以存储从-2147483648到2147483647的数。
BIGINT(M)	8	如果为无符号数，可以存储从0到18446744073709551615的数，否则可以存储从-92233720368547758 08到92233720368547758 07的数。
FLOAT(precision)	4或8	这里的precision是可以直达53的整数。如果precision<=24则转换为FLOAT，如果precision>24并且precision<=53则转换为DOUBLE。
FLOAT(M,D)	4	单精度浮点数。
DOUBLE(M,D), DOUBLE PRECISION, REAL	8	双精度浮点。
DECIMAL(M,D), DEC,NUMERIC,FIXED	M+1或M+2	±1.0 * 10e−28至±7.9 *10e28，28到29位有效
DATE	3	以YYYY-MM-DD的格式显示。
DATETIME	8	以YYYY-MM-DD HH:MM:SS的格式显示。
TIMESTAMP	8	以YYYY-MM-DD HH:MM:SS的格式显示。
TIME	3	以HH:MM:SS的格式显示。
YEAR	1	以YYYY的格式显示。
CHAR(M)	M	定长字符串。
VARCHAR(M)	最大M	变长字符串。M<=255.
TINYBLOB, TINYTEXT	最大255	TINYBLOB为大小写敏感，而TINYTEXT不是大小写敏感的。
BLOB, TEXT	最大64K	BLOB为大小敏感的，而TEXT不是大小写敏感的。

MEDIUMBLOB, MEDIUMTEXT	最大16M	MEDIUMBLOB为大小写敏感的，而MEDIUMTEXT不是大小敏感的。
LONGBLOB, LONGTEXT	最大4G	LONGBLOB为大小敏感的，而LONGTEXT不是大小敏感的。
ENUM(VALUE1,....)	1或2	最大可达65535个不同的值。
SET(VALUE1,....)	可达8	最大可达64个不同的值。

数据类型更详细的讲解请参考MYSQL帮助文档

1.5、SQL

SQL全称是：结构化查询语言(Structured Query Language)。

SQL语言包含4个部分

1. 数据定义语言（Data Definition Language--DDL）： 如CREATE, DROP,ALTER等语句
2. 数据操纵语言（Data Manipulation Language- -DML）： INSERT, UPDATE, DELETE语句
3. 数据查询语言（Data Query Language --DQL）： SELECT语句
4. 事务控制语言（Transaction Control Language--TCL）： 如COMMIT, ROLLBACK等语句

1.5.1、数据定义语言（DDL

- 创建数据库（CREATE DATABASE语句）

```
CREATE DATABASE mydatabase;

show databases; //查看当前服务器上存在的数据库(当前登录用户可见的)
use mydatabase; //访问数据库
```

- 创建表（CREATE TABLE语句）

```
create table student(
sid int(11) primary key auto_increment,name varchar(20),gender char(1),age
int(2),birth date
);

desc  student; //查看表结构

//创建表
```

```
CREATE TABLE `employee` (
  `eid` int(11) NOT NULL default '0',
  `name` varchar(11) NOT NULL,
  `dept` varchar(11) NOT NULL,
  `job` varchar(11) NOT NULL,
  `gender` varchar(5) NOT NULL,
  PRIMARY KEY  (`eid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;
```

> 主键：用来唯一代表一条记录的字段（主键值必须是唯一）

- 删除表（DROP TABLE语句）

```
DROP TABLE student;

//drop table 语句会删除该的所有记录及表结构

show create table student; --查看表结构创建语句

desc student; --查看表结构
```

- 修改表结构（ALTER TABLE语句）

```
alter table test add  column name varchar(10); --添加表列

alter table test rename test1; --修改表名

alter table test drop  column name; --删除表列

alter table test modify address char(10) --修改表列类型

alter table test change address address  char(40) --修改表列类型

alter table test change  column address address1 varchar(30)--修改表列名
```

1.5.2、数据操纵语言（DML）

- 添加数据（INSERT INTO...语句）

```
INSERT INTO student(name,gender,birth) values('Tom','男','1985-2-5');
```

- 修改数据（UPDATE ... SET语句）

```
UPDATE student SET name='LILY',gender='女',birth='1988-1-1' where id=1;
```

- 删除数据（DELETE FROM...语句）

```
DELETE FROM student; --删除所有记录 DELETE FROM student where id=1; --删除ID为1的记录
```

1.5.3、数据查询语言（DQL）

查询数据（SELECT ... FROM ...语句）

```
#查询表中的单个字段
SELECT last_name FROM employees;

#查询表中的多个字段
SELECT last_name,salary,email FROM employees;

#查询表中的所有字段
SELECT
  `employees`.`employee_id`,
  `employees`.`first_name`,
  `employees`.`last_name`,
  `employees`.`email`,
  `employees`.`phone_number`,
  `employees`.`job_id`,
  `employees`.`salary`,
  `employees`.`commission_pct`,
  `employees`.`manager_id`,
  `employees`.`department_id`,
  `employees`.`hiredate`
FROM
  employees;
# 方式二
SELECT * FROM employees;

#查询常量值
SELECT 100;
SELECT 'john';

#查询表达式
SELECT 10%3;

#查询函数
SELECT VERSION();

#7. 别名
SELECT 50%3 AS 结果;
SELECT last_name AS 姓,first_name AS 名 FROM employees;

#方式二:使用空格(省略 AS)
SELECT last_name 姓,first_name 名 FROM employees;
```

```sql
#案例:查询工资,显示结果为 员工 工资
SELECT salary "员工 工资" FROM employees;

#去重复
#案例:查询员工表中设计到的所有部门编号
SELECT DISTINCT department_id FROM employees;

#查询 ID 为 100 的员工信息
SELECT * FROM employees WHERE employee_id=100;

#查询部门id为空的员工信息
SELECT * FROM employees WHERE department_id IS NULL;

#查询表中的多个字段
SELECT first_name,last_name FROM employees;

#查询 employees_id 大于105 的员工信息
SELECT * FROM employees WHERE employee_id>105;

#查询 job_id 为 ST_MAN 并且 manager_id 为 100 的员工信息
SELECT * FROM employees WHERE job_id='ST_MAN' AND manager_id=100;

#查询 job_id 为 ST_MAN 或者 manager_id 大于 100 的员工信息
SELECT * FROM employees WHERE job_id='ST_MAN' OR manager_id>120;

#查询姓名的最后一个字符为“T”的员工信息
SELECT * FROM employees WHERE last_name LIKE '%T';

#查询姓名以“T”开头的员工信息
SELECT * FROM employees WHERE last_name LIKE 't%';

#查询姓名中包含“T”的员工信息
SELECT * FROM employees WHERE last_name LIKE '%t%';

#查询所有员工信息，并按工资降序排序（默认为升序：ASC）
SELECT * FROM employees ORDER BY salary DESC;

#多个排序条件：当第一个条件相同时，以第二个条件排序
SELECT * FROM employees ORDER BY salary DESC, department_id DESC;

#按部门分组查询各部门的人数
SELECT department_id, COUNT(department_id) AS 人数 FROM employees GROUP BY
department_id;

#HAVING条件语句一个HAVING子句(条件查询)必须位于GROUP BY子句之后，并位于ORDER BY子句之
前。
#按部门分组查询部门id为80的部门的人数
SELECT department_id, COUNT(department_id) AS 总数 FROM employees GROUP BY
department_id HAVING department_id=80;
```

```
#查询表的总记录数
SELECT COUNT(*) AS 总数 FROM employees;

#查询学生记录的前三条（从0位置开始找出3条）
SELECT * FROM employees LIMIT 0,3;
```

1.5.4、数据控制语言（TCL）

1.5.4.1、什么是事务

事务(Transaction)是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)。事务通常由高级数据库操纵语言或编程语言（如SQL，C++或Java）书写的用户程序的执行所引起，并用形如begin transaction和end transaction语句（或函数调用）来界定。事务由事务开始(begin transaction)和事务结束(end transaction)之间执行的全体操作组成。

例如：在关系数据库中，一个事务可以是一条SQL语句，一组SQL语句或整个程序。

事务是恢复和并发控制的基本单位。

事务应该具有4个属性：原子性、一致性、隔离性、持续性。这四个属性通常称为ACID特性。

- 原子性（atomicity）
  一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

- 一致性（consistency）
  事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

- 隔离性（isolation）
  一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

- 持久性（durability）
  持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

小结：

> 事务(Transaction)，也就是要么成功，要么失败，并恢复原状。

1.5.4.2、事务操作

- 设置默认事务提交方式

```
set autocommit = false –设置事务提交方式为“手动提交”
set autocommit = true –设置事务提交方式为“自动提交”
```

事务就是对数据库的多步操作，要么一起成功，要么一起失败

```
set autocommit = false;
update student set name='vince' where id=1;--更新数据
insert into student(name,gender,birth) values('yoyo','女','1981-03-20');--插入数据
commit;--手动提交事务

delete from student where id=1;--删除数据
insert into student(name,gender,birth) values('tony,'男','1985-02-26');--插入数据
rollback;--回滚事务

delete from student where id=1;--删除数据
savepoint point1;  --保存还原点
delete from student where id=1;--删除数据
savepoint point2;  --保存还原点
rollback to point2; --回滚到point2还原点
commit; --提交事务
```

1.6、函数

1.6.1、GROUP BY（聚合）函数

- **AVG ([DISTINCT] expr)**

返回expr 的平均值。 DISTINCT 选项可用于返回 expr的不同值的平均值。

```
SELECT department_id,AVG(salary) 平均薪资 FROM employees GROUP BY department_id;
SELECT gender, AVG(age) FROM student GROUP BY gender;
```

- **COUNT(expr)**
  返回SELECT语句检索到的行中非NULL值的数目。

```
SELECT COUNT(*) FROM student;  --返回检索行的数目，不论其是否包含 NULL值
SELECT COUNT(name) FROM student;  --返回SELECT语句检索到的行中非NULL值的数目
```

- **MIN ([DISTINCT] expr), MAX ([DISTINCT] expr)**

返回expr 的最小值和最大值

```
SELECT MIN(age),MAX(age) FROM student;
```

- **SUM ([DISTINCT] expr)**

返回expr 的总数

```
SELECT SUM(age) FROM student;
```

1.6.2、控制流程函数

- **CASE value WHEN [compare-value] THEN result [WHEN [compare-value] THEN result ...] [ELSE result] END**

如果没有匹配的结果值，则返回结果为ELSE后的结果，如果没有ELSE 部分，则返回值为 NULL。

```
SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END;
```

- **IF(expr1,expr2,expr3)**

如果 expr1 是TRUE (expr1 <> 0 and expr1 <> NULL)，则 IF()的返回值为expr2; 否则返回值则为 expr3。

```
SELECT IF(1<2,'yes ','no');
```

IFNULL(expr1,expr2)

假如expr1 不为 NULL，则 IFNULL() 的返回值为 expr1; 否则其返回值为 expr2

```
SELECT IFNULL(1,0);
```

- **NULLIF(expr1,expr2)**

如果expr1 = expr2 成立，那么返回值为NULL，否则返回值为 expr1

```
SELECT NULLIF(1,1);
```

1.6.3、字符串函数

- **ASCII (str)**

返回值为字符串str 的最左字符的数值。假如str为空字符串，则返回值为 0 。假如str 为NULL，则返回值为 NULL。 ASCII()用于带有从 0到255的数值的字符。

```
SELECT ASCII('dx');
```

- **BIN (N)**

返回值为N的二进制值的字符串表示

```
SELECT BIN(15);
```

BIT_LENGTH (str)

返回值为二进制的字符串str 长度

```
SELECT BIT_LENGTH('text');
```

- **CHAR_LENGTH(str)**

返回值为字符串str 的长度，长度的单位为字符

```
SELECT CHAR_LENGTH( 'Admiral');
```

- **FORMAT(X,D)**

将数字X 的格式写为'#,###,###.##',以四舍五入的方式保留小数点后 D 位，并将结果以字符串的形式返回。若D为0, 则返回结果不带有小数点，或不含小数部分。

```
SELECT FORMAT(12332.123456, 4);
```

- **INSERT (str,pos,len,newstr)**

返回字符串 str, 其子字符串起始于 pos 位置和长期被字符串 newstr取代的len 字符。 如果pos 超过字符串长度，则返回值为原始字符串。 假如len的长度大于其它字符串的长度，则从位置pos开始替换。若任何一个参数为null，则返回值为NULL。

```
SELECT INSERT('Quadratic', 3, 4, 'What');
SELECT INSERT('Quadratic', 3, 100, 'What');
```

- **INSTR(str,substr)**

返回字符串 str 中子字符串的第一个出现位置

```
SELECT INSTR('foobarbar', 'bar');
```

- **LEFT(str,len)**

返回从字符串str 开始的len 最左字符。

```
SELECT LEFT('foobarbar', 5);
```

- **LENGTH(str)**

返回值为字符串str 的长度，单位为字节。一个多字节字符算作多字节。这意味着对于一个包含5个2字节字符的字符串， LENGTH() 的返回值为 10, 而 CHAR_LENGTH()的返回值则为5。

```
SELECT LENGTH('text');
```

- **LTRIM(str)**

返回字符串 str ，其引导空格字符被删除。

```
SELECT LTRIM('  barbar');
```

- **TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str) TRIM(remstr FROM] str)**

返回字符串 str ， 其中所有remstr 前缀和/或后缀都已被删除。若分类符BOTH、LEADIN或TRAILING中没有一个是给定的,则假设为BOTH 。 remstr 为可选项，在未指定情况下，可删除空格。

```
SELECT TRIM('  bar   '); --去空格SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx'); --去左边的xSELECT TRIM(BOTH 'x' FROM 'xxxbarxxx'); --去左右两边的xSELECT TRIM(TRAILING 'xyz' FROM 'barxxyz'); --去右边的xyz
```

- **STRCMP(expr1,expr2)**

若所有的字符串均相同，则返回0，若根据当前分类次序，第一个参数小于第二个，则返回 -1，其它情况返回1。

```
SELECT STRCMP('text', 'text2'); --返回-1SELECT STRCMP('text2', 'text'); --返回1SELECT STRCMP('text', 'text');  --返回0
```

- **CONCAT (str1,str2,...)**

返回结果为连接参数产生的字符串。如有任何一个参数为NULL ，则返回值为 NULL。或许有一个或多个参数。 如果所有参数均为非二进制字符串，则结果为非二进制字符串。 如果自变量中含有任一二进制字符串，则结果为一个二进制字符串。一个数字参数被转化为与之相等的二进制字符串格式；若要避免这种情况，可使用显式类型 cast, 例如：

```
SELECT CONCAT(CAST(int_col AS CHAR), char_col)  mysql> SELECT CONCAT('My', 'S', 'QL');
```

1.6.4、日期和时间函数

- **DAYOFWEEK(date)**

返回日期date的星期索引(1=星期天，2=星期一, ……7=星期六)。

```
mysql> select DAYOFWEEK('1998-02-03');
```

- **WEEKDAY(date)**

返回date的星期索引(0=星期一，1=星期二, ……6= 星期天)。

```
mysql> select WEEKDAY('1997-10-04 22:23:00'); mysql> select WEEKDAY('1997-11-05');
```

- **DAYOFMONTH (date)**

返回date的月份中日期，在1到31范围内。

```
mysql> select DAYOFMONTH ('1998-02-03');
```

- **DAYOFYEAR(date)**

返回date在一年中的日数, 在1到366范围内。

```
mysql> select DAYOFYEAR('1998-02-03');
```

- **MONTH(date)**

返回date的月份，范围1到12。

```
mysql> select MONTH('1998-02-03');
```

- **DAYNAME(date)**

返回date的星期名字。

```
mysql> select DAYNAME("1998-02-05");
```

- **MONTHNAME(date)**

返回date的月份名字。

```
mysql> select MONTHNAME("1998-02-05");
```

- **QUARTER(date)**

返回date一年中的季度，范围1到4。

```
mysql> select QUARTER('98-04-01');
```

- **WEEK(date)**

对于星期天是一周的第一天的地方，有一个单个参数，返回date的周数，范围在0到52。2个参数形式WEEK()允许你指定星期是否开始于星期天或星期一。如果第二个参数是0，星期从星期天开始，如果第二个参数是1，从星期一开始。

```
mysql> select WEEK('1998-02-20'); mysql> select WEEK('1998-02-20',0); mysql>
select WEEK('1998-02-20',1);
```

- **YEAR(date)**

返回date的年份，范围在1000到9999。

```
mysql> select YEAR('98-02-03');
```

- **HOUR(time)**

返回time的小时，范围是0到23。

```
mysql> select HOUR('10:05:03');
```

- **MINUTE(time)**

返回time的分钟，范围是0到59。

```
mysql> select MINUTE('98-02-03 10:05:03');
```

- **SECOND(time)**

返回time的秒数，范围是0到59。

```
mysql> select SECOND('10:05:03');
```

- **PERIOD_ADD(P,N)**

增加N个月到阶段P（以格式YYMM或YYYYMM)。以格式YYYYMM返回值。注意阶段参数P不是日期值。

```
mysql> select PERIOD_ADD(9801,2);
```

- **PERIOD_DIFF(P1,P2)**

返回在时期P1和P2之间月数，P1和P2应该以格式YYMM或YYYYMM。注意，时期参数P1和P2不是日期值。

```
mysql> select PERIOD_DIFF(9802,199703);
```

- **ADDDATE(expr,days)**

expr是指定加到开始日期的间隔值一个表达式，expr是一个字符串；它可以以一个“-”开始表示负间隔。type是一个关键词，指明表达式应该如何被解释。（type关键词用法请参考帮助文档）

若 days 参数只是整数值，则将其作为天数值添加至 expr。

```
mysql> SELECT ADDDATE('1998-01-02', 31);SELECT DATE_ADD('1997-12-31
23:59:59',INTERVAL 1 SECOND);
```

- **SUBDATE(date,INTERVAL expr type)**

date是一个指定开始日期的DATETIME或DATE值，expr是指定从开始日期减去的间隔值一个表达式，expr是一个字符串；它可以以一个“-”开始表示负间隔。type是一个关键词，指明表达式应该如何被解释。（type关键词用法请参考帮助文档）

```
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
```

- **ADDTIME(expr,expr2)**

将 expr2添加至expr 然后返回结果。 expr 是一个时间或时间日期表达式，而expr2 是一个时间表达式。

```
mysql> SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
```

- **DATE(expr)**

提取日期或时间日期表达式expr中的日期部分。

```
mysql> SELECT DATE('2003-12-31 01:02:03');
```

- **TO_DAYS(date)**

给出一个日期date，返回一个天数(从0年的天数)。

```
mysql> select TO_DAYS(950501);mysql> select TO_DAYS('1997-10-07');
```

- **FROM_DAYS(N)**

给出一个天数N，返回一个DATE值。

```
mysql> select FROM_DAYS(729669);
```

- **CURDATE()**

以'YYYY-MM-DD'或YYYYMMDD格式返回今天日期值，取决于函数是在一个字符串还是数字上下文被使用。

```
mysql> select CURDATE();  -- YYYY-MM-DD 格式mysql> select CURDATE() + 0;  --
YYYYMMDD 格式
```

- **CURTIME()**

以'HH:MM:SS'或HHMMSS格式返回当前时间值，取决于函数是在一个字符串还是在数字的上下文被使用。

```
mysql> select CURTIME();  --HH:MM:SS 格式mysql> select CURTIME() + 0;  --HHMMSS
格式
```

- **NOW()**

以'YYYY-MM-DD HH:MM:SS'或YYYYMMDDHHMMSS格式返回当前的日期和时间，取决于函数是在一个字符串还是在数字的上下文被使用。

```
mysql> select NOW();  -- YYYY-MM-DD HH:MM:SS 格式mysql> select NOW() + 0;  --
YYYYMMDDHHMMSS 格式
```

- **SEC_TO_TIME(seconds)**

返回seconds参数，变换成小时、分钟和秒，值以'HH:MM:SS'或HHMMSS格式化，取决于函数是在一个字符串还是在数字上下文中被使用。

```
mysql> select SEC_TO_TIME(2378);  -- HH:MM:SS格式mysql> select
SEC_TO_TIME(2378) + 0;  --HHMMSS 格式
```

- **TIME_TO_SEC(time)**

返回time参数，转换成秒。

```
mysql> select TIME_TO_SEC('22:23:00'); mysql> select TIME_TO_SEC('00:39:38');
```

- **DATE_FORMAT(date,format)**

根据format 字符串安排date 值的格式。

以下说明符可用在 format 字符串中：

说明符	说明
%a	工作日的缩写名称 (Sun..Sat)
%b	月份的缩写名称 (Jan..Dec)
%c	月份，数字形式(0..12)
%D	带有英语后缀的该月日期 (0th, 1st, 2nd, 3rd, ...)
%d	该月日期, 数字形式 (00..31)
%e	该月日期, 数字形式(0..31)
%f	微秒 (000000..999999)
%H	小时(00..23)
%h	小时(01..12)
%I	小时 (01..12)
%i	分钟,数字形式 (00..59)
%j	一年中的天数 (001..366)
%k	小时 (0..23)
%l	小时 (1..12)

%M	月份名称 (January..December)
%m	月份, 数字形式 (00..12)
%p	上午（AM）或下午（ PM）
%r	时间 , 12小时制 (小时hh:分钟mm:秒数ss 后加 AM或PM)
%S	秒 (00..59)
%s	秒 (00..59)
%T	时间 , 24小时制 (小时hh:分钟mm:秒数ss)
%U	周 (00..53), 其中周日为每周的第一天
%u	周 (00..53), 其中周一为每周的第一天
%V	周 (01..53), 其中周日为每周的第一天 ; 和 %X同时使用
%v	周 (01..53), 其中周一为每周的第一天 ; 和 %x同时使用
%W	工作日名称 (周日..周六)
%w	一周中的每日 (0=周日..6=周六)
%X	该周的年份，其中周日为每周的第一天, 数字形式,4位数;和%V同时使用
%x	该周的年份，其中周一为每周的第一天, 数字形式,4位数;和%v同时使用
%Y	年份, 数字形式,4位数
%y	年份, 数字形式 (2位数)
%%	‘%’文字字符

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00','%D %y %a %d %m %b %j');
```

1.7、关联查询

1.7.1、多表连接查询

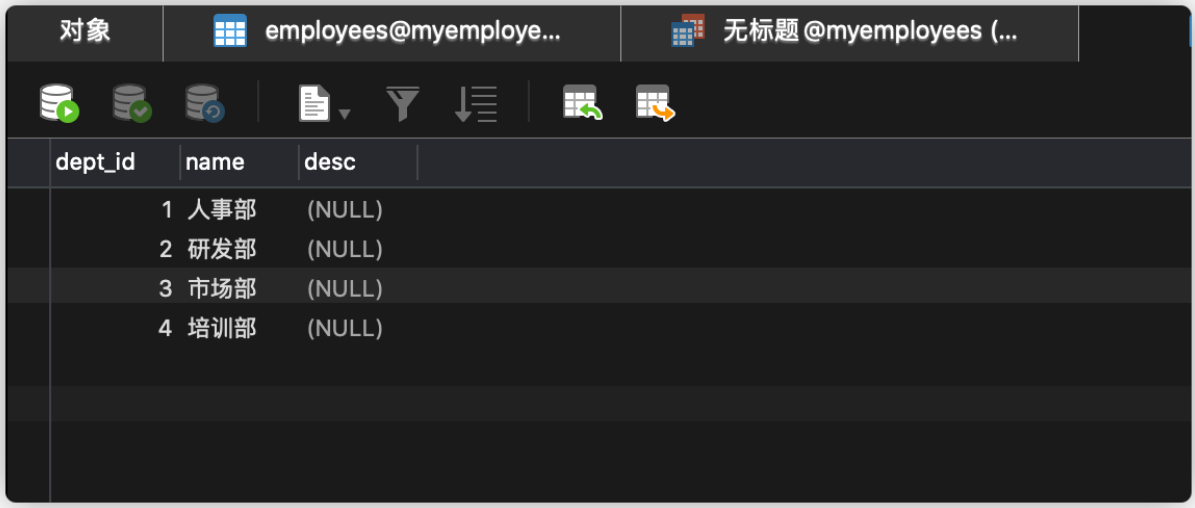
使用单个SELECT语句从多个表中取出相关的数据，通过多表之间的关系，构建相关数据的查询。多表连接通常是建立在相互关系的父子(主从)表上的。

SQL1999标准中多表连接的语法：

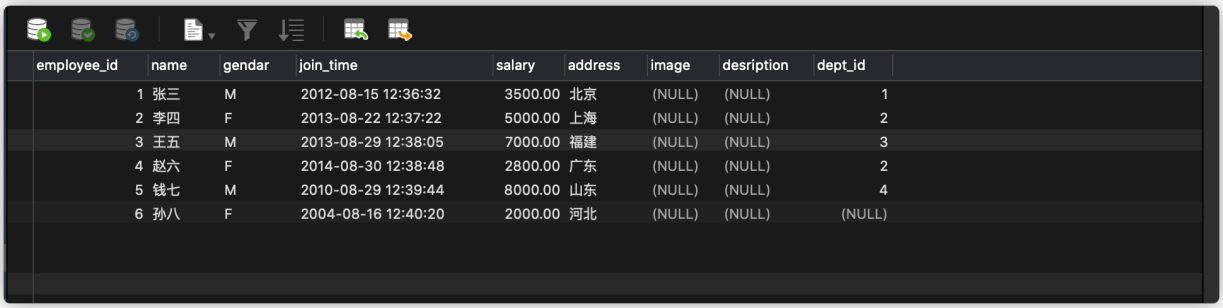
```
SELECT... FROM join_table
    JOIN_TYPE join_table
    ON  join_condition
WHERE where_condition


join_table：参与连接的表
JOIN_TYPE：连接类型：内连接、外连接、交叉连接、自连接
join_condition：连接条件
where_condition：where过滤条件
```

示例表:



dept_id	name	desc
1	人事部	(NULL)
2	研发部	(NULL)
3	市场部	(NULL)
4	培训部	(NULL)



employee_id	name	gendar	join_time	salary	address	image	desription	dept_id
1	张三	M	2012-08-15 12:36:32	3500.00	北京	(NULL)	(NULL)	1
2	李四	F	2013-08-22 12:37:22	5000.00	上海	(NULL)	(NULL)	2
3	王五	M	2013-08-29 12:38:05	7000.00	福建	(NULL)	(NULL)	3
4	赵六	F	2014-08-30 12:38:48	2800.00	广东	(NULL)	(NULL)	2
5	钱七	M	2010-08-29 12:39:44	8000.00	山东	(NULL)	(NULL)	4
6	孙八	F	2004-08-16 12:40:20	2000.00	河北	(NULL)	(NULL)	(NULL)

```
select e.name,e.salary,e.comm,d.dname from emp e join dept d on
e.deptid=d.did where e.comm is null;
--查询员工姓名，工资，奖金，部门姓称，并且员工奖金为null

select e.name,e.salary,e.comm,d.dname from emp e join dept d on
e.deptid=d.did where e.comm is not null;
--查询员工姓名，工资，奖金，部门姓称，并且员工奖金不为null

select e.name,e.salary,ifnull(e.comm,0),d.dname from emp e join dept d on
e.deptid=d.did where e.comm>0;
--查询员工姓名，工资，奖金，部门姓称，并且员工奖金大于0
```

```
select e.name,d.dname from emp e,dept d where d.dname='技术部'
and e.deptid=d.did;
--查询部门名称为“技术部”的员工姓名和部门名称
```