MySQL数据库介绍

讲师： 李 奇

什么是数据库

- 企业数据存储面临的问题：
  - 存储大量数据
  - 大量数据的检索和访问
  - 保证数据信息的一致和完整
  - 数据共享和安全
  - 通过分析整合，产生新的有用的信息(如提供决策支持)

- RDBMS：
  - 关系数据库管理系统
  - 关系数据库：所有的数据存储在不同的表中，使用主键或外键建立表间的关系。
  - RDBMS是这样一个软件：
    - 能让我们使用表、列和索引实现一个数据库
    - 保证各种表的行间的引用完整性
    - 自动更新索引
    - 解释一个SQL查询和组合来自不同表的信息
    - SQL:结构化查询语言(Structured Query Language)，在关系型数据库上执行数据操作、数据检索以及数据维护的标准语言。

创建数据库

- 创建数据库：create database 数据库名称;
  - 例：创建名为test的测试数据库

    create database test;

- 查看创建好的数据库：show create database 数据库名称;
  - 例： 查看创建好的test数据库

    show create database test;

- 查看所有数据库列表：show databases;

- 使用数据库：use 数据库名称;
  - 例： 使用创建好的test数据库

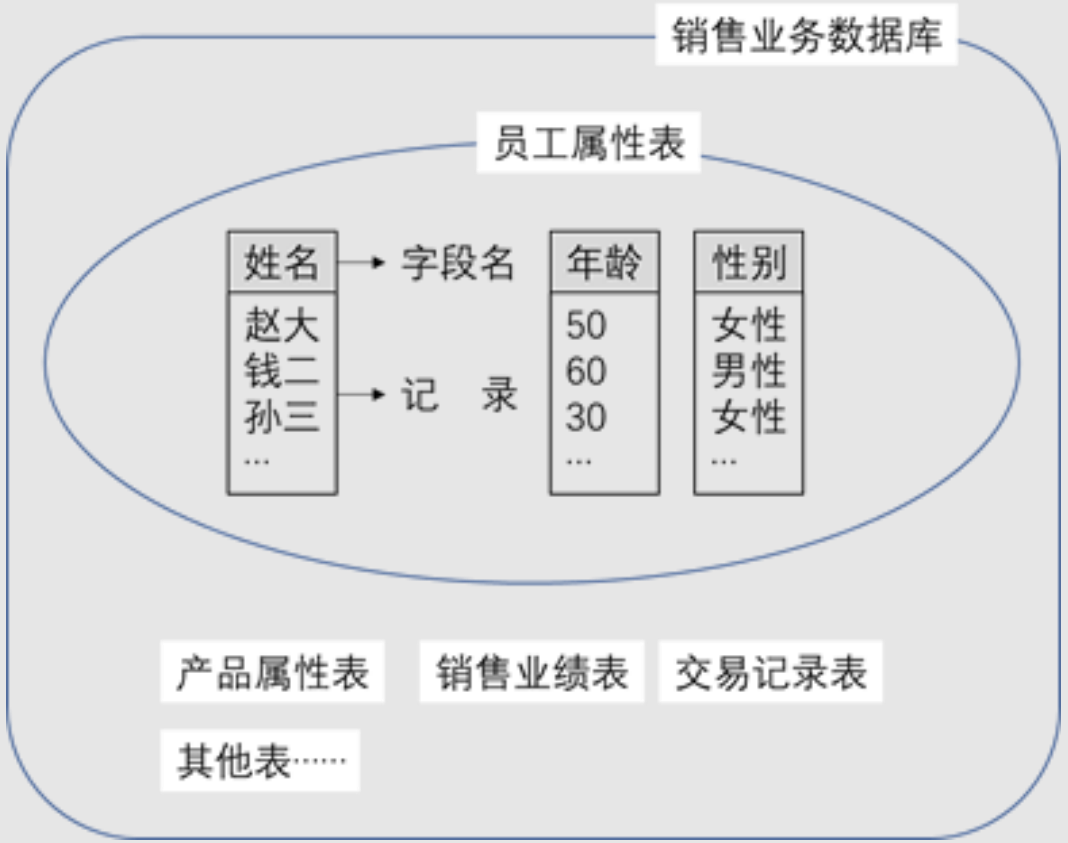
    use test;

- 删除数据库：drop database 数据库名称;
  - 例： 删除创建好的test数据库

    drop database test;

数据库基本结构

– 数据库：数据库是表的集合，带有相关的数据。
– 表：一个表是多个字段的集合。
– 字段：一个字段是一列数据，由字段名和记录组成。



数据表

- 数据库是由多个数据表构成的
- 每张数据表存储多个字段
- 每个字段由不同的字段名及记录构成，每个字段有自己的数据结构及约束条件

商品颜色表： GoodsColor

字段名	字段描述	数据类型	主键	外键	非空	唯一	自增	初始值
ColorID	颜色ID	VARCHAR(4)	N	N	Y	N	N	-
ColorNote	颜色注释（对应商品主表的商品属性）	VARCHAR(20)	N	N	Y	N	N	-
ColorSort	颜色排序	INT	N	N	Y	N	N	0
pt	更新时间	VARCHAR(9)	N	N	Y	N	N	-

商品颜色表：存储商品的颜色信息

创建数据表

- 创建数据表：create table 表名(…);
  - 例：用SQL语句创建以下员工信息表

部门ID	部门名称	员工数
p01	财务部	20
p02	销售部	100
p03	内审部	15
…	…	…

1. 使用test数据库: use test;
2. 创建员工信息表:

```
create table emp(
        depid char(3),
        depname varchar(20),
        peoplecount int
);
```

3. 查看表是否创建成功： show tables;
4. 删除数据表： drop table emp;

数据类型(1)

• 数值类型：

– INT：有符号的和无符号的。有符号大小-2147483648~2147483647，无符号大0~4294967295。宽度最多为11个数字- int(11)

– TINYINT:有符号的和无符号的。有符号大小-128~127，无符号大小为0~255。宽度最多为4个数字- tinyint(4)

– SMALLINT:有符号的和无符号的。有符号大小-32768~32767，无符号大小为0~65535。宽度最多为6个数字- smallint(6)

– MEDIUMINT:有符号的和无符号的。有符号大小-8388608~8388607，无符号大小为0~16777215。宽度最多为9个数字- mediumint(9)

– BIGINT:有符号的和无符号的。宽度最多为20个数字- bigint(20)

– FLOAT(M,D):只能为有符号的。默认为(10,2)

– DOUBLE(M,D):只能为有符号的。默认为(16,4)

– DECIMAL(M,D):只能为有符号的。

数据类型(2)

- 日期和时间类型
  - DATE:YYYY-MM-DD格式，在1000-01-01和9999-12-31之间。例如：1973-12-30
  - DATETIME:YYYY-MM-DD HH:MM:SS格式，位于1000-01-01 00:00:00和9999-12-31 23:59:59之间。例如：1973-12-30 15:30:00
  - TIMESTAMP:称为时间戳，在1970-01-01 00:00:00和2037-12-31 23:59:59之间。例如，1973年12月30日下午15:30，则在数据库中存储为:19731230153000
  - TIME: 以HH:MM:SS格式, -838:59:59~838:59:59
  - YEAR(2|4): 以2位或4位格式存储年份值。如果是2位，1970~2069；如果是4位，1901~2155。默认长度为4

数据类型(3)

- 字符串类型
  - CHAR(M)：固定长度字符串，长度为1-255。如果内容小于指定长度，右边填充空格。如果不指定长度，默认为1
  - VARCHAR(M)： 可变长度字符串，长度为1-255。定义该类型时必须指定长度
  - BLOB 或TEXT：最大长度65535。存储二进制大数据，如图片。不能指定长度。两者区别：BLOB 大小写敏感
  - TINYBLOB 和TINYTEXT：最大长度255。不能指定长度。
  - MEDIUMBLOB 或MEDIUMTEXT：最大长度16777215 字符
  - LONGBLOB 或LONGTEXT：最大长度4294967295 字符
  - ENUM：枚举。例如：ENUM(‘A’,’B’,’C’)。NULL 值也可

约束条件

– 约束是在表上强制执行的数据检验规则

– 用来保证创建的表的数据完整和正确

– MySQL数据库常用约束条件

约束条件	说明	语法
PRIMARY KEY	主键约束	字段名 数据类型 PRIMARY KEY
NOT NULL	非空约束	字段名 数据类型 NOT NULL
UNIQUE	唯一约束	字段名 数据类型 UNIQUE
AUTO_INCREMENT	自增字段	字段名 数据类型 AUTO_INCREMENT
DEFAULT	默认值	字段名 数据类型 DEFAULT 默认值

主键约束

主键约束：保证表中每行记录都不重复

主键，又称为”主码”，是数据表中一列或多列的组合。 主键约束要求主键列的数据必须是唯一的，并且不允许为空。 使用主键，能够惟一地标识表中的一条记录，并且可以结合外键来 定义不同数据表之间的关系，还可以加快数据库查询的速度

主键分为两种类型：

```
-- 单字段主键：
create table emp(
                depid char(3) primary key,
                depname varchar(20),
                peoplecount int
        );

-- 多字段联合主键：
create table emp(
                depid char(3),
                depname varchar(20),
                peoplecount int,
                primary key(depname,depid)
);
```

非空约束

非空约束，指的是字段的值不能为空：

– 语法：字段名　字段类型　not null

```
create table emp(
                depid char(3) primary key,
                depname varchar(20) not null,
                peoplecount int
        );
```

唯一性约束

唯一性约束，要求该列的值必须是唯一的：

– 允许为空，但只能出现一个空值;
– 一个表中可以有多个字段声明为唯一的;
– 唯一约束确保数据表的一列或几列不出现重复值;
– 语法：字段名 数据类型 unique

```
create table emp(
                    depid char(3) primary key,
                    depname varchar(20) not null,
                    peoplecount int unique
        );
```

默认约束

默认约束，指定某个字段的默认值：

– 如果新插入一条记录时没有为默认约束字段赋值，那么系统就会自动为这个字段赋值为默认约束设定的值

– 语法： 字段名 数据类型 default 默认值

```
create table emp(
        depid char(3) primary key,
        depname varchar(20) not null default '-',
        peoplecount int unique
);
```

自增字段

自增字段：一个表只能有一个自增字段，自增字段必须为主键的一部分。默认情况下从1开始自增

例： 创建含各种约束条件的数据表

```
CREATE TABLE example(id INT PRIMARY KEY AUTO_INCREMENT, -- 创建整数型自增主键
                     name VARCHAR(4) NOT NULL, -- 创建非空字符串字段
                     math INT DEFAULT 0, -- 创建默认值为0的整数型字段
                     minmax FLOAT UNIQUE – 创建唯一约束小数型字段
);
```

练习： 尝试用语言描述以下建表语句

```
create table fruits(
        f_id char(10) not null,
        s_id int not null default 0,
        f_name char(255) not null,
        f_price decimal(8,2) not null,
        primary key(f_id)
);
```

用insert into语句为表插入数据

语法： insert into 表名(字段1,字段2,…) values ……

```
-- 插入数据
insert into fruits(f_id,s_id,f_name,f_price)
values('a1',101,'apple',5.2),
('b1',101,'blackberry',10.2),
('bs1',102,'orange',11.2),
('bs2',105,'melon',8.2),
('t1',102,'banana',10.3),
('t2',102,'grape',5.3),
('o2',103,'coconut',9.2),
('c0',101,'cherry',3.2),
('a2',103,'apricot',25.2),
('l2',104,'lemon',6.4),
('b2',104,'berry',7.6),
('m1',106,'mango',15.6),
('m2',105,'xbabay',2.6),
('t4',107,'xbababa',3.6),
('b5',107,'xxxx',3.6);
```

练习：创建以下大气质量表

表名： Monthly_Indicator

字段名	字段描述	数据类型	主键	外键	非空	唯一	自增	初始值
city_name	城市名	VARCHAR(20)	Y	N	Y	N	N	-
month_key	月份	DATE	Y	N	Y	N	N	
aqi	AQI	INT(4)	N	N	Y	N	N	0
aqi_range	范围	VARCHAR(20)	N	N	Y	N	N	-
air_quality	质量等级	VARCHAR(20)	N	N	Y	N	N	-
pm25	PM2.5	FLOAT(6,2)	N	N	Y	N	N	0
pm10	PM10	FLOAT(6,2)	N	N	Y	N	N	0
so2	SO2	FLOAT(6,2)	N	N	Y	N	N	0
co	CO	FLOAT(6,2)	N	N	Y	N	N	0
no2	NO2	FLOAT(6,2)	N	N	Y	N	N	0
o3	O3	FLOAT(6,2)	N	N	Y	N	N	0
ranking	排名	INT(4)	N	N	Y	N	N	0

导入外部数据

导入外部文本文件：

```
-- 为Monthly_Indicator表导入外部txt文件
load data local infile '文件路径.txt'
        into table Monthly_Indicator
        fields terminated by '\t'
        ignore 1 lines;
```

检查表数据

对导入表中的数据一般从导入内容、导入数据总行数以及表结构三方面进行检查

```
-- 检查倒入内容Monthly_Indicator
Select * from Monthly_Indicator;

-- 检查导入数据总行数Monthly_Indicator
Select count(*) from Monthly_Indicator;

-- 检查表结构
Desc Monthly_Indicator;
```

修改数据表(1)

修改表指的是修改数据库中已经存在的数据表的结构：

– MySQL使用alter table语句修改数据表结构，包括： 修改表名，修改字段数据类型或字段名，增加和删除字段，修改字段的排列位置等

– 例：将数据表emp改名为empdep
```
alter table emp rename empdep;
```

– 例：将数据表empdep中depname字段的数据类型由varchar(20)修改成varchar(30)
```
alter table empdep modify depname varchar(30);
```

– 例：将数据表empdep中depname字段的字段名改为dep
```
alter table empdep change depname dep varchar(30);
```

– 例：将数据表empdep中dep字段的字段名改回为depname，并将该字段数据类型该会为varchar(20)
```
alter table empdep change dep depname varchar(20);
```

– 例：为数据表empdep添加新字段maname，新字段数据类型为varchar(10)，约束条件为非空
```
alter table empdep add maname varchar(10) not null;
```

修改数据表(2)

修改表指的是修改数据库中已经存在的数据表的结构：

– MySQL使用alter table语句修改数据表结构，包括： 修改表名，修改字段数据类型或字段名，增加和删除字段，修改字段的排列位置等

– 例：将数据表empdep中maname字段的排列顺序改为第一位

```
alter table empdep modify maname varchar(10) first;
```

– 例：将数据表emp中maname字段的排列顺序改到depid字段之后

```
alter table empdep modify maname varchar(10) after depid;
```

– 例：删除maname字段

```
alter table empdep drop maname;
```

SQL的数据查询功能

SELECT语句的语法

```
SELECT〈目标列组〉
    FROM〈数据源〉
    [WHERE〈元组选择条件〉]
    [GROUP BY〈分列组〉[HAVING 〈组选择条件〉]]
    [ORDER BY〈排序列1〉〈排序要求1〉 [，…n]]；
```

```
-- 对大气质量表进行有选择的查询
select city_name, avg(pm25), avg(pm10)
            from Monthly_Indicator
            where pm25 > 50
            group by city_name, month_key having city_name <> '北京’
            order by avg(pm25) desc;
```

Select语句的操作符

SELECT语句的操作符

- 算术操作符
+（加号）、—（减号）、*（乘号）和 /（除号）。

- 比较操作符
=（等于）、>（大于）、<（小于）、<=（小于等于）、>=（大于等于）、!=或<>（不等于）、!>（不大于）和 !<（不小于），共9种操作符。

- 逻辑操作符

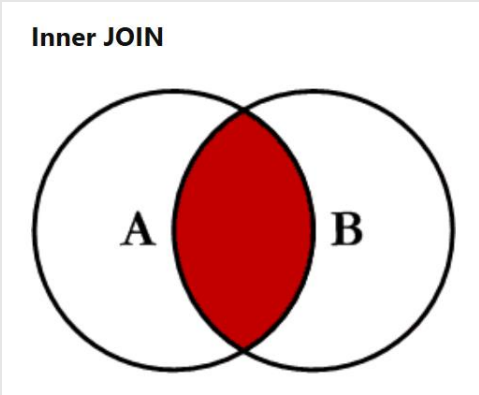
聚合类函数

函数名称	功能
AVG	按列计算平均值
SUM	按列计算值的总和
MAX	求一列中的最大值
MIN	求一列中的最小值
COUNT	按列值计个数

连接方式(1)

内连接：
按照连接条件合并两个表，返回满足条件的行。

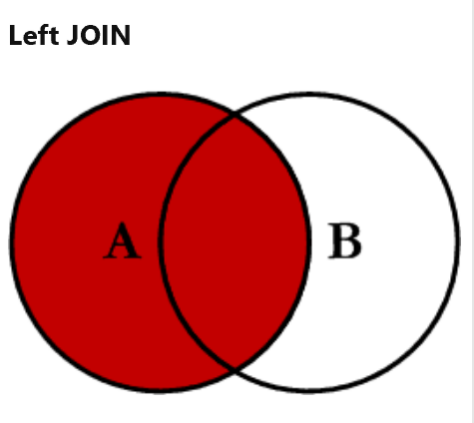
```
SELECT <select_list> FROM A
INNER JOIN B ON A.Key = B.Key;
```



连接方式(2)

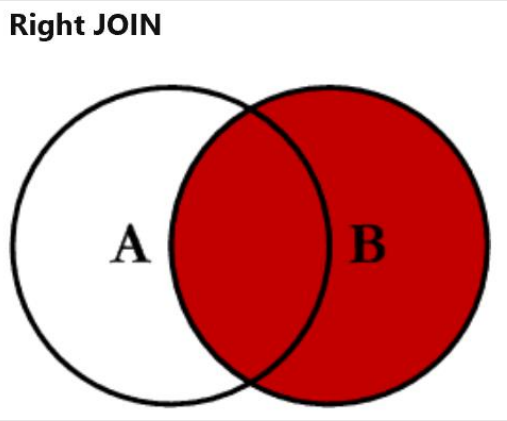
左连接：
结果中除了包括满足连接条件的行外，还包括左表的所有行

```
SELECT <select_list> FROM A
LEFT JOIN B ON A.Key = B.Key;
```

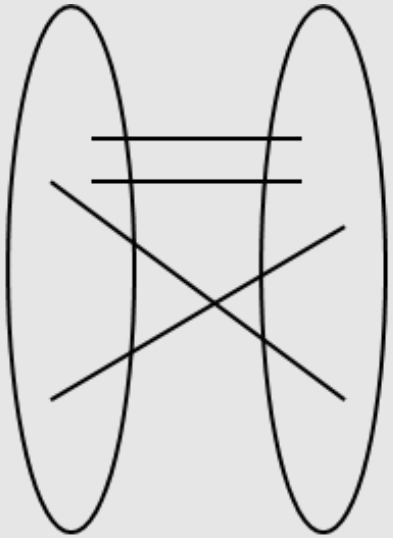


右连接：
结果中除了包括满足连接条件的行外，还包括右表的所有行

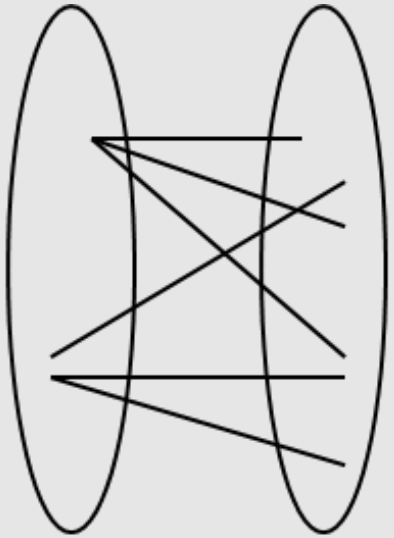
```
SELECT <select_list> FROM A
RIGHT JOIN B ON A.Key = B.Key;
```



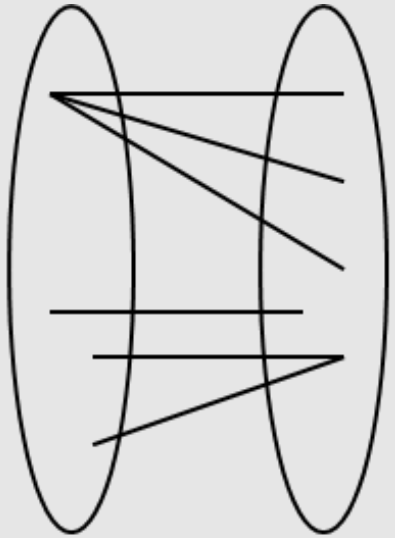
查询对应关系



(a)一对一联系



(b)一对多联系



(c)多对多联系

联合查询

union: 用于合并两个或多个 SELECT 语句的结果集，并消去表中任何重复行。

例： 用union合并t1与t2表

```
select t1.*  from t1
union
select t2.* from t2;
```

union all:用于合并两个或多个 SELECT 语句的结果集，保留重复行。

例： 用union all合并t1与t2表

```
select t1.*  from t1
union all
select t2.* from t2;
```

查询操作符列表

语义	操作符	使用格式或示例	示例解释
在[不在]其中	[NOT] IN	<字段>IN(<数据表|字查询>)	将字段值与数据表|子查询的结果集比较,看字段值在[不在]数据表或结果集中。
任何一个	ANY	<字段><比较>ANY(数据表|子查询)	测试字段值是否大于数据表或子查询结果集中的任何一个值。
全部(每个)	ALL	<字段><比较>ANY(数据表|子查询)	测试字段值是否大于数据表或子查询结果集中的每一个值。
[不]存在	EXISTS	EXIST(<子查询>)	测试子查询的结果集中有[没有]记录。
在[不在]范围	[NOT]BETWEEN…AND…	<字>[NOT]BETWEEN 小值 AND 大值	测试字段在[不在]给定的小值和大值指定的范围中
是[不是]空值	IS [NOT] NULL	<字段> IS [NOT] NULL	测试字段是[不是]空值
模式比较	[NOT] LIKE	<字段> [NOT] LIKE < 字符常数>	测试字段值是否与给定的字符模式匹配
与运算	AND	<条件1> AND <条件2>	测试条件1和条件2是否都满足要求
或运算	OR	<条件1> OR <条件2>	测试条件1和条件2是否有一个满足要求
非运算	NOT	NOT <条件>	测试条件是否不满足要求

distinct操作符

distinct: 用来消除重复记录。

例： 查询fruits表中所有不重复的s_id

```
select distinct s_id from fruits;
```

子查询

子查询：写在()中，把内层查询结果当做外层查询参照的数据表来用

例： 用in操作符与子查询语句来查询所有f_id对应的f_price在10元到20元之间的水果记录

```
select * from fruits where f_id in
(select f_id from fruits where f_price between 10 and 20);
```

例： 用any操作符与子查询语句来查询所有f_id对应的f_price在10元到20元之间的水果记录

```
select * from fruits where f_id =  any
(select f_id from fruits where f_price between 10 and 20);
```

例： 用all操作符与子查询语句来查询所有f_price大于20元的水果记录

```
select * from fruits where f_price > all
(select f_price from fruits where f_price < 20);
```

例： 用exists操作符与子查询语句来查询是否存在f_price大于30元的水果记录

```
select * from fruits where exists
(select * from fruits where f_price > 30);
```

as重命名与limit限制查询结果行数

as：可以将表名重新命名为别的名称使用，只在查询中有效

例： 用as将fruits表名重命名为f后使用

```
select f.* from fruits as f;
```

limit：查询后只显示limit指定数字的行数结果

例： 显示f_price金额最大的前三名水果记录

```
select * from fruits
order by f_price desc
limit 3;
```

常用的数学函数

常用的数学函数：主要用于处理数字值

函数	说明
ABS(x)	返回 x 的绝对值
BIN(x)	返回 x 的二进制（OCT 返回八进制，HEX 返回十六进制）
EXP(x)	返回值 e（自然对数的底）的 x 次方
GREATEST(x1,x2,...,xn)	返回集合中最大的值
LEAST(x1,x2,...,xn)	返回集合中最小的值
LN(x)	返回 x 的自然对数
LOG(x,y)	返回 x 的以 y 为底的对数
MOD(x,y)	返回 x/y 的模（余数）
PI()	返回 pi 的值（圆周率）
RAND()	返回 0 到 1 内的随机值,可以通过提供一个参数(种子)使 RAND()随机数生成器生成一个指定的值。
FLOOR(x)	返回小于 x 的最大整数值，（去掉小数取整）
CEILING(x)	返回大于 x 的最小整数值，（进一取整）
ROUND(x,y)	返回参数 x 的四舍五入的有 y 位小数的值，（四舍五入）
TRUNCATE(x,y)	返回数字 x 截短为 y 位小数的结果
SIGN(x)	返回数字 x 的符号的值（正数返回 1，负数返回-1，0 返回 0）
SQRT(x)	返回一个数的平方根

常用的字符串函数

常用的字符串函数：主要用于处理字符串值

函数	说明
ASCII(char)	返回字符的 ASCII 码值
BIT_LENGTH(str)	返回字符串的比特长度
CONCAT(s1,s2...,sn)	将 s1,s2...,sn 连接成字符串
CONCAT_WS(sep,s1,s2...,sn)	将 s1,s2...,sn 连接成字符串，并用 sep 字符间隔
INSERT(str,x,y,instr)	将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr，返回结果
FIND_IN_SET(str,list)	分析逗号分隔的 list 列表，如果发现 str，返回 str 在 list 中的位置
LCASE(str)或 LOWER(str)	返回将字符串 str 中所有字符改变为小写后的结果
UCASE(str)或 UPPER(str)	返回将字符串 str 中所有字符转变为大写后的结果
LEFT(str,x)	返回字符串 str 中最左边的 x 个字符
RIGHT(str,x)	返回字符串 str 中最右边的 x 个字符
LENGTH(str)	返回字符串 str 中的字符数
POSITION(substr,str)	返回子串 substr 在字符串 str 中第一次出现的位置
QUOTE(str)	用反斜杠转义 str 中的单引号
REPEAT(str,srchstr,rplcstr)	返回字符串 str 重复 x 次的结果
REVERSE(str)	返回颠倒字符串 str 的结果
LTRIM(str)	去掉字符串 str 开头的空格
RTRIM(str)	去掉字符串 str 尾部的空格
TRIM(str)	去除字符串首部和尾部的所有空格

常用的日期及时间函数

日期及时间函数：用来处理日期时间型数据

函数	说明
DATE_FORMAT(date,fmt)	依照指定的 fmt 格式格式化日期 date 值
FROM_UNIXTIME(ts,fmt)	根据指定的 fmt 格式，格式化 UNIX 时间戳 ts
MONTHNAME(date)	返回 date 的月份名(英语月份，如 October)
DAYNAME(date)	返回 date 的星期名(英语星期几，如 Saturday)
NOW()	返回当前的日期和时间
CURDATE()或 CURRENT_DATE()	返回当前的日期
CURTIME()或 CURRENT_TIME()	返回当前的时间
QUARTER(date)	返回 date 在一年中的季度(1~4)
WEEK(date)	返回日期 date 为一年中第几周(0~53)
DAYOFYEAR(date)	返回 date 是一年的第几天(1~366)
DAYOFMONTH(date)	返回 date 是一个月的第几天(1~31)
DAYOFWEEK(date)	返回 date 所代表的一星期中的第几天(1~7)
YEAR(date)	返回日期 date 的年份(1000~9999)
MONTH(date)	返回 date 的月份值(1~12)
DAY(date)	返回 date 的天数部分
HOUR(time)	返回 time 的小时值(0~23)
MINUTE(time)	返回 time 的分钟值(0~59)
SECOND(time)	返回 time 的秒值（0-59）
DATE(datetime)	返回 datetime 的日期值
TIME(datetime)	返回 datetime 的时间值

其他函数

其他函数：除上述函数之外的一些常用函数

函数	说明
GROUP_CONCAT(col)	返回由属于一组的列值连接组合而成的结果
CAST()	将一个值转换为指定的数据类型

※GROUP_CONCAT()函数：常与关键字 GROUP BY 一起使用，能够将分组后指定的字段值都显示出来。

例： 使用group_concat函数查询不同s_id下对应的所有f_name信息

```
SELECT s_id, GROUP_CONCAT(f_name) FROM fruits
GROUP BY s_id;
```

为字段赋值update…set 与 删除记录 delete

update…set：为字段赋值，语法为update 表名 set 字段名 = 值;

例： 使用concat函数在f_name字段值前添加'fruit_’信息

```
update fruits set f_name = concat('fruit_',f_name);
```

delete： 删除数据表中的数据，语法为DELETE FROM 表名 [WHERE Clause]，如果省略where的话则删除表中所有数据记录

例： 删除f_id为'b5'的数据记录

```
delete from fruits where f_id = 'b5';
```

单表查询练习：彩票数据核对练习（1）

彩票游戏规则：

彩票有10个刮奖区，每个刮奖区有一个图符和一个奖符，

彩票中奖金额 = 同一个刮奖区内的（图符倍数 * 奖符金额）

每张彩票售价5元

起始范围	位数	内容	示例	备注
1-11	11	本号-票号	0000001-000	100张彩票为1本，本号+票号确定唯一彩票
12-21	10	图符	abcdefghij	决定奖金倍数，m为1倍，n\o\p为2倍
22-31	10	奖符	ABCDEFGHIJ	代表奖金金额

单表查询练习：彩票数据核对练习（2）

1. 计算每张彩票的中奖金额
2. 求总中奖张数及金额
3. 求各不同奖符的张数及金额（奖符为5元、10元等）
4. 求中奖张数与总张数占比
5. 检查每个本号中有100张彩票
6. 检查每本彩票中最多只有一张中奖彩票金额超过50元
7. 检查每本彩票中最多只有连续7张无奖票

多表查询练习：电商数据查询练习

1. 倒序查询卖的金额最多的产品

2. 查询不同尺码下的产品销售数量

3. 查询不同颜色下的产品销售金额

4. 查询不同尺码下的不同颜色的产品销售金额

谢谢！

讲师姓名

12/15/2017