MySQL数据库

# 一、MySQL 数据类型

MySQL支持所有标准SQL数值数据类型。

数值类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
| 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 |

日期和时间类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **大小 (字节)** | **范围** | **格式** | **用途** |
| DATE | 3 | 1000-01-01/9999-12-31 | YYYY-MM-DD | 日期值 |
| TIME | 3 | '-838:59:59'/'838:59:59' | HH:MM:SS | 时间值或持续时间 |
| YEAR | 1 | 1901/2155 | YYYY | 年份值 |
| DATETIME | 8 | 1000-01-01 00:00:00/9999-12-31 23:59:59 | YYYY-MM-DD HH:MM:SS | 混合日期和时间值 |
| TIMESTAMP | 4 | 1970-01-01 00:00:00/2038  结束时间是第 **2147483647** 秒，北京时间 **2038-1-19 11:14:07**，格林尼治时间 2038年1月19日 凌晨 03:14:07 | YYYYMMDD HHMMSS | 混合日期和时间值，时间戳 |

字符串类型

|  |  |  |
| --- | --- | --- |
| **类型** | **大小** | **用途** |
| CHAR | 0-255字节 | 定长字符串 |
| VARCHAR | 0-65535 字节 | 变长字符串 |
| TINYBLOB | 0-255字节 | 不超过 255 个字符的二进制字符串 |
| TINYTEXT | 0-255字节 | 短文本字符串 |
| BLOB | 0-65 535字节 | 二进制形式的长文本数据 |
| TEXT | 0-65 535字节 | 长文本数据 |
| MEDIUMBLOB | 0-16 777 215字节 | 二进制形式的中等长度文本数据 |
| MEDIUMTEXT | 0-16 777 215字节 | 中等长度文本数据 |
| LONGBLOB | 0-4 294 967 295字节 | 二进制形式的极大文本数据 |
| LONGTEXT | 0-4 294 967 295字节 | 极大文本数据 |

# 二、MySQL指令

**创建库**

1. Create database database\_name;
2. Use database\_name;
3. Show databases;
4. Show tables;
5. Show columns from table\_name
6. show index from table\_name; #显示索引信息

**创建表**

1. Create table table\_name();

**删除表**

1. DROP TABLE table\_name;

**插入表数据**

1. Insert into table\_name (field) values (value);

**读取数据**

1. Select \* from table\_name;
2. Select column\_name from table\_name [where N];
3. SELECT field1, field2,...fieldN FROM table\_name1, table\_name2... [WHERE condition1 [AND [OR] condition2..... ];

**更改数据**

1. UPDATE table\_name SET field1=new-value1, field2=new-value2 [WHERE Clause];

**删除表中数据**

1. Delete from table\_name [where conditions];

**读取数据**

1. Select field1,field2 from table\_name where field1 LIKE condition1 [and [or]] filed2 = ‘aomevalue’;
2. SELECT expression1, expression2, ... expression\_n FROM tables [WHERE conditions]

UNION [ALL | DISTINCT]

SELECT expression1, expression2, ... expression\_n FROM tables [WHERE conditions];

1. Select field1, field2 from table\_name1, table\_name2 ordered by field1, [field2] [ASC [DESC]];
2. Select column\_name, function(column\_name)

From table\_name

Where column\_name operator value

GROUP BY column\_name [with rollup]; #with rollup 在分组统计的基础上再进行相同的统计

1. Select coalesce(a,b,c); 参数说明：如果a==null,则选择b；如果b==null,则选择c；如果a!=null,则选择a；如果a b c 都为null ，则返回为null（没意义）。

以下实例中如果名字为空我们使用总数代替：

Eg： select coalesce(name,’总数’), sum(sighin) as signin\_count from employee\_tb1 GROUP BY name WITH ROLLUP;

1. **连接**

在真正的应用中经常需要从多个数据表中读取数据。使用 MySQL 的 JOIN 在两个或多个表中查询数据。可以在 SELECT, UPDATE 和 DELETE 语句中使用 Mysql 的 JOIN 来联合多表查询。

INNER JOIN（内连接,或等值连接）：获取两个表中字段匹配关系的记录，求交集；

LEFT JOIN（左连接）：获取左表所有记录，即使右表没有对应匹配的记录，求左侧的表；

RIGHT JOIN（右连接）： 与 LEFT JOIN 相反，用于获取右表所有记录，即使左表没有对应匹配的记录，求右侧的表。

SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a INNER JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a LEFT JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a RIGHT JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

1. **关于选择Null的条件**

select \* , columnName1+ifnull(columnName2,0) from tableName;

columnName1，columnName2 为 int 型，当 columnName2 中，有值为 null 时，columnName1+columnName2=null， ifnull(columnName2,0) 把 columnName2 中 null 值转为 0。

MySQL 使用 SQL SELECT 命令及 WHERE 子句来读取数据表中的数据,但是当提供的查询条件字段为 NULL 时，该命令可能就无法正常工作。**IS NULL:** 当列的值是NULL,此运算符返回 true。

**IS NOT NULL:** 当列的值不为 NULL, 运算符返回 true。

**<=>:** 比较操作符（不同于=运算符），当比较的的两个值为 NULL 时返回 true。

1. **MySQL正则表达式**:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **^** | **$** | **.** | **[…]** | **[^…]** | **P1|P2|P3** | **\*** | **+** | **{n}** | **{n,m}** |

1. **MySQL事务**

MySQL 事务主要用于处理操作量大，复杂度高的数据。

在 MySQL 中只有使用了 Innodb 数据库引擎的数据库或表才支持事务。事务处理可以用来维护数据库的完整性，保证成批的 SQL 语句要么全部执行，要么全部不执行。

事务用来管理 insert,update,delete 语句.

事务是必须满足4个条件（ACID）：原子性（**A**tomicity，或称不可分割性）、一致性（**C**onsistency）、隔离性（**I**solation，又称独立性）、持久性（**D**urability）。

原子性：事务的执行必须完整；一致性：事务开始前和结束后，数据库完整性没有破坏；隔离性：允许多个事务并发执行，事务隔离分为不同的等级，包括未提交，读提交，可重复读，串行化；持久性：事务结束后，对数据的修改是永久的。

**备注**：在 MySQL 命令行的默认设置下，事务都是自动提交的，即执行 SQL 语句后就会马上执行 COMMIT 操作。因此要显式地开启一个事务必须使用命令 BEGIN 或 START TRANSACTION，或者执行命令 SET AUTOCOMMIT=0，用来禁止使用当前会话的自动提交。

**事务控制语句：**

BEGIN或START TRANSACTION；显式地开启一个事务。

COMMIT；也可以使用COMMIT WORK，不过二者是等价的。COMMIT会提交事务，并使已对数据库进行的所有修改成为永久性的；

ROLLBACK；有可以使用ROLLBACK WORK，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；

SAVEPOINT identifier；SAVEPOINT允许在事务中创建一个保存点，一个事务中可以有多个SAVEPOINT；

RELEASE SAVEPOINT identifier；删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；

ROLLBACK TO identifier；把事务回滚到标记点；

SET TRANSACTION；用来设置事务的隔离级别。InnoDB存储引擎提供事务的隔离级别有READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ和SERIALIZABLE。

**MYSQL 事务处理主要有两种方法：**

1、用 BEGIN, ROLLBACK, COMMIT来实现

**BEGIN** 开始一个事务

**ROLLBACK** 事务回滚

**COMMIT** 事务确认

2、直接用 SET 来改变 MySQL 的自动提交模式:

**SET AUTOCOMMIT=0** 禁止自动提交

**SET AUTOCOMMIT=1** 开启自动提交

1. **Alter 命令：**当我们需要修改数据表名或者修改数据表字段时，就需要使用到MySQL ALTER命令。

**对表字段修改**

删除表字段：alter table table\_name drop field\_name;

添加表字段：alter table table\_name add field\_name field\_type [fisrt [after field\_name]];

修改表字段：alter table table\_name modify field\_name new\_field\_type;

alter table table\_name change field\_name new\_field\_name new\_field\_type;

**对默认值修改**

修改默认值：alter table table\_name modify field\_name new\_field\_type [not Null] [default new\_default\_value];

删除默认值：alter table table\_name alter field\_name drop default;

**对表的修改**

修改表类型：Alter table\_name engine = new\_table\_type;

备注：表的类型有innodb, myisam

修改表名：alter table\_name rename to new\_table\_name;

1. MySQL索引

MySQL索引的建立对于MySQL的高效运行是很重要的，索引可以大大提高MySQL的检索速度。

**普通索引：**

创建索引：Create index index\_name on table\_name(field\_name(length));

**备注**：如果是CHAR，VARCHAR类型，length可以小于字段实际长度；如果是BLOB和TEXT类型，必须指定 length。

添加索引：Alter table table\_name add index index\_name(column\_name);

删除索引：drop index [index\_name] on table\_name;

**唯一索引：**

创建索引：create unique index index\_name on table\_name(field\_name

(length));

添加索引：alter table table\_name add unique index [index\_name] (field\_name(length));

**使用ALTER 命令添加和删除索引:**

ALTER TABLE tbl\_name ADD PRIMARY KEY (column\_list): 该语句添加一个主键，这意味着索引值必须是唯一的，且不能为NULL。

ALTER TABLE tbl\_name ADD UNIQUE index\_name (column\_list): 这条语句创建索引的值必须是唯一的（除了NULL外，NULL可能会出现多次）。

ALTER TABLE tbl\_name ADD INDEX index\_name (column\_list): 添加普通索引，索引值可出现多次。

ALTER TABLE tbl\_name ADD FULLTEXT index\_name (column\_list):该语句指定了索引为 FULLTEXT ，用于全文索引。

Eg: alter table table\_name add index index\_name(field\_name)

Alter table table\_name drop index index\_name;

**使用alter命令添加和删除主键：**

为了保证主要默认不为空（not Null）:

Alter table table\_name modify field\_name field\_type not Null;

然后添加主键：

Alter table table\_name add primary key (field\_name);

删除主键：

Alter table table\_name drop primary key;

**备注：**删除主键时只需指定PRIMARY KEY，但在删除索引时，你必须知道索引名。

**显示索引信息：**show index from table\_name;

1. 临时表

MySQL 临时表在我们需要保存一些临时数据时是非常有用的。临时表只在当前连接可见，当关闭连接时，Mysql会自动删除表并释放所有空间。

Create temporary table temporary\_table\_name ();

Insert into temporary\_table\_name ();

Select \* from temporary\_table\_name;

Drop table temporary\_table\_name;

1. 复制表

如果我们需要完全的复制MySQL的数据表，包括表的结构，索引，默认值等。 如果仅仅使用**CREATE TABLE ... SELECT**命令，是无法实现的。

完整的复制MySQL数据表：1）使用 **SHOW CREATE TABLE**命令获取创建数据表(CREATE TABLE) 语句，该语句包含了原数据表的结构，索引等；2）复制以下命令显示的SQL语句，修改数据表名，并执行SQL语句，通过以上命令 将完全的复制数据表结构；3）如果你想复制表的内容，你就可以使用 INSERT INTO ... SELECT 语句来实现。

1. MySQL元数据

查询结果信息： SELECT, UPDATE 或 DELETE语句影响的记录数。

数据库和数据表的信息： 包含了数据库及数据表的结构信息。

MySQL服务器信息： 包含了数据库服务器的当前状态，版本号等。

1. MySQL序列

MySQL 序列是一组整数：1, 2, 3, ...，由于一张数据表只能有一个字段自增主键， 如果你想实现其他字段也实现自动增加，就可以使用MySQL序列来实现。

**使用序列：**

MySQL 中最简单使用序列的方法就是使用 MySQL AUTO\_INCREMENT 来定义列。

在MySQL的客户端中你可以使用 SQL中的LAST\_INSERT\_ID( ) 函数来获取最后的插入表中的自增列的值。

**重置序列：**

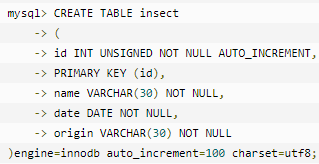
如果你删除了数据表中的多条记录，并希望对剩下数据的AUTO\_INCREMENT列进行重新排列，那么你可以通过删除自增的列，然后重新添加来实现。 不过该操作要非常小心，如果在删除的同时又有新记录添加，有可能会出现数据混乱。

Alter table table\_name drop field\_name; #备注：这个列是被定义为序列的

Alter table table\_name add field\_name field\_type Not Null auto\_increment fisrt;

设置序列开始值：

一般情况下序列的开始值为1，但如果你需要指定一个开始值100，那我们可以通过以下语句来实现：



或者创建了序列后，通过alter来修改：alter table table\_name auto\_increment = 100;

1. **MySQL处理重复数据**

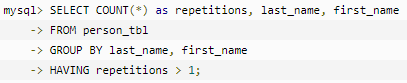
可以在MySQL数据表中设置指定的字段为**PRIMARY KEY（主键）**或者**UNIQUE（唯一）**索引来保证数据的唯一性。

Insert ignore into table\_name (field1,field2) values(value1,value2); #这个语句会忽略重复的数据，不插入；

Insert into table\_name (field1,field2) values(value1,value2);

**备注：**INSERT IGNORE INTO当插入数据时，在设置了记录的唯一性后，如果插入重复数据，将不返回错误，只以警告形式返回。 而REPLACE INTO into如果存在primary 或 unique相同的记录，则先删除掉。再插入新记录。

**统计重复数据：**



以上查询语句将返回 person\_tbl 表中重复的记录数。 一般情况下，查询重复的值，请执行以下操作：

确定哪一列包含的值可能会重复。

在列选择列表使用COUNT(\*)列出的那些列。

在GROUP BY子句中列出的列。

HAVING子句设置重复数大于1。

**过滤重复数据：**

Select distinct field1,field2 from table\_name;

Select field1,field2 from table\_name group by (field1,field2); #读取不重复的数据

**删除重复数据：**

首先将不重复的数据提取出来，重新创建一个表；然后删除原来的表；最后修改新表为原来的表名；

或者直接在表中添加索引和主键来删除重复记录：

Alter ignore table table\_name add primary key (field1,field2);

**备注：**括号中的field可以为一个或者多个

1. **MySQL及SQL注入**

如果通过网页获取用户输入的数据并将其插入一个MySQL数据库，那么就有可能发生SQL注入安全的问题。

所谓**SQL注入**，就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

**防止SQL注入，我们需要注意以下几个要点：**

1.永远不要信任用户的输入。对用户的输入进行校验，可以通过正则表达式，或限制长度；对单引号和 双"-"进行转换等。

2.永远不要使用动态拼装sql，可以使用参数化的sql或者直接使用存储过程进行数据查询存取。

3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息直接存放，加密或者hash掉密码和敏感的信息。

5.应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装

6.sql注入的检测方法一般采取辅助软件或网站平台来检测，软件一般采用sql注入检测工具jsky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN等。采用MDCSOFT-IPS可以有效的防御SQL注入，XSS攻击等。

1. MySQL导出数据

Select \* from table\_name into outfile ‘路径’

FIELDS TERMINATED BY ‘,’ ENCLOSED BY ‘”’, LINES TERMINATED BY ‘\r\n’;

**备注：**

**SELECT ... INTO OUTFILE 语句有以下属性:**

LOAD DATA INFILE是SELECT ... INTO OUTFILE的逆操作，SELECT句法。为了将一个数据库的数据写入一个文件，使用SELECT ... INTO OUTFILE，为了将文件读回数据库，使用LOAD DATA INFILE。

SELECT...INTO OUTFILE 'file\_name'形式的SELECT可以把被选择的行写入一个文件中。该文件被创建到服务器主机上，因此您必须拥有FILE权限，才能使用此语法。

输出不能是一个已存在的文件。防止文件数据被篡改。

需要有一个登陆服务器的账号来检索文件。否则 SELECT ... INTO OUTFILE 不会起任何作用。

在UNIX中，该文件被创建后是可读的，权限由MySQL服务器所拥有。这意味着，虽然你就可以读取该文件，但可能无法将其删除。

**导出表作为原始数据**

mysqldump 是 mysql 用于转存储数据库的实用程序。它主要产生一个 SQL 脚本，其中包含从头重新创建数据库所必需的命令 CREATE TABLE INSERT 等。使用 mysqldump 导出数据需要使用 --tab 选项来指定导出文件指定的目录，该目标必须是可写的。

开启服务后：

导出表

$ mysqldump –u –root –p database\_name table\_name > filename.txt

导出库

$ mysqldump –u –root –p database\_name > filename.txt

导出所有库

$ mysqldump –u –root –p –all-databases > filename.txt

导入远程的服务器上，保证两台服务器是相通的，是可以相互访问的：

$ mysqldump –u –root –p database\_name mysql –h other-host.com database\_name

1. MySQL导入服务

法1：行

Mysql –u user\_name –p database\_name < 要导入的数据库数据(filename.sql)

Eg: mysql –u root –p 123456 < mytestdump.sql

法2：行

Create database database\_name;

Use database\_name;

Set names utf8;

Source database\_name.sql;

法3：

Load data local infile ‘filename.txt’ into table table\_name;

1. MySQL函数

MySQL 字符串函数

MySQL 数字函数

MySQL 日期函数

MySQL 高级函数

1. MySQL运算符

算术运算符

比较运算符

逻辑运算符

位运算符

优先级