1. MySql登录

|  |  |
| --- | --- |
| 参数 | 描述 |
| -D, --database=name | 打开指定的数据库 |
| --delimiter=name | 指定分隔符 |
| -h, --host=name | 服务器名称 |
| -p, --password[=name] | 密码 |
| -p, --port=# | 端口号 |
| --prompt=name | 设置提示符 |
| -u, user=name | 用户名 |
| -V, --version | 输出版本号 |

2. MySql退出： exit; quit; 或者\q;

3. 修改MySQL提示符：连接上客户端后，通过·prompt 提示符·修改

|  |  |
| --- | --- |
| 参数 | 描述 |
| \D | 完整的日期 |
| \d | 当前的数据库名 |
| \h | 主机名 |
| \u | 当前用户 |

4. 常用MySQL命令

* SELECT VERSION(): 显示当前服务器版本
* SELECT NOW(): 显示当前日期
* SELECT USER(): 显示当前用户

5. 每个SQL语句必须以分号结尾。

6. 查看当前服务器下的数据表列表

SHWO {DATABASES | SCHEMAS} [LIKE ‘pattern’ | WHERE expr];

7. 其他命令：

SHOW WARNINGS; 查看警告信息

SHOW CREATE DATABASE {db\_name}: 查看数据库创建时信息

8. 创建数据库：

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db\_name [DEFAULT] CHARACTER SET [=] charset\_name;

9. 修改数据库

ALTER {DATABASE | SCHEMA} [db\_name] [DAFAULT] CHARACTER SET [=] charset\_name;

10. 数据类型是指列、存储过程参数、表达式和局部变量的数据特征，它决定了数据的存储格式，代表了不同的信息类型。**选择最合适的数据类型**。

|  |  |  |
| --- | --- | --- |
| **数据类型** | **存储范围** | **字节** |
| tinyint | 有符号值: -128 到 127  有符号值： 0 到 255 | 1 |
| samllint | 有符号值：-32768 到 32767  无符号值：0 到 65535 | 2 |
| mediumint | 有符号值：-223 到 223-1  无符号值：0 到 224-1 | 3 |
| int [unsigned] | 有符号值：-231 到 231-1  无符号值：0 到 232-1 | 4 |
| bigint | 有符号值：-263 到 263-1  无符号值：0 到 264-1 | 8 |

|  |  |
| --- | --- |
| 数据类型 | 描述 |
| float([M,D]) | M是数字总位数，D是小数点后面的位数。若被忽略，根据硬件的限制来保存值。单精度浮点数精确到大约7为小数 |

11. 日期时间类型

|  |  |
| --- | --- |
| 列类型 | 存储需求 |
| year | 1 |
| time | 3 |
| date | 3 |
| datetime | 8 |
| timestamp | 4 |

12. 字符型

|  |  |
| --- | --- |
| 列类型 | 存储需求 |
| char(M) | M个字节，0<=M<=255 |
| varchar(M) | L+1个字节，其中L<=M且0<=M<=65535 |
| tinytext | L+1个字节，其中L<28 |
| text | L+2个字节，其中L<216 |
| mediumtext | L+3个字节，其中L<224 |
| longtext | L+4个字节，其中L<232 |
| enum(‘val1’,”val2’,...) | 1或2个字节，取决于枚举值的个数（最多65535） |
| set(‘val1’,’val2’,...) | 1、2、3、4或者8个字节，取决于set成员数目（最多64个成员） |

1. SELECT DATABASE(): 查看当前打开的数据库

2. 创建表

|  |
| --- |
| **CREATE** **TABLE** **[IF** **NOT** **EXISTS]** table\_name **(**  column\_name data\_type**,**  **...**  **);**  **CREATE** **TABLE** tb1 **(**  id **TINYINT** UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY,** -- 自增的主键  username **VARCHAR(**20**)** **NOT** **NULL** **UNIQUE** **KEY,** -- 唯一约束  age **TINYINT(**1**)** UNSIGNED **NULL** **DEFAULT** 3**,** -- 当插入记录时，若没有明确赋值，则赋予默认值  salary **FLOAT(**8**,**2**)** UNSIGNED  **);** |

3. 查看数据表列表

|  |
| --- |
| SHOW TABLES **[FROM** **db\_name]** **[LIKE** `pattern` **|** **WHERE** expr**];**  SHOW COLUMNS **FROM** tb\_name**;** -- 查看数据表结构 |

4. 插入数据 INSERT

|  |
| --- |
| **INSERT** **[INTO]** tb\_name **[(col\_name,** **...)]** **VALUES(**val**,** **...)** -- 如果省略列名，所有的value都必须有  **INSERT** tb1 **VALUES(**'Tom'**,** 25**,** 7863.25**);** |

5. 自动编号，必须与主键组合使用。默认情况下，起始值为1，每次增量为1

6. 唯一约束： UNIQUE KEY

唯一约束可以保证记录的唯一性；

唯一约束的字段可以为空值(NULL)；

每张表可以存在多个唯一约束。

7. 约束

约束保证数据的完整性和一致性

约束分为表级约束和列级约束

NOT NULL, PRIMARY KEY, UNIQUE KEY, DEFAULT, FOREIGN KEY

8. 外键约束的要求

* 父表和子表必须使用相同的存储引擎，而且禁止使用临时表
* 数据的存储引擎只能为InnoDB
* 外键列和参照列必须具有相似的数据类型。其中数字的长度或是否有符号位必须相同；而字符的长度则可以不同
* 外键列和参照列必须创建索引，如果外键列不存在索引的话，MySQL将自动创建索引

9. 外键约束的参考

* CASCADE: 从父表删除或更新且自动删除或更新子表中匹配的行
* SET NULL：从父表删除或更新行，并设置子表中的外键列为NULL。如果使用该项，必须保证子表列中没有指定NOT NULL。
* RESTRICT:拒绝对父表的删除或更新操作
* NO ACTION：标准SQL的关键字，在MySQL中与RESTRICT相同

10. 表级约束与列级约束

列级约束：对于一个数据列建立约束

表级约束：对多个数据列建立的约束

列级约束即可以在列定义时声明，也可以在列定义后声明

表级约束只能在列定义后声明

11. 修改数据表

* 添加单列：FIRST(所有列的最前面) AFTER(某一列的后面)指定要插入列的位置

|  |
| --- |
| **ALTER** **TABLE** tb\_name **ADD** **[COLUMN]** **col\_name** column\_definition **[FIRST** **|** **AFTER]** |

* 添加多列：不能指定位置
* 删除列：**ALTER** **TABLE** tb\_name **DROP** col1**,** **DROP** col2**;**

10. 编辑数据表默认存储引擎

MySQL 配置文件 default-storage-engine=INNODB (my.ini)

字符函数

|  |  |  |
| --- | --- | --- |
| 函数名称 | 描述 | 例子 |
| CONCAT() | 字符连接 | CONCAT(f, l) |
| CONCAT\_WS() | 使用指定的分隔符连接 | CONCAT(‘-’,’m’,’n’,...) |
| FORMAT() | 数字格式化 | FORMAT(123.34,2) |
| LOWER | 将字符串转换成小写 |  |
| UPPER | 将字符串转换成大写 |  |
| LEFT() | 获取左侧字符 | LEFT(‘MySQL’,2) |
| RIGHT() | 获取右侧字符 |  |
| LENGTH() | 获取字符串长度 |  |
| LTRIM() | 删除前导空格 |  |
| RTRIM() | 删除后导空格 |  |
| TRIM() | 删除前后导空格 |  |
| SUBSTRING(str,start,  length) | 字符串截取 |  |
| [NOT] LIKE |  | LIKE ‘%1%%’ ESCAPE ‘1’ |
| REPLACE() | 字符串替换 |  |

数值运算符

|  |  |
| --- | --- |
| 函数名 | 描述 |
| CEIL() | 进一取整 |
| DIV | 整数除法 |
| FLOOR() | 舍一取整 |
| MOD | 取余运算 |
| POWER() | 幂运算 |
| ROUND() | 四舍五入 |
| TRUNCATE() | 数字截取 |

比较运算符和函数

|  |  |
| --- | --- |
| 名称 | 描述 |
| [NOT] BETWEEN ... AND ... | 是否在范围之内 |
| [NOT] IN() | 是否在列出值的范围内 |
| IS [NOT] NULL | 是否为空 |

日期函数

|  |  |
| --- | --- |
| 名称 | 描述 |
| NOW() | 当前日期和时间 |
| CURDATE() | 当前日期 |
| CURTIME() | 当前时间 |
| DATE\_ADD() | 日期变化 |
| DATEDIFF() | 日期差值 |
| DATE\_FORMAT() | 日期格式化 |

信息函数

|  |  |
| --- | --- |
| 名称 | 描述 |
| CONNECTION\_ID() | 连接ID |
| DATEBASE() | 当前的数据库 |
| LAST\_INSERT\_ID() | 最后插入记录的ID，插入多条，只返回的第一个 |
| USER() |  |
| VERSION() |  |

聚合函数

|  |  |
| --- | --- |
| 名称 | 描述 |
| AVG() | 平均值 |
| COUNT() | 计数 |
| MAX() | 最大值 |
| MIN() | 最小值 |
| SUM() | 求和 |

加密函数

|  |  |
| --- | --- |
| 名称 | 描述 |
| MD5() | 信息摘要算法 |
| PASSWORD() | 密码算法 |

自定义函数：

存储过程：SQL语句和控制语句的预编译集合，以一个名称存储并作为一个单元处理，省去了编译过程。

存储过程与函数的区别：

* 存储过程实现的功能要复杂一些；而函数的针对性更强
* 存储过程可以返回多个值；函数只能有一个返回值
* 存储过程一般独立来执行；而函数可以作为其他SQL语句的组成部分来执行

存储引擎

* MyISAM
* InnoDB
* Memory
* CSV
* Archive

并发控制：当多个连接对记录进行修改时保证数据的一致性和完整性。

事务：事务用于保证数据库的完整性：原子性；一致性；隔离性；持久性

外键：保证数据一致性

索引：对数据表中一列或多列的值进行排序的一种结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 特点 | MyISAM | InnoDB | Memory（内存中） | Archive |
| 存储限制 | 256TB | 64TB | 有 | 无 |
| 事务安全 | - | 支持 | - | - |
| 支持索引 | 支持 | 支持 | 支持 | - |
| 锁颗粒 | 表锁 | 行锁 | 表锁 | 行锁 |
| 数据压缩 | 支持 | - | - | 支持 |
| 支持外键 | - | 支持 | - | - |

修改数据引擎：

* 通过修改MySQL配置文件实现： --default-storage-engine = engine
* 创建数据表的命令实现： CREATE TABLE tb1 (...) ENGINE=engine
* 修改数据表的引擎：ALTER TABLE tb1 ENGIN=engine