**Mysql学习笔记**

**注：中括号内为可选输入**

**1. 添加数据库**

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

**2. 修改数据库**

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

**3. 删除数据库**

**DROP {DATABASE | SCHEMA}} [IF NOT EXISTS] db\_name;**

**4. 创建数据表**

**CREATE TABLE [IF NOT EXISTS] table\_name(**

**column\_name data\_type,**

**...**

**);**

**5. 查看数据表列表**

**SHOW TABLES [FROM db\_name] [LIKE 'pattern' | WHERE expr];**

**6. 查看数据表结构**

**SHOW COLUMNS FROM table\_name;**

**7. 插入记录**

**INSERT [INTO] table\_name [(column\_name,...)] VALUES(val,...);**

**INSERT [INTO] table\_name SET column\_name={expr | DEFAULT}, ...;(此方法可以使用子查询)**

**INSERT [INTO] table\_name [(column\_name,...)] SELECT ...;(此方法可以将查询结果插入到指定数据表)**

**8. 查找记录**

**SELECT select\_expr [, select\_expr ...]**

**[**

**FROM table\_references**

**[WHERE where\_condition]**

**[GROUP BY {column\_name | position} [ASC | DESC], ...]**

**[HAVING where\_condition]**

**[ORDER BY {column\_name | expr | position} [ASC | DESC], ...]**

**[LIMIT {[offset,] row\_count | row\_count OFFSET offset}]**

**];**

**查找表达式（select\_expr）:**

**每一个表达式表示想要的一列，必须至少有一个。**

**多个列之间以英文逗号分隔。**

**星号（\*）表示所有列。Table\_name.\*可以表示命名表的所有列。**

**查询表达式可以使用[AS] alias\_name为其赋予别名。**

**别名可用于GROUP BY,ORDER BY或HAVING子句。**

**9. 空与非空(CONSTRAINT)**

**· NULL,字段值可以为空**

**· NOT NULL,字段值禁止为空**

**10. AUTO\_INCREMENT**

**· 自动编号，且必须与主键组合使用**

**· 默认情况下，起始值为1，每次的增量为1**

**11. PRIMARY KEY(CONSTRAINT)**

**· 主键约束**

**· 每张数据表只能存在一个主键**

**· 主键保证记录的唯一性**

**· 主键自动为NOT NULL**

**12. UNIQUE KEY(CONSTRAINT)**

**· 唯一约束**

**· 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)**

**· 每张数据表可以存在多个唯一约束**

**13. DEFAULT(CONSTRAINT)**

**· 默认值**

**· 当插入记录时，如果没有明确为字段赋值，则自动赋予默认值**

**14. FOREIGN KEY(CONSTRAINT)**

**要求：**

**1. 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。**

**2. 数据表的存储引擎只能为InnoDB。**

**3. 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**

**4. 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。**

**参照操作：**

**1. CASCADE：从父表删除或更新行，自动删除或更新子表中匹配的行。**

**2. SET NULL：从父表删除或更新行，并设置子表中的外键列为NULL。如果使用该选项，必须保证子表列没有指定NOT NULL。**

**3. RESTRICT：拒绝对父表的删除或更新操作。**

**4. NO ACTION：标准SQL的关键字，在Mysql中与RESTRICT相同。**

**15. 表级约束与列级约束**

**· 对一个数据列建立的约束，称为列级约束。**

**· 对多个数据列建立的约束，称为表级约束。**

**· 列级约束既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。**

**16. 添加单列**

**ALTER TABLE table\_name ADD [COLUMN] column\_name column\_definition [FIRST | AFTER column\_name];**

**17. 添加多列**

**ALTER TABLE table\_name ADD [COLUMN] (column\_name column\_definition,...);**

**18. 删除列**

**ALTER TABLE table\_name DROP [COLUMN] column\_name;**

**19. 添加主键约束**

**ALTER TABLE table\_name ADD [CONSTRAINT [symbol]] PRIMARY KEY [index\_type] (index\_col\_name,...);**

**20. 添加唯一约束**

**ALTER TABLE table\_name ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index\_name] [index\_type] (index\_col\_name,...);**

**21. 添加外键约束**

**ALTER TABLE table\_name ADD [CONSTRAINT [symbol]] FOREIGN KEY [index\_name] (index\_col\_name,...) reference\_definition**

**22. 添加/删除默认约束**

**ALTER TABLE table\_name ALTER [COLUMN] column\_name {SET DEFAULT literal(默认值) | DROP DEFAULT}**

**23. 删除主键约束**

**ALTER TABLE table\_name DROP PRIMARY KEY;**

**24. 删除唯一约束**

**ALTER TABLE table\_name DROP {INDEX|KEY} index\_name;**

**25. 删除外键约束**

**ALTER TABLE table\_name DROP FOREIGN KEY fk\_symbol;**

**26. 修改列定义**

**ALTER TABLE table\_name MODIFY [COLUMN] column\_name column\_definition [FIRST | AFTER column\_name];**

**27. 修改列名称**

**ALTER TABLE table\_name CHANGE [COLUMN] old\_column\_name new\_column\_name column\_definition [FIRST | AFTER column\_name];**

**28. 数据表更名**

**ALTER TABLE old\_table\_name RENAME [TO|AS] new\_table\_name;**

**29. 新建记录为默认的且自动自增的字段赋值，可以赋予NULL或DEFAULT。**

**30. 更新记录**

**单表更新：**

**UPDATE [LOW\_PRIORITY] [IGNORE] table\_reference SET column\_name1={expr1|DEFAULT} [, column\_name2 = {expr2|DEFAULT}, ...] [WHERE where\_condition]**

**多表更新：**

**UPDATE table\_references SET col\_name1 = {expr1|DEFAULT} [, col\_name2 = {expr2|DEFAULT}] ... [WHERE where\_condition]**

**31. 删除记录**

**单表删除：**

**DELETE FROM table\_name [WHERE where\_condition]**

**多表删除：**

**DELETE table\_name[.\*] [,table\_name[.\*]] ... FROM table\_references [WHERE where\_condition]**

**32. 子查询**

**1. 子查询指嵌套在查询内部，且必须始终出现在圆括号内。**

**2. 子查询可以包含多个关键字或条件，如DISTINCT、GROUP BY、ORDER BY、LIMIT、函数等。**

**3. 子查询的外层查询可以是：SELECT、INSERT、UPDATE、SET、DO。**

**33. 表的参照**

**table\_reference {[INNER | CROSS] JOIN | {LEFT | RIGHT} [OUTER] JOIN} table\_reference ON conditional\_expr**

**34. 无限分类的数据表设计**

**CREATE TABLE tdb\_goods\_types(**

**type\_id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,**

**type\_name VARCHAR(20) NOT NULL,**

**parent\_id SMALLINT UNSIGNED NOT NULL DEFAULT 0**

**);**

**35. 字符函数**

|  |  |
| --- | --- |
| 函数名称 | 描述 |
| CONCAT() | **字符连接** |
| CONCAT\_WS()（第一个参数为分隔符） | **使用指定的分隔符进行字符连接** |
| FORMAT() | **数字格式化** |
| LOWER() | **转换成小写字母** |
| UPPER() | **转换成大写字母** |
| LEFT() | **获取左侧字符** |
| RIGHT() | **获取右侧字符** |
| LENGTH() | **获取字符串长度** |
| LTRIM() | **删除前导空格** |
| RTRIM() | **删除后续空格** |
| TRIM() | **删除前导和后续空格** |
| SUBSTRING() | **字符串截取** |
| [NOT] LIKE | **模式匹配（%(百分号)代表任意个字符，**  **\_(下划线)代表任意一个字符）** |
| REPLACE() | **字符串替换** |

**36. 数值运算符与函数**

|  |  |
| --- | --- |
| 名称 | 描述 |
| CEIL() | **向上取整** |
| DIV | **整数除法** |
| FLOOR() | **向下取整** |
| MOD | **取余数（模）** |
| POWER() | **幂运算** |
| ROUND() | **四舍五入** |
| TRUNCATE() | **数字截取** |

**37. 比较运算符和函数**

|  |  |
| --- | --- |
| 名称 | 描述 |
| [NOT] BETWEEN...AND... | **[不]在范围之内** |
| [NOT] IN() | **[不]在列出值之内** |
| IS [NOT] NULL | **[不]为空** |

**38. 日期时间函数**

|  |  |
| --- | --- |
| 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** |
| DATEFORMAT() | **日期格式化** |

**39. 信息函数**

|  |  |
| --- | --- |
| 名称 | 描述 |
| CONNECTION\_ID() | **连接ID** |
| DATABASE() | **当前数据库** |
| LAST\_INSERT\_ID() | **最后插入记录的ID号(一次插入多条记录，返回的是插入的多条记录中第一条记录的ID)** |
| USER() | **当前用户** |
| VERSION() | **版本信息** |

**40. 聚合函数（括号内接字段名称）**

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

**41. 加密函数**

|  |  |
| --- | --- |
| 名称 | 描述 |
| MD5() | **信息摘要算法（推荐）** |
| PASSWORD() | **密码算法（主要用于修改当前用户和其他用户的密码）** |

**42. 自定义函数**

**· 创建自定义函数**

**CREATE FUNCTION function\_name RETURNS {STRING|INTEGER|REAL|DECIMAL} routine\_body**

**· 关于函数体**

**（1）函数体由合法的SQL语句构成**

**（2）函数体可以是简单的SELECT或INSERT语句**

**（3）函数体如果为复合结构则使用BEGIN...END语句**

**（4）复合结构可以包含声明、循环、控制结构**

**· 删除函数**

**DELETE FUNCTION [IF EXISTS] function\_name**

**43. 存储过程**

**存储过程是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**

**创建存储过程：**

**CREATE [DEFINER = { user | CURRENT\_USER}] PROCEDURE sp\_name ([proc\_parameter[,...]]) [characteristic ...] routine\_body**

**Proc\_parameter: [IN | OUT | INOUT] param\_name type**

**IN: 表示该参数的值必须在调用存储过程时指定**

**OUT: 表示该参数的值可以被存储过程改变，并且可以返回**

**INOUT: 表示该参数在调用时指定，并且可以被改变和返回**

**routine\_body(过程体):**

**· 过程体由合法的SQL语句构成**

**· 过程体可以是大部分的SQL语句**

**· 过程体如果为复合结构则使用BEGIN...END语句**

**44. 调用存储过程**

**· CALL sp\_name([parameter[, ...]])**

**· CALL sp\_name[()]**

**45. 删除存储过程**

**DROP PROCEDURE [IF EXISTS] sp\_name**

**46. 存储过程的优点**

**· 增强SQL过程的功能和灵活性**

**· 实现较快的执行速度**

**· 减少网络流量**

**47.** **存储过程与自定义函数的区别:**

**· 存储过程实现的过程要复杂一些,而函数的针对性较强;**

**· 存储过程可以有多个返回值,而自定义函数只有一个返回值;**

**· 存储过程一般独立的来执行,而函数往往是作为其他SQL语句的一部分来使用;**

**48. mysql变量的术语分类：**

**1.用户变量：以"@"开始，形式为"@变量名"**

**用户变量跟mysql客户端是绑定的，设置的变量，只对当前用户使用的客户端生效**

**2.全局变量：定义时，以如下两种形式出现，set GLOBAL 变量名 或者 set @@global.变量名**

**对所有客户端生效。只有具有super权限才可以设置全局变量**

**3.会话变量：只对连接的客户端有效。**

**4.局部变量：作用范围在begin到end语句块之间。在该语句块里设置的变量**

**declare语句专门用于定义局部变量。set语句是设置不同类型的变量，包括会话变量和全局变量**

**49. 自定义函数体和存储过程的修改只能修改一些特性，并不能修改函数体。**

**50. 创建存储过程或者自定义函数时需要通过DELIMITER语句修改定界符。**

**51. 存储引擎**

**· Mysql可以将数据以不同的技术存储在文件（内存）中，这种技术就称为存储引擎。**

**· 每一种存储引擎使用不同的存储机制、索引技巧、锁定水平，最终提供广泛且不同的功能。**

**52. Mysql支持的存储引擎:**

**- MyISAM 存储限制可达256TB，支持索引、表级锁定、数据压缩（适用于事务的处理不多的情况）**

**- InnoDB 存储限制可达64TB，支持事务和索引，锁颗粒为行锁（适用于事务处理较多，需要有外键支持的情况）**

**- Memory**

**- CSV**

**- Archive**

**53. 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。**

**54. 锁**

**· 共享锁（读锁）：在同一个时间段内，多个用户可以读取同一个资源，读取过程中数据不会发生变化。**

**· 排他锁（写锁）：在任何时候只能有一个用户写入资源，当进行写锁时会阻塞其他的读锁或者写操作。**

**55. 锁颗粒**

**· 表锁，是一种开销最小的锁策略。（最多只会上一把锁）**

**· 行锁，是一种开销最大的锁策略。（有多少行就有可能上多少把锁）**

**56. 事务**

**· 事务用于保证数据库的完整性。**

**· 事务具有原子性，一致性，隔离性，持久性。**

**57. 索引**

**· 是对数据表中一列或多列的值进行排序的一种结构。**