7.MySQL SQL基础√

7.1 SQL介绍

结构化查询语言(Structured Query Language)简称SQL，结构化查询语
言是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关
系数据库系统；
SQL语句就是对数据库进行操作的一种语言。

7.2 SQL规范

为了使各类RDBMS数据库的SQL能够兼容，大部分的数据库的SQL都符合ANSI的SQL标准。
ANSI标准逐年：
1986： SQL-87最初由ANSI于1986年正式确定。
1989年：美国国家标准协会（ANSI）发布了第一套数据库查询语言标准，称
为SQL-89或FIPS 127-1。
1992年： ANSI发布了修订后的标准ANSI / ISO SQL-92或SQL2，它们比
SQLI更严格，增加了一些新功能。这些标准引入了合规水平，表明方言符合
ANSI标准的程度。
象的支持。取代了核心规范的合规水平，以及另外9个封装的附加规格。
象的支持。取代了核心规范的合规水平，以及另外9个封装的附加规格。
2003： ANSI发布SQL：2003，引入标准化序列，XML相关功能和标识列。
第一个RDBMS的创建者EFCodd博士于同年4月18日去世。
2008： ANSI发布SQL：2008，引入INSTEAD OF触发器以及TRUNCATE语
程序能够将XQuery集成到现有的SQL代码中。
https://dev.mysql.com/doc/refman/8.0/en/compatibility.ht <https://dev.mysql.com/doc/refman/8.0/en/compatibility.ht> ml
句。
2011： ANSI发布SQL：2011或ISO / IEC 9075：2011，ISO（1987）
的第七个修订版和SQL数据库查询语言的ANSI（1986）标准。
MySQL对于ANSI SQL的支持如下：
https://dev.mysql.com/doc/refman/8.0/en/compatibility.ht <https://dev.mysql.com/doc/refman/8.0/en/compatibility.ht> ml

7.3 SQL常用种类

客户端 mysql> help;

可以通过help命令获取Mysql客户端功能

?	(\?) Synonym for `help'	获取mysql客户端的帮助
clear	(\c) Clear the current input statement.	清除当前输入语句。
connect	(\r) Reconnect to the server. Optional arguments are db and host.	重新连接到服务器。可选参数是db和host。

delimiter (\d) Set statement delimiter. 重新定义命令结束符（存储过程会用到）
edit (\e) Edit command with $EDITOR. 打开编辑窗口
ego (\G) Send command to mysql server, display result vertically. 向mysql服务器发送命令，垂直显示结果。
exit (\q) Exit mysql. Same as quit. 退出mysql。和退出一样。
go (\g) Send command to mysql server. 向mysql服务器发送命令。（快速结束掉操作）
help (\h) Display this help. 显示此帮助。
nopager (\n) Disable pager, print to stdout. 取消pager的设定
notee (\t) Don't write into outfile.
pager (\P) Set PAGER [to_pager]. Print the query results via PAGER. page less 以linux中less查看文件的方式显示
print (\p) Print current command. 打印当前命令。
prompt (\R) Change your mysql prompt. 更改mysql提示符
quit (\q) Quit mysql. 离开数据库
source (\.) Execute an SQL script file. Takes a file name as an argument. 执行SQL脚本文件。以文件名作为参数。（导入sql脚本）
status (\s) Get status information from the server. 从服务器获取状态信息。
system (\!) Execute a system shell command. mysql中使用linux命令
tee (\T) Set outfile [to_outfile]. Append everything into given outfile. tee指定一个文件后，mysql操作记录在指定文件中
use (\u) Use another database. Takes database name as argument. 使用其他数据库。以数据库名称为参数。

服务端 mysql> help contents;(显示支持的sql语句类型)

Account Management（用户、权限管理）
Administration（系统管理类语句）
Components（组件应用）
Compound Statements（过程函数复合语句）
Contents（帮助目录）
Data Definition（数据定义） DDL
Data Manipulation（数据操作） DML
Data Types（数据类型）
Functions（内置函数）
Geographic Features（地理位置）
Help Metadata（帮助信息元数据）
Language Structure
Plugins（插件管理）
Storage Engines（存储引擎）
Table Maintenance（表维护）
Transactions（事务）
User-Defined Functions（自定义函数）

> Utility（实用工具）

7.4 sql_mode（mysql5.6版本之后sql语句进行严格模式）

sql_mode的作用

在MYSQL存储和应用数据时,能够保证数据时准确有效的.防止录入不规矩的数据

查看sql_mode

```
mysql> select @@sql_mode;
```

sql_mode8.0版本规范

ONLY_FULL_GROUP_BY：对于GROUP BY聚合操作，如果在SELECT中的列、HAVING或者ORDER BY子句的列，没有在GROUP BY中出现，或者不在函数中聚合，那么这个SQL是不合法的。
STRICT_TRANS_TABLES：进行数据的严格校验，错误数据不能插入，报error错误。如果不能将给定的值插入到事务表中，则放弃该语句。对于非事务表，如果值出现在单行语句或多行语句的第1行，则放弃该语句。
NO_ZERO_IN_DATE：在严格模式，不接受月或日部分为0的日期。
NO_ZERO_DATE：在严格模式，不要将 '0000-00-00'做为合法日期。
ERROR_FOR_DIVISION_BY_ZERO：在严格模式，在INSERT或UPDATE过程中，如果被零除(或MOD(X，0))，则产生错误(否则为警告)。
NO_ENGINE_SUBSTITUTION：如果需要的存储引擎被禁用或未编译，那么抛出错误。
更多详细信息，看官方文档 <https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html>

7.5 Mysql逻辑结构属性介绍

7.5.1 库，表，列（sql语句执行的逻辑对象）

mysql的逻辑结构就是我们登陆数据库接触到的库，表，列。让我们以库，表这种逻辑结构的方式来管理mysql底层数据。

> 库：库名，库属性（字符集，校对规则，表空间加密）
> 表：表名，表属性（存储引擎，字符集，校对，表空间加密）
> 列：列名，列属性，数据行

7.5.2 库表中的属性介绍

1.字符集

mysql和磁盘进行存储时的编码对应关系.每一种字符集可对应多种相对规则

> mysql> show charset; 查看数据库中支持的字符集类型
> 常用的是
> 1.UTF8(mysql8.0版本改名为UTF8mb3):字符最大长度3个字节 mysql客户端字符集默认utf8
> 2.UTF8mb4（推荐使用）：字符最大长度4个字节（emoji表情） ---5.7版本之后出的

2.相对规则（排序规则）

在存储数据时排序针对于大小写的敏感程度.

> mysql> show collation;

3.存储引擎

当前的数据库支持的存储引擎插件，默认是innodb引擎

mysql> show engines;
| FEDERATED 类型于oracle中dblink
| MEMORY
| InnoDB 默认的存储引擎
| PERFORMANCE_SCHEMA
| MyISAM 已经被innodb代替，存在是旧版本升级后的历史遗留
| MRG_MYISAM
| BLACKHOLE 黑洞 降低主从架构中主库的压力为中间加入黑洞作为中间库
| CSV
| ARCHIVE

4.加密表空间（不常用）

通过插件的方式对表空间进行加密，防止他人读取数据文件信息。

> ENCRYPTION='N'/'Y'
> 临时生效：
> INSTALL PLUGIN keyring_file soname 'keyring_file.so';
> mkdir -p /data/3306/mysql-keyring/
> chown -R mysql.mysql /data/3306/mysql-keyring/
> chmod 750 /data/3306/mysql-keyring/

> 永久生效

> 在my.cnf的[mysqld]段，加这两行

> early-plugin-load=keyring_file.so

> keyring_file_data=/data/3306/mysql-keyring/keyring

7.5.3 列中的属性信息

列约束

> primary key ： 主建约束，同时保证唯一性和非空.每个表只能有一个PK， 我们建议业务无关列（数字列）。

> foreign key：外键约束，用于限制两个表的关系，保证从表该字段的值来自于主表相关联的字段的值。

> not null ： 非空约束，保证字段的值不能为空

> default：默认约束，保证字段总会有值，即使没有插入值，都会有默认值

> unique：唯一，保证唯一性但是可以为空，比如手机号。

> auto_increment：自增长列

> unsigned： 无符号

> comment： 注释

数字类型

1.整数型

类型	占用字节	无符号范围	有符号范围	数据长度
tinyint	1	0-255	-128-127	3
smallint	2	0-65535	-32768~32767	5
mediumint	3	0~16777215	-8388608~8388607	8
int	4	0~2^32	-2^31~ 2^32-1	10
bigint	8	0~2^64	-2^63~ 2^63-1	20

最常用的是tingint int bigint

选择数据类型的关键3点：

- **合适的**
- **简短的**
- **完整的**

2.浮点型与定点数

对于精度不高的存储，我们会把小数转化为整数，在使用这个值时在转化为小数

> FLoat

> Float：表示不指定小数位的浮点数

> Float(M,D)：表示一共存储M个有效数字，其中小数部分占D位

> Float(10,2)：整数部分为8位，小数部分为2位

> Double

> Double又称之为双精度：系统用8个字节来存储数据，表示的范围更大，

> 10^308次方，但是精度也只有15位左右。

> Decimal： 最常用

> Decimal系统自动根据存储的数据来分配存储空间，每大概9个数就会分配四

> 个字节来进行存储，同时小数和整数部分是分开的。

> 定点数：能够保证数据精确的小数（小数部分可能不精确，超出长度会四舍五

> 入），整数部分一定精确

> Decimal(M,D)：M表示总长度，最大值不能超过65，D代表小数部分长度，

> 最长不能超过30。

字符串类型

定长字符char(64)

> 定长字符：指定长度之后，系统一定会分配指定的空间用于存储数据 ,如果空间没有用完，用NULL填充

> 基本语法：char(L)，L代表字符数（中文与英文字母一样），L长度为0到255

变成字符Varchar(64)

> 变长字符：指定长度之后，系统会根据实际存储的数据来计算长度，分配合适的长度（数据没有超出长度）

> 基本语法：Varchar(L)，L代表字符数，L的长度理论值位0到65535

> 因为varchar要记录数据长度（系统根据数据长度自动分配空间），所以每个varchar数据产生后，系统都会在数据后面增加1-2个字节的额外开销：是用来保存数据所占用的空间长度

> 如果数据本身小于255个字符：额外开销一个字节；如果大于255个，就开销两个字节

char和Varchar的区别

> 1.char一定会使用指定的空间，varchar是根据数据来定空间
> 2.char的插入数据效率理论上比varchar高.因为char是一次分配好空间大小，而Varchar会进行计算。

char和Varchar的使用场景

> 1.如果确定数据一定占用指定的长度，且不容易发生变化（身份证号，手机号，学生号）
> 2.如果不确定数据的长度我们就使用Varchar
> 3.如果数据长度超过255个字符，不用char和Varchar，会在底层使用text的方式进行存储

char和Varchar字符大小限制

字符串类型	字符串最大长度	字符集类型	额外占用	最多存储长度
char(255)	0~255	UTF8（占用3）	0	255×3=765
char(255)	0~255	UTF8mb4(占用4)	0	255×4=1020
vachar（65535）	0~65535	UTF8（占用3）	长度小于255占用1，大于占用2	65535×3-2
vachar（65535）	0~65535	UTF8mb4（占用4）	长度小于255占用1，大于占用2	65535×4-2

限制的原因是因为uft8mb4是5.7版本才出的，所以在之前版本中以UTF8字符集为准，char和varchar字符串类型长度以255为限制.
最大存储长度char(255) 255×3=765　　varchar(255) 255×3-1=764

5.6生成中限制常用

varchar(250) utf8 751字节 不超过 765字节

varchar(190) utf8mb4 761字节 不超过 765字节

Mysql不擅长存储大字段，通常处理大字段使用别的数据库，或者链接到副本，但还是提供了两种文本类型

Text文本类型

系统中提供了四种Text类型

> Tinytext：系统使用一个字节来保存，实际能够存储的数据为：2 ^ 8 + 1

> Text：使用两个字节保存，实际存储为：2 ^ 16 + 2

> Mediumtext：使用三个字节保存，实际存储为：2 ^ 24 + 3

> Longtext：使用四个字节保存，实际存储为：2 ^ 32 + 4

Blog文本类型

> 存储二进制文本（图片，文件），一般都不会使用blob来存储文件本身，通常是使用一个链接来指向对应的文件本身

注意：

1、 在选择对应的存储文本的时候，不用刻意去选择text类型，系统会自动根据存储的数据长度来选择合适的文本类型。

2、 在选择字符存储的时候，如果数据超过255个字符，那么一定选择text存储

ENUM枚举类型（省份）

介绍

> 枚举类型：在数据插入之前，先设定几个项，这几个项就是可能最终出现的数据结果。

语法

> 基本语法：enum(数据值1,数据值2...)

> 系统提供了1到2个字节来存储枚举数据：通过计算enum列举的具体值来选择实际的存储空间：如果数据值列表在255个以内，那么一个字节就够，如果超过255但是小于65535，那么系统采用两个字节保存。

使用场景

> 如果确定某个字段的数据只有那么几个值：如性别，男、女、保密，系统就可以在设定字段的时候规定当前字段只能存放固定的几个值.

Set集合类型

介绍

> 集合：是一种将多个数据选项可以同时保存的数据类型，本质是将指定的项按照对应的二进制位来进行控制：1表示该选项被选中，0表示该选项没有被选中。
> Set集合的意义：

> 1、 规范数据

> 2、 节省存储空间

语法

> 基本语法：set('值1','值2','值3'...)

> 系统为set提供了多个字节进行保存，但是系统会自动计算来选择具体的存储单元

> 1个字节：set只能有8个选项

> 2个字节：set只能有16个选项

> 3个字节：set只能表示24个选项

> 8个字节：set可以表示64个选项

ENUM与SET的相同与不同

> **相同:**
> Set和enum一样，最终存储到数据字段中的依然是数字而不是真实的字符串数据在存储的时候，如果被选中，那么对应的位的值就为1，否则为0

> 系统在进行存储的时候会自动将得到的最终的二进制颠倒过来，然后再进行转 换成十进制存储
> 不同
> Enum：单选框(只能插入一个)，写入多个选项范围，单选一个选项

> Set：复选框(可以插入多个)，写入多个选项范围，可多选选项

时间类型

类型	字节大小	取值范围	格式
DATE	3	1000-01-01/9999-12-31	YYYY-MM -DD
TIME	3	'-838:59:59'/'838:59:59'	HH:MM:SS
YEAR	1	1901/2155	YYYY
DATETIME	8	1000-01-0100:00:00/9999-12-31-23:59:59	YYYY-MM -DD HH:MM:SS
TIMESTAMP	4	1970-01-0100:00:00/2038-1-1911:14:07	YYYYMMDD HHMMSS

二进制类型

e856810e153c.png&title=7.MySQL%20SQL%E5%9F%BA%E7%A1%80%E2%88%9A%20%7C%207.1%20SQL%E4%BB%8B%E7%BB%8D%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E