第1章 SQL介绍

1.什么是SQL

```
1  属于关系型数据库产品中专用的语句。结构化查询语句。
```

2.SQL标准

```
1  SQL-89  \ SQL-92  \  SQL-99  \ SQL03 .....
```

3.SQL模式

作用:

```
1  影响到了SQL语句的执行行为。为了让数据库在存、取能够满足生活的常识、科学的逻辑，让这些
   数据有意义。
```

例子:

```
1  现实生活中常识:
2      日期:  1000-9999 年 1-12 月 1-31日
3  科学逻辑:
4      除法:除数不能为0
```

查看sql_mode:

```
mysql> select @@sql_mode;
ONLY_FULL_GROUP_BY          :  针对group by语句执行时的规范
STRICT_TRANS_TABLES         :  针对事务表启动严格模式
NO_ZERO_IN_DATE             :  2010-00-10
NO_ZERO_DATE                :  0000-00-00
ERROR_FOR_DIVISION_BY_ZERO  :  除数不能为0
NO_AUTO_CREATE_USER         :  是否自动创建用户
NO_ENGINE_SUBSTITUTION      :  建表是使用了一个不支持的存储引擎报错。
```

4.SQL的分类

```
DDL  :  数据定义语言
对于:  库 、表（元数据）的 增、删、改
建库、删库、修改库、建表、删表、修改表

DCL  :  数据控制语言
grant
revoke

DML  :  数据操作语言
表的数据行进行的增、删、改、查

DQL  :  数据查询语言
对于表数据行查看
对于元数据查看
```


第2章 数据库对象属性介绍

1.数据库对象

库的对象:

```
1  库名
2  库属性：字符集、校对规则
```

表的对象:

```
1  表名
2
3  列 ：
4  列名
5  列属性：数据类型、约束、其他属性
6
7  行
8  表属性：引擎、字符集、校对规则、其他
```

2.字符集

种类说明:

```
1  1.utf8     最大支持3字节的字符。不支持emoji字符
2  2.utf8mb4  最大支持4字节的字符。支持emoji字符
```

注意:

```
1  8.0之前，默认字符集latin1,8.0之后是utf8mb4。
2  我们建议使用utf8mb4.
```

设置方法:

```
默认字符集参数
建库
建表
```

3.列属性

3.1 数字类型

```
整数:
  tinyint   1字节  =  8位 = 00000000 - 11111111  = 2^8个  = 0  - 255 , -127-128
  int       4字节  =  32位                       = 2^32个 = 0  - 2^32-1 ,-2^31-2^31-1 ,10位数
  bigint    8字节                                          0  - 2^64-1 ,-2^63-2^63-1 ,20位数

浮点数:
  decimal(m,n)
```

3.2 字符串类型

```
char(N)    :
  N字符个数，最大不超过255
  定长的字符串类型。
  例如:  char(10) ,最多存10个字符,只要10个字符以内，都10个字符长度的存储空间。剩余用空格填充。

varchar(M) :
  M字符个数，最大不超过65535
  变长的字符串类型。会额外占用1-2字节存储字符长度。255字符之内，额外1字节，255以上，额外2字节
  例如:  varchar(10),最多存10个字符,按需分配存储空间。
  abc 3             =4
  asdasdadasd 1000  =1002
```

```
12
13  enum('bj','sh','tj','heibei','henan',.....)
14  district enum('bj','sh','tj','heibei','henan',.....)
```

3.3 时间类型

```
1  DATETIME
2  8字节
3  范围为从 1000-01-01 00:00:00.000000 至 9999-12-31 23:59:59.999999。
4
5  TIMESTAMP
6  4字节
7  1970-01-01 00:00:00.000000 至 2038-01-19 03:14:07.999999。
8  timestamp会受到时区的影响
```

4.列约束

```
1  主键 primary key  (PK)
2  1.一张表只能有一个，可以有多个列构成
3  2.特点:  非空+唯一
4  3.建议每张表都设置主键，有利于索引的应用，通常是使用自增的数字列更佳。
5
6  非空 not null
7  1.特点:  必须录入值。
8  2.建议:  每个列最好设置为，有利于索引的应用。
9
10 唯一 unique key
11 1.特点:  不能有重复值。
12
13 无符号 unsigned
14 1.特点:  针对数字列，无符号设定。
```

5.其他属性

```
表属性 :
    engine  :  存储引擎设置，默认是innodb，也是我们建议的。
    charset :  utf8mb4
    comment :  注释。

列属性 :
    default        :  默认值。一般是在not null 配合使用
    auto_increment:  数字列自增长。一般是在主键列配合使用
    comment        :  列的注释，建议每个列都有
```

第2章 DDL数据库定义语言

1.库定义

1.1 库定义规范

```
1. 库名不能数字开头
2. 库名要和业务有关
3. 库名不要有大写字符
   原因：为了多平台兼容。
4. 建库需要显示指定字符集。建议是utf8mb4。
5. 生产中禁用普通用户的drop database权限。
```

1.2 创建库

```
CREATE DATABASE oldguo CHARSET utf8mb4;
CREATE DATABASE school CHARSET utf8mb4;
```

1.3 修改库

```
CREATE DATABASE oldli ;
SHOW CREATE DATABASE oldboy;
SHOW CREATE DATABASE oldli;
ALTER DATABASE oldli CHARSET utf8mb4;
```

修改库客户端默认字符集

```
SHOW VARIABLES LIKE '%char%';
SET NAMES utf8mb4;
```

修改配置文件添加默认字符集参数

```
vim /etc/my.cnf
[mysqld]
character_set_server=utf8mb4;
```

1.4 查看库

```
show databases;
SHOW CREATE DATABASE oldboy;
```

1.5 删除库

```
DROP DATABASE oldli;
```

2.表定义

2.1 表定义规范

```
1. 建表
   a. 表名:
```

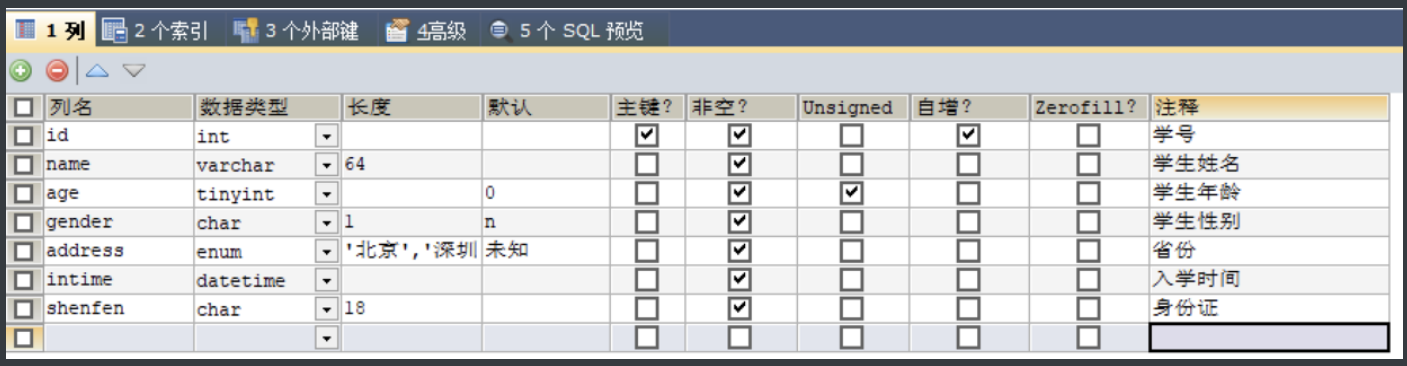
```
    不能数字开头
    业务有关
    不要大写字母
    不要超过18字符
    不能是关键字
  b. 存储引擎使用InnoDB
  c. 5.7版本以后，字符集使用utf8mb4
  d. 列名要和业务有关，不要超过18个字符
  e. 选择合适、足够、简短数据类型
  f. 建议每个列设置not null
  g. 每个列要有注释
  h. 每个表要有主键
  i. 针对not null 列，可以设定默认值。
  j. 表注释

2. 修改表
  a. 添加列，使用追加式添加列
  b. 修改列属性，尽量使用modify语句
  c. 修改表定义，建议在业务不繁忙期间进行。尽量采用pt-osc或者gh-ost工具减少业务影响。
```

推荐软件:

```
yearing   开源SQL审核工具。
inception SQL审核工具。
```

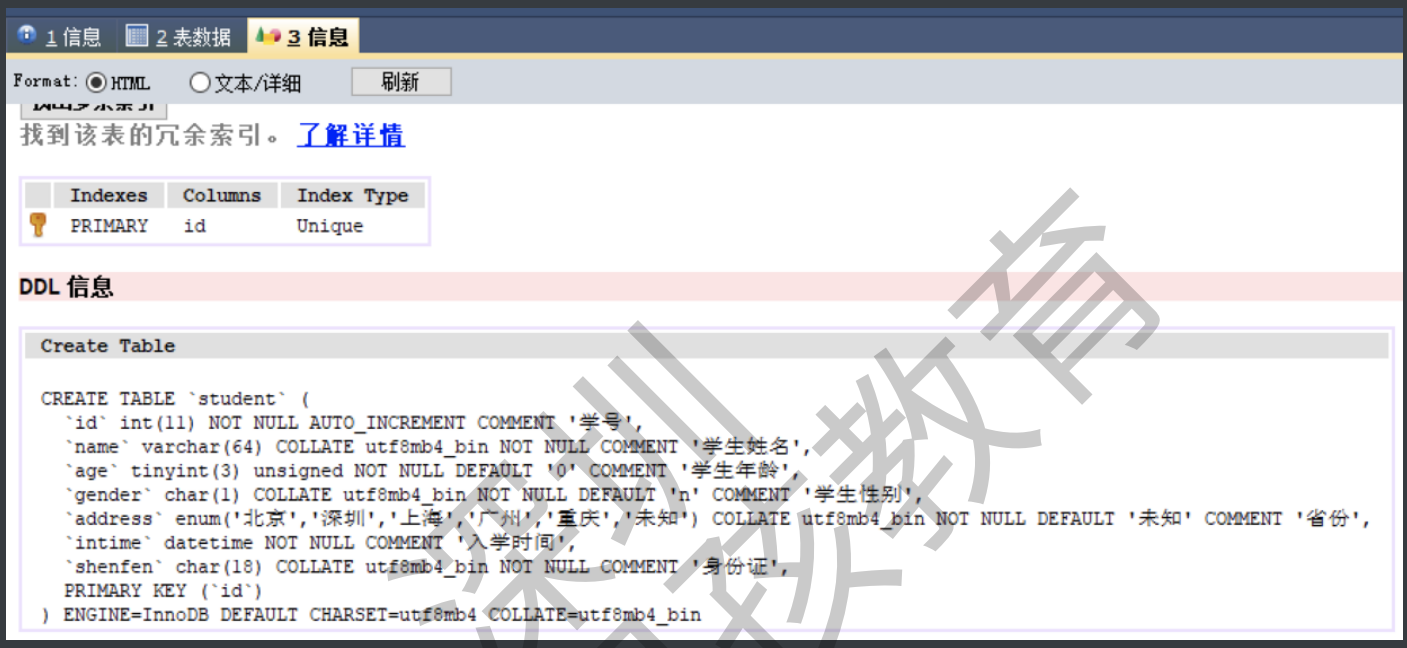
2.2 创建表

使用工具创建表:



列名	数据类型	长度	默认	主键?	非空?	Unsigned	自增?	Zerofill?	注释
id	int			☑	☑	☐	☑	☐	学号
name	varchar	64		☐	☑	☐	☐	☐	学生姓名
age	tinyint		0	☐	☑	☑	☐	☐	学生年龄
gender	char	1	n	☐	☑	☐	☐	☐	学生性别
address	enum	'北京','深圳	未知	☐	☑	☐	☐	☐	省份
intime	datetime			☐	☑	☐	☐	☐	入学时间
shenfen	char	18		☐	☑	☐	☐	☐	身份证
				☐	☐	☐	☐	☐	

使用工具查看建表语句:



建表语句如下:

```
CREATE TABLE `student` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '学号',
  `name` varchar(64) COLLATE utf8mb4_bin NOT NULL COMMENT '学生姓名',
  `age` tinyint(3) unsigned NOT NULL DEFAULT '0' COMMENT '学生年龄',
  `gender` char(1) COLLATE utf8mb4_bin NOT NULL DEFAULT 'n' COMMENT '学生性别',
  `address` enum('北京','深圳','上海','广州','重庆','未知') COLLATE utf8mb4_bin NOT NULL DEFAULT '未知' COMMENT '省份',
  `intime` datetime NOT NULL COMMENT '入学时间',
  `shenfen` char(18) COLLATE utf8mb4_bin NOT NULL COMMENT '身份证',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
```

2.3 查看表

```
use oldboy;
show tables;
show create table student;
desc student;
```

2.4 复制一张表

```
create table stu like student;
```

2.5 修改表定义

增加列：在student表中添加telnum列

```
use oldboy;
DESC `student`;
ALTER TABLE `student`
ADD COLUMN telnum CHAR(11) NOT NULL UNIQUE KEY DEFAULT '0' COMMENT '手机号' ;
```

不建议的方式：

在gender列后增加列

```
1  alter table oldboy.student
2  add column a CHAR(11) not null unique key default '0' comment '手机号'
   after gender ;
3  desc student;
```

在第一列添加列

```
1  alter table oldboy.student
2  add column b CHAR(11) not null unique key default '0' comment '手机号'
   first ;
3  desc student;
```

删除列：（不代表生产操作，危险！！！！）

```
1  alter table student drop  a;
2  alter table student drop  b;
3  alter table student drop  telnum;
```

2.6 修改表属性

修改表名：

```
1  ALTER TABLE student RENAME TO st;
```

修改引擎：

```
CREATE  TABLE t1(id INT) ENGINE=MYISAM;
SHOW CREATE TABLE t1;
ALTER TABLE t1 ENGINE=INNODB;
SHOW CREATE TABLE t1;
```

修改字符集：

```
CREATE  TABLE t2(id INT) CHARSET=utf8;
SHOW CREATE TABLE t2;
ALTER TABLE t2 CHARSET=utf8mb4;
SHOW CREATE TABLE t2;
```

2.7 修改列属性

修改列名：

```
ALTER TABLE st CHANGE shenfen cardnum CHAR(18) NOT NULL DEFAULT '0'
COMMENT '身份证';
```

修改默认值：

```
ALTER TABLE st CHANGE cardnum cardnum CHAR(18) NOT NULL DEFAULT '1'
COMMENT '身份证';
```

修改数据类型：

```
ALTER TABLE st MODIFY cardnum CHAR(20) NOT NULL DEFAULT '1' COMMENT '身份
证';
```

2.8 删除表

```
drop table stu;
```

第4章 DML数据操作语言

1.INSERT 插入表数据

标准:

```
1  INSERT INTO
2  st(id,NAME,age,gender,address,intime,cardnum,num)
3  VALUES(1,'张三',18,'m','北京','2020-09-06','666666',10);
4  SELECT * FROM st;
```

部分列录入:

```
1  INSERT INTO
2  st(NAME,intime,num)
3  VALUES('李四',NOW(),1111);
4  SELECT * FROM st;
```

修改时间列的默认值为NOW()

```
1  ALTER TABLE st MODIFY intime DATETIME NOT NULL DEFAULT NOW() COMMENT '入
   学时间';
2  DESC st;
```

再次插入数据:

```
1  INSERT INTO
2  st(NAME,num)
3  VALUES('王五',11112);
4  SELECT * FROM st;
```

省略写法:

```
desc st;
insert into
st
values(5,'张三',18,'m','北京','2020-04-27','666666',10);
select * from st;
```

2.UPDATE 修改表数据

更新数据行:

```
UPDATE st SET NAME='张六' WHERE id=3;
SELECT * FROM st;

update st set name='张qi' , age=21  where id=4;
select * from st;
```

3.DELETE/UPDATE/TRUNCATE 删除表数据

3.1 DELETE

```
DELETE FROM st WHERE id=3;
SELECT * FROM st;
```

3.2 伪删除

update 替代 dalete, 添加状态列，1代表存在，0代表删除

第一步：增加状态列

```
ALTER TABLE st ADD COLUMN state TINYINT NOT NULL DEFAULT 1 COMMENT '状态列,0是删除,1是存在';
DESC st;
```

第二步：使用update 替换 delete

```
原:
delete from st where id=4

修改后:
UPDATE st SET state=0 WHERE id=4;
```

第三步：替换原来查询业务语句

```
原:
select * from st;

改变后:
SELECT * FROM st WHERE state=1;
```

4.面试题

题目:

```
drop  table t1,truncate table t1,delete from t1 区别?
```

解答:

```
drop  table t1;
作用:
1.删除所有表数据，删除整个表段（rm ibd  ）,属于物理性质，会释放磁盘空间。
2.删除表定义（rm  frm ，元数据也会被删除）

truncate table t1;
作用:
1.保留表结构，清空表段中的数据页。属于物理删除，会释放磁盘空间。

delete from t1;
作用:
1.删除数据行。逐行删除。保留表结构，属于逻辑性质删除。只是标记删除，不会立即释放磁盘空间。
2.所以delete操作会产生碎片。
```

第5章 DQL数据查询语言

1.如何学习业务

```
1.查看表定义
desc city;

2.了解字段的意义
id            : 主键 1-N数字
name          : 城市名
countrycode   : 国家编码(三个字母? CHN,USA)

3.查看部分数据
select * from city where id<10;
district      : 区域 （省、州、县）
population    : 城市人口
```

```
4.查看解释说明
查看comment的解释说明
```

2.SELECT 查询数据

2.1 查询数据库服务器配置参数

```
select @@port;
select @@server_id;
select @@basedir;
select @@datadir;
select @@socket;
select @@innodb_flush_log_at_trx_commit;
```

替代方法:

```
show variables;
show variables like '%trx%';
```

2.2 查询内置函数

```
help Functions;
select DATABASE();
select NOW();
select USER();
select CONCAT("hello world");
select user,host from mysql.user;
SELECT CONCAT("数据库用户:",USER,"@",HOST,";") FROM mysql.user;
```

2.3 多子句执行顺序

```
select     列
from       表
where      条件
group by   列
having     条件
order by   列
limit      条件
```

2.4 查询表中所有数据(小表)

```
use world;
select id,name,countrycode,district,population
from city;
```

或者

```
select id,name,countrycode,district,population
from world.city;
```

或者

```
select * from city;
```

2.5 查询部分列数据

查询所有城市名及人口信息

```
select name,population from city;
```

查询city表中，所有中国的城市信息

```
select *  from city where countrycode = 'CHN';
```

查询人口数小于100人城市信息

```
SELECT * FROM city WHERE Population<100;
```

查询中国,人口数超过500w的所有城市信息

```
SELECT * FROM city WHERE countryCode='CHN' AND Population<5000000;
```

查询中国或美国的城市信息

```
SELECT * FROM city WHERE countryCode='CHN' OR countryCode='USA';

SELECT * FROM city WHERE countryCode IN ('CHN','USA');
```

查询人口数为100w-200w（包括两头）城市信息

```
SELECT * FROM city WHERE Population >= 1000000 AND Population <=
2000000;

SELECT * FROM city WHERE Population BETWEEN 1000000 AND 2000000;
```

查询中国或美国，人口数大于500w的城市

```
SELECT * FROM city WHERE (countryCode='CHN' OR countryCode='USA') AND
Population > 5000000;

SELECT * FROM city WHERE countryCode IN ('CHN','USA') AND Population >
5000000;
```

查询城市名为qing开头的城市信息

```
SELECT * FROM city WHERE NAME LIKE 'qing%';
```

3.GROUP BY 聚合函数

3.1 聚合函数

```
count()          统计数量
sum()            求和
avg()            平均数
max()            最大值
min()            最小值
group_concat()   列转行
```

3.2 group by 分组功能原理

```
1. 按照分组条件进行排序
2. 进行分组列的去重复
3. 聚合函数将其他列的结果进行聚合。
```

示意图：

3.3 group by练习

统计city表的行数

```
SELECT COUNT(*) FROM city;
```

统计中国城市的个数

```
SELECT COUNT(*) FROM city WHERE countryCode='CHN';
```

统计中国的总人口数

```
SELECT SUM(Population) FROM city WHERE countryCode='CHN';
```

统计每个国家的城市个数

```
SELECT countryCode,COUNT(NAME) FROM city GROUP BY countryCode;
```

统计每个国家的总人口数

```
SELECT countryCode,SUM(Population) FROM city GROUP BY countryCode;
```

统计中国每个省的城市个数及城市名列表

```
SELECT district, COUNT(NAME),GROUP_CONCAT(NAME)
FROM city
WHERE countrycode='CHN'  GROUP BY district;
```

4.HAVING 聚合判断

4.1 作用

```
主要应用在group by之后需要的判断。
```

4.2 练习

统计每个国家的总人口数，只显示总人口超过1亿人的信息

```
SELECT countrycode,SUM(population)
FROM city
GROUP BY countrycode
HAVING SUM(population)>100000000;
```

5.ORDER BY 聚合排序

查询所有城市信息，并按照人口数排序输出

```
SELECT * FROM city ORDER BY population;
```

查询中国所有的城市信息，并按照人口数从大到小排序输出

```
SELECT * FROM city WHERE countryCode='CHN' ORDER BY population DESC;
```

每个国家的总人口数，总人口超过5000w的信息,并按总人口数从大到小排序输出

```
SELECT countrycode,SUM(population)
FROM city
GROUP BY countrycode
HAVING SUM(population) > 50000000
ORDER BY SUM(population) DESC;
```

6.LIMIT 分页查询

查询中国所有的城市信息，并按照人口数从大到小排序输出，只显示前十名

```
select *
from city
where countrycode = 'CHN'
order by population desc
limit 10 ;
```

查询中国所有的城市信息，并按照人口数从大到小排序输出，只显示6-10名

```
select *
from city
where countrycode = 'CHN'
order by population desc
limit 5,5

select *
from city
where countrycode = 'CHN'
order by population desc
limit 5 offset 5;
```

解释:

```
-- limit M,N : 跳过M行, 显示N行
-- limit N offset M : 跳过M行, 显示N行
```

7.子句执行顺序

```
select select_list from where group_by having order by  limit
       4           1    2      3      5        6       7

select   select_list   4
from                   1
where                  2
group_by               3
having                 5
order by               6
limit                  7
```