Database

学习网站：每个语句有个 demo，形象点！

http://www.w3school.com.cn/sql/index.asp

数据库（database）种类：MySQL、SQL Server、Access、Oracle、Sybase、DB2 等等

SQL 是什么？

1. SQL，指结构化查询语言，全称是 Structured Query Language。
2. SQL 让您可以访问和处理数据库。
3. SQL 是一种 ANSI（美国国家标准化组织）标准的计算机语言。

SQL 能做什么？

1. SQL 面向数据库执行查询
2. SQL 可从数据库取回数据 ->查找
3. SQL 可在数据库中插入新的记录 ->增加
4. SQL 可更新数据库中的数据 ->修改
5. SQL 可从数据库删除记录 ->删除
6. SQL 可创建新数据库
7. SQL 可在数据库中创建新表
8. SQL 可在数据库中创建存储过程
9. SQL 可在数据库中创建视图
10. SQL 可以设置表、存储过程和视图的权限

首先明白几个概念

1. 数据库（database）：保存有组织的数据的容器（通常是一个文件或一组文件）
2. 表（table）某种特定类型数据的结构化清单。（存储在表中的数据是同一种类型的数据或清单）
3. 列（column）表中的一个字段。所有表都是由一个或多个列组成的。
4. 行（row）表中的一个记录
5. 主键（primary key）一列（或一组列），其值能够唯一标识表中每一行。（可以创建表时直接设置，也可以创建好再设置）

注：

1. 相同数据库中不能两次使用相同的表名，但在不同的数据库中完全可以使用相同的表名
2. 行中应该总是定义主键；若不设置主键，就可以重复添加相同的数据；

简单创建个数据库，顺下过程(SQL 不区分大小写)

```
1. CREATE  DATABASE  myDatabase; //创建一个名为 " myDatabase " 的数据库
2. use myDatabase;                //选择 SQL 模式中 myDatabase 数据库
3. create table Persons(          //创建一个表 Persons（标注其字段名及类型）
4.         PersonId   int,
5.         LastName   varchar(255),
```

```
6.          FirstName    varchar(255),
7.          age    int
8.  );
9.  insert into Persons (PersonId,LastName,FirstName,age) values (1, 'jiaqi', 'li', 24);
10. insert into Persons (PersonId,LastName,FirstName,age) values (2, 'hongxia', 'zhang',
    26);                                    //插入两条数据 values 是对应值（增加）
11. select * from Persons;                          //从 Persons 表中取数据*代表所有（查找）
12. update Persons    set age='40',PersonId='1' where FirstName='li';
13.                 //修改 FirstName='li'的行，age 改为 40，PersonId 改为 1（修改）
14. select * from Persons;
15. delete from Persons    where age='40';              //删除 age='40'的行(删除)
16. select * from Persons;
17. O(∩_∩)
```

11 行运行后的表：

Persons表

PersonId	LastName	FirstName	age
1	jiaqi	li	24
2	hongxia	zhang	26
2	hongxia	zhang	26
2	hongxia	zhang	26

rows in set (0.00 sec)

行

列（字段）

14 行运行后的表：

PersonId	LastName	FirstName	age
1	jiaqi	li	40
2	hongxia	zhang	26
2	hongxia	zhang	26
2	hongxia	zhang	26

4 rows in set (0.00 sec)

16 行运行后的表：

PersonId	LastName	FirstName	age
2	hongxia	zhang	26
2	hongxia	zhang	26
2	hongxia	zhang	26

3 rows in set (0.00 sec)

主键 PRIMARY KEY

1.创建表时就已经设置主键

```
CREATE TABLE Persons
(
   PersonId int NOT NULL,
```

```
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    age int,
    PRIMARY KEY (PersonId)        //设置主键为 PersonId
)
```

2.创建表时未设置主键

```
CREATE TABLE Persons
(
    PersonId int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    age int
)
ALTER TABLE Persons   ADD PRIMARY KEY (PersonId)   //设置主键（修改队列）
```

1.增加列 ALTER（字段）　　//之前操作（增删改查）其实是对行操作

ALTER TABLE 语句用于在已有的表中添加、删除或修改列。

（1）初始表 Persons：

```
+----------+----------+-----------+------+
| PersonId | LastName | FirstName | age  |
+----------+----------+-----------+------+
|        1 | jiaqi    | 1i        |   24 |
+----------+----------+-----------+------+
1 row in set (0.00 sec)
```

（2）ALTER TABLE Persons ADD address varchar(255);　　//执行后增加字段 address

```
+----------+----------+-----------+------+---------+
| PersonId | LastName | FirstName | age  | address |
+----------+----------+-----------+------+---------+
|        1 | jiaqi    | 1i        |   24 | NULL    |
+----------+----------+-----------+------+---------+
1 row in set (0.00 sec)
```

（3）ALTER TABLE Persons DROP COLUMN age;　　//执行后删除字段 address

```
+----------+----------+-----------+---------+
| PersonId | LastName | FirstName | address |
+----------+----------+-----------+---------+
|        1 | jiaqi    | 1i        | NULL    |
+----------+----------+-----------+---------+
1 row in set (0.00 sec)
```

（4）ALTER TABLE Persons　MODIFY COLUMN　address　varchar(50)；

//执行后修改字段 address 类型为 varchar(50)

2.删除 DROP 语句

DROP 可以轻松地删除索引、表和数据库。

```
DROP TABLE Persons;                  //删除表
TRUNCATE TABLE Persons;              //仅仅删除表内数据，不删除表本身
DROP DATABASE myDatabase;            //删除数据库
```

Important select 语句检索

1. Select distinct school from persons; // distinct 关键字：返回 school 不同的值
2. Select * from persons;　　//表中所有数据
3. Select school from persons limit 2;　//检索返回数据不多于 5 行
4. Select school from persons limit 2,2; //检索返回数据从行 2 开始，不多于 2 行
5. select persons.school from persons;//使用限定的字段，表名
6. select school,city from persons order by city;//按照城市名排序—默认 asc
7. select school,city from persons order by city desc;//按照城市名排序—降序
8. select school,city from persons where school='shandongdaxue';//过滤数据
9. order by 与 where 同时使用，要把 order by 放在 where 之后！
10. 过滤数据的组合使用，多条件过滤！
11. Select city from persons where city like 'nan%'; //通配符，找到 city 字段以 nan 开头的数据
12. 多个表检索。。。。。

创建表字段的几种约束：

NOT NULL	无值
UNIQUE	唯一标示
PRIMARY KEY	主键
FOREIGN KEY	外键
CHECK	用于限制列中的值的范围
DEFAULT	默认值

每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束

1. SQL join 用于根据两个或多个表中的列之间的关系，从这些表中查询数据
2. SQL INNER JOIN 关键字，在表中存在至少一个匹配时，INNER JOIN 关键字返回行。与 JOIN 是相同的。
3. LEFT JOIN 关键字会从左表 (table_name1) 那里返回所有的行，即使在右表 (table_name2) 中没有匹配的行。
4. RIGHT JOIN 关键字会右表 (table_name2) 那里返回所有的行，即使在左表 (table_name1) 中没有匹配的行。
5. FULL JOIN 关键字只要其中某个表存在匹配就会返回行。
6. UNION 操作符用于合并两个或多个 SELECT 语句的结果集。UNION 内部的 SELECT 语句必须拥有相同数量的列。列也必须拥有相似的数据类型。同时，每条 SELECT 语句中的列的顺序必须相同。UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。
7. SELECT INTO 语句从一个表中选取数据，然后把数据插入另一个表中。SELECT INTO 语句常用于创建表的备份复件或者用于对记录进行存档。

1. MySQL 查询某数据最值问题。

- SELECT MAX(字段名) FROM table_name LIMIT 0,1 最大
- SELECT MIN(字段名) FROM table_name LIMIT 0,1 最小
- SELECT * FROM table_name ORDER BY 字段名 DESC LIMIT 0,1 最大
- SELECT * FROM table_name ORDER BY 字段名 ASC LIMIT 0,1 最小

第二大值：

- SELECT MAX(字段名) FROM table_name WHERE (字段名) NOT IN (SELECT MAX(字段名) FROM table_name)
- SELECT * FROM table_name ORDER BY 字段名 DESC LIMIT 1,1 最大

后面以此类推！

2. 事务是什么？事务的四个特性。

①事务是应用程序中一系列严密的操作，所有操作必须成功完成，否则在每个操作中所作的所有更改都会被撤销。也就是事务具有原子性，一个事务中的一系列的操作要么全部成功，要么一个都不做。事务的结束有两种，当事务中的所有步骤全部成功执行时，事务提交。如果其中一个步骤失败，将发生回滚操作，撤销撤销之前到事务开始时的所有操作。事务通常以 BEGIN TRANSACTION 开始，以 COMMIT 或 ROLLBACK 结束。

②事务的 ACID。

事务具有四个特征：原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这四个特性简称为 ACID 特性。

原子性：事务是数据库的逻辑工作单位，它对数据库的修改要么全部执行，要么全部不执行，是一个不可分割的工作单位。

一致性：事务前后，数据库的状态都满足所有的完整性约束。事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。比如，当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。如果数据库系统在运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是不一致的状态。

隔离性：并发执行的事务是隔离的，一个不影响一个。如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请求，使得在同一时间仅有一个请求用于同一数据。通过设置数据库的隔离级别，可以达到不同的隔离效果。

持续性：也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。接下来的其它操作或故障不应该对其执行结果有任何影响。

事务中的所有操作要么全部执行,要么都不执行；如果事务没有原子性的保证，那么在发生系统 故障的情况下，数据库就有可能处于不一致状态。 因而，事务的原子性与一致性是密切相关的。

3. 事务的隔离级别。

数据库事务无非就两种：读取事务(SELECT)和修改事务(UPDATE、INSERT)。

在没有事务隔离控制的时候，多个事务在同一时刻对同一数据的操作可能就会影响到最终期望的结果，通常有四种情况：

(1)修改时允许修改—丢失更新

(2)修改时允许读取—脏读

(3)读取时允许修改—不可重复读

(4)读取时允许插入—幻读

以上四种情况描述完毕，前三种是对同一条数据的并发操作，对程序的结果可能产生致命影响，尤其是金融等实时性，准确性要求极高的系统，绝不容许这三中情况的出现，相比第四种情况不会影响数据的真实性，在很多情况下是允许的，如社交论坛等实时性要求不高的系统。

四种情况问题严重性降低，但性能开销增加！因为不同的系统允许不同级别情况，所以就出现了

事务的隔离级别—四种隔离级别：

(1)未授权读取(Read uncommitted)：一个事务更新时不允许更新，但允许读取。可以防止丢失更新，但不能防止脏读、不可重复读、幻读。(隔离级别最低)

(2)授权读取(Read committed)：一个事务更新时不允许读取，必须等到更新事务提交后才能读取，可以防止丢失更新和脏读，但不能防止不可重复读、幻读。(隔离级别次低)

(3)可重复读取(Repeatable read)：一个事务读取时，不允许更新，但允许插入。可以防止丢失更新、脏读、不可重复读，但不能防止幻读。

PS:这是 MySQL 的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会得到同样的数据行。InnoDB 存储引擎通过多版本并发控制(MVCC)机制解决了幻读！

(4)序列化(Serializable)：提供严格的事务隔离，它要求事务序列化执行，事务只能一个接一个地执行，不能并发执行。

隔离级别	脏读	不可重复读	幻读
读未提交（Read uncommitted）	V	V	V
读已提交（Read committed）	X	V	V
可重复读（Repeatable read）	X	X	V
可串行化（Serializable）	X	X	X

6. SELECT 语句的执行顺序：

SELECT 的执行顺序：

FROM ， WHERE ， GROUP BY ， HAVING, (SELECT) ORDER BY ， LIMIT