第二周:认识SQL并学习数据库的基础操作

1. 什么事关系型和非关系型数据库, 两者包含哪些种类的数据库以及区别是什么
2. 选择一种关系型数据库进行学习
3. 学习数据库中的字段类型并创建库和用户表, 需要包含所有字段类型
4. 学习数据库的增删改查, 记录学习过程(重点是sql语句的理解)

关于数据库的介绍

数据库常分为**关系型数据库**和**非关系型数据库**

关系型数据库:

1. 采用了关系模型来组织数据的数据库
2. 关系型数据库的最大特点就是事务的一致性
3. 简单来讲, 关系模型指的二维表格模型

优点:

1. 容易理解, 二维表格是非常贴近逻辑世界的一个概念, 关系模型相对网状和层次等其他模型来说更容易理解
2. 使用方便, 通用的SQL语言使得操作关系型数据库十分方便
3. 易于维护, 丰富的完整性(实体完整性, 参照完整性和用户定义的完整性)大大降低了数据冗余和数据不一致的概率
4. 支持SQL(结构化查询语言Structured Query Language), 可支持复杂查询

缺点:

1. 为维护一致性而导致读写性能较差
2. 固定的表结构
3. 有高并发读写的需求

常见关系型数据库: MariaDB, SQLite, SQL Server, MySQL, Oracle, PostgreSQL等

我的理解: 关系顾名思义, 数据之间存在依赖关系, 那么其数据之间必然存在约束性. 当一个表内的数据变化时必须得符合约束条件. 查询时可以通过一个表而发现与其他表的关系. 数据库在存储数据表时也存储了数据表之间的相互关系, 形成一个数据之间互相关联的数据组织.

非关系型数据库 :与关系型相对, 也就是其存储数据时并不存储两种数据间的关系, 也就是只是数据结构化存储的集合.因此该类并不局限于存储基础数据类型, 它用键值对的方式可以存贮文档和图片等多种格式. 由于其数据间无耦合性, 所以容易扩展. 该类数据库不支持SQL语言.在我看来与字典相似.

常见非关系型数据库: mongoDB, HBASE, redis, CouchDB, Cassandra, Neo4j等

此次先学习MySQL

1. MySQL

MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

RDBMS其作用是管理数据库, 其意义与操作系统相似, 用于建立、使用和维护数据库，简称DBMS。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性.

特点:

- 1.数据以表格的形式出现
- 2.每行为各种记录名称
- 3.每列为记录名称所对应的数据域
- 4.许多的行和列组成一张表单
- 5.若干的表单组成database

RDBMS 术语

在我们开始学习MySQL 数据库前，让我们先了解下RDBMS的一些术语：

- **数据库:** 数据库是一些关联表的集合。
- **数据表:** 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。
- **列:** 一列(数据元素) 包含了相同类型的数据, 例如邮政编码的数据。
- **行：**一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。
- **冗余**：存储两倍数据，冗余降低了性能，但提高了数据的安全性。
- **主键**：主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。
- **外键：**外键用于关联两个表。
- **复合键**：复合键（组合键）将多个列作为一个索引键，一般用于复合索引。
- **索引：**使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。
- **参照完整性:** 参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。
- 表头(header): 每一列的名称;
- 列(col): 具有相同数据类型的数据的集合;
- 行(row): 每一行用来描述某条记录的具体信息;
- 值(value): 行的具体信息, 每个值必须与该列的数据类型相同;
- **键(key)**: 键的值在当前列中具有唯一性。

MySQL 数据类型

MySQL中定义数据字段的类型对你数据库的优化是非常重要的。

MySQL支持多种类型，大致可以分为三类：数值、日期/时间和字符串(字符)类型。

数值类型

MySQL支持所有标准SQL数值数据类型。

这些类型包括严格数值数据类型(INTEGER、SMALLINT、DECIMAL和NUMERIC)，以及近似数值数据类型(FLOAT、REAL和DOUBLE PRECISION)。

关键字INT是INTEGER的同义词，关键字DEC是DECIMAL的同义词。

BIT数据类型保存位字段值，并且支持MyISAM、MEMORY、InnoDB和BDB表。

作为SQL标准的扩展，MySQL也支持整数类型TINYINT、MEDIUMINT和BIGINT。下面的表显示了需要的每个整数类型的存储和范围。

类型	大小	范围（有符号）	范围（无符号）	用途
TINYINT	1 字节	(-128，127)	(0，255)	小整数值
SMALLINT	2 字节	(-32 768，32 767)	(0，65 535)	大整数值
MEDIUMINT	3 字节	(-8 388 608，8 388 607)	(0，16 777 215)	大整数值
INT或INTEGER	4 字节	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整数值
BIGINT	8 字节	(-9,223,372,036,854,775,808，9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	极大整数值
FLOAT	4 字节	(-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	单精度浮点数值
DOUBLE	8 字节	(-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	双精度浮点数值
DECIMAL	对DECIMAL(M,D)，如果M>D，为M+2否则为D+2	依赖于M和D的值	依赖于M和D的值	小数值

日期和时间类型

表示时间值的日期和时间类型为DATETIME、DATE、TIMESTAMP、TIME和YEAR。

每个时间类型有一个有效值范围和一个"零"值，当指定不合法的MySQL不能表示的值时使用"零"值。

TIMESTAMP类型有专有的自动更新特性，将在后面描述。

类型	大小(字节)	范围	格式	用途
DATE	3	1000-01-01/9999-12-31	YYYY-MM-DD	日期值
TIME	3	'-838:59:59'/'838:59:59'	HH:MM:SS	时间值或持续时间
YEAR	1	1901/2155	YYYY	年份值
DATETIME	8	1000-01-01 00:00:00/9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	混合日期和时间值
TIMESTAMP	4	1970-01-01 00:00:00/2038 结束时间是第 **2147483647** 秒，北京时间 **2038-1-19 11:14:07**，格林尼治时间 2038年1月19日 凌晨 03:14:07	YYYYMMDD HHMMSS	混合日期和时间值，时间戳

字符串类型

字符串类型指CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM和SET。该节描述了这些类型如何工作以及如何在查询中使用这些类型。

类型	大小	用途
CHAR	0-255字节	定长字符串
VARCHAR	0-65535 字节	变长字符串
TINYBLOB	0-255字节	不超过 255 个字符的二进制字符串
TINYTEXT	0-255字节	短文本字符串
BLOB	0-65 535字节	二进制形式的长文本数据
TEXT	0-65 535字节	长文本数据
MEDIUMBLOB	0-16 777 215字节	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215字节	中等长度文本数据
LONGBLOB	0-4 294 967 295字节	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295字节	极大文本数据

CHAR 和 VARCHAR 类型类似，但它们保存和检索的方式不同。它们的最大长度和是否尾部空格被保留等方面也不同。在存储或检索过程中不进行大小写转换。

BINARY 和 VARBINARY 类似于 CHAR 和 VARCHAR，不同的是它们包含二进制字符串而不要非二进制字符串。也就是说，它们包含字节字符串而不是字符字符串。这说明它们没有字符集，并且排序和比较基于列值字节的数值值。

BLOB 是一个二进制大对象，可以容纳可变数量的数据。有 4 种 BLOB 类型：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。它们区别在于可容纳存储范围不同。

有 4 种 TEXT 类型：TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT。对应的这 4 种 BLOB 类型，可存储的最大长度不同，可根据实际情况选择。

1.1 登录 MySQL

当 MySQL 服务已经运行时, 我们可以通过 MySQL 自带的客户端工具登录到 MySQL 数据库中, 首先打开命令提示符, 输入以下格式的命名:

```
mysql -h 主机名 -u 用户名 -p
```

参数说明：

- **-h** : 指定客户端所要登录的 MySQL 主机名, 登录本机(localhost 或 127.0.0.1)该参数可以省略;
- **-u** : 登录的用户名;
- **-p** : 告诉服务器将会使用一个密码来登录, 如果所要登录的用户名密码为空, 可以忽略此选项。

如果我们要登录本机的 MySQL 数据库，只需要输入以下命令即可：

```
mysql -u root -p
```

按回车确认, 如果安装正确且 MySQL 正在运行, 会得到以下响应:

```
Enter password:
```

若密码存在, 输入密码登录, 不存在则直接按回车登录。登录成功后你将会看到 Welcome to the MySQL monitor... 的提示语。

然后命令提示符会一直以 mysq> 加一个闪烁的光标等待命令的输入, 输入 **exit** 或 **quit** 退出登录。

1.2 增加用户及授予权限

1.2.1直接在mysql数据库中的user表中添加新用户即可

以下为添加用户的的实例，用户名为guest，密码为guest123，并授权用户可进行 SELECT, INSERT 和 UPDATE操作权限：

```
root@host# mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> INSERT INTO user
          (host, user, password,
           select_priv, insert_priv, update_priv)
           VALUES ('localhost', 'guest',
           PASSWORD('guest123'), 'Y', 'Y', 'Y');
Query OK, 1 row affected (0.20 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT host, user, password FROM user WHERE user = 'guest';
+-----------+---------+------------------+
| host      | user    | password         |
+-----------+---------+------------------+
| localhost | guest | 6f8c114b58f2ce9e |
+-----------+---------+------------------+
1 row in set (0.00 sec)
```

在添加用户时，请注意使用MySQL提供的 PASSWORD() 函数来对密码进行加密。 你可以在以上实例看到用户密码加密后为： 6f8c114b58f2ce9e.

注意：在 MySQL5.7 中 user 表的 password 已换成了**authentication_string**。

注意：password() 加密函数已经在 8.0.11 中移除了，可以使用 MD5() 函数代替。

注意：在注意需要执行 **FLUSH PRIVILEGES** 语句。 这个命令执行后会重新载入授权表。

如果你不使用该命令，你就无法使用新创建的用户来连接mysql服务器，除非你重启mysql服务器。

你可以在创建用户时，为用户指定权限，在对应的权限列中，在插入语句中设置为 'Y' 即可，用户权限列表如下：

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Reload_priv

- Shutdown_priv
- Process_priv
- File_priv
- Grant_priv
- References_priv
- Index_priv
- Alter_priv

1.2.2 使用GRANT命令添加用户

以下命令会给指定数据库TUTORIALS添加用户 zara ，密码为 zara123 。

```
root@host# mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    -> ON TUTORIALS.*
    -> TO 'zara'@'localhost'
    -> IDENTIFIED BY 'zara123';
```

以上命令会在mysql数据库中的user表创建一条用户信息记录。

注意: MySQL 的SQL语句以分号 `;` 作为结束标识。

1.3 管理MySQL命令

该小节主要是切换数据库和数据表, 以及查看表头和查看数据库的性能和统计信息.

- **USE 数据库名** :
  选择要操作的Mysql数据库，使用该命令后所有Mysql命令都只针对该数据库。

  ```
  mysql> use RUNOOB;
  Database changed
  ```

- **SHOW DATABASES:**
  列出 MySQL 数据库管理系统的数据库列表。

  ```
  mysql> SHOW DATABASES;
  +--------------------+
  | Database           |
  +--------------------+
  | information_schema |
  | RUNOOB             |
  | cdcol              |
  | mysql              |
  | onethink           |
  | performance_schema |
  | phpmyadmin         |
  | test               |
  | wecenter           |
  | wordpress          |
  +--------------------+
  10 rows in set (0.02 sec)
  ```

- **SHOW TABLES:**
  显示指定数据库的所有表，使用该命令前需要使用 use 命令来选择要操作的数据库。

```
mysql> use RUNOOB;
Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_runoob |
+------------------+
| employee_tbl     |
| runoob_tbl       |
| tcount_tbl       |
+------------------+
3 rows in set (0.00 sec)
```

- **SHOW COLUMNS FROM 数据表:**
  显示数据表的属性，属性类型，主键信息 ，是否为 NULL，默认值等其他信息。

```
mysql> SHOW COLUMNS FROM runoob_tbl;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| runoob_id       | int(11)      | NO   | PRI | NULL    |       |
| runoob_title    | varchar(255) | YES  |     | NULL    |       |
| runoob_author   | varchar(255) | YES  |     | NULL    |       |
| submission_date | date         | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

- **SHOW INDEX FROM 数据表:**
  显示数据表的详细索引信息，包括PRIMARY KEY（主键）。

```
mysql> SHOW INDEX FROM runoob_tbl;
+------------+------------+----------+--------------+-------------+---------
--+-------------+----------+--------+------+------------+---------+---------
------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name |
Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
Index_comment |
+------------+------------+----------+--------------+-------------+---------
--+-------------+----------+--------+------+------------+---------+---------
------+
| runoob_tbl |          0 | PRIMARY  |            1 | runoob_id   | A
  |           2 |     NULL | NULL   |      | BTREE      |         |
      |
+------------+------------+----------+--------------+-------------+---------
--+-------------+----------+--------+------+------------+---------+---------
------+
1 row in set (0.00 sec)
```

- **SHOW TABLE STATUS LIKE [FROM db_name] [LIKE 'pattern'] \G:**
  该命令将输出Mysql数据库管理系统的性能及统计信息。

```
mysql> SHOW TABLE STATUS  FROM RUNOOB;   # 显示数据库 RUNOOB 中所有表的信息

mysql> SHOW TABLE STATUS from RUNOOB LIKE 'runoob%';     # 表名以runoob开头的表的信息
mysql> SHOW TABLE STATUS from RUNOOB LIKE 'runoob%'\G;   # 加上 \G，查询结果按列打印
```

1.4 创建数据库

1.4.1 直接新建数据库

登录MySQL服务后直接输入`CREATE DATABASE 数据库名;`建立新数据库

1.4.2 使用mysqladmin建立数据库

使用mysqladmin命令, 默认位于`/usr/bin/mysqladmin`

```
[root@host]# mysqladmin -u root -p create 数据库名称
Enter password:******
```

执行成功后会创建所需数据库.

1.4.3 使用php脚本创建数据库

PHP 使用 mysqli_query 函数来创建或者删除 MySQL 数据库。

该函数有两个参数，在执行成功时返回 TRUE，否则返回 FALSE。

- 语法

```
mysqli_query(connection,query,resultmode);
```

参数	描述
connection	必需。规定要使用的 MySQL 连接。
query	必需，规定查询字符串。
resultmode	可选。一个常量。可以是下列值中的任意一个： MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个） MYSQLI_STORE_RESULT（默认）

- 实例

以下实例演示了使用PHP来创建一个数据库：

- 创建数据库

```
<?php

$dbhost = 'localhost:3306';
// mysql服务器主机地址, 如果在my.cnf中设置了skip-networking则此处不加3306端口号

$dbuser = 'root';            // mysql用户名

$dbpass = '123456';          // mysql用户名密码
```

```
 $conn = mysqli_connect($dbhost, $dbuser, $dbpass);

if(! $conn ) {

         die('连接错误: ' . mysqli_error($conn));

}

 echo '连接成功';

 $sql = 'CREATE DATABASE RUNOOB';

$retval = mysqli_query($conn,$sql );

if(! $retval ) {

        die('创建数据库失败: ' . mysqli_error($conn));

}

echo "数据库 RUNOOB 创建成功\n";

 mysqli_close($conn);

?>
```

执行成功后，返回如下结果：

```
连接成功
数据库 RUNOOB 创建成功
```

如果数据库已存在，执行后，返回如下结果：

```
连接成功
创建数据库失败: Can't create database 'RUNOOB'; database exists
```

小结

使用root登录后，可以使用

```
CREATE DATABASE IF NOT EXISTS RUNOOB DEFAULT CHARSET utf8 COLLATE
utf8_general_ci;
```

创建数据库，该命令的作用：

- 1. 如果数据库不存在则创建，存在则不创建。
- 2. 创建RUNOOB数据库，并设定默认编码集为utf8
- 3. 数据库校对规则设置为utf8大小写不敏感, ci:不区分大小写, cs:区分大小写

如果指定了CHARSET X和COLLATE Y，那么采用字符集X和校对规则Y。
如果指定了CHARSET X而没有指定COLLATE Y，那么采用CHARSET X和CHARSET X的默认校对规则。
如果在CREATE TABLE语句中没有指定表字符集和校对规则，则使用数据库字符集和校对规则作为默认值。

分别修改数据库，表，字段编码：
`ALTER DATABASE db_name DEFAULT CHARSET utf8 COLLATE utf8_general_ci;`
`ALTER TABLE tbl_name DEFAULT CHARSET utf8 COLLATE utf8_general_ci;`
`ALTER TABLE tbl_name CHANGE 'column_name' 'column_name' 类型 CHARSET utf8 COLLATE utf8_general_ci;`
把表默认的字符集和所有字符列（CHAR,VARCHAR,TEXT）改为新的字符集：
`ALTER TABLE tbl_name CONVERT TO CHARSET character_name CHARSET utf8 COLLATE utf8_general_ci`

查看数据库编码：
`SHOW CREATE DATABASE db_name;`
查看表编码：
`SHOW CREATE TABLE tb_name;`
查看字段编码：

`SHOW FULL COLUMNS FROM tb_name;`

1.5 删除数据库

1.5.1 直接删除数据库

drop 命令格式：

```
drop database 数据库名;
```

1.5.2 使用mysqladmin删除数据库

以下实例删除数据库 (该数据库在前一章节已创建)：

```
[root@host]# mysqladmin -u root -p drop 数据库名
Enter password:******
```

执行以上删除数据库命令后，会出现一个提示框，来确认是否真的删除数据库：

```
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the '数据库名' database [y/N] y
Database "数据库名" dropped
```

1.5.3 使用php脚本删除数据库

PHP使用 mysqli_query 函数来创建或者删除 MySQL 数据库。

该函数有两个参数，在执行成功时返回 TRUE，否则返回 FALSE。

- 语法

```
mysqli_query(connection,query,resultmode);
```

参数	描述
connection	必需。规定要使用的 MySQL 连接。
query	必需，规定查询字符串。
resultmode	可选。一个常量。可以是下列值中的任意一个： MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个） MYSQLI_STORE_RESULT（默认）

- 实例

以下实例演示了使用PHP mysqli_query函数来删除数据库：

- 删除数据库

```
<?php

$dbhost = 'localhost';  // mysql服务器主机地址

$dbuser = 'root';            // mysql用户名

$dbpass = '123456';          // mysql用户名密码

$conn = mysqli_connect($dbhost, $dbuser, $dbpass);

 if(! $conn ) {

    die('连接失败: ' . mysqli_error($conn));

 }

echo '连接成功<br />';

 $sql = 'DROP DATABASE RUNOOB';

$retval = mysqli_query( $conn, $sql );

if(! $retval ) {

     die('删除数据库失败: ' . mysqli_error($conn));

}

echo "数据库 RUNOOB 删除成功\n";

mysqli_close($conn);

?>
```

执行成功后，数结果为：

```
连接成功
数据库 RUNOOB 删除成功
```

注意： 在使用PHP脚本删除数据库时，不会出现确认是否删除信息，会直接删除指定数据库，所以你在删除数据库时要特别小心

MySQL中常用的数据库操作命令

- 选择数据库: `USE 数据库名;`
- 创建数据表:

  ```
  CREATE TABLE table_name (column_name column_type);
  ```

  以下例子中我们将在 RUNOOB 数据库中创建数据表runoob_tbl, 注意是` :

  ```
  CREATE TABLE IF NOT EXISTS `runoob_tbl`(
     `runoob_id` INT UNSIGNED AUTO_INCREMENT,
     `runoob_title` VARCHAR(100) NOT NULL,
     `runoob_author` VARCHAR(40) NOT NULL,
     `submission_date` DATE,
     PRIMARY KEY ( `runoob_id` )
  )ENGINE=InnoDB DEFAULT CHARSET=utf8;
  ```

  - 实例解析：
  - 如果你不想字段为 **NULL** 可以设置字段的属性为 **NOT NULL**， 在操作数据库时如果输入该字段的数据为**NULL** ，就会报错。
  - AUTO_INCREMENT定义列为自增的属性，一般用于主键，数值会自动加1。
  - PRIMARY KEY关键字用于定义列为主键。 您可以使用多列来定义主键，列间以逗号分隔, 自增列必须是主键。
  - ENGINE 设置存储引擎，CHARSET 设置编码。
- 删除数据表: 首先选择数据库, 然后删除表 `DROP TABLE table_name ;`

```
delete from table where

直接删除表中的某一行数据，并且同时将该行的删除操作作为事务记录在日志中保存以便进行进行回滚操作。所以 delete 相比较 truncate 更加占用资源，数据空间不释放，因为需回滚。对 table 和 view 都能操作。

truncate table

一次性地从表中删除所有的数据(释放存储表数据所用的数据页来删除数据)并不把单独的删除操作记录记入日志保存(只在事务日志中记录 页的释放)，因此也不能回滚，不能恢复数据，在删除的过程中不会激活与表有关的删除触发器，占用资源更加少，速度更快。数据空间会释放，这个表和索引所占用的空间会恢复到初始大小。只能操作没有关联视图的 table。

truncate table 不能用于参与了索引视图的表。

drop table

删除的是整个表，包括表的结构，数据，定义。永久抹去，空间释放。对 table 和 view 都能操作。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器，对于外键（foreignkey ）约束引用的表，不能使用 truncate table，而应使用不带 where 子句的 delete 语句。
```

- 插入数据: value与field一一对应.

  ```
  INSERT INTO table_name ( field1, field2,...fieldN )
                         VALUES
                         ( valueA1, valueA2,...valueAN )(valueB1,
  valueB2,...valueBN);
  ```

  如果数据是字符型，必须使用单引号或者双引号，如："value"。

- 查询数据:

```
SELECT column_name,column_name
FROM table_name
[WHERE Clause]
[LIMIT N][ OFFSET M]
```

SELECT 命令可以读取一条或者多条记录。

你可以使用星号（*）来代替其他字段，SELECT语句会返回表的所有字段数据

你可以使用 WHERE 语句来包含任何条件。

你可以使用 LIMIT 属性来设定返回的记录数。

你可以通过OFFSET指定SELECT语句开始查询的数据偏移量。默认情况下偏移量为0。

- 删除表中的数据 `DELETE FROM table_name [WHERE Clause];`

- 更新修改表中的数据

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

可以同时更新一个或多个字段。

可以在 WHERE 子句中指定任何条件。

可以在一个单独表中同时更新数据。

当你需要更新数据表中指定行的数据时 WHERE 子句是非常有用的。

小结

关于数据库的使用可以简单理解为: **你找谁**, **你要对它做什么**.

- **可以做的事情**: 增加, 删除, 修改
- **怎么找**: 通过where子句指定查询的约束条件