Task01：初识数据库与 SQL

本章主要对数据库进行基本介绍，考虑易用性及普及度，课程主要使用 **MySQL** 进行介绍。

SQL 训练营页面地址：https://tianchi.aliyun.com/specials/promotion/aicampsql

天池龙珠计划训练营地址：https://tianchi.aliyun.com/specials/promotion/aicamps

一、初识数据库

数据库是将大量数据保存起来，通过计算机加工而成的可以 进行高效访问的数据集合。该数据集合称为数据库（Database，DB）。用来管理数据库的计算机系统称为数据库管理系统（Database Management System，DBMS）。

1.1 DBMS 的种类

DBMS 主要通过数据的保存格式（数据库的种类）来进行分类，现阶段主要有以下 5 种类型.

- 层次数据库（Hierarchical Database，HDB）
- 关系数据库（Relational Database，RDB）

  这种类型的 DBMS 称为关系数据库管理系统（Relational Database Management System，RDBMS）。比较具有代表性的 RDBMS 有如下 5 种。

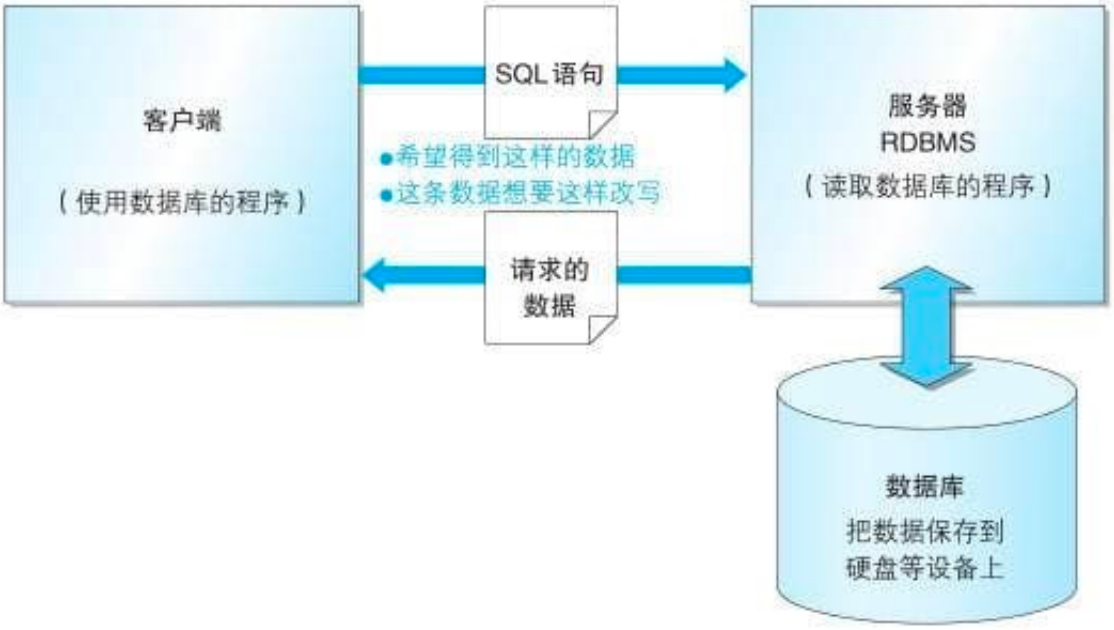
  - Oracle Database：甲骨文公司的 RDBMS
  - SQL Server：微软公司的 RDBMS
  - DB2：IBM 公司的 RDBMS
  - PostgreSQL：开源的 RDBMS
  - MySQL：开源的 RDBMS
- 面向对象数据库（Object Oriented Database，OODB）
- XML 数据库（XML Database，XMLDB）
- 键值存储系统（Key-Value Store，KVS），举例：MongoDB

本课程将向大家介绍使用 SQL 语言的数据库管理系统，也就是关系数据库管理系统（RDBMS）的操作方法。

1.2 RDBMS 的常见系统结构

使用 RDBMS 时，最常见的系统结构就是客户端 / 服务器类型（C/S 类型）这种结构（如下图）

图1-3　使用RDBMS时的系统结构



1.3 数据库安装（必须学习）

本次学习大家可以选择使用阿里云数据库服务器或者本地安装数据库进行学习，在下面对应的学习教程中也给告诉了大家如何创建本次学习需要的数据库表和数据，所以大家必须使用一个方式安装数据库，才能完成后面学习。

「Windows10专业版下安装Docker.pdf」，点击链接保存，或者复制本段内容，打开「阿里云盘」APP，无需下载极速在线查看，视频原画倍速播放。
链接：https://www.aliyundrive.com/s/GFqQ8ETaA1E

1.3.1 阿里云 MySQL 服务器使用介绍

节约篇幅，具体相关介绍以及给大家写到 pdf 里了，大家点击链接即可进入查看：

阿里云 MySQL 服务器使用介绍.pdf

优点： 操作使用方便，未来趋势（数据上云），导入、导出数据方便，运行速度快。
缺点： 需要付费购买，不过现在对开发者有优惠活动，基础版本 1 核 1G，存储空间 20G 的，目前优惠价半年只需 9.9 元，一杯奶茶钱不到。

1.3.2 本地 MySQL 环境搭建方法介绍

节约篇幅，具体相关介绍以及给大家写到 pdf 里了，大家点击链接即可进入查看：

本地 MySQL 环境搭建方法介绍.pdf

优点： 免费，增强动手能力。
缺点： 安装、配置麻烦，数据导入、导出耗时长。

二、初识 SQL

2.1 概念介绍

2.1.1 数据库基本概念

数据库是一个以某种有组织的方式存储的数据集合。数据库(database)是保存有组织的数据的容器。数据库管理系统(DBMS)是一种数据库软件，MySQL是一种DBMS，即它是一种数据库软件，作者使用的数据库管理系统是MySQL，除做特别说明外，作者使用的所有数据库软件都为MySQL。

表(table) 某种特定类型数据的结构化清单，是一种结构化的文件，可用来存储某种特定类型的数据。

模式(schema) 关于数据库和表的布局及特性的信息。

图1-6　表的示例(商品表)

商品编号	商品名称	商品种类	销售单价	进货单价	登记日期
0001	T恤衫	衣服	1000	500	2009-09-20
0002	打孔器	办公用品	500	320	2009-09-11
0003	运动T恤	衣服	4000	2800	
0004	菜刀	厨房用具	3000	2800	2009-09-20
0005	高压锅	厨房用具	6800	5000	2009-01-15
0006	叉子	厨房用具	500		2009-09-20
0007	擦菜板	厨房用具	880	790	2008-04-28
0008	圆珠笔	办公用品	100		2009-11-11

列名(数据的项目名称)

行(记录)

列(字段)

单元格

数据库中存储的表结构类似于 Excel 中的行和列，在数据库中，**行称为记录**，它相当于一条记录，**列称为字段**，它代表了表中存储的数据项目。

行和列交汇的地方称为**单元格**，一个单元格中只能输入一条记录。

主键(primary key)一列(或一组列)，其值能够唯一标识表中每一行。表中的任何列都可以作为主键，只要它满足以下条件：

任意两行都不具有相同的主键值;

每一行都必须具有一个主键值(主键列不允许NULL 值);

主键列中的值不允许修改或更新;

主键值不能重用(如果某行从表中删除，它的主键不能赋给以后的新行)

2.1.2 什么是 SQL

SQL是结构化查询语言(Structured Query Language)的缩写。SQL 是为操作数据库而开发的语言。国际标准化组织（ISO）为 SQL 制定了相应的标准，以此为基准的 SQL 称为标准 SQL（相关信息请参考专栏——标准 SQL 和特定的 SQL）。

SQL语句的优点

- SQL不是某个特定数据库供应商专有的语言。几乎所有重要的DBMS都支持SQL，所以，学习此语言使你几乎能与所有数据库打交道;
- SQL简单易学。它的语句全都是由描述性很强的英语单词组成，而且这些单词的数目不多;
- SQL尽管看上去很简单，但它实际上是一种强有力的语言，灵活使用其语言元素，可以进行非常复杂和高级的数据库操作。

完全基于标准 SQL 的 RDBMS 很少，通常需要根据不同的 RDBMS 来编写特定的 SQL 语句，原则上，本课程介绍的是标准 SQL 的书写方式。

根据对 RDBMS 赋予的指令种类的不同，SQL 语句可以分为以下三类.

- DDL

  DDL（Data Definition Language，数据定义语言） 用来创建或者删除存储数据用的数据库以及数据库中的表等对象。DDL 包含以下几种指令。

  - CREATE： 创建数据库和表等对象
  - DROP： 删除数据库和表等对象
  - ALTER： 修改数据库和表等对象的结构

- DML

  DML（Data Manipulation Language，数据操纵语言） 用来查询或者变更表中的记录。DML 包含以下几种指令。

    - SELECT： 查询表中的数据
    - INSERT： 向表中插入新数据
    - UPDATE： 更新表中的数据
    - DELETE： 删除表中的数据
- DCL

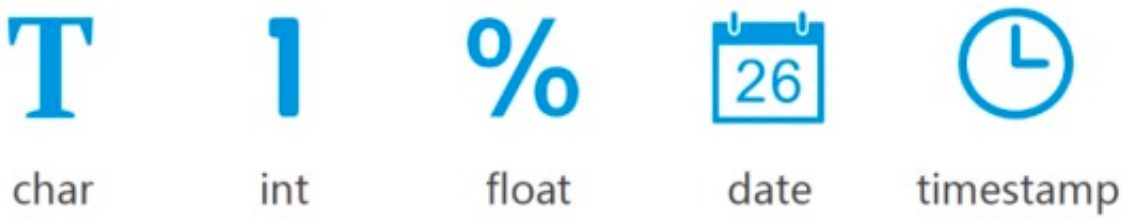
  DCL（Data Control Language，数据控制语言） 用来确认或者取消对数据库中的数据进行的变更。除此之外，还可以对 RDBMS 的用户是否有权限操作数据库中的对象（数据库表等）进行设定。DCL 包含以下几种指令。

    - COMMIT： 确认对数据库中的数据进行的变更
    - ROLLBACK： 取消对数据库中的数据进行的变更
    - GRANT： 赋予用户操作权限
    - REVOKE： 取消用户的操作权限

实际使用的 SQL 语句当中有 90% 属于 DML，本课程会以 DML 为中心进行讲解。

2.1.3 SQL数据类型

数据类型(datatype) 所容许的数据的类型。每个表列都有相应的数据类型，它限制(或容许)该列中存储的数据，常见的数据类型有字符串、数值、日期和时间、二进制数据类型。



- 字符串数据类型

数据类型	说　明
CHAR	1～255个字符的定长串。它的长度必须在创建时指定，否则MySQL假定为CHAR(1)
ENUM	接受最多64 K个串组成的一个预定义集合的某个串
LONGTEXT	与TEXT相同，但最大长度为4 GB
MEDIUMTEXT	与TEXT相同，但最大长度为16 K
SET	接受最多64个串组成的一个预定义集合的零个或多个串
TEXT	最大长度为64 K的变长文本
TINYTEXT	与TEXT相同，但最大长度为255字节
VARCHAR	长度可变，最多不超过255字节。如果在创建时指定为VARCHAR(n)，则可存储0到n个字符的变长串（其中n≤255）

- 数值数据类型

数据类型	说　明
BIT	位字段，1～64位。（在MySQL 5之前，BIT在功能上等价于TINYINT
BIGINT	整数值，支持–9223372036854775808～9223372036854775807（如果是UNSIGNED，为0～18446744073709551615）的数
BOOLEAN（或BOOL）	布尔标志，或者为0或者为1，主要用于开/关（on/off）标志
DECIMAL（或DEC）	精度可变的浮点值
DOUBLE	双精度浮点值
FLOAT	单精度浮点值
INT（或INTEGER）	整数值，支持–2147483648～2147483647（如果是UNSIGNED，为0～4294967295）的数
MEDIUMINT	整数值，支持–8388608～8388607（如果是UNSIGNED，为0～16777215）的数
REAL	4字节的浮点值
SMALLINT	整数值，支持–32768～32767（如果是UNSIGNED，为0～65535）的数
TINYINT	整数值，支持–128～127（如果为UNSIGNED，为0～255）的数

- 日期和时间数据类型

数据类型	说　明
DATE	表示1000-01-01～9999-12-31的日期，格式为YYYY-MM-DD
DATETIME	DATE和TIME的组合
TIMESTAMP	功能和DATETIME相同（但范围较小）
TIME	格式为HH:MM:SS
YEAR	用2位数字表示，范围是70（1970年）～69（2069年），用4位数字表示，范围是1901年～2155年

*二进制数据类型

数据类型	说　明
BLOB	Blob最大长度为64 KB
MEDIUMBLOB	Blob最大长度为16 MB
LONGBLOB	Blob最大长度为4 GB
TINYBLOB	Blob最大长度为255字节

2.2 SQL 的基本书写规则

- SQL 语句要以分号（;）结尾
- SQL 不区分关键字的大小写，但是插入到表中的数据是区分大小写的
- Windows 系统默认不区分表名及字段名的大小写
- Linux / Mac 默认严格区分表名及字段名的大小写(其实有的时候也不区分，但是最好是按照一定的规则来书写)
- 本教程已统一调整表名及字段名的为小写，以方便初学者学习使用。
- 常数的书写方式是固定的

  'abc', 1234, '26 Jan 2010', '10/01/26', '2010-01-26'...
- 单词需要用半角空格或者换行来分隔

SQL 语句的单词之间需使用半角空格或换行符来进行分隔，且不能使用全角空格作为单词的分隔符，否则会发生错误，出现无法预期的结果。

请大家认真查阅《附录 1 - SQL 语法规范》，养成规范的书写习惯。

2.3 数据库的创建（CREATE DATABASE 语句）

语法：

```
CREATE DATABASE < 数据库名称 > ;
```

创建本课程使用的数据库

```
CREATE DATABASE shop;
```

选择使用新创建的数据库

```
use shop;
```

一种更为专业和工程的做法：

设置数据库编码

```
CREATE DATABASE dbname DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
```

设置数据表编码

```
CREATE TABLE 'author' (
    'authorid' char(20) NOT NULL,
    'name' char(20) NOT NULL,
    'age' char(20) NOT NULL,
    'country' char(20) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1;
```

MySQL 中两种数据库引擎
ENGINE=MyISAM
ENGINE=InnoDB

2.4 表的创建（CREATE TABLE 语句）

语法：

```
CREATE TABLE < 表名 >
( < 列名 1> < 数据类型 > < 该列所需约束 > ,
  < 列名 2> < 数据类型 > < 该列所需约束 > ,
  < 列名 3> < 数据类型 > < 该列所需约束 > ,
  < 列名 4> < 数据类型 > < 该列所需约束 > ,
  .
  .
  .
```

```
    < 该表的约束 1> , < 该表的约束 2> ,……);
```

创建本课程用到的商品表

```
CREATE TABLE product(
    product_id CHAR(4) NOT NULL,
     product_name VARCHAR(100) NOT NULL,
     product_type VARCHAR(32) NOT NULL,
     sale_price INTEGER,
     purchase_price INTEGER,
     regist_date DATE,
     PRIMARY KEY(product_id)
 )  ;
```

2.5 命名规则

- 只能使用半角英文字母、数字、下划线（_）作为数据库、表和列的名称
- 名称必须以半角英文字母开头

商品表和 product 表列名的对应关系

商品表中的列名	Product表定义的列名
商品编号	product_id
商品名称	product_name
商品种类	product_type
销售单价	sale_price
进货单价	purchase_price
登记日期	regist_date

2.6 数据类型的指定

数据库创建的表，所有的列都必须指定数据类型，每一列都不能存储与该列数据类型不符的数据。

五种最基本的数据类型

- INTEGER 型

  用来指定存储整数的列的数据类型（数字型），不能存储小数。

- CHAR 型

  用来存储定长字符串，当列中存储的字符串长度达不到最大长度的时候，使用半角空格进行补足，由于会浪费存储空间，所以一般不使用。

- VARCHAR 型

  用来存储可变长度字符串，定长字符串在字符数未达到最大长度时会用半角空格补足，但可变长字符串不同，即使字符数未达到最大长度，也不会用半角空格补足。

- DATE 型

  用来指定存储日期（年月日）的列的数据类型（日期型）。

- float

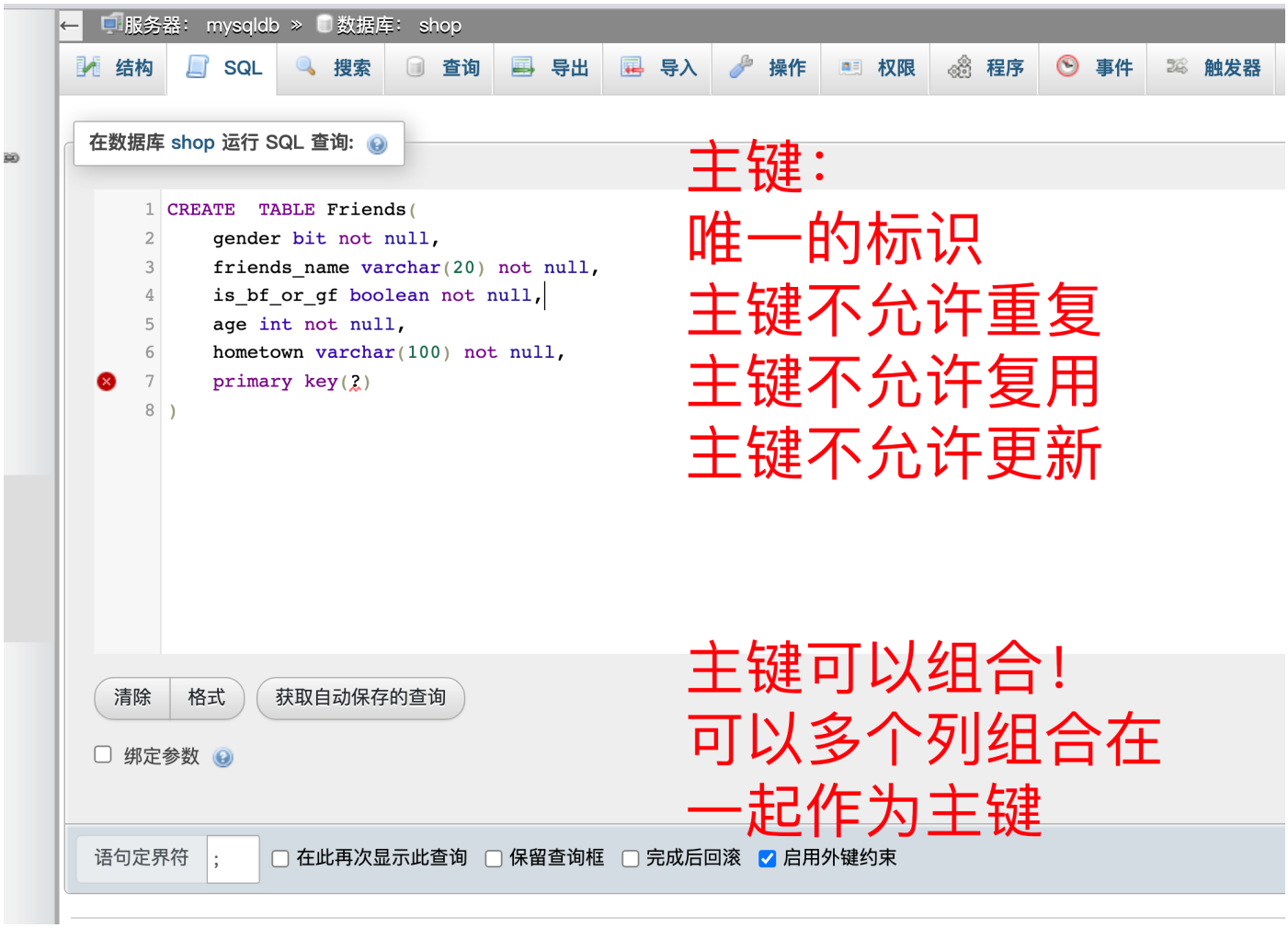
浮点小数。

2.7 约束的设置

约束是除了数据类型之外，对列中存储的数据进行限制或者追加条件的功能。

`NOT NULL` 是非空约束，即该列必须输入数据。

`PRIMARY KEY` 是主键约束，代表该列是唯一值，可以通过该列取出特定的行的数据。



2.8 表的删除和更新 (ALTER 、UPDATE)

2.8.1 删除表和表中的列的语法

```
DROP TABLE < 表名 > ;
```

删除 product 表

```
DROP TABLE product;
```

- 添加列的 ALTER TABLE 语句

```
ALTER TABLE < 表名 > ADD COLUMN < 列的定义 >;
```

添加一列可以存储 100 位的可变长字符串的 product_name_pinyin 列

```
ALTER TABLE product ADD COLUMN product_name_pinyin VARCHAR(100);
```

- 删除列的 ALTER TABLE 语句

```
ALTER TABLE < 表名 > DROP COLUMN < 列名 >;
```

删除 product_name_pinyin 列

```
ALTER TABLE product DROP COLUMN product_name_pinyin;
```

ALTER TABLE 语句和 DROP TABLE 语句一样，执行之后无法恢复。误添的列可以通过 ALTER TABLE 语句删除，或者将表全部删除之后重新再创建。

【扩展内容】

2.8.2 清空表内容

```
TRUNCATE TABLE TABLE_NAME;
```

优点：相比 drop/delete，truncate 用来清除数据时，速度最快。

2.8.3 数据的更新

基本语法：

```
    UPDATE < 表名 >
    SET < 列名 > = < 表达式 > [, < 列名 2>=< 表达式 2>...];
    WHERE < 条件 >;  -- 可选，非常重要。
    ORDER BY 子句;  -- 可选
    LIMIT 子句; -- 可选
```

> 使用 update 时要注意添加 where 条件，否则将会将所有的行按照语句修改

```
    -- 修改所有的注册时间
    UPDATE product
    SET regist_date = '2009-10-10';
    -- 仅修改部分商品的单价
    UPDATE product
    SET sale_price = sale_price * 10
    WHERE product_type = '厨房用具';
```

使用 UPDATE 也可以将列更新为 NULL（该更新俗称为 NULL 清空）。此时只需要将赋值表达式右边的值直接写为 NULL 即可。

```
    -- 将商品编号为 0008 的数据（圆珠笔）的登记日期更新为 NULL
```

```
    UPDATE product
    SET regist_date = NULL
    WHERE product_id = '0008';
```

和 INSERT 语句一样， UPDATE 语句也可以将 NULL 作为一个值来使用。**但是，只有未设置 NOT NULL 约束和主键约束的列才可以清空为 NULL。** 如果将设置了上述约束的列更新为 NULL，就会出错，这点与 INSERT 语句相同。

- 多列更新

UPDATE 语句的 SET 子句支持同时将多个列作为更新对象。

```
    -- 基础写法，一条 UPDATE 语句只更新一列
    UPDATE product
    SET sale_price = sale_price * 10
    WHERE product_type = '厨房用具';

    UPDATE product
    SET purchase_price = purchase_price / 2
    WHERE product_type = '厨房用具';
```

该写法可以得到正确结果，但是代码较为繁琐。可以采用合并的方法来简化代码。

```


    -- 合并后的写法
    UPDATE product
    SET sale_price = sale_price * 10,
        purchase_price = purchase_price / 2
    WHERE product_type = '厨房用具';
```

需要明确的是，SET 子句中的列不仅可以是两列，还可以是三列或者更多。

2.9 向 product 表中插入数据 (INSERT)

为了学习 `INSERT` 语句用法，我们首先创建一个名为 `productins` 的表，建表语句如下：

```
CREATE TABLE productins(
    product_id     CHAR(4)      NOT NULL,
    product_name   VARCHAR(100) NOT NULL,
    product_type   VARCHAR(32)  NOT NULL,
    sale_price     INTEGER      DEFAULT 0,
    purchase_price INTEGER ,
    regist_date    DATE ,
    PRIMARY KEY (product_id)
);
```

基本语法：

```
INSERT INTO < 表名 > (列 1, 列 2, 列 3, ……) VALUES (值 1, 值 2, 值 3, ……);
```

对表进行全列 `INSERT` 时，可以省略表名后的列清单。这时 `VALUES` 子句的值会默认按照从左到右的顺序赋给每一列。

```
-- 包含列清单
INSERT INTO productins (product_id, product_name, product_type,
sale_price, purchase_price, regist_date)VALUES ('0005', '高压锅', '厨房用具'
, 6800, 5000, '2009-01-15');
-- 省略列清单
INSERT INTO productins
VALUES ('0005', '高压锅', '厨房用具', 6800, 5000, '2009-01-15');
```

原则上，执行一次 `INSERT` 语句会插入一行数据。插入多行时，通常需要循环执行相应次数的 `INSERT` 语句。其实很多 `RDBMS` 都支持一次插入多行数据

```
-- 通常的 INSERT
INSERT INTO productins VALUES ('0002', '打孔器',
'办公用品', 500, 320, '2009-09-11');
INSERT INTO productins VALUES ('0003', '运动 T 恤',
'衣服', 4000, 2800, NULL);
INSERT INTO productins VALUES ('0004', '菜刀',
'厨房用具', 3000, 2800, '2009-09-20');


-- 多行 INSERT （DB2、SQL、SQL Server、 PostgreSQL 和 MySQL 多行插入）
INSERT INTO productins VALUES ('0002', '打孔器',
'办公用品', 500, 320, '2009-09-11'),
('0003', '运动 T 恤', '衣服', 4000, 2800, NULL),
('0004', '菜刀', '厨房用具', 3000, 2800, '2009-09-20');


-- Oracle 中的多行 INSERT
INSERT ALL INTO productins VALUES ('0002', '打孔器', '办公用品', 500, 320, '
2009-09-11')
INTO productins VALUES ('0003', '运动 T 恤', '衣服', 4000, 2800, NULL)
INTO productins VALUES ('0004', '菜刀', '厨房用具', 3000, 2800, '2009-09-20'
)
SELECT * FROM DUAL;
-- DUAL 是 Oracle 特有（安装时的必选项）的一种临时表 A。因此“SELECT * FROM DUAL”
部分也只是临时性的，并没有实际意义。
```

INSERT 语句中想给某一列赋予 `NULL` 值时，可以直接在 `VALUES` 子句的值清单中写入

`NULL`。想要插入 `NULL` 的列一定不能设置 `NOT NULL` 约束。

```
INSERT INTO productins (product_id, product_name, product_type,
sale_price, purchase_price, regist_date) VALUES ('0006', '叉子',
'厨房用具', 500, NULL, '2009-09-20');
```

还可以向表中插入 **默认值（初始值）**。可以通过在创建表的 `CREATE TABLE` 语句中设置 `DEFAULT` 约束来设定默认值。

```
CREATE TABLE productins
(product_id CHAR(4) NOT NULL,
 (略)
sale_price INTEGER
 (略)    DEFAULT 0, -- 销售单价的默认值设定为 0;
PRIMARY KEY (product_id));
```

可以使用 `INSERT … SELECT` 语句从其他表复制数据。

```
-- 将商品表中的数据复制到商品复制表中
INSERT INTO productcopy (product_id, product_name, product_type, sale_pric
e, purchase_price, regist_date)
SELECT product_id, product_name, product_type, sale_price,
purchase_price, regist_date
FROM Product;
```

> 本课程用表插入数据 sql 如下：

```
- DML ：插入数据
START 无TRANSACTION;
INSERT INTO product VALUES('0001', 'T 恤衫', '衣服', 1000, 500, '2009-09-20
');
INSERT INTO product VALUES('0002', '打孔器', '办公用品', 500, 320, '2009-09-
11');
INSERT INTO product VALUES('0003', '运动 T 恤', '衣服', 4000, 2800, NULL);
INSERT INTO product VALUES('0004', '菜刀', '厨房用具', 3000, 2800, '2009-09-
20');
INSERT INTO product VALUES('0005', '高压锅', '厨房用具', 6800, 5000, '2009-0
1-15');
INSERT INTO product VALUES('0006', '叉子', '厨房用具', 500, NULL, '2009-09-2
0');
INSERT INTO product VALUES('0007', '擦菜板', '厨房用具', 880, 790, '2008-04-
28');
INSERT INTO product VALUES('0008', '圆珠笔', '办公用品', 100, NULL, '2009-11
-11');
COMMIT;
```

2.10 自动增长属性

在表 shop.students 运行 SQL 查询:

```
1  CREATE TABLE students_new(
2      student_id int not null AUTO_INCREMENT,
3      student_name varchar(20) not null,
4      student_age int not null,
5      student_weight float not null DEFAULT 99.9,
6      PRIMARY KEY(student_id)
7  )
```

自动增长，当然也可以设定每次增长的步长，这个在分库分表的时候有用

SELECT * | SELECT | INSERT | UPDATE | DELETE | 清除 | 格式 | 获取自动保存的查询

绑定参数

有了自动增长，就可以全部给 0

```
1  INSERT INTO students_new(student_name,student_age) VALUES('王菲',19);
2
3  INSERT INTO students_new(student_id,student_name,student_age,student_weight) VALUES(5,'张玉婷',22,109.99);
4
5  INSERT INTO students_new(student_name,student_age) VALUES('丛老师',26);
6
7  INSERT INTO students_new VALUES(0,'杨小辉',3,42)
8
9
```

如果失败了怎么办?

SELECT * | SELECT | INSERT | UPDATE | DELETE | 清除 | 格式 | 获取自动保存的查询

绑定参数

auto_increament 插入失败，会跳过

语句定界符 ; 在此再次显示此查询 保留查询框 完成后回滚 启用外键约束

```
CREATE TABLE ai_test(
    ai_id int not null AUTO_INCREMENT,
    ai_unique varchar(20) not null,
```

```
    PRIMARY KEY(ai_id),
    CONSTRAINT ai_unique_ui UNIQUE (ai_unique)
)


INSERT INTO ai_test VALUES (0,'124');
INSERT INTO ai_test VALUES (0,'125');
INSERT INTO ai_test VALUES (0,'126');
INSERT INTO ai_test VALUES (0,'127');




INSERT INTO ai_test VALUES (0,'124');

INSERT INTO ai_test VALUES (0,'124');

INSERT INTO ai_test VALUES (0,'124');

INSERT INTO ai_test VALUES (0,'124');

INSERT INTO ai_test VALUES (0,'128');
```

```
INSERT INTO students_new(student_name,student_age) VALUES('王菲',19);

INSERT INTO students_new(student_id,student_name,student_age,student_weigh
t) VALUES(5,'张玉婷',22,109.99);

INSERT INTO students_new(student_name,student_age) VALUES('丛老师',26);

INSERT INTO students_new VALUES(0,'杨小辉',3,42)
```

ID 可以为负数

+ 选项

←T→	student_id	student_name	student_age	student_weight
编辑 复制 删除	-100	杨小辉	3	42
编辑 复制 删除	-1	杨小辉222	3	42
编辑 复制 删除	1	王菲	19	99.9
编辑 复制 删除	2	杨大辉	3	42
编辑 复制 删除	5	张玉婷	22	109.99
编辑 复制 删除	6	丛老师	26	99.9
编辑 复制 删除	7	杨小辉	3	42

三、练习题

3.1 创建表

编写一条 `CREATE TABLE` 语句，用来创建一个包含表 1-A 中所列各项的表 `Addressbook`（地址簿），并为 regist_no （注册编号）列设置主键约束

表 1-A　表 Addressbook （地址簿）中的列

列的含义	列的名称	数据类型	约束
注册编号	regist_no	整数型	不能为NULL、主键
姓名	name	可变长字符串类型（长度为128）	不能为NULL
住址	address	可变长字符串类型（长度为256）	不能为NULL
电话号码	tel_no	定长字符串类型（长度为10）	
邮箱地址	mail_address	定长字符串类型（长度为20）	

```
CREATE TABLE Addressbook(
   regist_no        INTEGER          NOT NULL,
   name             VARCHAR(128)     NOT NULL,
   address          VARCHAR(256)     NOT NULL,
```

```
    tel_no           CHAR(10)     ,
    mail_address     CHAR(20)     ,
    PRIMARY KEY (regist_no)
);
```

3.2 增加新的列

假设在创建练习 1.1 中的 Addressbook 表时忘记添加如下一列 `postal_code` （邮政编码）了，请把此列添加到 Addressbook 表中。

列名： postal_code

数据类型： 定长字符串类型（长度为 8）

约束： 不能为 NULL

```
-- [MySQL]

ALTER TABLE Addressbook ADD COLUMN postal_code CHAR(8) NOT NULL ;

-- [Oracle]
ALTER TABLE Addressbook ADD (postal_code CHAR(8)) NOT NULL;

-- [SQL Server]
ALTER TABLE Addressbook ADD postal_code CHAR(8) NOT NULL;

/*
 [DB2] 无法添加。
 在 DB2 中，如果要为添加的列设定 NOT NULL 约束，
 需要像下面这样指定默认值，或者删除 NOT NULL 约束，
 否则就无法添加新列。

*/
-- [DB2 修正版]
ALTER TABLE Addressbook ADD COLUMN postal_code CHAR(8) NOT NULL DEFAULT '0
000-
000';
```

3.3 删除表

编写 SQL 语句来删除 Addressbook 表。

```
DROP TALBE Addressbook;
```

3.4 恢复?

编写 SQL 语句来恢复删除掉的 Addressbook 表。

删除后的表无法使用命令进行恢复，请使用习题 3.1 答案中的 `CREATE TABLE` 语句再次创建所需的表。