第二章 初识数据库

1.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）**
  - 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

如上是5种具有代表性的RDBMS，其特点是由行和列组成的二维表来管理数据，这种类型的 DBMS 称为关系数据库管理系统（Relational Database Management System，RDBMS）。

- **键值存储系统（Key-Value　Store，KVS），举例：MongoDB**
- 面向对象数据库（Object　Oriented　Database，OODB）
- XML数据库（XML　Database，XMLDB）

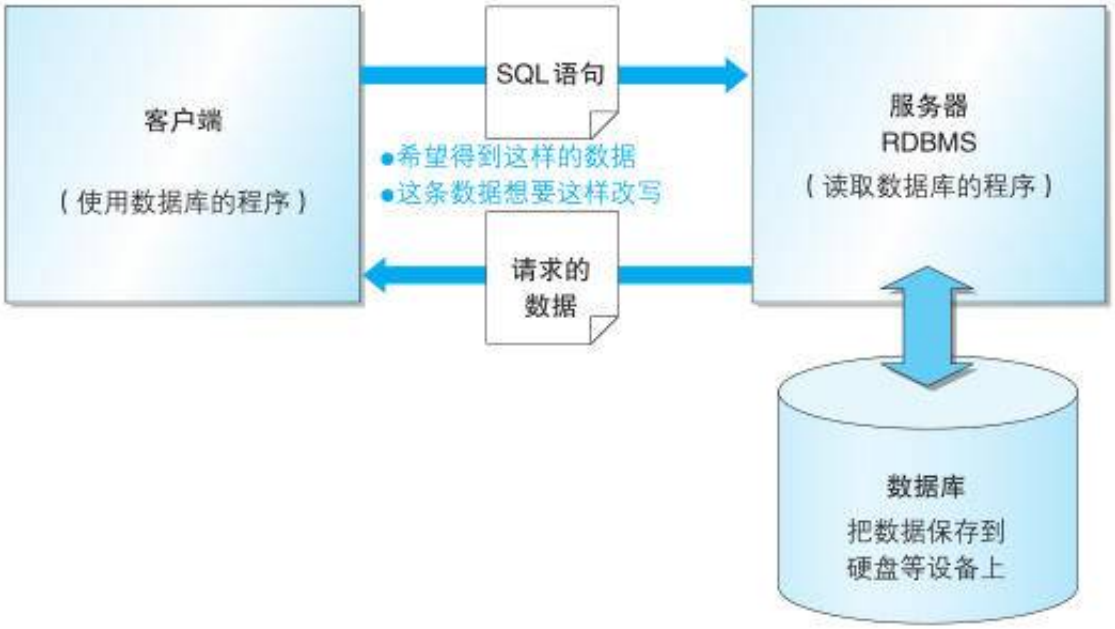
市场上比较常用的是关系型数据库mysql和非关系型数据库MongoDB。大家只要会使用其中一种即可。

本次向大家介绍使用 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

图1-3　使用RDBMS时的系统结构



2.初识 SQL

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。

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。

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
- **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。

2.1 SQL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
- win 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

参考资料：

- SQL编程风格https://zhuanlan.zhihu.com/p/27466166
- SQL Style Guidehttps://www.sqlstyle.guide/

2.2 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

2.3 数据类型

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。四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）。

2.4 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。

3.创建数据库及其数据，学习sql常用语句

3.1 数据库的创建（ CREATE DATABASE 语句）

语法：

```
CREATE DATABASE < 数据库名称 > ;
```

示例：

```
CREATE DATABASE shop;
```

3.2 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> ,……);
```

比如：

```
CREATE TABLE product   # product 表名
(product_id CHAR(4) NOT NULL,  # product_id 列名、CHAR(4)数据类型、NOT NULL 约束不能为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));
```

3.3 表的删除（DROP TABLE）

语法：

```
DROP TABLE < 表名 > ;
```

删除 product 表，需要特别注意的是，**删除的表是无法恢复的，只能重新插入**，请执行删除操作时要特别谨慎。实际场景无法重新插入，所以不要轻易删出数据库数据，除非实时备份并不在线上数据库操作。

```
DROP TABLE product;
```

3.4 表数据的更新（ALTER TABLE）

- 添加列的 ALTER TABLE 语句

```
-- 语法：

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

-- 比如：向 product 表中添加一列可以存储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 >;

-- 比如：向 product 表中删除 product_name_pinyin 列

ALTER TABLE product DROP COLUMN product_name_pinyin;
```

- 删除表中特定的行（语法）

```
-- 一定注意添加 WHERE 条件，否则将会删除所有的数据
DELETE FROM product WHERE COLUMN_NAME='XXX';
```

- 清空表内容

```
TRUNCATE TABLE TABLE_NAME;
```

注意事项：

- ALTER TABLE 语句和 DROP TABLE 语句一样，执行之后无法恢复。误添加的列可以通过 ALTER TABLE 语句删除，或者将表全部删除之后重新再创建。
- 相比drop / delete，truncate用来清除数据时，速度最快。

3.5 表数据的更新（UPDATE）

```
-- 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
```

使用示例如下：

- 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。

```
-- 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';
```

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

- 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';
-- ----------------------------------------------------
--             合并后的写法，一条UPDATE语句只更新多列
-- ----------------------------------------------------
UPDATE product
   SET sale_price = sale_price * 10,
       purchase_price = purchase_price / 2
 WHERE product_type = '厨房用具';
```

以上2种写法等效。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。

3.6 插入数据（INSERT）

注意事项：**表必须存在才能插入数据，即表存在即插入数据，不存在先创建在插入**

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
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');
```

3.7 索引

- 索引的作用
  MySQL索引的建立对于MySQL的高效运行是很重要的，索引可以大大提高MySQL的检索速度。

打个比方，如果合理的设计且使用索引的 MySQL 是一辆兰博基尼的话，那么没有设计和使用索引的 MySQL 就是一个人力三轮车。

拿汉语字典的目录页（索引）打比方，我们可以按拼音、笔画、偏旁部首等排序的目录（索引）快速查找到需要的字。

索引创建了一种有序的数据结构，采用二分法搜索数据时，其复杂度为 1 ，1000多万的数据只要搜索23次，其效率是非常高效的。

- 如何创建索引

创建表时可以直接创建索引，语法如下：

```
CREATE TABLE mytable(

ID INT NOT NULL,

username VARCHAR(16) NOT NULL,

INDEX [indexName] (username(length))

);
```

也可以使用如下语句创建：

```
-- 方法1
CREATE INDEX indexName ON table_name (column_name)

-- 方法2
ALTER table tableName ADD INDEX indexName(columnName)
```

- 索引分类
  - 主键索引:建立在主键上的索引被称为主键索引，一张数据表只能有一个主键索引，索引列值不允许有空值，通常在创建表时一起创建。
  - 唯一索引:建立在UNIQUE字段上的索引被称为唯一索引，一张表可以有多个唯一索引，索引列值允许为空，列值中出现多个空值不会发生重复冲突。
  - 普通索引:建立在普通字段上的索引被称为普通索引。
  - 前缀索引:前缀索引是指对字符类型字段的前几个字符或对二进制类型字段的前几个bytes建立的索引，而不是在整个字段上建索引。前缀索引可以建立在类型为char、varchar、binary、varbinary的列上，可以大大减少索引占用的存储空间，也能提升索引的查询效率。
  - 全文索引:利用“分词技术”实现在长文本中搜索关键字的一种索引。
  - 单列索引:建立在单个列上的索引被称为单列索引。
  - 联合索引（复合索引、多列索引）

3.7 本次练习使用的数据

```
-- 创建数据库
CREATE DATABASE shop;

-- 创建数据表
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));

-- DML ： 插入数据
STARTTRANSACTION;
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, '2009-01-
15');
INSERT INTO product VALUES('0006', '叉子', '厨房用具', 500, NULL, '2009-09-20');
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;
```

4.练习题

（1）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 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）	

answers:

```
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));
```

(2)假设在创建练习(1)中的 Addressbook 表时忘记添加如下一列 postal_code （邮政编码）了，请编写 SQL 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

answers:

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

(3)请补充如下 SQL 语句来删除 Addressbook 表。

```
(  ) table Addressbook;
```

answers:
DROP

（4）是否可以编写 SQL 语句来恢复删除掉的 Addressbook 表?
answers:不可以。