第一章：初识数据库

本章主要讲解数据库安装和数据库基本介绍，考虑易用性及普及度，本课程采取mysql进行教学。

1.1 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）。

1.1.1 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

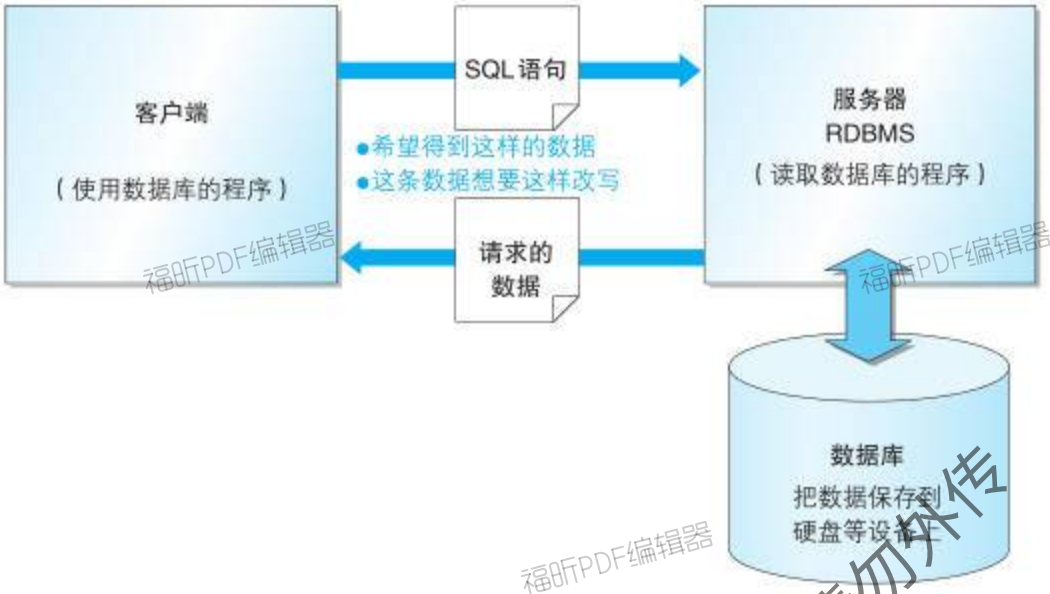
  如上是5种具有代表性的RDBMS，其特点是由行和列组成的二维表来管理数据，这种类型的 DBMS 称为关系数据库管理系统（Relational Database Management System，RDBMS）。
- 面向对象数据库（Object Oriented Database，OODB）
- XML数据库（XML Database，XMLDB）
- 键值存储系统（Key-Value Store，KVS），举例：MongoDB

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。

1.1.2 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）

图1-3　使用RDBMS时的系统结构



1.2 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，本书同样会以 DML 为中心进行讲解。

1.2.1 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
- 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
  * 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 `./materials/附录1 - SQL 语法规范` ，养成规范的书写习惯。

1.2.2 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;
```

1.2.3 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));
```

1.2.4 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

表1-3　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

1.2.5 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。

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）。

1.2.6 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。

1.2.7 表的删除和更新

- 删除表的语法：

```
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。

```
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;
```

- 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';
```

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。
【扩展内容】

- 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。

- 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。

1.2.8 向 product 表中插入数据

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
-- 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
```

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
```

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_price, purchase_price, regis
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
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;
```

1.2.9 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$log_2(N)$，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。

语

法

：`SELECT * FROM article WHERE MATCH (col1, col2, ...) AGAINST (expr [ search _ modifier`

1、MySQL 5.6 以前的版本，只有 MyISAM 存储引擎支持全文索引；

2、MySQL 5.6 及以后的版本，MyISAM 和 InnoDB 存储引擎均支持全文索引;

3、只有字段的数据类型为 char、varchar、text 及其系列才可以建全文索引。

4、如果可能，请尽量先创建表并插入所有数据后再创建全文索引，而不要在创建表时就直接创建全文索引，因为前者比后者的全文索引效率要高。

- 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）

建立在多个列上的索引被称为联合索引，又叫复合索引、组合索引。

练习题

1.1

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)	

1.2

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

1.3 填空题

请补充如下 SQL 语句来删除 Addressbook 表。

```
(    ) table Addressbook;
```

1.4 判断题

是否可以编写 SQL 语句来恢复删除掉的 Addressbook 表？