

旺旺老师 版权所有

QQ : 22713528

Email : shudaizi@163.com

Blog: <http://blog.sina.com.cn/liuhongwanglaoshi>



旺旺老师简介

刘红旺，82 年生于山西，02 年来西安求学，一年后辍学从事软件开发工作，三年后投身软件教育至今。至此，旺旺拥有七年工作经验，四年软件教育经验。旺旺兴趣广泛，近期一直学习经济学与心理学。旺旺老师课程特点：通俗易懂，风趣幽默。

旺旺老师软件工程师教程

SQLServer 电子书





城市
陕西，西安

QQ
22713528

电子邮件
shudaizi@163.com

BLOG
<http://blog.sina.com.cn/liuhongwanglao>
[shi](#)

说明：本教材配套视频教程会同期发布，欢迎大家访问旺旺老师的 blog 了解详情，同时，如果您对本教程有什么好的建议，也欢迎写信给旺旺老师，更欢迎您加旺旺老师 QQ 与他交流。

目录

- ❖ 数据库设计
 - ❖ 数据库设计在软件开发中的步骤
 - ❖ 三大范式
- ❖ 数据库实现
 - ❖ DDL,DML,DCL
 - ❖ 建库，建表，加约束
- ❖ 查询
 - ❖ 标准查询
 - ❖ 联接
 - ❖ 子查询
- ❖ T-SQL 编程
 - ❖ 事务，索引，视图
- ❖ 存储过程,ado 调用存储过程

再说明：

现在市面上的 java 教材有两种：

一种是学院派老师编写的，他们是主流，你看到十本 java 书，有九本半是这样的。他们的作者拥有让人敬仰的称谓，如某某教授，某某专家；他们写的书都是大部头，动辄上千上万；他们文风严谨，遣词专业；他们案例深奥，让人深思。总之，我很敬仰他们，因为我当初也是读着这样的书学 java 的。但现在看来，这样的书并不适合入门，情节大都是这样的，当我们怀着莫大的热情准备开始学习 java,买了一本久仰的《java 编程宝典》（有如此书，无从得之，是旺旺杜撰的书名），他很厚很重，很专业，你用毅力恒心支持看了几十页，才相信，java 的确是难学，后来，你就不怎么看它了。直到过了很多天，也许你已从事 java 方面的开发工作，一天，从书堆里无意的见到它，拍拍灰尘看看，够厚，就作为工具书备用吧。

还有一种是像旺旺这样草根阶层（好听点叫实战派）编写的，这样的书凤毛麟角，因为大家都在忙于工作，不像旺旺这样打了鸡血精力充沛的无法发泄。他们的称谓一般都不接地气，不谈也罢；他们书也不怎么厚，能上千页的更少之又少；他们文风朴实，贴近大众；他们案例生动，通俗易懂，风趣幽默。（没办法，其实旺旺老师讲课还一般，最厉害的还是给自己脸上贴金）。OK,如果你初学 java,你需要的就是一本这样的书，所以《旺旺老师 JavaSE 教程》的目标人群是：初学 java 的读者。

还有如果您已认真完善的学习过 java,或已从事 java 方面的开发工作，那也可以看看本书的部分章节，旺旺一些幽默风趣的案例也许能给您带来一些惊喜。

数据库设计

软件生命周期

- 1 需求分析：分析客户的业务和数据处理需求；
- 2 概要设计：设计数据库的 E-R (Entity-Relationship) 模型图
 - 2.1 ER 图：矩形是实体，椭圆是属性，菱形是关系
 - 2.2 关系的种类
 - 2.2.1 一对一：网站快速注册基本信息表，完备详细信息对应详细信息表
 - 2.2.2 一对多：主外键关系。班级对学生，学校对班级等等
 - 2.2.3 多对一：
 - 2.2.4 多对多：两个一对多形成一个多对多。一般用三张表来实现，学生表，课程表，成绩表，一个学生可以参加多门课程的考试，一门课程可以被多个学生考。比如，商品表，客户表，订单表，一个商品可以被多个客户购买，一个客户可以购买多个商品。
- 3 详细设计：形成数据字典
- 4 编码：选定具体的数据库如 oracle, sqlserver, db2, mysql 等等，编写具体的 sql 脚本实现
- 5 测试
- 6 部署维护

三大范式

定义：范式是衡量数据库设计是否标准合理的规范

第一范式：原子性：每列都是不可分割的最小逻辑单元，

第二范式：在满足第一范式的基础上，除去主键列的其他列都依赖于主键列，保证每张表描述一件事情。

第三范式：在满足第二范式的基础上，除去主键列的其他列都不传递依赖于主键列。

注意：在我们设计数据库的过程中，不但要考虑范式，还要考虑效率，比如姓名列就不满足第一范式，国外人姓名较长一般满足，国人姓名较短，一般不满足。

数据库实现

SQL 分类

- DDL(data define language):数据定义语言: create Create database[table, index, view, procedure] drop alter
- DML(data management language):数据操作（管理）语言 ,insert update delete select
- DCL(data control language):数据控制语言, 创建登陆用户，给用户赋予权限
GRANT select, insert, update ON stuInfo TO zhangsanDBUser
GRANT create table TO S26301DBUser

SQL 扩展

T-SQL: sqlserver 特有

P/L-SQL:oracle 特有

SQL 建库建表准备知识

建库，建表等 DDL 语言后加: go

两张特殊表（视图）：sysdatabases，其实是系统视图，存储当前数据库的详细信息，包括创建时间，创建人，主数据文件位置等信息，整个数据库只有一个，在 master 下。

Sysobjects（视图），每个数据库一张，存储当前数据库对象信息，对象包括表，视图，约束：

特殊的函数：exists 参数是一个 select 类别的 sql 语句，如果有结果返回 true, 否则返回 false。

```
Boolean exists(String sql) {  
    If(查询出结果)  
        Return true;  
    Else  
        Return false;  
}
```

SQL 建库

```
USE master --设置当前数据库为master，以便访问sysdatabases表  
GO  
IF EXISTS (SELECT * FROM sysdatabases WHERE name ='stuDB')  
    DROP DATABASE stuDB  
  
CREATE DATABASE stuDB  
    ON PRIMARY --默认就属于PRIMARY主文件组，可省略  
(  
    NAME='stuDB_data', --主数据文件的逻辑名  
    FILENAME='D:\project\stuDB_data.mdf', --主数据文件的物理名  
    SIZE=5mb, --主数据文件初始大小  
    MAXSIZE=100mb, --主数据文件增长的最大值  
    FILEGROWTH=15% --主数据文件的增长率  
)  
LOG ON  
(  
    NAME='stuDB_log',  
    FILENAME='D:\project\stuDB_log.ldf',  
    SIZE=2mb,  
    FILEGROWTH=1MB  
)  
GO
```

SQL 建表

约束类别

- 主键约束 (Primary Key Constraint)：要求主键列数据唯一，并且不允许为空
- 唯一约束 (Unique Constraint)：要求该列唯一，允许为空，但只能出现一个空值。
- 检查约束 (Check Constraint)：某列取值范围限制、格式限制等，如有关年龄的约束
- 默认约束 (Default Constraint)：某列的默认值，如我们的男性学员较多，性别默认为“男”
- 外键约束 (Foreign Key Constraint)：用于两表间建立关系，需要指定引用主表的那列，那个表是外键表，就修改那个表。

```
ALTER TABLE 表名
    ADD CONSTRAINT 约束名 约束类型 具体的约束说明
```

```
ALTER TABLE stuInfo
    ADD CONSTRAINT PK_stuNo PRIMARY KEY (stuNo)
ALTER TABLE stuInfo
    ADD CONSTRAINT UQ_stuID UNIQUE (stuID)
ALTER TABLE stuInfo
    ADD CONSTRAINT DF_stuAddress
        DEFAULT ('地址不详') FOR stuAddress
ALTER TABLE stuInfo
    ADD CONSTRAINT CK_stuAge
        CHECK(stuAge BETWEEN 15 AND 40)
ALTER TABLE stuMarks
    ADD CONSTRAINT FK_stuNo
        FOREIGN KEY(stuNo) REFERENCES stuInfo(stuNo)
GO
```

T-SQL 编程

变量分类

1 全局变量：两个@, 只读，系统定义赋值

1.1 @@error: 最后一个 sql 语句错误号，没错误 0

1.2 @@identity: 最后一次插入的标示号

1.3 @@rowcount: 最后一个 sql 语句影响的行数

1.4 @@transcount: 当前打开的事物数

2 局部变量：一个@

2.1 定义: declare @变量名 类型 declare @a int

2.2 赋值

2.2.1 Set @a = 10

2.2.2 根据查询结果赋值: select @a=列名 from 表名 where 条件 保证只返回一行数据

流程控制

1 If else begin end

2 while

3 CASE WHEN 条件 1 THEN 结果 1

ELSE 其他结果

END

条件可以是表达式，如@age>16, 类似以前的阶梯式 if

查询

标准查询

```
Select
    colName1, colName2, colName3 *****
    --聚合函数
from
    tableName
where
    colName1= ' ' or
    colName2 in(value1, value2,value3) and
    colName3 between 18 and 36 -- colName3>=18 and colName3<=36
group by - 分组，一般有 group by 都会使用聚合函数
    colName
having
    条件筛选
Order by
    colName asc/desc
```

- 1 聚合函数: max min sum/count=avg
- 2 group by 与 having: 有 having 必须一定有 group by, 有 group by 不一定有 having
- 3 where 与 having 区别: where 是对表里原始数据进行的筛选, having 是对分组使用聚合函数计算后的数据进行的筛选。

联接查询

使用环境: 当我们在一个结果集中显示多张表数据的情况下, 只能使用联接(子查询做不到, 因为子查询只能显示一张表的数据)。

- 1 内联接: 改变表出现顺序没有影响


```
Select
    t1.col, t2.col,**
from
    table1 t1
inner join
    table2 t2
on
    (t1.主键=t2.外键)
```

另一种写法：不推荐使用

```
Select
    t1.col, t2.col,**
from
    table1 t1
where
    t1.主键 = t2.外键
```

2 外联接：改变表出现的顺序有影响，因为前面出现的是左表

```
Select
    t1.col, t2.col,**
from
    table1 t1 - 前面出现的是左表
left(right) join
    table2 t2 --后面出现的是右表
on
    (t1.主键=t2.外键)
```

3 左外联接

3.1.1 把内连接的数据给显示出来

3.1.2 看左表中是否还有没匹配的数据，有的话原样列出，右表用 null 补齐

3.1.3 总归会显示左表的所有数据，右表与之匹配的数据

3.2 右外联接：与左外联接相反

3.3 完全外连接=左联接+右联接

4 交叉联接：一般不使用，返回数据的行数=左表行数*右表数据行数

子查询

- 1 定义：一个查询的结果作为另外查询的条件
- 2 使用范围：当结果集中只显示一张表的数据
- 3 使用方法：

3.1 `Select * from tableName where col=(select colName from tableName2 where 提交)`

如果子查询使用的是=, >, <等, 那要保证子查询返回的是一行一列数据

3.2 `Select * from tableName where in (select colName from tableName2 where 提交)`

如果子查询使用的是 in 关键字, 可以返回多行数据, 但只能是一列

3.3 `Select * from tableName where not in (select colName from tableName2 where 提交)`

可以查询没有参加考试的学生信息等在外键表没出现过的主键数据, 这个用左联接也可以实现, 左边是主键表, 在 where 中追加 右键表主键 is null 条件。

事务

1. 定义：事务(TRANSACTION)是作为单个逻辑工作单元执行的一系列操作，这些操作作为一个整体一起向系统提交，**要么全部执行、要么全不执行**，事务是一个不可分割的工作逻辑单元。
2. 特点：
 - a) 原子性 (Atomicity)：事务是一个完整的操作。事务的各步操作是不可分的（原子的）；**要么全部执行、要么全不执行**
 - b) 永久性 (Durability) 持久性：事务完成后，它对数据库的修改被永久保持，事务日志能够保持事务的永久性
 - c) 隔离性 (Isolation)：对数据进行修改的所有并发事务是彼此隔离的，这表明事务必须是独立的，它不应以任何方式依赖于或影响其他事务
 - d) 一致性 (Consistency)：当事务完成时，数据必须处于一致状态
3. 使用情况：insert, update, delete, 在应用程序中，当你执行一个操作，比如点击一个

按钮，操作了多张表或者说执行了多条 insert, update, delete 语句，那一定要使用事务。

4. 分类：

a) 隐式事务：自动提交事务，默认使用，把执行的每条 sql 语句作为一个独立的事务，

b) 显式事务：

i. Sql 脚本 `begin transaction rollback commit`

ii. C# 在 connection 上做事务

```
SqlTransaction t = Con.beginTransaction();
```

```
SqlCommand command = new SqlCommand(sql, con, t);
```

```
t.Commit();
```

```
t.Rollback();
```

iii. Java 在 connection 上做事务

```
con.setAutoCommit(false);
```

```
con.commit();
```

```
con.rollback();
```

索引

1. 定义：类似于目录，是为提高查询效率，速度服务的。

2. 分类：

a) 聚集索引：每个表只有一个，当给表添加主键约束时，会自动添加一个聚集索引，这也是主键索引，也是唯一索引。

b) 非聚集索引：每个表可以有多个（249）

3. 语法：`create index indexName`

4. 注意：带索引的表在数据库中需要更多的存储空间（存储索引页），虽然索引能提高查询速度，但也带来了额外的开销，比如当新增修改数据时，不但操作数据页，还要操作索引页。

5. 建索引规则：

- a) 适用
 - i. 该列用于频繁搜索
 - ii. 该列用于对数据进行排序
- b) 不适用
 - i. 列中仅包含几个不同的值。
 - ii. 表中仅包含几行。为小型表创建索引可能不太划算，因为 SQL Server 在索引中搜索数据所花的时间比在表中逐行搜索所花的时间更长

视图

1. 定义：视图是一张虚拟表，它表示一张表的部分数据或多张表的综合数据，其结构和数据是建立在对表的查询基础上。视图中并不存放数据，而是存放在视图所引用的原始表（基表）中。同一张原始表，根据不同用户的不同需求，可以创建不同的视图。
2. 语法 `create view view name as select * *** go`
3. 注意：视图数据是只读的，只能查看，不能新增修改删除。
4. 优点
 - a) 筛选表中的行
 - b) 防止未经许可的用户访问敏感数据
 - c) 降低数据库的复杂程度
 - d) 将多个物理数据库抽象为一个逻辑数据库

存储过程

1. 定义：存储在 sqlserver 服务器上，用 T-SQL 语句编写的可以包含复杂业务逻辑的函数（方法，过程），与普通方法不同，没有返回值，但可以多个输出参数实现返回值功能。

2. 分类：

- a) 系统存储过程 sp xp 开头

xp_cmdshell 可以执行 DOS 命令下的一些的操作

```
EXEC xp_cmdshell DOS 命令 [NO_OUTPUT]
```

- b) 自定义存储过程

```
CREATE PROC[EDURE] 存储过程名
```

```
    @参数 1 数据类型 = 默认值 OUTPUT,
```

```
    ..... ,
```

```
    @参数 n 数据类型 = 默认值 OUTPUT
```

```
AS
```

```
SQL 语句
```

```
GO
```

3. 其他：

- a) 因为存储过程可以使用 T-SQL 编程，也就是说里面可以包含流程控制等语句，所以能完成一些复杂的业务逻辑功能，但现在大部分 java 程序讲的是与具体的数据库相分离，而存储过程与数据库相绑定，所以编写 java 不推荐使用存储过程，但.net 平台还建议大量使用。

- b) 在业界，存储过程是衡量一个程序员技术是否牛叉的一个标志，一些爱装 13 的程序员常常自我炫耀自己写了一个几千行的存储过程。

- c) 其他优点：

- i. 执行速度更快

- ii. 允许模块化程序设计

- iii. 提高系统安全性

iv. 减少网络流通量

4. ado.net 调用存储过程方法：

```
CREATE TABLE bank
(
    cid int identity primary key,
    customerName CHAR(10),  --顾客姓名
    currentMoney MONEY      --当前余额
)
GO
INSERT INTO bank(customerName,currentMoney)
VALUES(' zhangsan',1000)
INSERT INTO bank(customerName,currentMoney)
VALUES(' lisi',1000)
create proc zhuanzhang
@money int,
@outId int,
@inId int,
@outMoney int output,
@inMoney int output
as
    update bank set currentMoney=currentMoney+@money
    where cid = @inId
    update bank set currentMoney=currentMoney-@money
    where cid = @outId
    select @outMoney=currentMoney from bank
    where cid = @outId
    select @inMoney=currentMoney from bank
    where cid = @inId
go
```

下面介绍 ado.net 如何调用存储过程：

```
SqlCommand command = new SqlCommand();

command.Connection = con;//设置连接

command.CommandType = CommandType.StoredProcedure;//设置执行类别

command.CommandText = "zhuanzhang";//设置调用存储过程名

//定义输入参数

SqlParameter sp1 = new SqlParameter("@money", money);
SqlParameter sp2 = new SqlParameter("@outId", outId);
SqlParameter sp3 = new SqlParameter("@inId", inId);

command.Parameters.Add(sp1);//注册输入参数

command.Parameters.Add(sp2);

command.Parameters.Add(sp3);

//定义输出参数

SqlParameter outMoneyParam = new SqlParameter();
outMoneyParam.ParameterName = "@outMoney";
outMoneyParam.SqlDbType = SqlDbType.Int; //输出参数类型
outMoneyParam.Direction = ParameterDirection.Output;
command.Parameters.Add(outMoneyParam);//注册输出参数

SqlParameter inMoneyParam = new SqlParameter();
inMoneyParam.ParameterName = "@inMoney";
inMoneyParam.SqlDbType = SqlDbType.Int;
inMoneyParam.Direction = ParameterDirection.Output;
command.Parameters.Add(inMoneyParam);

//执行存储过程

command.ExecuteNonQuery();

//得到输出参数

this.lblIn.Text = inMoneyParam.Value.ToString();
this.lblOut.Text = outMoneyParam.Value.ToString();
```

版本	修改内容	时间
1.0	创建	2010-03-24