SQL语言共分为四大类：数据查询语言DQL，数据操纵语言DML，数据定义语言DDL，数据控制语言DCL。

数据查询语言DQL基本结构是由SELECT子句，FROM子句，WHERE  
数据操纵语言DML主要有三种形式：1) 插入：INSERT，2) 更新：UPDATE，3) 删除：DELETE

数据定义语言DDL用来创建数据库中的各种对象-----表、视图、  
索引、同义词、聚簇等如：  
CREATE TABLE/VIEW/INDEX/SYN/CLUSTER

数据控制语言DCL用来授予或回收访问数据库的某种特权，并控制  
数据库操纵事务发生的时间及效果，对数据库实行监视等。如：  
1) GRANT：授权。

2) ROLLBACK [WORK] TO [SAVEPOINT]：回退到某一点。

3) COMMIT [WORK]：提交。

在数据库的插入、删除和修改操作时，只有当事务在提交到数据  
库时才算完成。在事务提交前，只有操作数据库的这个人才能有权看  
到所做的事情，别人只有在最后提交完成后才可以看到。

提交数据有三种类型：显式提交、隐式提交及自动提交。下面分  
别说明这三种类型。

(1) 显式提交  
用COMMIT命令直接完成的提交为显式提交。其格式为：  
SQL>COMMIT；

(2) 隐式提交  
用SQL命令间接完成的提交为隐式提交。这些命令是：  
ALTER，AUDIT，COMMENT，CONNECT，CREATE，DISCONNECT，DROP，  
EXIT，GRANT，NOAUDIT，QUIT，REVOKE，RENAME。

(3) 自动提交  
若把AUTOCOMMIT设置为ON，则在插入、修改、删除语句执行后，  
系统将自动进行提交，这就是自动提交。其格式为：  
SQL>SET AUTOCOMMIT ON；

InnoDB存储引擎提供了具有提交、回滚和崩溃恢复能力的事务安全

索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。更通俗的说，数据库索引好比是一本书前面的目录，能加快数据库的查询速度。

一般数据库默认都会为主键生成索引

索引分为聚簇索引和非聚簇索引两种，聚簇索引是按照数据存放的物理位置为顺序的，而非聚簇索引就不一样了；聚簇索引能提高多行检索的速度，而非聚簇索引对于单行的检索很快。

MySQL索引的类型

**1. 普通索引**

这是最基本的索引，它没有任何限制，比如上文中为title字段创建的索引就是一个普通索引，MyIASM中默认的BTREE类型的索引，也是我们大多数情况下用到的索引。

|  |  |
| --- | --- |
| 01 | –直接创建索引 |
| 02 | CREATE INDEX index\_name ON table(column(length)) |

|  |  |
| --- | --- |
| 03 | –修改表结构的方式添加索引 |
| 04 | ALTER TABLE table\_name ADD INDEX index\_name ON (column(length)) |

|  |  |
| --- | --- |
| 05 | –创建表的时候同时创建索引 |
| 06 | CREATE TABLE `table` ( |

|  |  |
| --- | --- |
| 07 | `id` int(11) NOT NULL AUTO\_INCREMENT , |
| 08 | `title` char(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL , |

|  |  |
| --- | --- |
| 09 | `content` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL , |
| 10 | `time` int(10) NULL DEFAULT NULL , |

|  |  |
| --- | --- |
| 11 | PRIMARY KEY (`id`), |
| 12 | INDEX index\_name (title(length)) |

|  |  |
| --- | --- |
| 13 | ) |
| 14 | –删除索引 |

|  |  |
| --- | --- |
| 15 | DROP INDEX index\_name ON table |

**2. 唯一索引**

与普通索引类似，不同的就是：索引列的值必须唯一，但允许有空值（注意和主键不同）。如果是组合索引，则列值的组合必须唯一，创建方法和普通索引类似。

|  |  |
| --- | --- |
| 01 | –创建唯一索引 |
| 02 | CREATE UNIQUE INDEX indexName ON table(column(length)) |

|  |  |
| --- | --- |
| 03 | –修改表结构 |
| 04 | ALTER TABLE table\_name ADD UNIQUE indexName ON (column(length)) |

|  |  |
| --- | --- |
| 05 | –创建表的时候直接指定 |
| 06 | CREATE TABLE `table` ( |

|  |  |
| --- | --- |
| 07 | `id` int(11) NOT NULL AUTO\_INCREMENT , |
| 08 | `title` char(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL , |

|  |  |
| --- | --- |
| 09 | `content` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL , |
| 10 | `time` int(10) NULL DEFAULT NULL , |

|  |  |
| --- | --- |
| 11 | PRIMARY KEY (`id`), |
| 12 | UNIQUE indexName (title(length)) |

|  |  |
| --- | --- |
| 13 | ); |

**3. 全文索引（FULLTEXT）**

MySQL从3.23.23版开始支持全文索引和全文检索，FULLTEXT索引仅可用于 MyISAM 表；他们可以从CHAR、VARCHAR或TEXT列中作为CREATE TABLE语句的一部分被创建，或是随后使用ALTER TABLE 或CREATE INDEX被添加。////对于较大的数据集，将你的资料输入一个没有FULLTEXT索引的表中，然后创建索引，其速度比把资料输入现有FULLTEXT索引的速度更为快。不过切记对于大容量的数据表，生成全文索引是一个非常消耗时间非常消耗硬盘空间的做法。

|  |  |
| --- | --- |
| 01 | –创建表的适合添加全文索引 |
| 02 | CREATE TABLE `table` ( |

|  |  |
| --- | --- |
| 03 | `id` int(11) NOT NULL AUTO\_INCREMENT , |
| 04 | `title` char(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL , |

|  |  |
| --- | --- |
| 05 | `content` text CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL , |
| 06 | `time` int(10) NULL DEFAULT NULL , |

|  |  |
| --- | --- |
| 07 | PRIMARY KEY (`id`), |
| 08 | FULLTEXT (content) |

|  |  |
| --- | --- |
| 09 | ); |
| 10 | –修改表结构添加全文索引 |

|  |  |
| --- | --- |
| 11 | ALTER TABLE article ADD FULLTEXT index\_content(content) |
| 12 | –直接创建索引 |

|  |  |
| --- | --- |
| 13 | CREATE FULLTEXT INDEX index\_content ON article(content) |

**4. 单列索引、多列索引**

多个单列索引与单个多列索引的查询效果不同，因为执行查询时，MySQL只能使用一个索引，会从多个索引中选择一个限制最为严格的索引。

MySQL索引的优化

上面都在说使用索引的好处，但过多的使用索引将会造成滥用。因此索引也会有它的缺点：虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表进行INSERT、UPDATE和DELETE。因为更新表时，MySQL不仅要保存数据，还要保存一下索引文件。建立索引会占用磁盘空间的索引文件。一般情况这个问题不太严重，但如果你在一个大表上创建了多种组合索引，索引文件的会膨胀很快。索引只是提高效率的一个因素，如果你的MySQL有[**大数据**](http://lib.csdn.net/base/20)量的表，就需要花时间研究建立最优秀的索引，或优化查询语句。下面是一些总结以及收藏的MySQL索引的注意事项和优化方法。

事实上，我们可以通过前面聚集索引和非聚集索引的定义的例子来理解上表。如：返回某范围内的数据一项。比如您的某个表有一个时间列，恰好您把聚合索引建立在了该列，这时您查询2004年1月1日至2004年10月1日之间的全部数据时，这个速度就将是很快的，因为您的这本字典正文是按日期进行排序的，聚类索引只需要找到要检索的所有数据中的开头和结尾数据即可；而不像非聚集索引，

**视图**

视图就是一个存在于数据库中的虚拟表。

视图本身没有数据，只是通过执行相应的select语句完成获得相应的数据。

视图是从一个或几个基本表（或视图）导出的表。它与基本表不同，是一个虚表。[**数据库**](http://lib.csdn.net/base/14)只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。所以基本表中的数据发生变化，从视图中查询出的数据也就随之改变了。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。  
既然视图的定义是基于基本表的，哪为什么还要定义视图呢？这是因为合理地使用视图能够带来许多好处：  
1、 视图能简化用户操作  
    视图机制使用户可以将注意力集中在所关心地数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，并且可以简化用户的的数据查询操作。例如，那些定义了若干张表连接的视图，就将表与表之间的连接操作对用户隐藏起来了。换句话说，用户所作的只是对一个虚表的简单查询，而这个虚表是怎样得来的，用户无需了解。  
2、 视图使用户能以多种角度看待同一数据  
    视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户共享同一个数据库时，这种灵活性是非常必要的。  
3、 视图对重构数据库提供了一定程度的逻辑独立性  
    数据的物理独立性是指用户的应用程序不依赖于数据库的物理结构。数据的逻辑独立性是指当数据库重构造时，如增加新的关系或对原有的关系增加新的字段，用户的应用程序不会受影响。层次数据库和网状数据库一般能较好地支持数据的物理独立性，而对于逻辑独立性则不能完全的支持。  
在关许数据库中，数据库的重构造往往是不可避免的。重构数据库最常见的是将一个基本表“垂直”地分成多个基本表。例如：将学生关系Student（Sno，Sname，Ssex，Sage，Sdept），  
分为SX（Sno，Sname，Sage）和SY（Sno，Ssex，Sdept）两个关系。这时原表Student为SX表和SY表自然连接的结果。如果建立一个视图Student：  
CREATE VIEW Student（Sno，Sname，Ssex，Sage，Sdept）  
AS  
SELECT SX.Sno，SX.Sname，SY.Ssex，SX.Sage，SY.Sdept  
FROM SX，SY  
WHERE SX.Sno=SY.Sno；  
    这样尽管数据库的逻辑结构改变了（变为SX和SY两个表了），但应用程序不必修改，因为新建立的视图定义为用户原来的关系，使用户的外模式保持不变，用户的应用程序通过视图仍然能够查找数据。  
当然，视图只能在一定程度上提供数据的逻辑独立，比如由于视图的更新是有条件的，因此应用程序中修改数据的语句可能仍会因为基本表构造的改变而改变。  
4、视图能够对机密数据提供安全保护  
    有了视图机制，就可以在设计数据库应用系统时，对不同的用户定义不同的视图，使机密数据不出现在不应该看到这些数据的用户视图上。这样视图机制就自动提供了对机密数据的安全保护功能。例如，Student表涉及全校15个院系学生数据，可以在其上定义15个视图，每个视图只包含一个院系的学生数据，并只允许每个院系的主任查询和修改本原系学生视图。  
5、适当的利用视图可以更清晰地表达查询  
    例如经常需要执行这样的查询“对每个学生找出他获得最高成绩的课程号”。可以先定义一个视图，求出每个同学获得的最高成绩：  
CREATE VIEW VMGRADE  
AS  
SELECT Sno，MAX(Grade) Mgrade  
FROM SC  
GROUP BY Sno；  
然后用如下的查询语句完成查询：  
SELECT SC.Sno，Cno FROM SC，VMGRADE WHERE SC.Sno = VMGRADE.Sno AND SC.Grade = VMGRADE.Mgrade；

存储过程

**【存储过程的概念】**

存储过程（**Stored Procedure**）是一组**为了完成特定功能的SQL语句集**。经编译后存储在[**数据库**](http://lib.csdn.net/base/14)中。

存储过程是数据库中的一个重要对象，**用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它**。

存储过程是由 **流控制**和 **SQL语句**书写的过程，**这个过程经编译和优化后存储在数据库服务器中**。

存储过程 **可由应用程序通过一个调用来执行，而且允许用户声明变量。**

同时，存储过程**可以接收和输出参数、返回执行存储过程的状态值，也可以嵌套调用**。

**【存储过程的功能】**

这类语言主要提供以下功能，让用户可以设计出符合引用需求的程序：

① 变量说明

② ANSI兼容的SQL命令(如Select,Update….)

③ 一般流程控制命令(if…else…、while….)

④ 内部函数

**【存储过程的优点】**

① 存储过程的能力**大大增强了SQL语言的功能和灵活性**。

存储过程可以用流控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的 运算。

② 可**保证数据的安全性和完整性**。

理由1：通过存储过程可以使没有权限的用户在控制之下间接地存取数据库，从而保证数据的安全。

理由2：通过存储过程可以使相关的动作在一起发生，从而可以维护数据库的完整性。

③ 在运行存储过程前，数据库已对其进行了语法和句法分析，并给出了优化执行方案。

这种已经编译好的过程**可极大地改善SQL语句的性能**。

**由于执行SQL语句的大部分工作已经完成，所以存储过程能以极快的速度执行**。

④ 可以降低网络的通信量。

理由：**客户端调用存储过程只需要传存储过程名和相关参数即可**，与传输SQL语句相比自然数据量少了很多。

⑤ 使体现企业规则的运算程序放入数据库服务器中，以便：

1). 集中控制。

2). 当企业规则发生变化时在服务器中改变存储过程即可，无须修改任何应用程序。

**企业规则的特点是要经常变化，如果把体现企业规则的运算程序放入应用程序中，则当企业规则发生变化时，就需要修改应用程序工作量非常之大（修改、发行和安装应用程序）。**

如果把体现企业规则的运算放入存储过程中，则当企业规则发生变化时，只要修改存储过程就可以了，应用程序无须任何变化。

**【存储过程优点概括】**

① **存储过程只在创造时进行编译，以后每次执行存储过程都不需再重新编译，而一般SQL语句每执行一次就编译一次,所以使用存储过程可提高数据库执行速度。**

② 当对数据库进行复杂操作时(如对多个表进行Update,Insert,Query,Delete时)，**可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用**。

③ 存储过程**可以重复使用**,可减少数据库开发人员的工作量。

④ 安全性高,**可设定只有某些用户才具有对指定存储过程的使用权**

**【存储过程缺点】**

① 调试麻烦，但是用 PL/SQL Developer 调试很方便！弥补这个缺点。

② 移植问题，数据库端代码当然是与数据库相关的。但是如果是做工程型项目，基本不存在移植问题。

③ 重新编译问题，因为后端代码是运行前编译的，如果带有引用关系的对象发生改变时，受影响的存储过程、包将需要重新编译（不过也可以设置成运行时刻自动编译）。

④ 如果在一个程序系统中大量的使用存储过程，到程序交付使用的时候随着用户需求的增加会导致数据结构的变化，接着就是系统的相关问题了，最后如果用户想维护该系统可以说是很难很难、而且代价是空前的，维护起来更麻烦。

**【存储过程的种类】**

① 系统存储过程：以sp\_开头,用来进行系统的各项设定.取得信息.相关管理工作。

② 本地存储过程：用户创建的存储过程是由用户创建并完成某一特定功能的存储过程，事实上一般所说的存储过程就是指本地存储过程。

③ 临时存储过程：分为两种存储过程：

一是本地临时存储过程，以井字号(#)作为其名称的第一个字符，则该存储过程将成为一个存放在tempdb数据库中的本地临时存储过程，且只有创建它的用户才能执行它;

二是全局临时存储过程，以两个井字号(##)号开始，则该存储过程将成为一个存储在tempdb数据库中的全局临时存储过程，全局临时存储过程一旦创建，以后连接到服务器的任意用户都可以执行它，而且不需要特定的权限。

④ 远程存储过程：在SQL Server2005中，远程存储过程(Remote Stored Procedures)是位于远程服务器上的存储过程，通常可以使用分布式查询和EXECUTE命令执行一个远程存储过程。

⑤ 扩展存储过程：扩展存储过程(Extended Stored Procedures)是用户可以使用外部程序语言编写的存储过程，而且扩展存储过程的名称通常以xp\_开头。

**【Oracle 数据库的一个分页存储过程示例】**

**create or replace procedure fenye(**

tableName in varchar2,--*表名*pageSize in number,--*1页显示的记录数*pageNow in number,--*当前为第几页*myrows out number, --*总记录数(一定要小心代码中不能有Oracle的保留字，保留字是以青色显示的)*myPageCount out number,--*总页数*p\_cursor out testpackage.test\_cursor--*返回的记录集***) is***--定义部分  
--定义一个sql语句字符串*v\_sql varchar2(1000);  
*--定义两个整数*v\_begin number**:=**(pageNow-1)\*pageSize+1;  
v\_end number**:=**pageNow\*pageSize;  
**begin***--执行部分(排序的话在最内层加条件)*v\_sql**:=**'select \* from (select t1.\*,rownum rn from (select \* from '|| tableName  
   ||' order by sal) t1 where rownum<='|| v\_end ||') where rn>='|| v\_begin;  
*--把游标和sql关联***open** p\_cursor **for**v\_sql;  
*--要计算myrows和myPageCount  
--组织一个sql*v\_sql**:=**'select count(\*) from '|| tableName;  
*--执行sql，并把返回的值，赋给myrows***execute immediate** v\_sql **into**myrows**;***--计算myPageCount***if** mod(myrows,pageSize)=0 **then**  myPageCount**:=**myrows/pageSize;  
**else**  myPageCount**:=**myrows/pageSize+1;  
**end if;***--关闭游标(关了的话运行*[***java***](http://lib.csdn.net/base/17)*程序竟然报游标已经关闭的错误！)  
--close p\_cursor;***end;**

**【java调用存储过程的示例】**

**[java]** [view plain](http://blog.csdn.net/yang3wei/article/details/6281255) [copy](http://blog.csdn.net/yang3wei/article/details/6281255)

1. **package org.bruce.test.oracle.procedure;**
3. **import java.sql.CallableStatement;**
4. **import java.sql.Connection;**
5. **import java.sql.DriverManager;**
6. **import java.sql.ResultSet;**
7. **import java.sql.SQLException;**
9. **public class TestOracleProc6 {**
10. **public static void main(String[] args) {**
11. **Connection ct = null;**
12. **CallableStatement cs = null;**
13. **int pageNow = 1;**
14. **try {**
15. **Class.forName("oracle.jdbc.driver.OracleDriver");   // 1.加载驱动**
16. **ct = DriverManager.getConnection("jdbc:oracle:thin:@127.0.0.1:1521:MYORA1","SCOTT","m123");**
17. **cs = ct.prepareCall("{call my\_fenye(?,?,?,?,?,?)}");**
19. **cs.setString(1, "emp"); // 取emp表**
20. **cs.setInt(2, 5);    // 每页取 5 条记录**
21. **cs.setInt(3, pageNow);  // 当前页为第 pageNow 页**
22. **cs.registerOutParameter(4, oracle.jdbc.OracleTypes.INTEGER); // 注册总记录数**
23. **cs.registerOutParameter(5, oracle.jdbc.OracleTypes.INTEGER); // 注册总页数**
24. **cs.registerOutParameter(6, oracle.jdbc.OracleTypes.CURSOR); // 注册返回的结果集**
26. **cs.execute();**
28. **int rowNum = cs.getInt(4); // 取出总记录数(getInt(4)中4，是由该参数的位置决定的**
29. **int pageCount = cs.getInt(5);**
30. **ResultSet rs = (ResultSet)cs.getObject(6);**
32. **System.out.println("总共有多少行记录：" + rowNum);**
33. **System.out.println("总共有多少页：" + pageCount);**
34. **System.out.println("当前为多少页：" + pageNow);**
35. **while(rs.next()) {**
36. **System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getFloat(6));**
37. **}**
38. **} catch(Exception e) {**
39. **e.printStackTrace();**
40. **} finally {**
41. **try { // 关闭各个打开的资源...**
42. **if(cs != null) {**
43. **cs.close();**
44. **}**
45. **if(ct != null) {**
46. **ct.close();**
47. **}**
48. **} catch (SQLException e) {**
49. **e.printStackTrace();**
50. **}**
51. **}**
52. **}**
53. **}**

**常用SQL语句**

**1、说明：创建数据库**  
**CREATE DATABASE database-name**   
**2、说明：删除数据库  
drop database dbname**  
**4、说明：创建新表**

**根据已有的表创建新表：**  
**A：create table tab\_new like tab\_old (使用旧表创建新表)**  
**B：create table tab\_new as select col1,col2… from tab\_old definition only**  
**5、说明：删除新表  
drop table tabname**   
**6、说明：增加一个列  
Alter table tabname add column col type**  
**注**：列增加后将不能删除。DB2中列加上后数据类型也不能改变，唯一能改变的是增加varchar类型的长度。  
**7、说明：添加主键**： **Alter table tabname add primary key(col)**说明：删除主键：**Alter table tabname drop primary key(col)**  
**8、说明：创建索引**：**create [unique] index idxname on tabname(col….)**删除索引：**drop index idxname**  
注：索引是不可更改的，想更改必须删除重新建。  
**9、说明：创建视图**：create view viewname as select statement   
**删除视图**：drop view viewname  
**10、说明：几个简单的基本的sql语句  
选择：**select \* from table1 where 范围  
**插入：**insert into table1(field1,field2) values(value1,value2)  
**删除：**delete from table1 where 范围 **更新**：update table1 set field1=value1 where 范围  
**查找**：select \* from table1 where field1 like ’%value1%’ ---like的语法很精妙，查资料!  
**排序**：select \* from table1 order by field1,field2 [desc]  
**总数**：select count as totalcount from table1  
**求和**：select sum(field1) as sumvalue from table1  
**平均**：select avg(field1) as avgvalue from table1  
**最大**：select max(field1) as maxvalue from table1  
**最小**：select min(field1) as minvalue from table1  
**11、说明：几个高级查询运算词  
A： UNION 运算符**   
UNION 运算符通过组合其他两个结果表（例如 TABLE1 和 TABLE2）并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。   
**B： EXCEPT 运算符   
EXCEPT**运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时 (EXCEPT ALL)，不消除重复行。  
**C： INTERSECT 运算符  
INTERSECT**运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 **ALL**随 INTERSECT 一起使用时 (INTERSECT ALL)，不消除重复行。   
**注：**使用运算词的几个查询结果行必须是一致的。   
**12、说明：使用外连接**A、**left （outer） join**：   
左外连接（左连接）：结果集几包括连接表的匹配行，也包括左连接表的所有行。   
SQL: select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c  
**B：right （outer） join:**  
右外连接(右连接)：结果集既包括连接表的匹配连接行，也包括右连接表的所有行。   
**C：full/cross （outer） join**：   
全外连接：不仅包括符号连接表的匹配行，还包括两个连接表中的所有记录。  
**12、分组:Group by:**一张表，一旦分组 完成后，查询后只能得到组相关的信息。  
    **组相关的信息：**（统计信息） count,sum,max,min,avg **分组的标准)  
    在SQLServer中分组时：不能以text,ntext,image类型的字段作为分组依据  
   在selecte统计函数中的字段，不能和普通的字段放在一起；**

**13、对数据库进行操作：  
   分离数据库**：**sp\_detach\_db;附加数据库**：**sp\_attach\_db**后接表明，附加需要完整的路径名  
**14.如何修改数据库的名称:**  
sp\_renamedb 'old\_name', 'new\_name'

**二、提升**

**1、说明：复制表(只复制结构,源表名：a 新表名：b) (Access可用)  
法一：**select \* into b from a where 1<>1（仅用于SQlServer）  
**法二：**select top 0 \* into b from a  
**2、说明：拷贝表(拷贝数据,源表名：a 目标表名：b) (Access可用)**insert into b(a, b, c) select d,e,f from b;

**3、说明：跨数据库之间表的拷贝(具体数据使用绝对路径) (Access可用)**insert into b(a, b, c) select d,e,f from b in ‘具体数据库’ where 条件  
例子：..from b in '"&Server.MapPath(".")&"\data.mdb" &"' where..

**4、说明：子查询(表名1：a 表名2：b)**select a,b,c from a where a IN (select d from b ) 或者: select a,b,c from a where a IN (1,2,3)

**5、说明：显示文章、提交人和最后回复时间**select a.title,a.username,b.adddate from table a,(select max(adddate) adddate from table where table.title=a.title) b

**6、说明：外连接查询(表名1：a 表名2：b)**select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c

**7、说明：在线视图查询(表名1：a )**select \* from (SELECT a,b,c FROM a) T where t.a > 1;

**8、说明：between的用法,between限制查询数据范围时包括了边界值,not between不包括**select \* from table1 where time between time1 and time2  
select a,b,c, from table1 where a not between 数值1 and 数值2

**9、说明：in 的使用方法**select \* from table1 where a [not] in (‘值1’,’值2’,’值4’,’值6’)

**10、说明：两张关联表，删除主表中已经在副表中没有的信息**delete from table1 where not exists ( select \* from table2 where table1.field1=table2.field1 )

**11、说明：四表联查问题：**select \* from a left inner join b on a.a=b.b right inner join c on a.a=c.c inner join d on a.a=d.d where .....

**12、说明：日程安排提前五分钟提醒**SQL: select \* from 日程安排 where datediff('minute',f开始时间,getdate())>5

**13、说明：一条sql 语句搞定数据库分页**select top 10 b.\* from (select top 20 主键字段,排序字段 from 表名 order by 排序字段 desc) a,表名 b where b.主键字段 = a.主键字段 order by a.排序字段  
**具体实现：**  
关于数据库分页：

  declare @start int,@end int

  @sql  nvarchar(600)

  set @sql=’select top’+str(@end-@start+1)+’+from T where rid not in(select top’+str(@str-1)+’Rid from T where Rid>-1)’

  exec sp\_executesql @sql

**注意：在top后不能直接跟一个变量，所以在实际应用中只有这样的进行特殊的处理。Rid为一个标识列，如果top后还有具体的字段，这样做是非常有好处的。因为这样可以避免 top的字段如果是逻辑索引的，查询的结果后实际表中的不一致（逻辑索引中的数据有可能和数据表中的不一致，而查询时如果处在索引则首先查询索引）**

**14、说明：前10条记录**select top 10 \* form table1 where 范围

**15、说明：选择在每一组b值相同的数据中对应的a最大的记录的所有信息(类似这样的用法可以用于论坛每月排行榜,每月热销产品分析,按科目成绩排名,等等.)**select a,b,c from tablename ta where a=(select max(a) from tablename tb where tb.b=ta.b)

**16、说明：包括所有在 TableA中但不在 TableB和TableC中的行并消除所有重复行而派生出一个结果表**(select a from tableA ) except (select a from tableB) except (select a from tableC)

**17、说明：随机取出10条数据**select top 10 \* from tablename order by **newid()**

**18、说明：随机选择记录**select newid()

**19、说明：删除重复记录  
1),**delete from tablename where id not in (select max(id) from tablename group by col1,col2,...)  
**2)**,select distinct \* into temp from tablename  
  delete from tablename  
  insert into tablename select \* from temp  
**评价： 这种操作牵连大量的数据的移动，这种做法不适合大容量但数据操作  
3),例如：在一个外部表中导入数据，由于某些原因第一次只导入了一部分，但很难判断具体位置，这样只有在下一次全部导入，这样也就产生好多重复的字段，怎样删除重复字段**

alter table tablename  
--添加一个自增列  
add  column\_b int identity(1,1)  
 delete from tablename where column\_b not in(  
select max(column\_b)  from tablename group by column1,column2,...)  
alter table tablename drop column column\_b

**20、说明：列出数据库里所有的表名**select name from sysobjects where type='U' // U代表用户

**21、说明：列出表里的所有的列名**select name from syscolumns where id=object\_id('TableName')

**22、说明：列示type、vender、pcs字段，以type字段排列，case可以方便地实现多重选择，类似select 中的case。**select type,sum(case vender when 'A' then pcs else 0 end),sum(case vender when 'C' then pcs else 0 end),sum(case vender when 'B' then pcs else 0 end) FROM tablename group by type  
**显示结果：  
type vender pcs**电脑 A 1  
电脑 A 1  
光盘 B 2  
光盘 A 2  
手机 B 3  
手机 C 3

**23、说明：初始化表table1**

TRUNCATE TABLE table1

**24、说明：选择从10到15的记录**select top 5 \* from (select top 15 \* from table order by id asc) table\_别名 order by id desc