# DataBase\_Need to recite everyday

## ORACLE QUICK REFERENCE

* SQL> set linesize 120
* SQL> set pagesize 30
* SQL> set serveroutput on
* SQL>select \* from tab; //show all table include views
* SQL>desc emp;
* SQL> SELECT DISTINCT OWNER FROM ALL\_OBJECTS
* SQL> select tablespace\_name, table\_name from user\_tables ///too much
* SQL〉select name from v$database ？？？not work
* SQL> select tablespace\_name from user\_tablespaces; //not what I want see
* SQL>Conn / as sysdba
* select username from v$session where username is not null;

## MySQL QUICK REFERENCE

Fast command in mysql:

# >Net start MySQL

# **>NET STOP MySQL**

>mysql -u root –p

{

Laptop:Pc-pc

Open service > sugarMysql -> start

>mysql –uroot –ppass

}

c:\mysql\bin\mysqladmin-u root -p shutdown

D:\MySQL Server 5.6\bin\mysqld –install

D:\MySQL Server 5.6\bin\net start mysql

If error 1067 go check and fix my.ini

Retry

D:\MySQL Server 5.6\bin\net start mysql

mysql> desc mytab

If you are already running [**mysql**](http://dev.mysql.com/doc/refman/5.0/en/mysql.html), you can execute an SQL script file using the **source**command or **\.** command:

mysql> **source *file\_name***

mysql> **\. *file\_name***

# MySQL> create database elltest;

Mysql> create table mytab(

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(30) NOT NULL,

password VARCHAR(32) NOT NULL,

age INT NOT NULL

);

Mysql> SHOW DATABASES;

Mysql> insert into orderss values(2,'tester');

# mysql> mysql -D samp\_db -u root -p < createtable.sql

## MongoDB QUICK REFERENCE

Server cmd console

c:\>cd c:\

c:\>mkdir data

c:\>cd data

c:\data>mkdir db

c:\data>cd db

c:\data\db>

C:\app\MongoDB\Server\3.2\bin\mongod.exe --dbpath c:\data\db

Client cmd console

> mongo

MongoDB shell version: 3.0.6

connecting to: test

……

由于它是一个JavaScript shell，您可以运行一些简单的算术运算:

> 2 + 2

4

>

**db** 命令用于查看当前操作的文档（数据库）：

> db

test

>

> use runoob

switched to db runoob

> db

runoob

>

如果你想查看所有数据库，可以使用 **show dbs** 命令：

> show dbs

local 0.078GB

test 0.078GB

>

可以看到，我们刚创建的数据库 runoob 并不在数据库的列表中， 要显示它，我们需要向 runoob 数据库插入一些数据。

> db.runoob.insert({"name":"菜鸟教程"})

(--display) --- WriteResult({ "nInserted" : 1 })

> show dbs

local 0.078GB

runoob 0.078GB

test 0.078GB

>

## 

## sqlserver、Mysql、Oracle三种数据库的优缺点总结

这篇文章主要介绍了sqlserver、Mysql、Oracle三种数据库的优缺点总结,需要的朋友可以参考下

**一、sqlserver**  
优点：  
易用性、适合分布式组织的可伸缩性、用于决策支持的数据仓库功能、与许多其他服务器软件紧密关联的集成性、良好的性价比等；  
为 数据管理与分析带来了灵活性，允许单位在快速变化的环境中从容响应，从而获得竞争优势。从数据管理和分析角度看，将原始数据转化为商业智能和充分利用 Web带来的机会非常重要。作为一个完备的数据库和数据分析包，SQLServer为快速开发新一代企业级商业应用程序、为企业赢得核心竞争优势打开了胜 利之门。作为重要的基准测试可伸缩性和速度奖的记录保持者，SQLServer是一个具备完全Web支持的数据库产品，提供了对可扩展标记语言 (XML)的核心支持以及在Internet上和防火墙外进行查询的能力；  
  
缺点：  
开放性 ：SQL Server 只能windows上运行没有丝毫开放性操作系统系统稳定对数据库十分重要Windows9X系列产品偏重于桌面应用NT server只适合小型企业而且windows平台靠性安全性和伸缩性非常有限象unix样久经考验尤其处理大数据库；  
伸缩性并行性 ：SQL server 并行实施和共存模型并成熟难处理日益增多用户数和数据卷伸缩性有限；  
安全性：没有获得任何安全证书。  
性能 ：SQL Server 多用户时性能佳 ；  
客户端支持及应用模式： 客户端支持及应用模式。只支持C/S模式，SQL Server C/S结构只支持windows客户用ADO、DAO、OLEDB、ODBC连接；  
使用风险：SQL server 完全重写代码经历了长期测试断延迟许多功能需要时间来证明并十分兼容；  
**二、Oracle**  
优点：  
开放性：Oracle 能所有主流平台上运行（包括 windows）完全支持所有工业标准采用完全开放策略使客户选择适合解决方案对开发商全力支持；  
可伸缩性,并行性：Oracle 并行服务器通过使组结点共享同簇工作来扩展windownt能力提供高用性和高伸缩性簇解决方案windowsNT能满足需要用户把数据库移UNIXOracle并行服务器对各种UNIX平台集群机制都有着相当高集成度；  
安全性：获得最高认证级别的ISO标准认证。    
性能：Oracle 性能高 保持开放平台下TPC-D和TPC-C世界记录；  
客户端支持及应用模式：Oracle 多层次网络计算支持多种工业标准用ODBC、JDBC、OCI等网络客户连接   
使用风险：Oracle 长时间开发经验完全向下兼容得广泛应用地风险低   
  
缺点：  
对硬件的要求很高；  
价格比较昂贵；  
管理维护麻烦一些；  
操作比较复杂，需要技术含量较高；

**三、MySql**  
优点：  
体积小、速度快、总体拥有成本低，开源；  
支持多种操作系统；  
是开源数据库，提供的接口支持多种语言连接操作  
MySql的核心程序采用完全的多线程编程。线程是轻量级的进程，它可以灵活地为用户提供服务，而不过多的系统资源。用多线程和C语言实现的MySql能很容易充分利用CPU；  
MySql有一个非常灵活而且安全的权限和口令系统。当客户与MySql服务器连接时，他们之间所有的口令传送被加密，而且MySql支持主机认证；  
支持ODBC for Windows， 支持所有的ODBC 2.5函数和其他许多函数， 可以用Access连接MySql服务器， 使得应用被扩展；  
支持大型的数据库， 可以方便地支持上千万条记录的数据库。作为一个开放源代码的数据库，可以针对不同的应用进行相应的修改。  
拥有一个非常快速而且稳定的基于线程的内存分配系统，可以持续使用面不必担心其稳定性；   
MySQL 同时提供高度多样性，能够提供很多不同的使用者介面，包括命令行客户端操作，网页浏览器，以及各式各样的程序语言介面，例如 C+，Perl，Java，PHP，以及Python。你可以使用事先包装好的客户端，或者干脆自己写一个合适的应用程序。MySQL可用于 Unix，Windows，以及OS/2等平台，因此它可以用在个人电脑或者是服务器上；  
  
缺点：  
不支持热备份；  
MySQL最大的缺点是其安全系统，主要是复杂而非标准，另外只有到调用mysqladmin来重读用户权限时才发生改变；  
没有一种存储过程(Stored Procedure)语言，这是对习惯于企业级数据库的程序员的最大限制；  
MySQL的价格随平台和安装方式变化。Linux的MySQL如果由用户自己或系统管理员而不是第三方安装则是免费的，第三方案则必须付许可费。Unix或Linux 自行安装 免费 、Unix或Linux 第三方安装 收费；

## T-SQL

1, DDL(creat,alert,drop), DML(select,insert,update,delete)-crud(retrieve,create,update,delete),

DCL(grant,revoke)

2, batch ;/begin…end/if…else/while…begin…end/return/waitfor/goto

3, 自定义变量： declare @v int; set@v=50; select @v=sum(qty) from saleitem where

4,错误处理： raiseerror

Try…catch(only ms sql server)

Print

5, develop and tools: Query Analyzer/Management Studio/osql.exe/sqlcom.exe/

Powerdesigner

Rollback

# **Transact-SQL Formatting Standards (Coding Styles)**

Here’s one other example. This time, the keywords are uppercase, the data types lowercase, and the object names camel case:

IF OBJECT\_ID(‘ProductDocs’, ‘U’) IS NOT NULL

DROP TABLE ProductDocs

GO

CREATE TABLE ProductDocs

(

  DocID int NOT NULL IDENTITY,

  DocTitle nvarchar(50) NOT NULL,

  DocFileName nvarchar(400) NOT NULL,

  CONSTRAINT PK\_ProductDocs\_DocID PRIMARY KEY CLUSTERED (DocID ASC)

)

GO

# **GO (Transact-SQL)**

SQL Server applications can send multiple Transact-SQL statements to an instance of SQL Server for execution as a batch. The statements in the batch are then compiled into a single execution plan. Programmers executing ad hoc statements in the SQL Server utilities, or building scripts of Transact-SQL statements to run through the SQL Server utilities, use GO to signal the end of a batch.

Applications based on the ODBC or OLE DB APIs receive a syntax error if they try to execute a GO command. The SQL Server utilities never send a GO command to the server.

USE AdventureWorks2012;

GO

DECLARE @MyMsg VARCHAR(50)

SELECT @MyMsg = 'Hello, World.'

GO -- @MyMsg is not valid after this GO ends the batch.

-- Yields an error because @MyMsg not declared in this batch.

PRINT @MyMsg

GO

SELECT @@VERSION;

-- Yields an error: Must be EXEC sp\_who if not first statement in

-- batch.

sp\_who

GO

IF OBJECT\_ID(‘ProductDocs’, ‘U’) IS NOT NULL

DROP TABLE ProductDocs

GO

CREATE TABLE ProductDocs

(

   DocID int NOT NULL IDENTITY,

   DocTitle nvarchar(50) NOT NULL,

   DocFileName nvarchar(400) NOT NULL,

   CONSTRAINT PK\_ProductDocs\_DocID PRIMARY KEY CLUSTERED (DocID ASC)

)

GO

------------------------------------

DECLARE @c varchar(10), @d varchar(10)

SET @c = ‘one’

SET @d = ‘two’

IF (@c = @d) OR (@c = ‘two’)

   BEGIN

      PRINT ‘The first condition is correct.’

   END

ELSE

   BEGIN

      PRINT ‘The first condition is incorrect.’

END

**例 25 查询全部学生的学生名和所学课程号及成绩。（连接查询）**

        SELECT SNAME, CNO, GRADE

        FROM   STUDENTS, ENROLLS

        WHERE STUDENTS.SNO ＝ ENROLLS.SNO

**例 40 找出选修了全部课程的学生的姓名。**

        本查询可以改为：查询这样一些学生，没有一门课程是他不选修的。

(是一种思想，一种有效的方法，如果不知‘选全部’是多少，那就‘没有一个不选’)

        SELECT SNAME

        FROM   STUDENTS

        WHERE NOT EXISTS

               (SELECT \*

                FROM COURSES

                WHERE NOT EXISTS

                        (SELECT \*

                         FROM ENROLLS

                         WHERE ENROLLS.SNO ＝ STUDENTS.SNO

                             AND ENROLLS.CNO ＝ COURSES.CNO))

## MySQL

Fast command in mysql:

# >Net start MySQL

# **>NET STOP MySQL**

>mysql -u root –p

c:\mysql\bin\mysqladmin-u root -p shutdown

D:\MySQL Server 5.6\bin\mysqld –install

D:\MySQL Server 5.6\bin\net start mysql

If error 1067 go check and fix my.ini

Retry

D:\MySQL Server 5.6\bin\net start mysql

mysql> desc mytab

If you are already running [**mysql**](http://dev.mysql.com/doc/refman/5.0/en/mysql.html), you can execute an SQL script file using the **source**command or **\.** command:

mysql> **source *file\_name***

mysql> **\. *file\_name***

# MySQL> create database elltest;

# Mysql> create table mytab(

# id INT AUTO\_INCREMENT PRIMARY KEY,

# name VARCHAR(30) NOT NULL,

# password VARCHAR(32) NOT NULL,

# age INT NOT NULL

# );

# Mysql> SHOW DATABASES;

# mysql> show tables

# mysql> mysql -D samp\_db -u root -p < createtable.sql

left join right join inner join

mysql>select \* from emp e right join user u on e.ename=u.name;

**MySQL**:

**ALTER TABLE Customer MODIFY Address char(100);**

select \* from table limit 10

**Oracle**:

**ALTER TABLE Customer MODIFY Address char(100);**

**SQL Server**:

**ALTER TABLE Customer ALTER COLUMN Address char(100);**

SELECT User FROM mysql.user;

create user 'lele'@'localhost' identified by 'some\_pass';

GRANT ALL ON mldn.\* TO 'lele'@'localhost';

revoke select on mldn.\* from 'lele'@'localhost';

|  |
| --- |
| MySQL JDBC Driver and URL Information Listed below are connection examples for MySQL:   **MySQL Connector/J Driver**  **DRIVER CLASS:** com.mysql.jdbc.Driver   **DRIVER LOCATION:** Simply provide the location of the jar file containing  the MySQL JDBC Drivers. These drivers can be obtained from MySQL.  See the MySQL web site for more information.   Example: mysql-connector-java-3.1.11-bin.jar   **JDBC URL FORMAT:** jdbc:mysql://<host>:<port>/<database\_name>   The default port for MySQL is 3306. Usually, if the default port  is being used by the database server, the :<port> value  of the JDBC url can be omitted.   **ADDITIONAL JDBC CONNECTION PROPERTIES**   Additional connection properties can be appended to the JDBC URL.  The format to add connection properties is the following:   jdbc:mysql://<host>:<port>/<database\_name>?property1=value1&property2=value2   Here are examples of connection properties for the MySQL Connector/J driver.  Consult your driver documentation for more examples of connection properties.   **connectTimeout** (Timeout for socket connect (in milliseconds), with 0 being no timeout.  Only works on JDK-1.4 or newer. Defaults to 0.)   **socketTimeout** (Timeout on network socket operations (0, the default means no timeout).)   **autoReconnect** (Should the driver try to re-establish stale and/or dead connections.)   **Examples:**   jdbc:mysql://neptune.acme.com:3306/test   jdbc:mysql://127.0.0.1:3306/test   jdbc:mysql://localhost:3306/mysql?connectTimeout=0&socketTimeout=0&autoReconnect=true |

Regular Expression 7

速记理解技巧[编辑](http://baike.baidu.com/link?url=2MdrQtAL-dZrvKRoo55EEJYbtE_8pxjl6GAfxkp5jlhUORrrJiVqO4ptWVN3BnbXh638K285OIloeF8MJQ_rS_)

|  |  |  |  |
| --- | --- | --- | --- |
| **.** | **[ ]** | **^** | **$** |

四个字符是所有语言都支持的正则表达式，所以这个四个是基础的正则表达式。正则难理解因为里面有一个等价的概念，这个概念大大增加了理解难度，让很多初学者看起来会蒙，如果把等价都恢复成原始写法，自己书写正则就超级简单了，就像说话一样去写你的正则了：

**等价：**  
　　?,\*,+,\d,\w 都是等价字符  
　　？等价于匹配长度{0,1}  
　　\*等价于匹配长度{0,}   
　　+等价于匹配长度{1,}  
　　\d等价于[0-9]  
　　\w等价于[A-Za-z\_0-9]。

**常用运算符与表达式：**^ 开始  
　　（） 域段  
　　[] 包含,默认是一个字符长度  
　　[^] 不包含,默认是一个字符长度  
　　{n,m} 匹配长度   
　　. 任何单个字符(\. 字符点)  
　　| 或  
　　\ 转义  
　　$ 结尾  
　　[A-Z] 26个大写字母  
　　[a-z] 26个小写字母  
　　[0-9] 0至9数字

[A-Za-z0-9] 26个大写字母、26个小写字母和0至9数字  
　　， 分割  
　　.  
　　  
　　**分割语法：**  
　　[A,H,T,W] 包含A或H或T或W字母  
　　[a,h,t,w] 包含a或h或t或w字母  
　　[0,3,6,8] 包含0或3或6或8数字

　　**语法与释义：**基础语法 "^([]{})([]{})([]{})$"  
　　正则字符串 = "开始（[包含内容]{长度}）（[包含内容]{长度}）（[包含内容]{长度}）结束"   
　　  
　　?,\*,+,\d,\w 这些都是简写的,完全可以用[]和{}代替，在(?:)(?=)(?!)(?<=)(?<!)(?i)(\*?)(+?)这种特殊组合情况下除外。  
　　初学者可以忽略?,\*,+,\d,\w一些简写标示符，学会了基础使用再按表自己去等价替换

**实例：**  
　　字符串；<tel:086-0666-88810009999>  
　　原始正则："^tel:[0-9]{1,3}-[0][0-9]{2,3}-[0-9]{8,11}$"   
　　速记理解：开始 "<tel:普通文本>"[0-9数字]{1至3位}"-普通文本"[0数字][0-9数字]{2至3位}"-普通文本"[0-9数字]{8至11位} 结束"  
　　等价简写后正则写法："^tel:\d{1,3}-[0]\d{2,3}-\d{8,11}$" ，简写语法不是所有语言都支持。

PL/SQL也是一种程序语言，叫做过程化SQL语言（Procedural Language/SQL）。PL/SQL是[Oracle数据库](http://baike.baidu.com/view/1685727.htm)对SQL语句的扩展。在普通SQL语句的使用上增加了编程语言的特点，所以PL/SQL就是把数据操作和查询语句组织在PL/SQL代码的过程性单元中，通过逻辑判断、循环等操作实现复杂的功能或者计算的程序语言。

是Oracle对标准[数据库语言](http://baike.baidu.com/view/702712.htm)SQL的过程化扩充，它将数据库技术和[过程化程序设计语言](http://baike.baidu.com/view/495710.htm)联系起来，是一种应用开发语言，可使用循环，分支处理数据，**将SQL的数据操纵功能与过程化语言**[**数据处理**](http://baike.baidu.com/view/51860.htm)**功能结合起来.PL/SQL的使用**，使SQL成为一种高级程序设计语言，支持高级语言的块操作，条件判断，[循环语句](http://baike.baidu.com/view/961969.htm)，嵌套等，与数据库核心的[数据类型](http://baike.baidu.com/view/675645.htm)集成，使SQL 的程序设计效率更高.

⒈1 PL/SQL的作用

使用PL/SQL可以编写具有很多高级功能的程序，虽然通过多个SQL语句可能也能实现同样的功能，但是相比而言，PL/SQL具有更为明显的一些优点：

⒈能够使一组SQL语句的功能更具模块化程序特点；

⒉采用了过程性语言控制程序的结构；

⒊可以对程序中的错误进行自动处理，使程序能够在遇到错误的时候不会被中断；

⒋具有较好的可移植性，可以移植到另一个[Oracle数据库](http://baike.baidu.com/view/1685727.htm)中；

⒌集成在数据库中，调用更快；

⒍减少了网络的交互，有助于提高程序性能。

通过多条SQL语句实现功能时，每条语句都需要在[客户端](http://baike.baidu.com/view/930.htm)和[服务端](http://baike.baidu.com/view/1087294.htm)传递，而且每条语句的执行结果也需要在网络中进行交互，占用了大量的网络带宽，消耗了大量网络传递的时间，而在网络中传输的那些结果，往往都是中间结果，而不是我们所关心的。

而使用PL/SQL程序是因为程序代码存储在数据库中，程序的分析和执行完全在数据库内部进行，用户所需要做的就是在[客户端](http://baike.baidu.com/view/930.htm)发出调用PL/SQL的执行命令，数据库接收到执行命令后，在数据库内部完成整个PL/SQL程序的执行，并将最终的执行结果返馈给用户。在整个过程中网络里只传输了很少的数据，减少了[网络传输](http://baike.baidu.com/view/1542295.htm)占用的时间，所以整体程序的执行性能会有明显的提高。

⒈2 PL/SQL程序的基本结构

PL/SQL块由四个基本部分组成：声明、执行体开始、[异常处理](http://baike.baidu.com/view/1072586.htm)、执行体结束。

下面是四个部分的基本结构：

DECLARE —— 可选部分

[变量](http://baike.baidu.com/view/296689.htm)、[常量](http://baike.baidu.com/view/346799.htm)、[游标](http://baike.baidu.com/view/176618.htm)、用户定义异常的声明

……

BEGIN —— 必要部分

SQL语句和PL/SQL语句构成的执行程序

……

EXCEPTION —— 可选部分

程序出现异常时，捕捉异常并处理异常

……

END；—— 必须部分

在数据库执行PL/SQL程序时，PL/SQL语句和SQL语句是分别进行解析和执行的。PL/SQL块被数据库内部的PL/SQL引擎提取，将SQL语句取出送给Oracle的SQL引擎处理，两种语句分别在两种引擎中分析处理，在数据库内部完成数据交互、处理过程。

## SQL server

例 1 对于表的教学管理数据库中的表 STUDENTS ，可以定义如下：

   CREATE TABLE STUDENTS

   (SNO      NUMERIC (6, 0) NOT NULL,

   SNAME    CHAR (8) NOT NULL ,

   AGE      NUMERIC(3,0),

   SEX      CHAR(2) ,

   BPLACE CHAR(20),

   PRIMARY KEY(SNO))

例 2 对于表的教学管理数据库中的表 ENROLLS ，可以定义如下：

        CREATE TABLE ENROLLS

        (SNO      NUMERIC(6,0) NOT NULL ,

        CNO     CHAR(4) NOT NULL ,

        GRADE   INT ,

        PRIMARY KEY(SNO,CNO) ,

        FOREIGN KEY(SNO) REFERENCES STUDENTS(SNO) ,

        FOREIGN KEY(CNO) REFERENCES COURSES(CNO) ,

        CHECK ((GRADE IS NULL) OR (GRADE BETWEEN 0 AND 100)))

例 3 根据表的 STUDENTS 表，建立一个只包含学号、姓名、年龄的女学生表。

        CREATE TABLE GIRL

        AS SELECT SNO, SNAME, AGE

        FROM STUDENTS

       WHERE SEX=' female ';

例 4 删除教师表 TEACHER 。

        DROP TABLE TEACHER

例 5 在教师表中增加住址列。

       ALTER TABLE TEACHERS

       ADD (ADDR CHAR(50))

例 6 把 STUDENTS 表中的 BPLACE 列删除，并且把引用 BPLACE 列的所有视图和约束也一起删除。

        ALTER TABLE STUDENTS

        DROP BPLACE CASCADE

例 7 补充定义 ENROLLS 表的主关键字。

       ALTER TABLE ENROLLS

       ADD PRIMARY KEY (SNO,CNO) ；

视图操作（虚表）

例 9 建立一个只包括教师号、姓名和年龄的视图 FACULTY 。 ( 在视图定义中不能包含 ORDER BY 子句 )

        CREATE VIEW FACULTY

        AS SELECT TNO, TNAME, AGE

        FROM TEACHERS

例 10 从学生表、课程表和选课表中产生一个视图 GRADE\_TABLE ， 它包括学生姓名、课程名和成绩。

        CREATE VIEW GRADE\_TABLE

        AS SELECT SNAME,CNAME,GRADE

        FROM STUDENTS,COURSES,ENROLLS

        WHERE STUDENTS.SNO ＝ ENROLLS.SNO AND

        COURSES.CNO=ENROLLS.CNO

例 11 删除视图 GRADE\_TABLE

        DROP VIEW GRADE\_TABLE RESTRICT

索引操作

例 12 在学生表中按学号建立索引。

        CREATE  UNIQUE  INDEX  ST

        ON STUDENTS (SNO,ASC) ? errors

CREATE  UNIQUE  INDEX  ST ON STUDENTS (SNO) --this work

例 13 删除按学号所建立的索引。

        DROP INDEX ST on students

数据库模式操作

例 14 创建一个简易教学数据库的数据库模式  TEACHING\_DB ，属主为 ZHANG 。

        CREATE SCHEMA TEACHING\_DB  AUTHRIZATION  ZHANG

例 15 删除简易教学数据库模式 TEACHING\_DB 。（（ 1 ）选用 CASCADE ，即当删除数据库模式时，则本数据库模式和其下属的基本表、视图、索引等全部被删除。（ 2 ）选用 RESTRICT ，即本数据库模式下属的基本表、视图、索引等事先已清除，才能删除本数据库模式，否则拒绝删除。）

        DROP SCHEMA TEACHING\_DB CASCADE

单表操作

例 16 找出 3 个学分的课程号和课程名。

         SELECT CNO, CNAME

         FROM 　 COURSES

         WHERE 　 CREDIT ＝ 3

例 17 查询年龄大于 22 岁的学生情况。

         SELECT \*

         FROM   STUDENTS

         WHERE AGE ＞ 22

例 18   找出籍贯为河北的男生的姓名和年龄。

         SELECT SNAME, AGE

         FROM 　 STUDENTS

         WHERE 　 BPLACE ＝ ' 河北 '  AND SEX ＝ ' 男 '

例 19 找出年龄在 20 ～ 23 岁之间的学生的学号、姓名和年龄，并按年龄升序排序。 (ASC （升序）或 DESC （降序）声明排序的方式，缺省为升序。 )

         SELECT SNO, SNAME, AGE

         FROM   STUDENTS

         WHERE AGE BETWEEN 20 AND 23

         ORDER BY  AGE

例 20 找出年龄小于 23 岁、籍贯是湖南或湖北的学生的姓名和性别。（条件比较运算符＝、＜ 和逻辑运算符 AND （与），此外还可以使用的运算符有：＞（大于）、＞＝（大于等于）、＜＝（小于等于）、＜＞（不等于）、 NOT （非）、 OR （或）等。

谓词 LIKE 只能与字符串联用，常常是 “ ＜列名＞  LIKE pattern” 的格式。特殊字符 “\_” 和 “%” 作为通配符。

谓词 IN 表示指定的属性应与后面的集合（括号中的值集或某个查询子句的结果）中的某个值相匹配，实际上是一系列的 OR （或）的缩写。谓词 NOT IN 表示指定的属性不与后面的集合中的某个值相匹配。

谓词 BETWEEN 是 “ 包含于 … 之中 ” 的意思。）

        SELECT SNAME, SEX

        FROM   STUDENTS

        WHERE AGE ＜ 23 AND BPLACE LIKE' 湖％ '

        或

        SELECT SNAME, SEX

        FROM   STUDENTS

        WHERE AGE ＜ 23 AND BPLACE IN （ ' 湖南 ' ， ' 湖北 ' ）

例 22 找出学生表中籍贯是空值的学生的姓名和性别。（在 SQL 中不能使用条件：＜列名＞＝ NULL 。在 SQL 中只有一个特殊的查询条件允许查询 NULL 值：）

       SELECT SNAME, SEX

       FROM   STUDENTS

       WHERE BPLACE IS NULL

多表操作

例 23 找出成绩为 95 分的学生的姓名。（子查询）

        SELECT SNAME

        FROM 　 STUDENTS

        WHERE 　 SNO ＝

              (SELECT SNO

               FROM   ENROLLS

               WHERE GRADE ＝ 95)

例 24 找出成绩在 90 分以上的学生的姓名。

       SELECT SNAME

        FROM   STUDENTS

        WHERE SNO IN

                (SELECT SNO

                FROM ENROLLS

                WHERE GRADE ＞ 90)

        或

        SELECT SNAME

        FROM   STUDENTS

        WHERE SNO ＝ ANY

                (SELECT SNO

                FROM ENROLLS

                WHERE GRADE ＞ 90)

例 25 查询全部学生的学生名和所学课程号及成绩。（连接查询）

        SELECT SNAME, CNO, GRADE

        FROM   STUDENTS, ENROLLS

        WHERE STUDENTS.SNO ＝ ENROLLS.SNO

例 26 找出籍贯为山西或河北，成绩为 90 分以上的学生的姓名、籍贯和成绩。（当构造多表连接查询命令时，必须遵循两条规则。第一，连接条件数正好比表数少 1 （若有三个表，就有两个连接条件 ) ；第二，若一个表中的主关键字是由多个列组成，则对此主关键字中的每一个列都要有一个连接条件（也有少数例外情况））

        SELECT SNAME, BPLACE, GRADE

        FROM   STUDENTS, ENROLLS

        WHERE BPLACE IN (‘ 山西 ’ ， ‘ 河北 ’) AND GRADE ＞＝ 90 AND 　 STUDENTS.SNO=ENROLLS.SNO

例 28 查出课程成绩在 80 分以上的女学生的姓名、课程名和成绩。（ FROM 子句中的子查询）

        SELECT SNAME,CNAME, GRADE

        FROM   (SELECT SNAME, CNAME , GRADE

                        FROM STUDENTS, ENROLLS,COURSES

                        WHERE SEX ＝ ' 女 ')

        AS TEMP (SNAME, CNAME,GRADE)

        WHERE GRADE ＞ 80

 ??????????????????????????????????????not right

 This way work as

mysql> select sname,cname,grade from students s,enrolls e,courses c where s.sno=

e.sno and e.cno=c.cno and s.sex='f' and e.grade>=80;

表达式与函数的使用

例 29 查询各课程的学时数。（算术表达式由算术运算符＋、－、 \* 、／与列名或数值常量所组成。）

        SELECT CNAME,COURSE\_TIME ＝ CREDIT\*16

        FROM   COURSES

例 30 找出教师的最小年龄。（内部函数： SQL 标准中只使用 COUNT 、 SUM 、 AVG 、 MAX 、 MIN 函数，称之为聚集函数（ Set Function ）。 COUNT 函数的结果是该列统计值的总数目， SUM 函数求该列统计值之和， AVG 函数求该列统计值之平均值， MAX 函数求该列最大值， MIN 函数求该列最小值。）

        SELECT MIN(AGE)

        FROM   TEACHERS

例 31 统计年龄小于等于 22 岁的学生人数。（统计）

        SELECT COUNT(\*)

        FROM   STUDENTS

        WHERE AGE < ＝ 22

例 32 找出学生的平均成绩和所学课程门数。

        SELECT SNO, AVG(GRADE), COURSES ＝ COUNT(\*)

        FROM   ENROLLS

        GROUP BY SNO

例 34 找出年龄超过平均年龄的学生姓名。

SELECT SNAME

FROM STUDENTS

WHERE AGE ＞

      (SELECT AVG(AGE)

        FROM   STUDENTS)

例 35 找出各课程的平均成绩，按课程号分组，且只选择学生超过 3 人的课程的成绩。（ GROUP BY 与 HAVING

        GROUP BY 子句把一个表按某一指定列（或一些列）上的值相等的原则分组，然后再对每组数据进行规定的操作。

        GROUP BY 子句总是跟在 WHERE 子句后面，当 WHERE 子句缺省时，它跟在 FROM 子句后面。

        HAVING 子句常用于在计算出聚集之后对行的查询进行控制。）

         SELECT CNO, AVG(GRADE), STUDENTS ＝ COUNT(\*)

         FROM ENROLLS

         GROUP BY CNO

         HAVING COUNT(\*) >= 3

相关子查询

例 37 查询没有选任何课程的学生的学号和姓名。（当一个子查询涉及到一个来自外部查询的列时，称为相关子查询（ Correlated Subquery) 。相关子查询要用到存在测试谓词 EXISTS 和 NOT EXISTS ，以及 ALL 、 ANY （ SOME ）等。）

        SELECT SNO, SNAME

        FROM   STUDENTS

        WHERE NOT EXISTS

              (SELECT \*

              FROM ENROLLS

              WHERE ENROLLS.SNO=STUDENTS.SNO)

例 38   查询哪些课程只有男生选读。

        SELECT DISTINCT CNAME

        FROM   COURSES C

        WHERE ' 男 ' ＝ ALL

               (SELECT SEX

                FROM ENROLLS ， STUDENTS

                WHERE ENROLLS.SNO=STUDENTS.SNO AND

                      ENROLLS.CNO=C.CNO)

例 39 要求给出一张学生、籍贯列表，该表中的学生的籍贯省份，也是其他一些学生的籍贯省份。

        SELECT SNAME, BPLACE

        FROM   STUDENTS A

        WHERE EXISTS

               (SELECT \*

                 FROM STUDENTS B

                 WHERE A.BPLACE=B.BPLACE AND

                       A.SNO < > B.SNO)

例 40 找出选修了全部课程的学生的姓名。

        本查询可以改为：查询这样一些学生，没有一门课程是他不选修的。

        SELECT SNAME

        FROM   STUDENTS

        WHERE NOT EXISTS

               (SELECT \*

                FROM COURSES

                WHERE NOT EXISTS

                        (SELECT \*

                         FROM ENROLLS

                         WHERE ENROLLS.SNO ＝ STUDENTS.SNO

                             AND ENROLLS.CNO ＝ COURSES.CNO))

关系代数运算

例 41 设有某商场工作人员的两张表：营业员表 SP\_SUBORD 和营销经理表 SP\_MGR ，其关系数据模式如下：

        SP\_SUBORD (SALPERS\_ID, SALPERS\_NAME, MANAGER\_ID, OFFICE)

        SP\_MGR (SALPERS\_ID, SALPERS\_NAME, MANAGER\_ID, OFFICE)

        其中，属性 SALPERS\_ID 为工作人员的编号 , SALPERS\_NAME 为工作人员的姓名 , MANAGER\_ID 为所在部门经理的编号 , OFFICE 为工作地点。

若查询全部商场工作人员，可以用下面的 SQL 语句：

        (SELECT \* FROM SP\_SUBORD)

        UNION

        (SELECT \* FROM SP\_MGR)

        或等价地用下面的 SQL 语句：

        SELECT \*

        FROM (TABLE SP\_SUBORD UNION TABLE SP\_MGR)

（ 2 ） INTERSECT

         (SELECT \* FROM SP\_SUBORD)

         INTERSECT

         (SELECT \* FROM SP\_MGR)

        或等价地用下面的 SQL 语句：

        SELECT \*

        FROM (TABLE SP\_SUBORD INTERSECT TABLE SP\_MGR)

        或用带 ALL 的 SQL 语句：

        (SELECT \* FROM SP\_SUBORD)

       INTERSECT ALL

        (SELECT \* FROM SP\_MGR)

        或

        SELECT \*

        FROM (TABLE SP\_SUBORD INTERSECT ALL TABLE SP\_MGR)

（ 3 ） EXCEPT

        (SELECT \* FROM SP\_MGR)

        EXCEPT

        (SELECT \* FROM SP\_SUBORD)

        或等价地用下面的 SQL 语句：

        SELECT \*

        FROM (TABLE SP\_MGR EXCEPT TABLE SP\_ SUBORD)

        或用带 ALL 的 SQL 语句：

        (SELECT \* FROM SP\_MGR)

        EXCEPT ALL

        (SELECT \* FROM SP\_SUBORD)

例 42 查询籍贯为四川、课程成绩在 80 分以上的学生信息及其成绩。（自然连接）

        (SELECT \* FROM STUDENTS

         WHERE BPLACE=‘ 四川 ’)

        NATURAL JOIN

        (SELECT \* FROM ENROLLS

         WHERE GRADE >=80)

例3.43          列出全部教师的姓名及其任课的课程号、班级。

（外连接与外部并外连接允许在结果表中保留非匹配元组，空缺部分填以 NULL 。外连接的作用是在做连接操作时避免丢失信息。

        外连接有 3 类：

（ 1 ）左外连接（ Left Outer Join ）。连接运算谓词为 LEFT [OUTER] JOIN ，其结果表中保留左关系的所有元组。

（ 2 ）右外连接（ Right Outer Join ）。连接运算谓词为 RIGHT [OUTER] JOIN ，其结果表中保留右关系的所有元组。

（ 3 ）全外连接（ Full Outer Join ）。连接运算谓词为 FULL [OUTER] JOIN ，其结果表中保留左右两关系的所有元组。）

          SELECT TNAME, CNO, CLASS

          FROM TEACHERS LEFT OUTER JOIN TEACHING USING (TNO)

SQL 的数据操纵

例 44 把教师李映雪的记录加入到教师表 TEACHERS 中。（插入）

        INSERT INTO TEACHERS

        VALUES(1476 ， ' 李映雪 ' ， 44 ， ' 副教授 ')

例 45 成绩优秀的学生将留下当教师。

        INSERT INTO TEACHERS (TNO ， TNAME)

        SELECT DISTINCT SNO ， SNAME

        FROM   STUDENTS ， ENROLLS

        WHERE STUDENTS.SNO ＝ ENROLLS.SNO AND GRADE ＞＝ 90

例 47 把所有学生的年龄增加一岁。（修改）

        UPDATE STUDENTS

        SET AGE ＝ AGE+1

例 48 学生张春明在数据库课考试中作弊，该课成绩应作零分计。

        UPDATE ENROLLS

        SET GRADE ＝ 0

        WHERE CNO ＝ 'C1' AND

             ' 张春明 ' ＝

             (SELECT SNAME

             FROM STUDENTS

             WHERE STUDENTS.SNO=ENROLLS.SNO)

例 49 从教师表中删除年龄已到 60 岁的退休教师的数据。（删除）

        DELETE FROM TEACHERS

        WHERE AGE ＞＝ 60

SQL 的数据控制

例 50 授予 LILI 有对表 STUDENTS 的查询权。（表／视图特权的授予

        一个 SQL 特权允许一个被授权者在给定的数据库对象上进行特定的操作。授权操作的数据库对象包括：表 / 视图、列、域等。授权的操作包括： INSERT 、 UPDATE 、 DELETE 、 SELECT 、 REFERENCES 、 TRIGGER 、 UNDER 、 USAGE 、 EXECUTE 等。其中 INSERT 、 UPDATE 、 DELETE 、 SELECT 、 REFERENCES 、 TRIGGER 有对表做相应操作的权限，故称为表特权。）

        GRANT SELECT ON STUDENTS

        TO LILI

        WITH GRANT OPTION

例 51 取消 LILI 的存取 STUDENTS 表的特权。

        REVOKE ALL

        ON STUDENTS

        FROM LILI CASCADE

不断补充中：  
1.  模糊查找:  
它判断列值是否与指定的字符串格式相匹配。可用于char、varchar、text、ntext、datetime和smalldatetime等类型查询。   
　　可使用以下通配字符：   
　　百分号%：可匹配任意类型和长度的字符，如果是中文，请使用两个百分号即%%。   
　　下划线\_：匹配单个任意字符，它常用来限制表达式的字符长度。   
　　方括号[]：指定一个字符、字符串或范围，要求所匹配对象为它们中的任一个。[^]：其取值也[] 相同，但它要求所匹配对象为指定字符以外的任一个字符。   
　　 例如：   
 　　限制以Publishing结尾，使用LIKE '%Publishing'   
　　限制以A开头：LIKE '[A]%'   
　　限制以A开头外：LIKE '[^A]%'  
  
2.更改表格　  
         ALTER TABLE table\_name    
        ADD COLUMN column\_name DATATYPE    
        说明：增加一个栏位（没有删除某个栏位的语法。)   
        ALTER TABLE table\_name   
        ADD PRIMARY KEY (column\_name)   
        说明：更改表得的定义把某个栏位设为主键。   
        ALTER TABLE table\_name   
        DROP PRIMARY KEY (column\_name)   
        说明：把主键的定义删除。

3.group by

在select 语句中可以使用group by 子句将行划分成较小的组，然后，使用聚组函数返回每一个组的汇总信息，另外，可以使用having子句限制返回的结果集。group by 子句可以将查询结果分组，并返回行的汇总信息Oracle 按照group by 子句中指定的表达式的值分组查询结果。

在带有group by 子句的查询语句中，在select 列表中指定的列要么是group by 子句中指定的列，要么包含聚组函数

select max(sal),job emp group by job;  
(注意max(sal),job的job并非一定要出现，但有意义)

查询语句的select 和group by ,having 子句是聚组函数唯一出现的地方，在where 子句中不能使用聚组函数。

select deptno,sum(sal) from emp where sal>1200 group by deptno having sum(sal)>8500 order by deptno;

当在gropu by 子句中使用having 子句时，查询结果中只返回满足having条件的组。在一个sql语句中可以有where子句和having子句。having 与where 子句类似，均用于设置限定条件  
  
where 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，条件中不能包含聚组函数，使用where条件显示特定的行。  
having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件显示特定的组，也可以使用多个分组标准进行分组。

查询每个部门的每种职位的雇员数  
select deptno,job,count(\*) from emp group by deptno,job;

4.外连接与内连接

有时候，即使在连接的表中没有相应的行，用户可能想从一张表中看数据，Oracle提供了外连接实现该功能。  
内连接是指连接查询只显示完全满足连接条件的记录，即等值连接，外连接的查询结果是内连接查询结果的扩展。外连接不仅返回满足连接条件的所有记录而且也返回了一个表中那些在另一个表中没有匹配行的记录。外连接的操作符是“+”。“+”号放在连接条件中信息不完全的那一边（即没有相应行的那一边）。运算符“+”影响NULL行的建立。建一行或多行NULL来匹配连接的表中信息完全的行。

外连接运算符“+”只能出现在where子句中表达式的一边。

假如在多张表之间有多个连接条件，外连接运算符不能使用or,in逻辑运算符与其它条件组合。

假如emp表中deptno=10的ename为空值，dept表中deptno=20的loc为空值：

1.

select   
ename,dept.deptno,loc   
from   
emp,dept   
where   
emp.depno(+)=dept.deptno;

如果在dept.deptno中有的数值在emp.deptno中没有值，则在做外连接时，  
结果中ename会产生一个空值。(emp.deptno=10)

2.

select   
ename,dept.deptno,loc   
from   
emp,dept   
where   
emp.depno=dept.deptno(+);

如果在emp.deptno中有的数值在dept.deptno中没有值，则在做外连接时，  
结果中loc会产生一个空值。。(dept.deptno=20)

5.自连接

自连接是指同一张表的不同行间的连接。该连接不受其他表的影响。用自连接可以比较同一张表中不同行的某一列的值。因为自连接查询仅涉及到某一张表与其自身的连接。所以在from子句中该表名出现两次，分别用两个不同的别名表示，两个别名当作两张不同的表进行处理，与其它的表连接一样，别名之间也使用一个或多个相关的列连接。为了区分同一张表的不同行的列，在名前永别名加以限制。

select   
worker.ename,  
manager.ename manager   
from   
emp worker,  
emp manager  
where  
work.mgr=manager.empno;

6.集合运算

基合运算符可以用于从多张表中选择数据。

①UNION运算  
用于求两个结果集合的并集（两个结果集合的所有记录），并自动去掉重复行。

select ename,sal from account where sal>2000  
union  
select ename,sal from research where sal>2000  
union  
select ename,sal from sales where sal>2000;

注：ename,sal 是必须一致的。   
  
②UNION ALL运算  
用于求两个结果集合的并集（两个结果集中的所有记录），并且不去掉重复行。

select ename,sal from account where sal>2000  
union  
select ename,sal from research where sal>2000  
union  
select ename,sal from sales where sal>2000;

③INTERSECT运算  
intersect运算返回查询结果中相同的部分。  
? not work in mysql  
各部门中有哪些相同的职位？

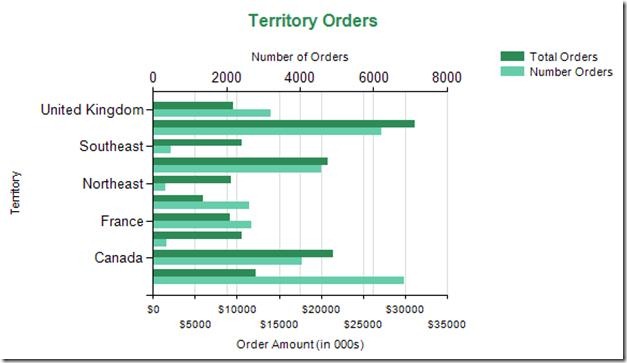
select Job from account  
intersect   
select Job from research  
intersect  
select Job from sales;

④MINUS运算  
minus返回两个结果集的差集。（在第一个结果集中存在的，而在第二个结果集中不存在的行。）

有那些职位是财务部中有，而在销售部门中没有？

select Job from account  
minus  
select Job from sales;

## SSRS

[](http://images.cnblogs.com/cnblogs_com/CareySon/201202/201202291535162002.jpg)

    为了完成我们的图表报表,打开BIDS,创建一个新的报表项目后创建报表。针对AdventureWorks数据库写如下查询语句.

[select](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=select&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) t.Name [as](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=as&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) TerritoryName

, [SUM](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=SUM&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99)(o.TotalDue) [as](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=as&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) TotalOrders

, [COUNT](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=COUNT&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99)(\*) [as](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=as&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) NumberOrders

[from](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=from&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) Sales.SalesOrderHeader o

[inner](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=inner&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) [join](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=join&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) Sales.Customer c

[on](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=on&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) o.CustomerID=c.CustomerID

[inner](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=inner&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) [join](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=join&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) Sales.SalesTerritory t

[on](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=on&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) c.TerritoryID=t.TerritoryID

[**group**](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=group&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99)[**by**](http://search.microsoft.com/default.asp?so=RECCNT&siteid=us%2Fdev&p=1&nq=NEW&qu=by&IntlSearch=&boolean=PHRASE&ig=01&i=09&i=99) **t.Name**

/\*

declare @ytempdb1 nvarchar(50)

declare @ytempdb2 nvarchar(50)

declare @ytempdb3 nvarchar(50)

\*/

declare @ytempdb1 nvarchar(50),@ytempdb2 nvarchar(50),@ytempdb3 nvarchar(50)

set @ytempdb1='www.ytempdb.com'

print @ytempdb1

select @ytempdb1='www.enet.com.cn'

print @ytempdb1

select @ytempdb1='www.enet.com.cn',@ytempdb2='www.cdbaba.com.cn',@ytempdb3='www.sian.com.cn'

print @ytempdb1

print @ytempdb2

print @ytempdb3

insert into students values('2211','tim','2014-01-01 00:00:00.000','89 warden')

print @@identity

print @@servername

use YTEMPDB

declare @myname nvarchar(50)

set @myname='tim'

select \* from students where name=@myname and id=4

/\*0.000000025 2.5e-8

\*/

/\* 1

declare @a int

declare @b int

declare @c int \*/

declare @a nvarchar(10)

declare @b nvarchar(10)

declare @c nvarchar(10)

set @a=1111

set @b=2222

set @c=@a+@b

print @c

Go -- Go 分隔后， 变量失效

print @c

/\*

result

1,3333

2,11112222

\*/

if exists (select \* from dbo.sysobjects where name='teachers')

drop table teachers

go

create table teachers

(

id int identity(201201,1),--自动编号

name nvarchar(10),

age int,

EnTime datetime,

IsMarry bit,

tel nvarchar(10),

)

--表变量

declare @myTable table(id int, studnum nvarchar(20), Name nvarchar(20))

insert into @myTable select id, Studnum, Name from students

select \* from @myTable

--自定义数据类型

exec sp\_addtype newChar, 'char(4)','not null'

--判断语句

declare @a int

declare @b int

declare @c int

set @a=1111

set @b=5555

set @c=3333

declare @temp int

if (@a>@b)

set @temp=@a

else

set @temp=@b

if (@temp>@c)

print @temp

else

print @c

Go

--case 条件语句

select \*,sex,sex=

case

when sex='True' then 'nan'

when sex='False' then 'nu'

end

from students

--循环语句

declare @mysum int

declare @i int

set @i=1

set @mysum=0

while (@i<101)

begin

set @mysum=@mysum+@i

set @i=@i+1

if(@i=5)

-- break -- 退出循环

continue --退出本次循环

print @i

end

print @mysum

go

declare @a int

select @a=100

flag1:

print @a

select @a=@a+1

while @a<105 goto flag1

print '-----------------'

print @a

-- c22 局部临时表 会话级别 只存在于存储过程的创建时,

create table #temp\_stu

(

num nvarchar(20),

name nvarchar(20)

)

insert into #temp\_stu values('20120202','servly')

select \* from #temp\_stu

go

-- c23 全局临时表

create table ##temp\_stu2

(

num nvarchar(20),

name nvarchar(20)

)

insert into ##temp\_stu2 values('20120202','servly')

select \* from ##temp\_stu2

use YTEMPDB

drop table ##temp\_stu2

select studnum, name into #temp\_stu3 from students

select \* from #temp\_stu3

列别名

select studnum as number,name as ming ,sex as xingbie from students

-- c26,27 列计算，总分，平均分

select studnum,name, chinese as yuwen, math as shuxue, english as yingyu, chinese+math+english as total, (chinese+math+english)/3 as average from students

--c28 排序

select studnum,name, chinese as yuwen, math as shuxue, english as yingyu, chinese+math+english as total, (chinese+math+english)/3 as average from students order by total desc

--c29 连接列值

select \*, name1+name2 as xingming from teachers

select \* from students where age<20 and id=2

select \* from students where age<>20

select \* from students where age between 18 and 20

select \* from students where age not between 18 and 20

select \* from students where entertime between '2011-3-3' and '2013-4-4'

go

--运算符

select \* from students where age in(18,19,22)

select \* from students where name in('tim','leen')

select \* from students where name not in('tim','leen')

select \* from students where classname='class1' and sex is null

select top 3 studnum,name, chinese as yuwen, math as shuxue, english as yingyu, chinese+math+english as total, (chinese+math+english)/3 as average from students order by total asc

--随机查询

select top 3 \* from students order by newid()

--发现重复记录

select distinct studnum,name,sex from students where classname='class1'order by name

go

--显示行号

select columnnum=identity(int,1,1), studnum,name,sex into #temprownum1 from students

select \* from #temprownum1

go

select \* from News where Classid in (select ID from Newsclass where ParentID=1)

go

--选取第3行到地8行数据 (8-3)+1=6

select \* from (select top 6 \* from (select top 8 \* from students order by ID asc ) as students1 order by ID desc) as students2 order by ID asc

select \* from students

go

--模糊查询 通配符

select \* from students where Name like 't%'

select \* from students where Name like '%n'

select \* from students where Name like 'l%n'

select \* from students where Name like 't%[^m}'

select \* from students where studnum like '552\_' --? \_不工作

select \* from students where EnName like '[f-l]%'-- c to r 之间

select \* from students where EnName like '[flm]%' -- f,l,m 开头

select \* from students where EnName like '%[nyu]' -- nyu结尾

select \* from students where EnName like '%[^nyu]' -- 不是nyu结尾

select \* from students where age like '[^1-1]\_' -- 开头不为1

select \* from students where age like '[^1-2]\_' -- 开头不为1

select \* from students where age like '[1-2]\_' -- 十位为1，2， 个位任意

select \* from students where EnName like '%100%' -- 100

select \* from students where EnName like '%100/%%' escape '/' -- 100%

select \* from students where (EnName+Name+classname like '%[ty1]%') -- 多个字段的匹配查询

go

--c52,53 类型转换 cast,convert

use YTEMPDB

select \*, english2=

case

when english is NULL then 'no score'

when english is not NULL then cast(english as Nvarchar(20))

end

from students

go

--SQL convert style 日期格式

select Convert(nvarchar(20),GETDATE(),106) as testdate

select convert(datetime, '2012-04-04 07:07:00') as timetostring

--去除尾部的空格

select rtrim(' helllll lo world ') as newtitle

select ltrim(' helllll lo world ') as newtitle

go

--substring

select SUBSTRING('I will well verse the SQL skill',3,4) as title

--stuff 插入字符

select STUFF('hello world',6,3,'sea')

select STUFF('hello world',7,0,'sea')

select STUFF('hello world',1,0,'sea')

go

declare @yyjcw nvarchar(50)

set @yyjcw='hello world'

select STUFF(@yyjcw,1,0,'sea ')

select len(@yyjcw) as titlelongth

--大小写， 客户端与服务器，验证码对比

select lower('asdasFGGGdfa') as title

select upper('asdasFGGGdfa') as title

select \*, upper(Enname) as newnamee from students

--去掉特殊符号

select replace('sdfsdsf love asdfs love sfdasdf love','love','heart')

go

select charindex('-','0281-88888888') as pos

select substring('0281-88888888',charindex('-','0281-88888888')+1,100) as newtitle

select substring('0281-88888888',0,charindex('-','0281-88888888')) as newtitle

go

--data function

select getdate() as mytime

select year(getdate()) as myyear

select month(getdate()) as mymonth

select day(getdate()) as myday

select Convert(nvarchar(20),getdate(),8) as mytime

select datepart(hour,getdate()) as myhour

select datepart(hh,getdate()) as myhour

select datepart(minute,getdate()) as myminuts

select datepart(second,getdate()) as mysecond

select datepart(millisecond, getdate()) as mymillisecond

----

select datename(DW,getdate()) as whatday

select datename(ww,getdate()) as whichweekinoneyear

select datename(ww,'2012-05-05') as whichweekinoneyear

select datename(YY,'2012-05-05') as Myear

select datename(M,'2012-05-05') as MYMONTH

select datename(D,'2012-05-07') as MYDAY

select datename(DY,'2012-05-07') as MYDAYOFYEAR

select datediff(d,'2012-05-06','2012-05-08') as daydifferent

select datediff(hh,'2012-05-06 12:00:00','2012-05-08 13:00:00') as hourdifferent

select entertime, datediff(d,entertime,getdate()) as enrolldays from students

--

delete from students where datediff(yy,entertime,getdate())>3

select entertime, dateadd(hh,2,dateadd(d,3,entertime)) as newentertime from students

select entertime, dateadd(hh,2,dateadd(yy,3,entertime)) as newentertime from students

go

--动态排序

declare @myorder int

set @myorder=2

select \* from students order by case @myorder

when 1 then chinese

when 2 then english

when 3 then math

end

asc

--c69 聚合函数，多行函数，组合函数, 适当的索引可加快聚合函数的运行

--ignore NULL,saft to use

use YTEMPDB

select sum(chinese) as chinese from students

select avg(math) as math from students

select count(\*) from students where classname='class1'

select count(distinct(classname)) from students

select min(age) from students

select max(age) from students

--\*\*\*聚合查询的结果作为子查询的条件进行筛选

select \* from students where age=(select max(age) from students)

select classname, count(classname) as classstudentnum from students group by classname

--\*\*\*where 单条记录 having 多条记录 聚合函数

select classname, count(classname) as classstudentnum from students group by classname having count(classname)>2

select classname, count(classname) as classstudentnum from students group by classname having count(classname)>2 order by classstudentnum desc

go

--insert

insert into NewsClass (ParentID,ClassName,[Description]) values (1,'IT news','IT news')

insert into NewsClass values (1,'IT news222','IT news')

insert into NewsClass (ClassName) values ('IT news')

insert into NewsClass values (1,'IT news222',NULL)

--design setup default

insert into NewsClass (ParentID,ClassName) values (1,'IT news')

go

create table cust

(

id int,

name nvarchar(50),

tel nvarchar(50),

addre nvarchar(50),

descriptions nvarchar(100) not null default('no yet'),

)

insert into cust(id,name,tel,addre) values(1,'lele','12121212121','one squere')

--批量插入数据

insert into cust(id,name,tel,addre)

select 1,'lele','12121212121','one squere'

union all select 2,'lele2','12121212121','one squere'

union all select 3,'lele3','12121212121','one squere'

CREATE TABLE [dbo].[goodstudents](

[id] [int] IDENTITY(1,1) NOT NULL,

[studnum] [nvarchar](50) NULL,

[name] [nvarchar](50) NULL,

[age] [int] NULL,

[classname] [nvarchar](50) NOT NULL

CONSTRAINT [PK\_goodstudents] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

insert into goodstudents select studnum,name,age,classname from students where chinese>80

go

use YTEMPDB

update NewsClass set ParentID=1,[Description]='IT news 333' where id=12 --keyword 加中括号

update NewsClass set ClassName=ClassName+'1' where parentID=1

update News set Title=Title+'nation news 'where Classid in (select ID from NewsClass where ParentID=2)

Go

update students set Grade=case

when (chinese+math+english)/3<60 then 'no pass'

when (chinese+math+english)/3>=60 and (chinese+math+english)/3<80 then 'pass'

when (chinese+math+english)/3>=80 then 'excellent'

end

go

--delete

delete from students where ID=1005

go

--保留ID大的，其他删去

Delete from goodstudents where ID NOT in(select MAX(id) as id from goodstudents group by studnum)

go

--删除完全重复的行

select distinct \* into #tempgood from goodstudents

delete from goodstudents

insert into goodstudents select \* from #tempgood

go

--游标 游标变量

declare cur\_stu cursor for

select \* from students

--for read only

for update of studnum

declare @stu cursor

set @stu=cur\_stu --游标变量

open @stu

fetch next from @stu

while @@FETCH\_STATUS=0

begin

fetch next from @stu

end

close @stu

deallocate @stu

go

----sample2

declare @\_StuNum nvarchar(50)

declare @\_Age int

declare cur\_stu cursor for

select studnum,age from students

--for read only

for update of studnum

declare @stu cursor

set @stu=cur\_stu --游标变量

open @stu

fetch next from @stu into @\_StuNum,@\_Age

print @\_StuNum

print @\_Age

while @@FETCH\_STATUS=0

begin

fetch next from @stu into @\_StuNum,@\_Age

print @\_StuNum

print @\_Age

end

close @stu

deallocate @stu

---存储过程

--创建

create Procedure GetStuInf

as

begin

select \* from students

end

--修改

ALTER Procedure [dbo].[GetStuInf]

as

begin

select \* from students where classname='class1'

end

---

create procedure [dbo].[GetStuInfoByClassName]

@className nvarchar(50)

as

begin

select \* from students where classname=@className

end

---

ALTER procedure [dbo].[GetStuInfoByClassName]

@className nvarchar(50),

@age int

as

begin

select \* from students where classname=@className and age>@age

end

-- 传出参数 output

create procedure GetOutInfo

@ClassName nvarchar(50),

@Age int,

@maxscore int output,

@avgscore int output,

@stucount int output

as

begin

select \* from students where classname=@ClassName and age>@Age

select @maxscore=MAX(chinese) from students where classname=@ClassName and age>@Age

select @avgscore=avg(chinese) from students where classname=@ClassName and age>@Age

select @stucount=count(\*) from students where classname=@ClassName and age>@Age

end

declare @max\_score int

declare @avg\_score int

declare @count\_stu int

exec GetOutInfo 'class1',18, @max\_score output,@avg\_score output,@count\_stu output

select 'maxscore'=@max\_score,'avg'=@avg\_score,'num'=@count\_stu

--存储过程返回值

create procedure GetReturnValue

@ClassName nvarchar(50),

@Age int

as

begin

declare @maxscore int

select @maxscore=MAX(chinese) from students where classname=@ClassName and age>@Age

return @maxscore

end

declare @myReturnValue int

exec @myReturnValue=GetReturnValue'class1',18

select 'returevalue'=@myReturnValue

--

exec sp\_rename 'getreturevalue','getReturnValue'

exec sp\_monitor

go

---函数 不能执行update insert

------标量值函数

create function myAdd(@a int,@b int)

returns int

as

begin

declare @c int

set @c=@a+@b

return @c

end

declare @a1 int

declare @b1 int

declare @c1 int

set @a1=1

set @b1=5

set @c1=ytempdb.dbo.myAdd(@a1,@b1)

print cast(@c1 as nvarchar(10))

---表值函数

create function GetClassMember(@classname nvarchar(50))

returns table

as

return(select \* from students where classname=@classname)

select \* from GetClassMember('class1')

go

drop function GetClassMember

---触发器 特殊的存储过程，3种触发器 insert update delete 事件

use YTEMPDB

create trigger TriggerDeleteNewsclass

on NewsClass

after delete

as

delete News from News, deleted where News.classid=deleted.ID

go

use YTEMPDB

create trigger TriggerUpdateNewsClassCountsAfterInsert

on News

after insert

as

begin

declare @totalcount int --表变量里面总共有的记录条数

declare @rownum int --当前是多少行

declare @tempid int --当前栏目的id

declare @tempcounts int --当前栏目文章数

declare @temptb1 table(indexid int identity(1,1) not NULL, id int)

insert into @temptb1 select id from NewsClass

set @totalcount=@@ROWCOUNT

set @rownum=1

while @rownum<=@totalcount

begin

select @tempid=id from @temptb1 where indexid=@rownum

select @tempcounts=count(\*) from news where classid=@tempid

update NewsClass set NewsCounts=@tempcounts where id=@tempid

set @rownum=@rownum+1

end

delete from newsclass where id=16 --触发另一个触发器

end

go

---updata trigger嵌套

create trigger triggerUpdateNewsClassCountAfterUpdate

on News

after update

as

begin

declare @totalcount int --表变量里面总共有的记录条数

declare @rownum int --当前是多少行

declare @tempid int --当前栏目的id

declare @tempcounts int --当前栏目文章数

declare @temptb1 table(indexid int identity(1,1) not NULL, id int)

insert into @temptb1 select id from NewsClass

set @totalcount=@@ROWCOUNT

set @rownum=1

while @rownum<=@totalcount

begin

select @tempid=id from @temptb1 where indexid=@rownum

select @tempcounts=count(\*) from news where classid=@tempid

update NewsClass set NewsCounts=@tempcounts where id=@tempid

set @rownum=@rownum+1

end

end

--触发器递归 a--a\* SQL default ferbident

--not work exec sp\_'ytempdb','recursive triggers',true

--trigger have 2 special table: deleted inserted

---事物 like bat sp, in one event all transaction must all run or all fail

begin try

begin transaction

insert into teachers(name1,name2,age)values('mike','liu',37)

insert into students(name,age,classname)values('dodo',15,'class2')

commit transaction

end try

begin catch

rollback

end catch

go

begin transaction

insert into teachers(name1,name2,age)values('mike','liu',37)

insert into students(name,age,classname)values('dodo',15,'class2')

save transaction save1

insert into teachers(name1,name2,age)values('mike2','liu2',32)

insert into students(name,age,classname)values('dodo2',19,'class1')

rollback transaction save1

commit transaction

go

--????-并发控制 concurrence control

begin tran

select \* from News with(holdlock)

waitfor delay '00:00:10'

commit tran

begin tran

update News set classid=3 where ID=1004

select \* from News

commit tran

--view

--view's view

--数据的完整性 约束

--标志 主键约束

create table tbadmin

(

id int constraint pk\_id primary key,

adminloginname nvarchar(20),

adminpassword nvarchar(20),

adminGrade int,

adminMeno nvarchar(50)

)

alter table teachers

add constraint pk\_id1

primary key(id)

alter table teachers

--drop constraint pk\_id1

add constraint pk\_id3

primary key(tel)

go

--外键约束

alter table News

add constraint fk\_Classid

foreign key(Classid)

references Newsclass(id)

go

create table stuInfo

(

stuNum int primary key,

stuName nvarchar(10),

sex char(2),

ClassName nvarchar(20)

)

create table stuScore

(

stuNum int foreign key references stuInfo(stuNum),

chinese float,

english float,

Math float,

)

GO

--唯一约束

alter table Cust

add constraint uq\_Cust

unique(IDcard)

go

--check constraint 检查约束

alter table cust with check

add constraint ck\_Cust

check(tel like '010-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

go

alter table cust

add constraint df\_cust

default 'no commens' for addre

## MySQL与Oracle的语法区别详细对比

Oracle和mysql的一些简单命令对比在本文中将会涉及到很多的实例，感兴趣的你不妨学习一下，就当巩固自己的知识了

Oracle和mysql的一些简单命令对比   
1)　SQL> select to\_char(sysdate,'yyyy-mm-dd') from dual;   
　　SQL> select to\_char(sysdate,'hh24-mi-ss') from dual;   
　　mysql> select date\_format(now(),'%Y-%m-%d');   
　　mysql> select time\_format(now(),'%H-%i-%S');   
　　日期函数   
　　增加一个月：   
　　SQL> select to\_char(add\_months(to\_date ('20000101','yyyymmdd'),1),'yyyy-mm-dd') from dual;   
　　结果：2000-02-01   
　　SQL> select to\_char(add\_months(to\_date('20000101','yyyymmdd'),5),'yyyy-mm-dd') from dual;   
　　结果：2000-06-01   
　　mysql> select date\_add('2000-01-01',interval 1 month);   
　　结果：2000-02-01   
　　mysql> select date\_add('2000-01-01',interval 5 month);   
　　结果：2000-06-01   
　　截取字符串：   
　　SQL> select substr('abcdefg',1,5) from dual;   
　　SQL> select substrb('abcdefg',1,5) from dual;   
　　结果：abcdemysql> select substring('abcdefg',2,3);   
　　结果：bcd   
　　mysql> select mid('abcdefg',2,3);   
　　结果：bcd   
　　mysql> select substring('abcdefg',2);   
　　结果：bcdefg   
　　mysql> select substring('abcdefg' from 2);   
　　结果：bcdefg   
2) 在MySQL中from 后的表如果是(select.......)这种，那么后面必须有别名   
3) 连接字符串在Oracle中用|| ,SqlServer中用+,MySQL中用concat('a','b','c')   
  
4)   
在SqlServer中的写法：

复制代码 代码如下:

declare @id varchar(50);   
set @id='4028e4962c3df257012c3df3b4850001';   
select \* from sims\_sample\_detect where ID= @id;

在MySQL中的写法：

复制代码 代码如下:

set @a = 189;   
select \* from bc\_article where id = @a //不用declare

在Orcale中的写法：   
  
5)MySQL存储过程：

复制代码 代码如下:

DELIMITER $$   
DROP PROCEDURE IF EXISTS `SIMS`.`transaction\_delSampleInfo`$$   
CREATE DEFINER=`root`@`%` PROCEDURE `transaction\_delSampleInfo`(in sampleInfoId varchar(50))   
BEGIN   
start transaction;   
update sims\_sample\_info set del='1' where ID = sampleInfoId;   
update sims\_sample\_detect set del='1' where SAMPLE\_ID\_PARENT = sampleInfoId;   
update sims\_sample\_detect\_info set del='1' where DETECT\_ID in(   
select ID from sims\_sample\_detect where SAMPLE\_ID\_PARENT = sampleInfoId   
);   
commit;   
END$$   
DELIMITER ;

变量名不能跟列名相同，否则效果为1=1，且MySQL不区分大小写。   
  
6）mysql 游标   
mysql没有像orcale的动态游标，只有显示游标，例子如下：

复制代码 代码如下:

DELIMITER $$   
DROP PROCEDURE IF EXISTS `test`.`liyukun`$$   
CREATE DEFINER=`ids`@`localhost` PROCEDURE `liyukun`(out z int)   
BEGIN   
declare count1 int;   
DECLARE done INT DEFAULT 0;   
declare v\_haoma varchar(50);   
declare v\_yingyeting varchar(100);   
DECLARE cur1 CURSOR FOR select haoma,yingyeting from eryue where id<2;   
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;   
//这里和oracle有区别，Oracle的PL/SQL的指针有个隐性变   
量%notfound，Mysql是通过一个Error handler的声明来进行判断的   
OPEN cur1;   
cur1: LOOP   
FETCH cur1 INTO v\_haoma,v\_yingyeting;   
IF done=1 THEN //如果没有数据了，则离开   
LEAVE cur1;   
ELSE   
select count(\*) into count1 from year2012 where haoma=v\_haoma ;   
if(count1=0) then   
insert into year2012(haoma, yingyeting)   
values(v\_haoma,v\_yingyeting);   
else   
set z = z+1;   
update year2012 set eryue = ‘100' where haoma=v\_haoma;   
end if;   
END IF;   
END LOOP cur1;   
CLOSE cur1;   
END$$   
DELIMITER ;

执行：

复制代码 代码如下:

call liyukun(@a);   
select @a;

7) mysql的group by 语句可以select 没有被分组的字段，如   
select id,name,age from A group by age 这样   
但是在orcale和sqlserver中是会报错的。这个取出的id,name所在的行是每个分组中的第一行数据。   
8)orcale用decode()来转换数据，mysql,sqlserver用case when:   
case t.DETECT\_RESULT when '2402' then t.SAMPLEID end (必须有end)   
9)mysql: 两个select 出的数据相减：   
(COUNT(distinct(t.SAMPLEID))-   
CONVERT((COUNT(distinct(case t.DETECT\_RESULT when '2402' then t.SAMPLEID end))), SIGNED)) AS NEGATIVE   
FROM `view\_sims\_for\_report` t   
10）convert,cast用法   
mysql将varchar转为int   
convert(字段名, SIGNED)   
字符集转换 : CONVERT(xxx USING gb2312)   
类型转换和SQL Server一样,就是类型参数有点不同 : CAST(xxx AS 类型) , CONVERT(xxx,类型)   
可用的类型　   
二进制,同带binary前缀的效果 : BINARY   
字符型,可带参数 : CHAR()   
日期 : DATE   
时间: TIME   
日期时间型 : DATETIME   
浮点数 : DECIMAL   
整数 : SIGNED   
无符号整数 : UNSIGNED   
11)如果从mysql数据库中取的时候没有乱码，而在Java的List中得到的是乱码的话，那么可能是SQL语句中的有字段不是 varchar的数据类型，这时需要转换convert(字段名, 类型)转换一下，Orcale则用ToChar函数   
12)Orcale的大字段用clob,图像用blob,clob字段在Hibernate的映射文件中用String就可以   
13) mysql,orcale,sqlserver 语句执行顺序   
开始->FROM子句->WHERE子句->GROUP BY子句->HAVING子句->ORDER BY子句->SELECT子句->LIMIT子句->最终结果   
每个子句执行后都会产生一个中间结果 ，供接下来的子句使用，如果不存在某个子句，就跳过。   
14) LPAD函数   
1在oracle的数据库里有个函数 LPAD(String a,int length,String addString).   
2作用：把addString添加到a的左边，length 是返回值的长度。   
3例子

复制代码 代码如下:

A ： SQL> select lpad('test',8,0) from dual;   
LPAD('TEST',8,0)   
----------------   
0000test   
B： select lpad('test',8) from dual;   
LPAD('TEST',8)   
--------------   
test 注：不写最后一个参数，函数会默认在返回值左边加一个空格。   
C： SQL> select lpad('test',2,0) from dual;   
LPAD('TEST',2,0)   
----------------   
te   
D：SQL> select lpad('test',3) from dual;   
LPAD('TEST',3)   
--------------   
tes

15)Orcale中没有TOP，是通过   
select \* from (select \* from A order by id desc) where rownum=1   
注：不能直接写 select \* from A where rownum=1 order by id desc 因为语句执行的顺序是先where再order by ,如果这样写就无法按id的排序来取第一个了。   
不能写rownum=2或rownum>1这样，因为Orcale 默认必须包含第一条。   
如果非要取第二条的话，可以写成：

复制代码 代码如下:

select \* from (select id,rownum as row\_num from lws\_q\_bl\_result r where r.sample\_id = 'B10226072') where row\_num=2

16)Orcale,MySql while循环比较   
Orcale:

复制代码 代码如下:

while num<10   
loop   
str := to\_char(num);   
num := num+1;   
end loop;

也可以：

复制代码 代码如下:

for num in 1..10 --这样的缺陷是无法间隔取值   
loop   
str := to\_char(num);   
end loop;

mysql:

复制代码 代码如下:

while num<10   
do   
str := to\_char(num);   
num := num+1;   
end while;

17)orcale 生成唯一序列是 select sys.guid() from dual ，mysql是 select uuid() from dual   
  
18)MySql和Orcale的ID自增   
MySql由于是在数据库中实现ID自增，所以如果想返回插入一条序列的该条ID，只能用如下方法：

复制代码 代码如下:

public int insertSign(final SpaceSign sign) throws Exception {   
try{   
KeyHolder keyHolder = new GeneratedKeyHolder();   
final String sql = "insert into space\_sign(userId,userName,nickName,contentText,contentHtml,isPublic,commentCount,userIp,status,insertTime)" +   
" values(?,?,?,?,?,?,?,?,?,?)";   
template.update(new PreparedStatementCreator() {   
public PreparedStatement createPreparedStatement(Connection connection) throws SQLException {   
PreparedStatement ps = connection.prepareStatement(sql,Statement.RETURN\_GENERATED\_KEYS);   
ps.setInt(1, sign.getUserId());   
ps.setString(2, sign.getUserName());   
ps.setString(3, sign.getNickName());   
ps.setString(4, sign.getContentText());   
ps.setString(5, sign.getContentHtml());   
ps.setInt(6, sign.getIsPublic());   
ps.setInt(7,sign.getCommnetCount());   
ps.setString(8, sign.getUserIp());   
ps.setInt(9, sign.getStatus());   
ps.setTimestamp(10, new java.sql.Timestamp(sign.getInsertTime().getTime()));   
return ps;   
}   
}, keyHolder);   
Long generatedId = keyHolder.getKey().longValue();   
return generatedId.intValue();   
} catch (Exception e) {   
// TODO Auto-generated catch block   
e.printStackTrace();   
throw new SQLException("发表签名失败", e);   
}   
}

由于Orcale的ID是在插入该条数据之前就通过select SEQ\_BLOG\_ID.nextval from dual 获得的，所以直接返回既可。ps:SEQ\_BLOG\_ID为在数据库中设置的sequence。

# [**一个题目涉及到的50个Sql语句**](http://blog.csdn.net/htl258/article/details/3945715)

 Student(S#,Sname,Sage,Ssex) 学生表   
Course(C#,Cname,T#) 课程表   
SC(S#,C#,score) 成绩表   
Teacher(T#,Tname) 教师表   
  
问题：   
**1**、查询“**001**”课程比“**002**”课程成绩高的所有学生的学号；   
  select a.S# from (select s#,score from SC where C#='001') a,(select s#,score   
  from SC where C#='002') b   
  where a.score>b.score and a.s#=b.s#;   
**2**、查询平均成绩大于60分的同学的学号和平均成绩；   
    select S#,avg(score)   
    from sc   
    group by S# having avg(score) >**60**;   
**3**、查询所有同学的学号、姓名、选课数、总成绩；   
  select Student.S#,Student.Sname,count(SC.C#),sum(score)   
  from Student left Outer join SC on Student.S#=SC.S#   
  group by Student.S#,Sname   
**4**、查询姓“李”的老师的个数；   
  select count(distinct(Tname))   
  from Teacher   
  where Tname like '李%';   
**5**、查询没学过“叶平”老师课的同学的学号、姓名；   
    select Student.S#,Student.Sname   
    from Student    
    where S# not in (select distinct( SC.S#) from SC,Course,Teacher where  SC.C#=Course.C# andTeacher.T#=Course.T# and Teacher.Tname='叶平');   
**6**、查询学过“**001**”并且也学过编号“**002**”课程的同学的学号、姓名；   
  select Student.S#,Student.Sname from Student,SC where Student.S#=SC.S# and SC.C#='001'and exists(Select \* from SC as SC\_2 where SC\_2.S#=SC.S# and SC\_2.C#='002');   
**7**、查询学过“叶平”老师所教的所有课的同学的学号、姓名；   
  select S#,Sname   
  from Student   
  where S# in (select S# from SC ,Course ,Teacher where SC.C#=Course.C# and Teacher.T#=Course.T# andTeacher.Tname='叶平' group by S# having count(SC.C#)=(select count(C#) from Course,Teacher  whereTeacher.T#=Course.T# and Tname='叶平'));   
**8**、查询课程编号“**002**”的成绩比课程编号“**001**”课程低的所有同学的学号、姓名；   
  Select S#,Sname from (select Student.S#,Student.Sname,score ,(select score from SC SC\_2 whereSC\_2.S#=Student.S# and SC\_2.C#='002') score2   
  from Student,SC where Student.S#=SC.S# and C#='001') S\_2 where score2 <score;   
**9**、查询所有课程成绩小于60分的同学的学号、姓名；   
  select S#,Sname   
  from Student   
  where S# not in (select Student.S# from Student,SC where S.S#=SC.S# and score>**60**);   
**10**、查询没有学全所有课的同学的学号、姓名；   
    select Student.S#,Student.Sname   
    from Student,SC   
    where Student.S#=SC.S# group by  Student.S#,Student.Sname having count(C#) <(select count(C#) fromCourse);   
  
**11**、查询至少有一门课与学号为“**1001**”的同学所学相同的同学的学号和姓名；   
    select S#,Sname from Student,SC where Student.S#=SC.S# and C# in select C# from SC where S#='1001';   
**12**、查询至少学过学号为“**001**”同学所有一门课的其他同学学号和姓名；   
    select distinct SC.S#,Sname   
    from Student,SC   
    where Student.S#=SC.S# and C# in (select C# from SC where S#='001');   
**13**、把“SC”表中“叶平”老师教的课的成绩都更改为此课程的平均成绩；   
    update SC set score=(select avg(SC\_2.score)   
    from SC SC\_2   
    where SC\_2.C#=SC.C# ) from Course,Teacher where Course.C#=SC.C# and Course.T#=Teacher.T# andTeacher.Tname='叶平');   
**14**、查询和“**1002**”号的同学学习的课程完全相同的其他同学学号和姓名；   
    select S# from SC where C# in (select C# from SC where S#='1002')   
    group by S# having count(\*)=(select count(\*) from SC where S#='1002');   
**15**、删除学习“叶平”老师课的SC表记录；   
    Delect SC   
    from course ,Teacher    
    where Course.C#=SC.C# and Course.T#= Teacher.T# and Tname='叶平';   
**16**、向SC表中插入一些记录，这些记录要求符合以下条件：没有上过编号“**003**”课程的同学学号、**2**、   
    号课的平均成绩；   
    Insert SC select S#,'002',(Select avg(score)   
    from SC where C#='002') from Student where S# not in (Select S# from SC where C#='002');   
**17**、按平均成绩从高到低显示所有学生的“数据库”、“企业管理”、“英语”三门的课程成绩，按如下形式显示： 学生ID,,数据库,企业管理,英语,有效课程数,有效平均分   
    SELECT S# as 学生ID   
        ,(SELECT score FROM SC WHERE SC.S#=t.S# AND C#='004') AS 数据库   
        ,(SELECT score FROM SC WHERE SC.S#=t.S# AND C#='001') AS 企业管理   
        ,(SELECT score FROM SC WHERE SC.S#=t.S# AND C#='006') AS 英语   
        ,COUNT(\*) AS 有效课程数, AVG(t.score) AS 平均成绩   
    FROM SC AS t   
    GROUP BY S#   
    ORDER BY avg(t.score)    
**18**、查询各科成绩最高和最低的分：以如下形式显示：课程ID，最高分，最低分   
    SELECT L.C# As 课程ID,L.score AS 最高分,R.score AS 最低分   
    FROM SC L ,SC AS R   
    WHERE L.C# = R.C# and   
        L.score = (SELECT MAX(IL.score)   
                      FROM SC AS IL,Student AS IM   
                      WHERE L.C# = IL.C# and IM.S#=IL.S#   
                      GROUP BY IL.C#)   
        AND   
        R.Score = (SELECT MIN(IR.score)   
                      FROM SC AS IR   
                      WHERE R.C# = IR.C#   
                  GROUP BY IR.C#   
                    );   
**19**、按各科平均成绩从低到高和及格率的百分数从高到低顺序   
    SELECT t.C# AS 课程号,max(course.Cname)AS 课程名,isnull(AVG(score),**0**) AS 平均成绩   
        ,**100** \* SUM(CASE WHEN  isnull(score,**0**)>=**60** THEN **1** ELSE **0** END)/COUNT(\*) AS 及格百分数   
    FROM SC T,Course   
    where t.C#=course.C#   
    GROUP BY t.C#   
    ORDER BY **100** \* SUM(CASE WHEN  isnull(score,**0**)>=**60** THEN **1** ELSE **0** END)/COUNT(\*) DESC   
**20**、查询如下课程平均成绩和及格率的百分数(用"1行"显示): 企业管理（**001**），马克思（**002**），OO&UML （**003**），数据库（**004**）   
    SELECT SUM(CASE WHEN C# ='001' THEN score ELSE **0** END)/SUM(CASE C# WHEN '001' THEN **1** ELSE**0** END) AS 企业管理平均分   
        ,**100** \* SUM(CASE WHEN C# = '001' AND score >= **60** THEN **1** ELSE **0** END)/SUM(CASE WHEN C# = '001'THEN **1** ELSE **0** END) AS 企业管理及格百分数   
        ,SUM(CASE WHEN C# = '002' THEN score ELSE **0** END)/SUM(CASE C# WHEN '002' THEN **1** ELSE **0**END) AS 马克思平均分   
        ,**100** \* SUM(CASE WHEN C# = '002' AND score >= **60** THEN **1** ELSE **0** END)/SUM(CASE WHEN C# = '002'THEN **1** ELSE **0** END) AS 马克思及格百分数   
        ,SUM(CASE WHEN C# = '003' THEN score ELSE **0** END)/SUM(CASE C# WHEN '003' THEN **1** ELSE **0**END) AS UML平均分   
        ,**100** \* SUM(CASE WHEN C# = '003' AND score >= **60** THEN **1** ELSE **0** END)/SUM(CASE WHEN C# = '003'THEN **1** ELSE **0** END) AS UML及格百分数   
        ,SUM(CASE WHEN C# = '004' THEN score ELSE **0** END)/SUM(CASE C# WHEN '004' THEN **1** ELSE **0**END) AS 数据库平均分   
        ,**100** \* SUM(CASE WHEN C# = '004' AND score >= **60** THEN **1** ELSE **0** END)/SUM(CASE WHEN C# = '004'THEN **1** ELSE **0** END) AS 数据库及格百分数   
  FROM SC   
**21**、查询不同老师所教不同课程平均分从高到低显示   
  SELECT max(Z.T#) AS 教师ID,MAX(Z.Tname) AS 教师姓名,C.C# AS 课程ＩＤ,MAX(C.Cname) AS 课程名称,AVG(Score) AS 平均成绩   
    FROM SC AS T,Course AS C ,Teacher AS Z   
    where T.C#=C.C# and C.T#=Z.T#   
  GROUP BY C.C#   
  ORDER BY AVG(Score) DESC   
**22**、查询如下课程成绩第 **3** 名到第 **6** 名的学生成绩单：企业管理（**001**），马克思（**002**），UML （**003**），数据库（**004**）   
    [学生ID],[学生姓名],企业管理,马克思,UML,数据库,平均成绩   
    SELECT  DISTINCT top **3**   
      SC.S# As 学生学号,   
        Student.Sname AS 学生姓名 ,   
      T1.score AS 企业管理,   
      T2.score AS 马克思,   
      T3.score AS UML,   
      T4.score AS 数据库,   
      ISNULL(T1.score,**0**) + ISNULL(T2.score,**0**) + ISNULL(T3.score,**0**) + ISNULL(T4.score,**0**) as 总分   
      FROM Student,SC  LEFT JOIN SC AS T1   
                      ON SC.S# = T1.S# AND T1.C# = '001'   
            LEFT JOIN SC AS T2   
                      ON SC.S# = T2.S# AND T2.C# = '002'   
            LEFT JOIN SC AS T3   
                      ON SC.S# = T3.S# AND T3.C# = '003'   
            LEFT JOIN SC AS T4   
                      ON SC.S# = T4.S# AND T4.C# = '004'   
      WHERE student.S#=SC.S# and   
      ISNULL(T1.score,**0**) + ISNULL(T2.score,**0**) + ISNULL(T3.score,**0**) + ISNULL(T4.score,**0**)   
      NOT IN   
      (SELECT   
            DISTINCT   
            TOP **15** WITH TIES   
            ISNULL(T1.score,**0**) + ISNULL(T2.score,**0**) + ISNULL(T3.score,**0**) + ISNULL(T4.score,**0**)   
      FROM sc   
            LEFT JOIN sc AS T1   
                      ON sc.S# = T1.S# AND T1.C# = 'k1'   
            LEFT JOIN sc AS T2   
                      ON sc.S# = T2.S# AND T2.C# = 'k2'   
            LEFT JOIN sc AS T3   
                      ON sc.S# = T3.S# AND T3.C# = 'k3'   
            LEFT JOIN sc AS T4   
                      ON sc.S# = T4.S# AND T4.C# = 'k4'   
      ORDER BY ISNULL(T1.score,**0**) + ISNULL(T2.score,**0**) + ISNULL(T3.score,**0**) + ISNULL(T4.score,**0**) DESC);  
  
**23**、统计列印各科成绩,各分数段人数:课程ID,课程名称,[100-85],[85-70],[70-60],[ <60]   
    SELECT SC.C# as 课程ID, Cname as 课程名称   
        ,SUM(CASE WHEN score BETWEEN **85** AND **100** THEN **1** ELSE **0** END) AS [100 - 85]   
        ,SUM(CASE WHEN score BETWEEN **70** AND **85** THEN **1** ELSE **0** END) AS [85 - 70]   
        ,SUM(CASE WHEN score BETWEEN **60** AND **70** THEN **1** ELSE **0** END) AS [70 - 60]   
        ,SUM(CASE WHEN score < **60** THEN **1** ELSE **0** END) AS [60 -]   
    FROM SC,Course   
    where SC.C#=Course.C#   
    GROUP BY SC.C#,Cname;   
  
**24**、查询学生平均成绩及其名次   
      SELECT **1**+(SELECT COUNT( distinct 平均成绩)   
              FROM (SELECT S#,AVG(score) AS 平均成绩   
                      FROM SC   
                  GROUP BY S#   
                  ) AS T1   
            WHERE 平均成绩 > T2.平均成绩) as 名次,   
      S# as 学生学号,平均成绩   
    FROM (SELECT S#,AVG(score) 平均成绩   
            FROM SC   
        GROUP BY S#   
        ) AS T2   
    ORDER BY 平均成绩 desc;   
    
**25**、查询各科成绩前三名的记录:(不考虑成绩并列情况)   
      SELECT t1.S# as 学生ID,t1.C# as 课程ID,Score as 分数   
      FROM SC t1   
      WHERE score IN (SELECT TOP **3** score   
              FROM SC   
              WHERE t1.C#= C#   
            ORDER BY score DESC   
              )   
      ORDER BY t1.C#;   
**26**、查询每门课程被选修的学生数   
  select c#,count(S#) from sc group by C#;   
**27**、查询出只选修了一门课程的全部学生的学号和姓名   
  select SC.S#,Student.Sname,count(C#) AS 选课数   
  from SC ,Student   
  where SC.S#=Student.S# group by SC.S# ,Student.Sname having count(C#)=**1**;   
**28**、查询男生、女生人数   
    Select count(Ssex) as 男生人数 from Student group by Ssex having Ssex='男';   
    Select count(Ssex) as 女生人数 from Student group by Ssex having Ssex='女'；   
**29**、查询姓“张”的学生名单   
    SELECT Sname FROM Student WHERE Sname like '张%';   
**30**、查询同名同性学生名单，并统计同名人数   
  select Sname,count(\*) from Student group by Sname having  count(\*)>**1**;;   
**31**、1981年出生的学生名单(注：Student表中Sage列的类型是datetime)   
    select Sname,  CONVERT(char (**11**),DATEPART(year,Sage)) as age   
    from student   
    where  CONVERT(char(**11**),DATEPART(year,Sage))='1981';   
**32**、查询每门课程的平均成绩，结果按平均成绩升序排列，平均成绩相同时，按课程号降序排列   
    Select C#,Avg(score) from SC group by C# order by Avg(score),C# DESC ;   
**33**、查询平均成绩大于85的所有学生的学号、姓名和平均成绩   
    select Sname,SC.S# ,avg(score)   
    from Student,SC   
    where Student.S#=SC.S# group by SC.S#,Sname having    avg(score)>**85**;   
**34**、查询课程名称为“数据库”，且分数低于60的学生姓名和分数   
    Select Sname,isnull(score,**0**)   
    from Student,SC,Course   
    where SC.S#=Student.S# and SC.C#=Course.C# and  Course.Cname='数据库'and score <**60**;   
**35**、查询所有学生的选课情况；   
    SELECT SC.S#,SC.C#,Sname,Cname   
    FROM SC,Student,Course   
    where SC.S#=Student.S# and SC.C#=Course.C# ;   
**36**、查询任何一门课程成绩在70分以上的姓名、课程名称和分数；   
    SELECT  distinct student.S#,student.Sname,SC.C#,SC.score   
    FROM student,Sc   
    WHERE SC.score>=**70** AND SC.S#=student.S#;   
**37**、查询不及格的课程，并按课程号从大到小排列   
    select c# from sc where scor e <**60** order by C# ;   
**38**、查询课程编号为003且课程成绩在80分以上的学生的学号和姓名；   
    select SC.S#,Student.Sname from SC,Student where SC.S#=Student.S# and Score>**80** and C#='003';   
**39**、求选了课程的学生人数   
    select count(\*) from sc;   
**40**、查询选修“叶平”老师所授课程的学生中，成绩最高的学生姓名及其成绩   
    select Student.Sname,score   
    from Student,SC,Course C,Teacher   
    where Student.S#=SC.S# and SC.C#=C.C# and C.T#=Teacher.T# and Teacher.Tname='叶平' andSC.score=(select max(score)from SC where C#=C.C# );   
**41**、查询各个课程及相应的选修人数   
    select count(\*) from sc group by C#;   
**42**、查询不同课程成绩相同的学生的学号、课程号、学生成绩   
  select distinct  A.S#,B.score from SC A  ,SC B where A.Score=B.Score and A.C# <>B.C# ;   
**43**、查询每门功成绩最好的前两名   
    SELECT t1.S# as 学生ID,t1.C# as 课程ID,Score as 分数   
      FROM SC t1   
      WHERE score IN (SELECT TOP **2** score   
              FROM SC   
              WHERE t1.C#= C#   
            ORDER BY score DESC   
              )   
      ORDER BY t1.C#;   
**44**、统计每门课程的学生选修人数（超过10人的课程才统计）。要求输出课程号和选修人数，查询结果按人数降序排列，查询结果按人数降序排列，若人数相同，按课程号升序排列    
    select  C# as 课程号,count(\*) as 人数   
    from  sc    
    group  by  C#   
    order  by  count(\*) desc,c#    
**45**、检索至少选修两门课程的学生学号   
    select  S#    
    from  sc    
    group  by  s#   
    having  count(\*)  >  =  **2**   
**46**、查询全部学生都选修的课程的课程号和课程名   
    select  C#,Cname    
    from  Course    
    where  C#  in  (select  c#  from  sc group  by  c#)    
**47**、查询没学过“叶平”老师讲授的任一门课程的学生姓名   
    select Sname from Student where S# not in (select S# from Course,Teacher,SC whereCourse.T#=Teacher.T# and SC.C#=course.C# and Tname='叶平');   
**48**、查询两门以上不及格课程的同学的学号及其平均成绩   
    select S#,avg(isnull(score,**0**)) from SC where S# in (select S# from SC where score <**60** group by S# havingcount(\*)>**2**)group by S#;   
**49**、检索“**004**”课程分数小于60，按分数降序排列的同学学号   
    select S# from SC where C#='004'and score <**60** order by score desc;   
**50**、删除“**002**”同学的“**001**”课程的成绩   
delete from Sc where S#='001'and C#='001';