1 前言

该部分经验主要是在首都国际机场，海口梅兰国际机场系统的开发过程中得到的。环境是使用 IBM s80 机器，AIX4.3 操作系统，4G 内存，DB2 数据库(UDB 7.2 版本)，

存储设备是 EMC 磁盘阵列，12 对硬盘，做 RAID 1，即可用 12 个硬盘，每个 36G。其中 4 个被用做 bcv(也是一个镜像系统，和工作库中的数据一模一样，可以用于快速创建一个和工作环境一样的开发环境。具体不是很懂)。实际可用 8 个物理硬盘，每个划为 4 个 9G 盘。

2 DB2 专有名词解释

2.1 Instance(实例)

相当于 Informix 的 Informix Server 的概念，在一台机器上可以有多个相互独立的实例，并同时运行．每个实例可以管理若干个数据库，一个数据库只属于一个实例．

2.2 DB2 Administration Server(管理服务器)

与 DB2 Administration Client 对称．一个 DB2 数据库如果需要远端的管理，就需要在有 DB2 数据库的机器上有管理服务进程以接收远端的管理客户进程的请求．一般来讲，一个在 R/6000 上的 DB2，由于 AIX 一般无图形界面，最好在局域网内有一台有图形界面的机器(例如装有 Win 98 或 Win NT)来对其进行远程管理．因为用带图形界面的 DB2 控制中心，可以很方便的查看 DB2 的状态，详细形象的监控 DB2 的性能，对 DB2 的配置参数进行精确的调整，而这些都是用 DB2 的命令行难以实现的．控制中心提供的 Smart Guide 功能，更可以让数据库管理员不用关心数据库内部实现的细节，而对数据库进行较精确的调整．

2.3 Container(容器)

与 Informix 中的 chunk 概念基本一样．但 DB2 数据库管理进程在向容器内写数据时，所有一个表空间内的容器是均衡着写入的．并且这种均衡是实时的，例如在一个表空间中加入一个容器后，该容器所处的表空间中其它容器的数据会很快的均衡到该容器来．

2.4 DRDA

分布式关系数据库结构 Distributed Relational Database Architecture

2.5 DARI

数据库应用远程接口 Database Application Remote Interface

2.6 SPM

Synchronous Point Management，相当于 Informix 的 check point

2.7 FCM

Fast Communication Management，用于数据库分区间通信

2.8 ADSM

ADSTAR Distributed Storage Manager

2.9 DCE

Distributed Compute Environment

3 DB2 编程

1.1 建存储过程时 Create 后一定不要用 TAB 键

create procedure
的 create 后只能用空格,而不可用 tab 健，否则编译会通不过。

切记，切记。

1.2 使用临时表

要注意,临时表只能建在 user tempory tables space 上,如果 database 只有 system tempory table space 是不能建临时表的。

另外，DB2 的临时表和 sybase 及 oracle 的临时表不太一样，DB2 的临时表是在一个 session 内有效的。所以，如果程序有多线程，最好不要用临时表，很难控制。

建临时表时最好加上 with replace 选项，这样就可以不显示的 drop 临时表，建临时表时如果不加该选项而该临时表在该 session 内已创建且没有 drop,这时会发生错误。

注意：一旦 rollback，该临时表将不存在。

临时表有好几种定义方式。但如果是对 not null 及 default 值有什么要求的话，最好还是使用完整字段列表来定义。因为有一次，我使用了 like table including column default 来定义，但 default 还是没有按预料的那样带过来.

如下例，可以作为常用的临时表的定义方式。

```
declare global    temporary table tmp_tb_clear_match_detail (
        tradedate                    char(8) not null          --    业务日期
)
    with replace on commit preserve rows not logged ;
```

on commit preserve：是在 commit 时不将临时表的内容释放。

临时表中也可以使用自增字段：

declare global temporary table tt(aa char(1),bb int generated always as identity) not logged

经过本人测试,对临时表做插入比做 update 速度要快很多,插入 50000 条记录是用 15 秒,再对该表中插入 1000 条记录，用时不到 1 秒，而 update 其中 1000 条，用时 60 秒。

临时表中不能建索引，很不好用。

1.3 从数据表中取指定前几条记录

```
select    *    from tb_market_code fetch first 1 rows only
```

但下面这种方式不允许

```
select market_code into v_market_code
    from tb_market_code fetch first 1 rows only;
```

选第一条记录的字段到一个变量以以下方式代替

```
declare v_market_code char(1);
declare cursor1 cursor for select market_code from tb_market_code
fetch first 1 rows only for update;
open cursor1;
fetch cursor1 into v_market_code;
close cursor1;
```

1.4 游标的使用

注意 commit 和 rollback

使用游标时要特别注意如果没有加 with hold 选项,在 Commit 和 Rollback 时,该游标将被关闭。Commit 和 Rollback 有很多东西要注意。特别小心

游标的两种定义方式

一种为

```
declare continue handler for not found
    begin
        set v_notfound = 1;
end;

declare cursor1 cursor with hold for select market_code from
tb_market_code      for update;
open cursor1;
set v_notfound=0;
fetch cursor1 into v_market_code;
while v_notfound=0 Do
        --work
        set v_notfound=0;
        fetch cursor1 into v_market_code;
end while;
close cursor1;
```

这种方式使用起来比较复杂,但也比较灵活。特别是可以使用 with hold

选项。如果循环内有 commit 或 rollback 而要保持该 cursor 不被关闭，只能使用这种方式。

另一种为

```
pcursor1: for loopcs1 as    cousor1    cursor    as
      select    market_code    as market_code
       from tb_market_code
       for update
 do
 end for;
```

这种方式的优点是比较简单，不用（也不允许）使用 open,fetch,close。但不能使用 with hold 选项。如果在游标循环内要使用 commit,rollback 则不能使用这种方式。如果没有 commit 或 rollback 的要求，推荐使用这种方式(看来 For 这种方式有问题)。

修改游标的当前记录的方法

```
update tb_market_code set market_code='0' where current of cursor1;
```

不过要注意将 cursor1 定义为可修改的游标

```
declare cursor1 cursor for select market_code from tb_market_code
for update;
```

for update 不能和 GROUP BY、 DISTINCT、 ORDER BY、 FOR READ ONLY 及 UNION, EXCEPT, or INTERSECT 但 UNION ALL 除外）一起使用。

1.5 类似 decode 的转码操作

oracle 中有一个函数 select decode(a1,'1','n1','2','n2','n3') aa1 from
db2 没有该函数，但可以用变通的方法

```
select case a1
    when '1' then 'n1'
    when '2' then 'n2'
    else 'n3'
  end as aa1 from
```

1.6 类似 charindex 查找字符在字串中的位置

Locate('y','dfdasfay')

查找’y’ 在’dfdasfay’中的位置。

1.7 类似 datedif 计算两个日期的相差天数

days(date(‘2001-06-05’)) – days(date(‘2001-04-01’))
days 返回的是从 0001-01-01 开始计算的天数

下面一个例子是取该天所在的周的星期一的日：
date(days('2001-08-20')-dayofweek('2001-08-20')+2)

1.8 写 UDF 的例子

C 写见 sqllib\samples\cli\udfsrv.c

1.9 创建含 identity 值(即自动生成的 ID)的表

建这样的表的写法
CREATE TABLE test
(t1 SMALLINT NOT NULL
GENERATED ALWAYS AS IDENTITY
(START WITH 500, INCREMENT BY 1),
t2 CHAR(1));
在一个表中只允许有一个 identity 的 column.

1.10 预防字段空值的处理

```
SELECT
DEPTNO , DEPTNAME , COALESCE (MGRNO , 'ABSENT') , ADMRDEPT
FROM DEPARTMENT
```

COALESCE 函数返回()中表达式列表中第一个不为空的表达式，可以带多个表达式。

和 oracle 的 isnull 类似，但 isnull 好象只能两个表达式。

1.11 取得处理的记录数

declare v_count int;

```
update   tb_test        set       t1='0'
      where    t2='2';
--检查修改的行数,判断指定的记录是否存在
get diagnostics    v_ count=ROW_COUNT;
只对 update,insert,delete 起作用.
不对 select into  有效
```

1.12 从存储过程返回结果集(游标)的用法

```
1、建一 sp 返回结果集
CREATE PROCEDURE DB2INST1.Proc1 (    )
 LANGUAGE SQL
      result sets 2(返回两个结果集)
-----------------------------------------------------------------------
-- SQL  存储过程
-----------------------------------------------------------------------
P1: BEGIN
                declare c1 cursor    with return to caller for
                     select    market_code
                     from       tb_market_code;
                --指定该结果集用于返回给调用者
                declare c2 cursor    with return to caller for
                     select    market_code
                     from       tb_market_code;
                 open c1;
                 open c2;
END P1


2、建一 SP 调该 sp 且使用它的结果集

CREATE PROCEDURE DB2INST1.Proc2 (
     out out_market_code char(1))
      LANGUAGE SQL
     -----------------------------------------------------------------------
     -- SQL  存储过程
     -----------------------------------------------------------------------
P1: BEGIN

       declare loc1,loc2 result_set_locator varying;
     --建立一个结果集数组
     call proc1;
```

```
--调用该 SP 返回结果集。
associate result set locator(loc1,loc2) with procedure proc1;
--将返回结果集和结果集数组关联
  allocate cursor1 cursor for result set loc1;
  allocate cursor2 cursor for result set loc2;
--将结果集数组分配给 cursor
fetch    cursor1 into out_market_code;
--直接从结果集中赋值
close cursor1;

END P1
```

3、动态 SQL 写法

```
    DECLARE CURSOR C1 FOR STMT1;
    PREPARE STMT1 FROM
        'ALLOCATE C2 CURSOR FOR RESULT SET ?';
```

4、注意：

一、如果一个 sp 调用好几次，只能取到最近一次调用的结果集。

二、allocate 的 cursor 不能再次 open，但可以 close，是 close sp 中的对应 cursor。

1.13 类型转换函数

```
select cast ( current time as char(8)) from tb_market_code
```

1.14 存储过程的互相调用

目前,c sp 可以互相调用。

Sql sp 可以互相调用，

Sql sp 可以调用 C sp，

但 C sp 不可以调用 Sql sp(最新的说法是可以)

1.15 C 存储过程参数注意

```
create procedure pr_clear_task_ctrl(
                        IN IN_BRANCH_CODE char(4),
                        IN IN_TRADEDATE      char(8),
                        IN IN_TASK_ID        char(2),
                        IN IN_SUB_TASK_ID char(4),
                        OUT OUT_SUCCESS_FLAG INTEGER )
```

DYNAMIC RESULT SETS 0
LANGUAGE C
PARAMETER STYLE GENERAL WITH NULLS(如果不是这样，sql 的 sp 将不能调用该用 c 写的存储过程，产生保护性错误)
该参数的实际意义是，如果不是 with nulls 则 sql 在调用该存储过程时，如果有一个参数为 null 的话，存储过程的调用会出错。
NO DBINFO
FENCED
MODIFIES SQL DATA
EXTERNAL NAME 'pr_clear_task_ctrl!pr_clear_task_ctrl'@

1.16 存储过程 fence 及 unfence

fence 的存储过程单独启用一个新的地址空间,而 unfence 的存储过程和调用它的进程使用同一个地址空间。
一般而言，fence 的存储过程比较安全。
但有时一些特殊的要求，如要取调用者的 pid，则 fence 的存储过程会取不到，而只有 unfence 的能取到。

1.17 SP 错误处理用法

如果在 SP 中调用其它的有返回值的，包括结果集、临时表和输出参数类型的 SP，DB2 会自动发出一个 SQLWarning。而在我们原来的处理中对于 SQLWarning 都会插入到日志，这样子最后会出现多条 SQLCODE=0 的警告信息。
处理办法：
定义一个标志变量，比如 DECLARE V_STATUS INTEGER DEFAULT 0,
在 CALL SPNAME 之后, SET V_STATUS = 1,

```
DECLARE CONTINUE HANDLER FOR SQLWARNING
BEGIN
    IF V_STATUS <> 1 THEN
        --警告处理，插入日志
        SET V_STATUS = 0;
    END IF;
END;
```

1.18 values 的使用

如果有多个 set 语句给变量付值，最好使用 values 语句，改写为一句。这样可以提高效率。

但要注意，values 不能将 null 值付给一个变量。
values(null) into out_return_code;
这个语句会报错的。

1.19 给 select 语句指定隔离级别

select * from tb_head_stock_balance with ur

1.20 atomic 及 not atomic 区别

atomic 是将该部分程序块指定为一个整体,其中任何一个语句失败,则整个程序块都相当于没做,包括包含在 atomic 块内的已经执行成功的语句也相当于没做，有点类似于 transaction。

1.21 C 及 SQL 存储过程名称都要注意长度

C 的存储过程要注意：C 的过程名称长度可以到 128 位，但是由于 syscat.packages 的系统表中的 pkgname 这个字段只有 8 位长，而 C 的过程的名称默认情况下就是作为 pkgname 的，所以 C 过程的前 8 位最好是要保证唯一的，否则如果 pkgname 重名则会互相覆盖，只有后建的过程才可以用，如果不能确认，就特意将 pkgname 另外取名字。

```
db2 "create procedure pr_clear_call(
        in    in_instname      char(18),
        in    in_database          char(18),
        in    in_user               char(18),
        in    in_password          char(30),
        out OUT_SUCCESS_FLAG int,
        out OUT_RETURN_MESSAGE char(128))
    DYNAMIC RESULT SETS 0
    LANGUAGE C
    PARAMETER STYLE GENERAL WITH NULLS
EXTERNAL NAME '$DB2PATH/function/$1/clearcal!pr_clear_call' FENCED"
```

SQL 的过程名称没什么限制，但我们在调用一个名字较长的过程时，常出现莫名其妙的现象，好象没有被调用一样，最后将名字改短了，之后恢复正常。

1.22 怎样获得自己的数据库连接句柄

SQL 目前好象还没有什么办法。
C 的可以使用 CLI 编程得到，在 samples\cli\dbconn.c 中有例子。实际上就可以将 pid 及 ppid 及连接句柄记录下来写到库中，便于管理。

1.23 类似于 ORACLE 的 Name pipe

有时在做事物的时候,有些类似于登录信息等是和事物的成功与失败是无关的,无论结果如何都应该把这些内容记录下来。

Oracle 有一种 name pipe 的机制，可以将信息输出到数据库外一个指定的文件中去，然后在写一个 c 的服务程序不断轮循这个文件，读出其中的信息，再写回到数据库中。

Db2 的 Sql 是不能实现这个功能的，但是可以使用 sqc 写 c 的存储过程来实现这个功能。

1.24 类似于 ORACLE 的 TRUNCATE 清表但不记日志的做法

db2 "alter table tmp_testalt activate not logged initially with empty table "
但这个表定义的时候一定要有 not logged initially 选项

1.25 用 cli 编程批量的 insert

据说比 import 要快很多,下面是例子

/*HongTao, you should create a table fetchscrolltable(col1 char(13),col2 char(13)) first for running this sample program. Any concern, pls feel free
to call me!
(See attached file: tbread.c)

```
*/


/*****************************************************************
**********************
**
** Source File Name = tbread.c    %I%
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**       PURPOSE :
**       Shows how to read tables.
**
** For more information on programming in CLI see the:
**          - "Building CLI Applications" section of the Application
Building Guide, and the
**         - CLI Guide and Reference.
**
** For more information on the SQL language see the SQL Reference.
**
*****************************************************************
**********************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlcli1.h>
#include "utilcli.h"

/* methods to perform select */
#define ROWSET_SIZE 5
int TbSelectUsingFetchScrollColWise( SQLHANDLE) ;
```

```
int main( int argc, char * argv[] )
{     SQLRETURN     sqlrc = SQL_SUCCESS;
      int           rc = 0;
      SQLHANDLE     henv;    /* environment handle */
      SQLHANDLE     hdbc;    /* connection handles */

      char          dbAlias[SQL_MAX_DSN_LENGTH + 1] ;
      char          user[MAX_UID_LENGTH + 1] ;
      char          pswd[MAX_PWD_LENGTH + 1] ;

      /* checks the command line arguments */
      rc = CmdLineArgsCheck1( argc, argv, dbAlias, user, pswd );
      if ( rc != 0 ) return( rc ) ;

      printf("\n\nTABLES: HOW TO READ TABLES.\n");

      /* initialize the CLI application */
      rc = CLIAppInit( dbAlias, user, pswd, &henv, &hdbc,
                       (SQLPOINTER)SQL_AUTOCOMMIT_ON);
      if ( rc != 0 ) return( rc ) ;


      rc = TbSelectUsingFetchScrollColWise( hdbc) ;

      rc = CLIAppTerm( &henv, &hdbc, dbAlias);
      return( rc ) ;
}                                          /* end main */




int TbSelectUsingFetchScrollColWise( SQLHANDLE hdbc)
{     SQLRETURN     sqlrc = SQL_SUCCESS;
      int           rc = 0;
      SQLHANDLE     hstmt ;    /* statement handle */
      SQLHANDLE     hstmtTable ; /* to create a test table */

      SQLINTEGER    rowNb;
      SQLCHAR       stmtInsert[100];

      SQLUINTEGER     rowsFetchedNb;
      SQLUSMALLINT    rowStatus[ROWSET_SIZE];
      static        char                    ROWSTATVALUE[][26]        =
{ "SQL_ROW_SUCCESS", \
```

```
"SQL_ROW_SUCCESS_WITH_INFO", \

"SQL_ROW_ERROR", \

"SQL_ROW_NOROW" };
        int i;

        struct
        {     SQLINTEGER ind[ROWSET_SIZE] ;
              SQLCHAR val[ROWSET_SIZE][15] ;
        } col1, col2 ;




        /* set AUTOCOMMIT on */
        sqlrc = SQLSetConnectAttr( hdbc,
                                              SQL_ATTR_AUTOCOMMIT,

(SQLPOINTER)SQL_AUTOCOMMIT_ON, SQL_NTS) ;
        DBC_HANDLE_CHECK( hdbc, sqlrc);

        /* allocate a statement handle */
        sqlrc    =    SQLAllocHandle(    SQL_HANDLE_STMT,    hdbc,
&hstmtTable ) ;
        DBC_HANDLE_CHECK( hdbc, sqlrc);

            sprintf((char*) stmtInsert,
                  "delete from    fetchScrollTable ");
              sqlrc = SQLExecDirect( hstmtTable, stmtInsert, SQL_NTS ) ;
              STMT_HANDLE_CHECK( hstmtTable, sqlrc);


        /* allocate a statement handle for FetchScroll */
        sqlrc = SQLAllocHandle( SQL_HANDLE_STMT, hdbc, &hstmt ) ;
        DBC_HANDLE_CHECK( hdbc, sqlrc);


        sqlrc = SQLPrepare( hstmt, "Insert into fetchScrollTable values
(?, ?)", SQL_NTS ) ;
        STMT_HANDLE_CHECK( hstmt, sqlrc);

        /* set the required statement attributes */
```

```
        printf("       Set the required statement attributes.\n");


    for (i=0; i<ROWSET_SIZE ; i++) {
        sprintf((char *)(col1.val[i]), "col1%d", i);
        sprintf((char *)(col2.val[i]), "col2%d", i);
        /*printf("%s,%s\n",col1.val,col2.val);*/
    }

    sqlrc = SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT,
SQL_C_CHAR,
                             SQL_CHAR, 15, 0, col1.val, 15,
NULL);
    STMT_HANDLE_CHECK( hstmt, sqlrc);
    sqlrc = SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT,
SQL_C_CHAR,
                             SQL_CHAR, 15, 0, col2.val, 15,
NULL);
    STMT_HANDLE_CHECK( hstmt, sqlrc);


    sqlrc = SQLSetStmtAttr( hstmt,
                            SQL_ATTR_PARAMSET_SIZE ,
                            (SQLPOINTER)ROWSET_SIZE,
                            0);
    STMT_HANDLE_CHECK( hstmt, sqlrc);

        printf("       insert into the test table.\n");
        /*sqlrc=SQLBulkOperations(hstmt, SQL_ADD) ;*/
          sqlrc = SQLExecute( hstmt ) ;
        STMT_HANDLE_CHECK( hstmt, sqlrc);


        /* free the statement handle */
        sqlrc = SQLFreeHandle( SQL_HANDLE_STMT, hstmt ) ;
        STMT_HANDLE_CHECK( hstmt, sqlrc);


        /* free the statement handle */
        sqlrc = SQLFreeHandle( SQL_HANDLE_STMT, hstmtTable ) ;
        STMT_HANDLE_CHECK( hstmtTable, sqlrc);

        return(rc);
}
```

4 DB2 一些不好的限制

4.1 临时表不能建索引

对临时表 update 记录时极慢,一个临时表中有 50000 条记录,update 其中的 1000 条,花了 1 分钟。

4.2 cursor 不能定义为 with ur(可以但…)

在 7.2.2 版本下，一个存储过程内只有一个 cursor 可以使用 with ur，如果有两个 cursor 可以任意给一个使用 with ur，但不能两个都用，否则编译通不过。

4.3 cursor order by 以后不能 for update

这样使我只能使用关键字来 update 又老是要做 table scan.

4.4 程序中间不能自由改变隔离级别

4.5 update 不能用一个表中的记录为条件修改另一个表中的记录。

这样导致我不得不用 cursor，速度慢。

4.6 如果显示调用存储过程时传 null 值要注意

注意将 null 用大写表示，NULL
否则 db2 不认为是空。

5 DB2 编程性能注意

5.1 大数据的导表的使用(export,load,import)(小心)

5.1.1 import 的用法

db2 import from gh1.out of DEL messages err.txt insert into db2inst1.tb_dbf_match_ha

db2 import from gh1.out of del modified by chardel; messages err.txt insert into db2inst1.tb_dbf_match_ha
(column 的分割符号改为;)

注意要加 schma

Import 使用要特别注意：

Import 缺省使用 RR 隔离级别，且对表加锁，而且是加 X 锁。使用 import 一定要想清楚。

如果 import 时使用了 replace 选项，连 select * from tbtmp with ur 都不可以选记录，也会报错为 timeout。不知道是怎么回事。好象这张表被加了 Z 超级排它锁一样。

另外，import 时对 syscolumns 加了一些行锁，对 systables 加了表锁 IS 锁，而 IS 和 X 是不相容的。因此，这时候就不能创建和 drop 表了。

个人认为：

如果要并发，还是使用 insert 比较好，虽然 insert 比较慢，但不会锁表，可以多个进程并发执行。

5.1.2 性能比较

应该是 export 后再 load 性能更好，因为 load 不写日志。
比 select into 要好。

5.1.3 export 用法

db2 "export to tb_head_pending_buyback.del of del select * from tb_head_pending_buyback"

5.2 SQL 语句尽量写复杂 SQL

尽量使用大的复杂的 SQL 语句,将多而简单的语句组合成大的 SQL 语句对性能会有所改善。

DB2 的 SQL Engieer 对复杂语句的优化能力比较强，基本上不用当心语句的性能问题。

Oracle 则相反，推荐将复杂的语句简单化，SQL Engieer 的优化能力不是特别好。

这是因为每一个 SQL 语句都会有 reset SQLCODE 和 SQLSTATE 等各种操作，会对数据库性能有所消耗。

一个总的思想就是尽量减少 SQL 语句的个数。

5.3 SQL SP 及 C SP 的选择

首先，C 的 sp 的性能比 sql 的 sp 的要高。

一般而言，SQL 语句比较复杂，而逻辑比较简单，sql sp 与 c sp 的性能差异会比较小，这样从工作量考虑，用 SQL 写比较好。

而如果逻辑比较复杂，SQL 比较简单，用 c 写比较好。

5.4 查询的优化(HASH 及 RR_TO_RS)

db2set DB2_HASH_JOIN=Y (HASH 排序优化)

指定排序时使用 HASH 排序，这样 db2 在表 join 时，先对各表做 hash 排序，再 join，这样可以大大提高性能。

剧沈刚说做实验，7 个一千万条记录表的做 join 取 10000 条记录，再没有索引的情况下 72 秒。

db2set DB2_RR_TO_RS=Y

该设置后，不能定义 RR 隔离级别，如果定义 RR，db2 也会自动降为 RS. 这样，db2 不用管理 Next key，可以少管理一些东西，这样可以提高性能。

5.5 避免使用 count(*) 及 exists 的方法

1、首先要避免使用 count(*)操作，因为 count(*)基本上要对表做全部扫描一遍，如果使用很多会导致很慢。
2、exists 比 count(*)要快，但总的来说也会对表做扫描，它只是碰到第一条符合的记录就停下来。

如果做这两中操作的目的是为
select into 服务的话，就可以省略掉这两步。
直接使用 select into 选择记录中的字段。

如果是没有记录选择到的话，db2 会将 sqlcode=100 和 sqlstate='20000'
如果是有多条记录的话，db2 会产生一个错误。

程序可以创建 continue handler for exception
continue handler for not found
来检测。
这是最快速的方法。

3、如果是判断是不是一条,可以使用游标来计算，用一个计数器，累加，达到预定值后就离开。这个速度也比 count(*) 要快，因为它只要扫描到预定值就不再扫描了，不用做全表的 scan，不过它写起来比较麻烦。

5.6 Commit 的次数要适当

据马宏伟的测试，
一个 50 万条记录的表，有索引，update 或 insert 一条记录是 6 毫秒。
Commit 一条要 3 毫秒，所以也不能 commit 太频繁。

5.7 Insert 和 update 速度比较

如果表内数据少，有索引，update 要快
数据多，有索引，insert 要快。
但还有看索引的多少，及 update 的字段，如果 update 不对索引及主键改变，速度很可能会比 insert 要快。

5.8 使用临时表取代一条一条插入

如果程序是要一条一条记录的插入到一个数据量很大的表中，如果是使用一个临时表作为一个中间表，先插入到中间表，再一次插入大表中。

我是用 50 个进程并行的，改动前后的速度比较是 50 分钟--》4 分钟。是分解的国泰君安深圳席位的数据，共 9 万条数据，137 个席位。计算比较复杂。

改临时表后，锁的等待变得很少，数据库及 cpu 的压力就可以上去。

5.9 循环次数很多时注意减少执行语句(附例子)

下面是两个过程,可以比较一下

过程 1:

```
DROP PROCEDURE pr_test1 @

CREATE PROCEDURE pr_test1(
OUT out_return_code INTEGER )                    --返回值
LANGUAGE SQL
Proc: BEGIN
    declare     v_nobr_flag          int  default 0;
    declare     v_fetch_brnext       int;
    declare     v_fetch_headnext     int;
    declare     v_error_flag      char(1);
    declare     v_error_desc      char(1);
    declare     m_br_branch_code        char(1);
    declare     m_br_sub_branch_code    char(1);

DoWork:BEGIN

PFetch: while v_nobr_flag<100000
    do

        set v_nobr_flag=v_nobr_flag+1;
          set v_fetch_brnext=0;
        set v_fetch_headnext=0;
        set v_error_flag,=’’;
        set v_error_desc=’’;
        set m_br_branch_code=’’;
        set m_br_sub_branch_code=’’;
```

```
end while PFetch;
END DoWork;
END Proc
@
```

过程 2:

```
DROP PROCEDURE pr_test2 @

CREATE PROCEDURE pr_test2(
OUT out_return_code INTEGER )                    --返回值
LANGUAGE SQL
Proc: BEGIN
    declare     v_nobr_flag          int   default 0;
    declare     v_fetch_brnext       int;
    declare     v_fetch_headnext     int;
    declare     v_error_flag      char(1);
    declare     v_error_desc      char(1);
    declare     m_br_branch_code        char(1);
    declare     m_br_sub_branch_code    char(1);
DoWork:BEGIN

PFetch: while v_nobr_flag<100000
    do

        set v_nobr_flag=v_nobr_flag+1;

        values(0,0,",",",")
        into v_fetch_brnext,
            v_fetch_headnext,
            v_error_flag,
            v_error_desc,
            m_br_branch_code,
            m_br_sub_branch_code;

end while PFetch;
END DoWork;
END Proc
@
```

本人实验,pr_test1 用时 34 秒,pr_test2 用时 15 秒。

5.10 看程序执行时间及结果 db2batch

db2batch –d -zbdb –f test.sql –r test.res
有点像 sybase 的 isql。
也有多个选项可以用，help。

5.11 看程序或语句具体的执行计划 shell（改写后的语句）

goplan.sh

```
#!/bin/ksh
#
#
[ $# -eq 1 ] || {
    echo "$0: missing argument."
    exit
}
id=$1
output=${id}.out
output2=db2exp.${id}.out1
set -x
db2batch -d zbdb -f ${id}.sql -r ${output} -o o 5 e 1
db2exfmt -d zbdb -t -w -1 > ${output2}
```

5.12 两个表做 join 的不同方式的区别

这里几个例子是查询两个表中的记录有哪些在一个表中存在而在另一个表中不存在。
具体的执行计划可以用上面的 goplan.sh 来看。

以下是比较简单的方式，其实如果要核对的项很多，可能采用做两个 cursor，排序后自己比较可能会更快。

5.12.1 not in 方式

```
select
        head.sub_branch_code,
        0,
```

```
        '1',
        '总部有，营业部没有',
    '1',

    head.capital_account,
    head.account_head_type,
    head.account_branch_type
from table(
    select * from kstar.tb_head_capital_account
        where branch_code='1104'
    and increment_flag<>'2'
    and (branch_code,sub_branch_code,capital_account) not in
        (select branch_code,sub_branch_code,capital_account
            from kstar.tb_brclose_capital_account
            where branch_code='1104'
        )

    ) head;
```

这种方式非常的慢，据编写优化程序的 db2 研究人员回答，这部分内核根本就没有做什么优化。这个语句在一个营业部有 20000 条记录的情况下根本不可行。

5.12.2 except 方式

```
select
    head.sub_branch_code,
    0,
        '1',
        '总部有，营业部没有',
    '1',

    head.capital_account,
    head.account_head_type,
    head.account_branch_type
from table(
    select * from kstar.tb_head_capital_account
        where branch_code='1104'
    and increment_flag<>'2'

    except

    select b.* from kstar.tb_head_capital_account b,kstar.tb_brclose_capital_account c
```

```
            where b.branch_code='1104'
            and c.branch_code='1104'
            and b.sub_branch_code=c.sub_branch_code
            and b.capital_account=c.capital_account
            and b.increment_flag<>'2'
        ) head;
```

这种系统会做有限的优化，速度还可以。

5.12.3 not exist 方式

```
    select
        head.sub_branch_code,
        0,
            '1',
            '总部有，营业部没有',
        '1',

        head.capital_account,
        head.account_head_type,
        head.account_branch_type
from table(
        select * from kstar.tb_head_capital_account a
            where branch_code='1104'
        and increment_flag<>'2'
        and not exists
            (select sub_branch_code,capital_account
                from kstar.tb_brclose_capital_account b
                where branch_code='1104' and
                a.sub_branch_code=b.sub_branch_code
                and a.capital_account=b.capital_account
            )

        ) head;
```

这中方式做 join 的速度很快，甚至都用到了 Hash join。

6 其他系统和 DB2 的交互

6.1 DELPHI 中从 db2 取 bigint 的数据

由于 delphi 不能识别 bigint，在返回结果在 delphi 中显示的时候最好将 bigint 转为 double 或者 decimal。

7 DB2 表及 sp 管理

7.1 权限管理

7.1.1 数据库权限控制

syscat.DBAUTH

管理用户在整个 database 上能否创建 table,sp,load 等的权限。

db2 "GRANT CREATETAB,BINDADD,CONNECT,CREATE_NOT_FENCED, LOAD,DBADM,IMPLICIT_SCHEMA ON DATABASE TO USER kstar"

也可以是 To Group db2admin

To Public

收回权限

db2 "revoke CREATETAB,BINDADD,CONNECT,CREATE_NOT_FENCED, LOAD,DBADM,IMPLICIT_SCHEMA ON DATABASE from user kstar"

7.1.2 schema 权限控制

syscat. SCHEMAAUTH

db2 "GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA KINGSTAR TO USER kstar"

收回权限

db2 "revoke CREATEIN,DROPIN,ALTERIN on SCHEMA KINGSTAR from user kstar"

7.1.3 tablespace 权限控制

syscat. TBSPACEAUTH
管理用户可以对 tablespace 的权限，如是否可以访问等。

db2 "GRANT USE OF TABLESPACE BAK_TABSPACE TO PUBLIC"

收回权限
db2 "revoke use on tablespace tbsname from PUBLIC"

7.1.4 table 权限控制

syscat.tabauth
管理用户可以对 table 的权限，如更改、删除数据、插入数据等。

db2 "GRANT alter,control,delete,index,insert,select,update(column 列表),references (column 列表) on table tabname TO PUBLIC"

收回权限
db2 "revoke alter,control,delete,index,insert,select,update(column 列表),references (column 列表) on table tabname from PUBLIC"

7.1.5 package 权限控制

SYSCAT.PACKAGEAUTH
管理用户可以对 package 的权限，如 bind 等。

db2 "GRANT bind,control,execute on package pkgname TO PUBLIC"
收回权限
db2 "revoke bind,control,execute on package pkgname from PUBLIC"

7.2 建存储过程会占用很多的系统资源（特别是 io）

在大通的升级过程中，偶然发现在构建 sp 的时候，db2instance 所在的磁盘非常的繁忙，io 达到 100，且整个系统 io 等待达到 50%。
构建 sp 很耗系统资源。

7.3 看存储过程文本

select text from syscat.procedures where procname='PROC1';

7.4 看表结构

describe table syscat.procedures
describe select * from syscat.procedures
注意要加 schema 名称。

7.5 看表的索引信息

db2 "select colnames from syscat.indexes where tabname= 'TB_CLEAR_MATCH_DETAIL' "

7.6 查看各表对 sp 的影响(被哪些 sp 使用)

select PROCNAME from SYSCAT.PROCEDURES where SPECIFICNAME in(select dname from sysibm.sysdependencies where bname in (select PKGNAME from syscat.packagedep where bname='TB_BRANCH'))

7.7 查看 sp 使用了哪些表

select bname from syscat.packagedep where btype='T' and pkgname in(select bname from sysibm.sysdependencies where dname in (select specificname from syscat.procedures where procname='PR_CLEAR_MATCH_DIVIDE_SHA'))

7.8 查看 function 被哪些 sp 使用

select PROCNAME from SYSCAT.PROCEDURES where SPECIFICNAME in(select dname from sysibm.sysdependencies where bname in (select PKGNAME from syscat.packagedep where bname in (select SPECIFICNAME from SYSCAT.functions where funcname='GET_CURRENT_DATE')))

使用 function 时要注意，如果想 drop 掉该 function 必须要先将调用该 function 的其它存储过程全部 drop 掉。

必须先创建 function，调用该 function 的 sp 才可以创建成功。

7.9 查 sp 的 ID 号

select * from syscat.packages a,syscat.procedures b where substr(b.implementation,1,8)=a.pkgname and procname="with ur

7.10 从 sp 的 id 号查存储过程名称

SELECT * FROM SYSCAT.PACKAGES A,SYSCAT.PROCEDURES B WHERE SUBSTR(B.IMPLEMENTATION,1,8)=A.PKGNAME AND A.PKGNAME='P3227010' WITH UR

7.11 创建及使用 summary table

例:

```
create table tb_whttest
(aa char(1),
  bb int
);


create summary table st_whttest
              (aa,bb_sum,colcount,bbcount)
              as
       ( select aa, sum(bb),count(*),count(bb)
          from tb_whttest
              group by aa
       )
              data initially deferred refresh immediate
              enable query optimization
```

说明：

使用 summary table 有很多限制。这里是一个可以用的例子，因为 bb 可以为空，就必须有 count(bb)。

其他一些具体的规定查 sql_reference。

7.12 修改表结构

一次给一个表增加多个字段

db2 "alter table tb_test add column t1 char(1) add column t2 char(2) add column t3 int"

drop 及创建主键
db2 "alter table kstar.tb_increment_balance_his drop PRIMARY KEY "
db2 "ALTER TABLE KSTAR.tb_increment_balance_his add PRIMARY KEY (branch_code,enddate,startdate,sub_branch_code,capital_account, currency_code)"

7.13 给一个表改名

db2 rename table tb_branch to tb_branch_bak

7.14 得到一个表或库的相关脚本

db2look -d gtjazb -t tb_branch -e -o out.log –p
可以得到和该表相关的全部脚本，但不包括 trigger.

也可以得到整个数据库的脚本

7.15 在对表操作的性能下降后对表做整理

db2 reorg table db2inst1.tb_brclose_stock_balance

db2 "reorg table kstar.tb_orders_total use TMPSYSTBS16"
指定系统临时表空间 tablespace

带 index 的：
db2 "reorg table kstar.tb_posi_stat index kstar.posi_stat_idx0 use TMPSYSTBS16"

对表的存储做整理,在表使用了一段时间后，特别是 update 和 delete 比较多时，数据在数据库内的存储会很乱，带来的直接的问题就是运行速度会很慢。在这种情况下就应该对这张表整理一下。

但要注意：
在做 reorg 时会要把数据全部放到系统临时表空间中，所以要注意把临时表空间开得足够大，同时还要考虑文件系统的限制。

db2 reorgchk on table db2inst1.tb_brclose_stock_balance
因为做一次 reorg 需要的时间会比较长，有时不知道是不是应该对该表重新

整理，可以先检查一下。

共会有 6 个指标，分 base 表和 index 各有 3 个指标，在每个的最后有一列，reorg，如果是’---‘则不需要做 reorg，如果有*则说明需要做 reorg。

db2 runstats on table db2inst1.tb_brclose_stock_balance

db2 runstats on table db2inst1.tb_brclose_stock_balance and detailed indexes all

detailed 选项是统计表数据的相关的物理分布。

这个是重新统计该表的数据信息，因为 DB2 在做查询优化的时候是根据表的统计信息来选择合适的执行计划的。如果数据在变动比较大的时候，就应该做一下表数据的重新统计。

做完 runstatus 后在将过程再 bind 一次。

使用脚本来完成：

db2 "select 'db2 reorgchk on table kstar.'||tabname from syscat.tables where tabschema='KSTAR' and tabname not like '%_BACKUP_%' with ur">reorgchk.sh

db2 "select 'db2 reorg table kstar.'||tabname from syscat.tables where tabschema='KSTAR' and type='T' and tabname not like '%_BACKUP_%' with ur">reorg.sh

db2 "select 'db2 runstats on table kstar.'||tabname || ' and indexes all' from syscat.tables where tabschema='KSTAR' and type='T' and tabname not like '%_BACKUP_%' with ur">runstatus.sh

db2 "select 'rebind package '||pkgname||';' from syscat.packages where pkgschema='KSTAR' with ur " >rebind.sql

7.16 查看语句的执行计划

dynexpln -d gtjazb -f test.sql -o test.out -g -z ';'
sql 语句放在 test.sql 中，结果输出到 test.out。

7.17 查看 sp 的执行计划

db2expln -c kstar -d zbdb -o test.out -p P2806220 -s 0
-p 是存储过程的 id

建过索引或 runstats 后,要重新绑定过程.
Db2 “rebind package P2806220”

7.18 更改存储过程的隔离级别

C 的存储过程：
在 bind 的时候指定隔离级别

SQL 的存储过程：
在构建的时候更改环境变量的设置.

7.19 取全部表的大小

```
drop PROCEDURE pr_gettbsize@

CREATE PROCEDURE pr_gettbsize
        (in i_tbsname varchar(100),
         in i_schema varchar(100),
         out o_status smallint)
         LANGUAGE SQL

Proc: BEGIN

   declare v_cnt integer default 0;
   declare v_tabname, v_space varchar(100);
   declare sqlcode integer default 0;
   declare stmt varchar(1000);

   declare c1 CURSOR for
      SELECT tabname from syscat.tables
      where      = upper(i_tbsname) and
              TABSCHEMA = upper(i_schema)
      order by tabname;

   OPEN C1;

   fetch_loop:
   loop
      fetch c1 into v_tabname ;
      if sqlcode <>0 then
          leave fetch_loop;
```

```
                END if;

                set stmt =      'insert into tb_tbsize '||
                                '(ndate ,tabname , tablen ,tabcnt, tb_schema,
tb_tbsname) '||
                                'SELECT current date, '||chr(39)||v_tabname||chr(39)||',
'||
                                '(SELECT sum(length) from syscat.columns '||
                                'where                  tabschema               =
upper('||chr(39)||i_schema||chr(39)||') and '||
                                'tabname = '||chr(39)||v_tabname||chr(39)||'), '||
                                '(SELECT count(*) from '||i_schema||'.'||v_tabname||'),'||

   chr(39)||i_schema||chr(39)||','||chr(39)||i_tbsname||chr(39)||
                                ' from syscat.tables '||
                                'where                  tabschema               =
upper('||chr(39)||i_schema||chr(39)||') and '||
                                'tabname = '||chr(39)||v_tabname||chr(39);
                prepare s1 from stmt;
                 execute s1;
          END loop fetch_loop;

          COMMIT;
END proc
@
```

8 DB2 系统管理

8.1 DB2 EE 及 WORKGROUP 版本的区别

Workgroup 版本和 EE 的区别主要在于两方面：

1、并行性 workgroup 比较差

2、主机连接（即和大型机的连接性能上）比较差

3、只能用于 4 个 CPU 以下的环境。

8.2 怎样判断 DB2 实例的版本号和修补级别？

用 db2level 命令。在 DB2 5.2 及以上版本中，在安装每个 DB2 实例时，即会 装入 db2level 程序。db2level 命令的输出提供了有关 DB2 实例的版本及

修补级别的详细信息。

命令输出如下所示：
DB21085I Instance "" uses DB2 code
release "" with level identifier
"" and informational tokens "", ""
and "".

例如：
DB21085I Instance "DB2" uses DB2 code release "SQL05020"
with level identifier "02070103" and informational tokens
"DB2 v5.2.0.30","c990717" and "WR21119".

下面解释以下这些信息：
= DB2 DB2 的实例名
= SQL05020 Release 号 05，Version 号 02，Module 号 0
= 02070103 内部使用的 DB2 版本号
= DB2 v5.2.0.30 实例的版本信息
= c990717 代码的级别信息
= WR2119 修补的级别信息
注：db2level 执行程序不能在不同的系统之间拷贝使用。
并且此程序只显示正式支持的修补级别信息。

对于 DB2 版本 5.0 和 2.0，可用如下方法获得版本信息：
OS/2：syslevel 命令

NT：查询 regedit 变量：HKEY_LOCAL_MACHINE | SOFTWARE | IBM | DB2 | DB2
universal database xx edition |
CurrentVersion

AIX：用 dump -H
$HOME/sqllib/lib/libdb2e.a

Solaris: cat 命令查看文件信息 /opt/IBMdb2/V5.0/cfg/bldlevel or
"ldd -s
$HOME/sqllib/lib/libdb2e.so |
grep engn|grep search|uniq"

HP: cat 命令查看文件信息
/opt/IBMdb2/V5.0/cfgbldlevel

8.3 DB2 客户端安装时选择语言

在缺省情况下，是按照操作系统的语言来安装的，如中文 windows 下装的就是中文的 db2.

如果要在中文 windows 中安装英文 db2,则要到命令行的模式下

setup /ien 即可

setup /icn 简体中文

setup /itw 繁体中文

可以看 readme.txt 中的内容

8.4 DB2 安装

8.4.1 AIX 中自动启动 db2

在 aix 的/etc/目录下创建文件 rc.db2，属性为

-rwxr-xr-- 1 root system rc.db2

内容为：

```
#!/bin/bsh
################################################################################
#
# Licensed Materials - Property of IBM
#
# 5648-B90
# (C) COPYRIGHT International Business Machines Corp. 1993, 1999
#
# 5648-B91
# (C) COPYRIGHT International Business Machines Corp. 1993, 1999
#
# 5648-B95
# (C) COPYRIGHT International Business Machines Corp. 1993, 1999
#
# 5648-B97
# (C) COPYRIGHT International Business Machines Corp. 1993, 1999
#
# 5648-B99
# (C) COPYRIGHT International Business Machines Corp. 1993, 1999
#
# All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
#
#################################################################
##############
#
# NAME: rc.db2
#
# FUNCTION: rc.db2 - auto start all DB2 services on boot
#
# USAGE: rc.db2
#
# NOTE: To avoid your system from failing on reboot, do not change this
#       file in any way.
#
#       This script is designed to be executed on reboot. Do the following
#       to enable auto-starting all DB2 Instances and Administration Server:
#
#       1) copy this file as /etc/rc.db2
#       2) add the following line to /etc/inittab:
#
#            db:2:once:/etc/rc.db2 > /dev/console 2>&1 # Start DB2 services
#
#################################################################
##############

# Default DB2 product directory
DB2DIR="/usr/lpp/db2_07_01"

if [ -x ${DB2DIR?}/instance/db2istrt ]; then
   ${DB2DIR?}/instance/db2istrt
fi

if [ -x /usr/lpp/db2_05_00/instance/db2istrt ]; then
   /usr/lpp/db2_05_00/instance/db2istrt
fi

if [ -x /usr/lpp/db2_06_01/instance/db2istrt ]; then
   /usr/lpp/db2_06_01/instance/db2istrt
fi


if [ -x ${DB2DIR?}/instance/dlfmlist ]; then
   ${DB2DIR?}/instance/dlfmlist 1>/dev/null 2>/dev/null
   if [ $? -eq 0 ]; then
      if [ -x ${DB2DIR?}/instance/dlfmstrt ]; then
```

```
                        ${DB2DIR?}/instance/dlfmstrt
                        exit 0
                    fi
        fi
fi

if [ -x /usr/lpp/db2_05_00/instance/dlfmlist ]; then
        /usr/lpp/db2_05_00/instance/dlfmlist 1>/dev/null 2>/dev/null
        if [ $? -eq 0 ]; then
                if [ -x /usr/lpp/db2_05_00/instance/dlfmstrt ]; then
                        /usr/lpp/db2_05_00/instance/dlfmstrt
                        exit 0
                fi
        fi
fi

if [ -x /usr/lpp/db2_06_01/instance/dlfmlist ]; then
        /usr/lpp/db2_06_01/instance/dlfmlist 1>/dev/null 2>/dev/null
        if [ $? -eq 0 ]; then
                if [ -x /usr/lpp/db2_06_01/instance/dlfmstrt ]; then
                        /usr/lpp/db2_06_01/instance/dlfmstrt
                        exit 0
                fi
        fi
fi

#--------------------------------------------------------------------
# Exit successfully.
#--------------------------------------------------------------------
exit 0
```

8.4.2 AIX 中用户使用 db2 的环境

在.profile 文件中加入以下语句：

```
# The following three lines have been added by UDB DB2.
if [ -f /home/db2inst1/sqllib/db2profile ]; then
        . /home/db2inst1/sqllib/db2profile
fi
```

export LIBPATH=$LIBPATH:/usr/lpp/db2_07_01/lib:/usr/lpp/db2_07_01/java12

DOC_LANG=en_US; export DOC_LANG

8.4.3 在 win98 下安装 db2 报 Jdbc 错误

在 Windows 98 下安装 db2 7.1 或其他版本，如果有 Jdbc 错误或者是 Windwos 98 不能启动，则将 autoexec.bat 中的内容用如下内容替换：

C:\PROGRA~1\TRENDP~1\PCSCAN.EXE C:\ C:\WINDOWS\COMMAND\ /NS /WIN95
rem C:\WINDOWS\COMMAND.COM /E:32768
REM [Header]

REM [CD-ROM Drive]

REM [Miscellaneous]

REM [Display]

set PATH=%PATH%;C:\MSSQL\BINN;C:\PROGRA~1\SQLLIB\BIN;C:\PROGRA~1\SQLLIB\FUNCTION;C:\PROGRA~1\SQLLIB\SAMPLES\REPL;C:\PROGRA~1\SQLLIB\HELP
IF EXIST C:\PROGRA~1\IBM\IMNNQ\IMQENV.BAT CALL C:\PROGRA~1\IBM\IMNNQ\IMQENV.BAT
IF EXIST C:\PROGRA~1\IBM\IMNNQ\IMNENV.BAT CALL C:\PROGRA~1\IBM\IMNNQ\IMNENV.BAT
set DB2INSTANCE=DB2
set CLASSPATH=.;C:\PROGRA~1\SQLLIB\java\db2java.zip;C:\PROGRA~1\SQLLIB\java\runtime.zip;C:\PROGRA~1\SQLLIB\java\sqlj.zip;C:\PROGRA~1\SQLLIB\bin
set MDIS_PROFILE=C:\PROGRA~1\SQLLIB\METADATA\PROFILES
set LC_ALL=ZH_CN
set INCLUDE=C:\PROGRA~1\SQLLIB\INCLUDE;C:\PROGRA~1\SQLLIB\LIB;C:\PROGRA~1\SQLLIB\TEMPLATES\INCLUDE
set LIB=C:\PROGRA~1\SQLLIB\LIB
set DB2PATH=C:\PROGRA~1\SQLLIB
set DB2TEMPDIR=C:\PROGRA~1\SQLLIB
set VWS_TEMPLATES=C:\PROGRA~1\SQLLIB\TEMPLATES

set VWS_LOGGING=C:\PROGRA~1\SQLLIB\LOGGING
set VWSPATH=C:\PROGRA~1\SQLLIB
set VWS_FOLDER=IBM DB2
set ICM_FOLDER=信息目录管理器

win
其实更大的可能是 path 太长而实际上没有生效，在命令行下看 path 中是否包含了 db2 的目录。

8.4.4 将一台机器上的数据库复制到另外一台机器

将一台机器上的数据库复制到另外一台机器上，只要数据库对象的定义以及表里面的数据能原封不动复制就可以了。
视图和触发器以及一些外键也要复制。

方法一：
首先用 db2look 生成 DDL 脚本，然后执行脚本生成所有数据库对象，然后用 db2move dbname load 即可！
这种方式有人用过。
方法二：
如果你的 DB2 数据库版本是 V7.2,补丁是 4 版本的话——恭喜：你就可以使用使用”db2recoke“进行复制数据库了。
这种方法没有得到验证。

8.4.5 在 WIN2000 下编译本地 sp 设置

在 sqllib\function\routine\sr_cpath.bat 这个文件的内容需要配置，如果没有装 vc5 或 vc6，则要安装一个。
这是因为 db2 的 sp 编译要使用 c 编译器。

8.5 启动支持远程管理数据库服务（db2admin）

db2admin stop
db2admin start

8.6 安装另一个 instance 要注意的地方

因为是第二个 instance，安装时原来可以缺省安装的变成要自己配置。

8.6.1 通讯配置

db2 缺省使用两个端口，50000 及 50001。
要配两个端口服务，
zbtcp 60000/tcp
zbtcpi 60001/tcp

更改 instance 配置，指定使用服务端口
db2 “update dbm cfg using SVCENAME zbtcp”

8.6.2 更改文件权限

-rwsrwxrwx 1 db2inst1 db2iadm1 25019 Sep 11 12:12 db2aud
-rwxrwxrwx 1 db2inst1 db2iadm1 4096 Sep 11 12:12 db2audit.cfg
lrwxrwxrwx 1 root system 36 Sep 11 12:12 db2chkau -> /usr/lpp/db2_0
7_01/security/db2chkau
-r-s--x--x 1 root db2iadm1 12526 Sep 11 12:12 db2chpw
-r-s--x--x 1 root db2iadm1 19972 Sep 11 12:12 db2ckpw
-rwxrwsrwx 1 db2inst1 db2iadm1 33027 Sep 11 12:12 db2flacc
注意按照上面的列表更改权限。
因为 db2 的用户是交由系统来管理的，db2ckpw 等需要用 root 的身份执行来检查权限。

8.7 Db2 的 C 编译报没有 licsence

修改/etc/vac.cfg
看是用的什么编译器,在里面加一个选项
options = -qlanglvl=extended,-qnoro,-qnoroconst,-qnolm

8.8 Db2 的进程管理

Db2 的进程，除了 db2wdog 外，其他程序都不应该是由 root 来管理。如果一个数据 down 过一次以后，如果有有关数据库的进程的父进程是 1 的话，都是死掉的进程，应该把它杀掉。否则有可能影响系统的稳定运行。

下面是 db2 的一些常见的程序：
Db2sysc db2 引擎

8.9 创建 Database

create database head using codeset IBM-eucCN territory CN;
这样可以支持中文。

8.10 Database 的备份

Db2 ”Backup db gtjazb user db2inst1 using db2inst1 to filebak\dbbacup”
这里是一个完整的备份 shell：

```
db2admin stop
db2stop force
db2admin start
db2start
db2 backup database zbdb to /dev/rmt0 with 2 buffers buffer 1024
```

说明：
/dev/rmt0 是磁带机，直接备到磁带机上。

8.11 Tablespace

8.11.1 创建临时表空间

用户临时表空间

```
db2 “CREATE USER TEMPORARY TABLESPACE TMPUSRTBS16
        PAGESIZE 16 K
        MANAGED BY SYSTEM
        USING ('/gtja_emc/gtjadb/NODE0000/SQL00001/TMPUSR16001.00')
        EXTENTSIZE 32
        PREFETCHSIZE 64
        BUFFERPOOL USER16KBP;”
```

注意目录下不能有任何文件，所以如果是建在文件系统的 mount 点上的时候，就是 lost 的目录也不能要。

只有有了用户临时表空间,才可以使用临时表。

系统临时表空间
drop tablespace TMPSYSTBS16;

```
CREATE SYSTEM TEMPORARY TABLESPACE TMPSYSTBS16
        PAGESIZE 16 K
    MANAGED BY DATABASE
     USING (DEVICE '/dev/rutemptbs1' 5400M,
            DEVICE '/dev/rutemptbs2' 5400M
            )
        EXTENTSIZE 32
        PREFETCHSIZE 128
        BUFFERPOOL USER16KBP;
```

8.11.2 将 Tablespace 授权给用户使用

db2 "GRANT USE OF TABLESPACE TMPUSRTBS16 TO PUBLIC"
请注意，不授权的话，用户使用将会出错的。
只有系统临时表空间可以不授权.

8.11.3 看 Tablespace 信息

db2 list tablespaces
db2 list tablespaces show detail
db2 list tablespace containers for X(tablespace id)
db2 list tablespace containers for X show detail

只有 DMS 可以增加 containers.

8.11.4 去掉 tag

--以下命令是为了去除 TABLESPACE 的 TAG, 仅在创建 tbs 出现错误已经占用时使用
--需要 su - root
db2untag -f rparatbs1

8.12 手工做数据库别名配置及去除该别名配置

db2 catalog db gtjazb on /gtjadb2(目录)
db2 uncatalog db gtjazb

8.13 手工做数据库远程(别名)配置

db2 catalog tcpip node node1 remote 172.28.200.200 server 50000
db2 catalog db head as test1 at node node1

然后既可使用：
db2 connect to test1 user … using …
连上 head 库了

8.14 停止启动数据库实例

db2start
db2stop (force)

8.15 连接数据库及看当前连接数据库

连接数据库
db2 connect to head user db2inst1 using db2inst1

当前连接数据库
db2 connect

8.16 停止启动数据库 head

db2 activate db head
db2 deactivate db head
要注意的是，如果有连接，使用 deactivate db 不起作用。
如果是用 activate db 启动的数据库，一定要用 deactivate db 才会停止该数据库。（当然如果是 db2stop 也会停止）。
使用 activate db，这样可以减少第一次连接时的等待时间。
Database 如果不是使用 activate db 启动而是通过连接数据库而启动的话，当所有的连接都退出后，db 也就自动停止。

8.17 查看及停止数据库当前的应用程序

查看应用程序：

db2 list applications show detail

授权标识 | 应用程序名 | 应用程序句柄 | 应用程序标识 | 序号# | 代理程序 | 协调程序 | 状态 | 状态更改时间 | DB 名 | DB 路径| | 节点号 | pid / 线程

其中：1、应用程序标识的第一部分是应用程序的 IP 地址，不过是已 16 进制表示的。

2、pid/线程即是在 unix 下看到的线程号。

停止应用程序：

db2 "force application(236)"
db2 “force application all”

其中:该 236 是查看中的应用程序句柄。

db2 list applications|grep -i trans_db | wc –l
看有多少个 trans_db 的连接。

Grep –i I 是不区分大小写。
Wc 计算个数，-l 是计算行数，即进程数。

8.18 查看本 instance 下有哪些 database

db2 LIST DATABASE DIRECTORY [on /home/db2inst1]

8.19 查看及更改数据库 head 的配置

设置的时候要注意，一个 database 可用的最大内存数只有 2G,配置后要计算一下使用的内存数，注意不要超过 1.6G，还要留一些内存给数据库其它的来用。

如果要使用 2G 以外的内存，将 NUM_ESTORE_SEGS 参数由 0->1，这样就可以使用 2G 以外的内存做为二级缓存。

请注意，在大多数情况下，更改了数据的配置后，只有在所有的连接全部断掉后才会生效。

查看数据库 head 的配制

db2 get db cfg for head

更改数据库 head 的某个设置的值

8.19.1 设置使用 2G 以外的内存

db2 update db cfg for head using NUM_ESTORE_SEGS 1

参数由 0->1，这样就可以使用 2G 以外的内存做为二级缓存。

8.19.2 更改 Buffer pool 的大小

db2 update db cfg for head using BUFFPAGE 20480

但如果要上面这个设置值生效，需要查看一下 SYSCAT.BUFFERPOOLS 这张表，里面的 bufferpool 的 npages 字段值至少有一条记录的为-1。

db2 alter bufferpool ibmdefaultbp size -1

Bufferpool 的大小这个指标是个很重要的指标，它实际上分配的一个内存区，数据的绝大部分操作都是在 bufferpool 中来进行。如果服务器仅仅是给这个数据库用，bufferpool 的大小应该要占到机器总物理内存的 50%-75%的比例。

如果内存够的话，理论上 bufferpool.是越大越好。

8.19.3 更改 dbheap 的大小

db2 update db cfg for head using DBHEAP 4096

该值是对 db 的。

Dbheap>catalogcache_sz+logbufsz

8.19.4 改 catalogcache 的大小

db2 update db cfg for head using catalogcache 2048

和表的数量和字段数量有关，如果表及字段较多，最好将该指标改大一些。该值的大小可以开为和建表 script 的大小相当，再稍大一点。当然，如果有动态创建表的话，根据实际情况可能要开得更大一些。

8.19.5 改事务 buff 的大小

db2 update db cfg for head using LOGBUFSZ 512

该指标对数据库并行比较有影响，如果并发的较多，最好将该指标

改大一些。

数据库写日志先写在这个内存中，如果没有写满，每秒自动刷新一次，内存用满也会刷新一次，开的值最好不要让它被用满而被迫写到磁盘。

因此，如果一个事务的日志比较多时，最好能开大一些。

8.19.6 改工具堆大小

UTIL_HEAP_SZ
这个指标值是用于对 import,export,load 等工具来分配内存的。

8.19.7 改排序堆的大小

db2 update db cfg for head using SORTHEAP 2048

将排序堆的大小改为 2048 个页面，查询比较多的应用最好将该值设置比较大一些。

该指标值是对每个连接分配的内存，如果连接数比较多，注意不要开得太大。如果看到了 sort overflow 的话，可以将改值调大一些。
这个内存是只在用的时候才申请，平时是不会申请的。

8.19.8 改 stmtheap 的大小

db2 update db cfg for head using STMTHEAP 4096

该指标值是对每个连接分配的内存，如果连接数比较多，注意不要开得太大。
该数据值和解释语句有关，如果太小，可能比较大的语句会解释不了。
这个内存是只在用的时候才申请，平时是不会申请的。

8.19.9 改事务日志的大小

db2 update db cfg for head using logfilsiz 40000

该项内容的大小要和数据库的事物处理相适应，如果事物比较大，应该要将该值改大一点。否则很容易处理日志文件满的错误。这是指单个文件的大小。

日志文件的大小，大概是同时没有 commit 的数据量的两倍大小。

还有个两个指标值是指日志文件的个数

LOGPRIMARY 基本日志文件数
LOGSECOND 备用日志文件数

在正常情况下，只用到基本日志文件,则日志大小为 logfilsiz* LOGPRIMARY

在基本日志文件都用完而不够用的情况下，就会去使用备用日志文件。

备用日志文件在事务 commit 以后，系统就会自动的将备用日志文件释放掉。

db2 update db cfg for head using LOGPRIMARY 3
db2 update db cfg for head using LOGSECOND 2

db2 7.2 版本中, LOGPRIMARY+ LOGSECOND<=128

马宏伟建议，日志文件开多一点，而每个日志文件开小一点，这样性能会好一些。好象 AIX 对大文件的管理不是很好。

8.19.10 改锁的相关参数的大小

db2 update db cfg for head using LOCKLIST 40000

这个是整个 db 的最大锁资源占的内存。锁资源的消耗是每条共享锁占 36 个字节，独占锁占用 72 个字节。锁资源的大小要考虑应用来设置。

db2 update db cfg for head using MAXLOCKS 10

这个参数是设定单个应用程序能够使用的最大锁资源，这是个百分比的值。实际上单个应用程序能够使用的锁资源的大小为

LOCKLIST* MAXLOCKS

db2 update db cfg for head using LOCKTIMEOUT 60

这个参数是设定应用程序锁等待的最大时间。单位是秒，这个值的设定要比较适当，对并发较多的情况下，锁等待可能是不可避免的，如果设定不适当，可能会发生太多的 time out 错误。

db2 update db cfg for head using DLCHKTIME 10000

这个参数是设定系统检测死锁发生的时间，单位是毫秒，不要设得太小，耗系统资源且没太多必要。

8.19.11 出现程序堆内存不足时修改程序堆内存大小

db2 update db cfg for head using applheapsz 4096

该值不能太小,否则会没有足够的内存来运行应用程序。

但也不能太大，因为该内存是对每个连接分配的,这个的内存是应用程序使用得最多和频繁的内存。

这个内存是只在用的时候才申请，平时是不会申请的。

8.19.12 NUM_IOCLEANERS 及 NUM_IOSERVERS 数量设置

NUM_IOCLEANERS 设为 cpu 即可
NUM_IOSERVERS 设为 cpu 的个数+2

8.19.13 成组 commit 设置 MINICOMMIT

马宏伟说就是设为 1 最好。可能是因为 db2 设计还没有特别好。

8.19.14 设置连接数的相关参数 MAXAPPLS

MAXAPPLS 按需要设置,对于目前的应用为 500
AVG_APPLS 平均活动应用程序数,数据库会根据这个数值来采取不同的资源分配方式。50 可能比较合理。

8.19.15 设置包缓冲区 PCKCACHESZ

这块区域是放置程序包的，最好不要让它有溢出。
缺省是 MAXAPPLS*8*4K
但我发现了溢出，聂华建议开大一点，它是对整个 database 的,考虑设到 MAXAPPLS*16*4K(32M)

8.20 日志管理

由于我们现在经常发生数据库日志满的,由没有什么办法能够比较快速的找到是哪个数据库的应用占用了比较多的日志而没有 commit

看 db2dialog.log 虽然可以看到是哪个应用最后将日志写满，但不能确认是哪个应用写得比较多

8.20.1 更改日志文件的存放路径

db2 update db cfg gtjazb using NEWLOGPATH /dev/rloglv
在下一次启动 database，就会将该日志放到新的指定地方，指定

的路径可以是文件系统，也可以是 Raw device，/dev/ 的都是裸设备。

8.20.2 监控应用程序日志使用情况

db2 update dbm cfg using DFT_MON_UOW on
再用
db2 get snapshot for applications on head
可看到应用程序的各种信息，包括日志使用情况，commit,rollback 等等。

8.20.3 循环日志和归档日志

循环日志比较容易管理，但如果是 7*24 的应用，如果要备份，循环日志不能是 online backup。所以如果是 7*24 的 oltp 应用只能采用归档日志。

归档日志在 online backup 时也要注意，从开始备份到备份结束的这个日志文件不能删除，否则会到恢复的时候恢复不回来。

8.20.4 循环日志日志满的原因

循环日志因为是日志文件可以循环使用的，使用起来比较方便。它导致满有两种可能：

1、日志文件全部被用完而没有。这种情况导致的日志满可能很小。

2、由于循环日志用了一个循环后，辅助日志也用完了。如果这个时候，以前使用这个日志的程序还占用着着段日志而没有 commit，这时候数据将会申请不到日志空间而报日志满。

8.21 查看及更改数据库实例的配置

查看数据库实例配置

db2 get dbm cfg

更改数据库实例配制

8.21.1 打开对锁定情况的监控。

db2 update dbm cfg using dft_mon_lock on

8.21.2 更改诊断错误捕捉级别

db2 update dbm cfg using diaglevel 3

0 为不记录信息
1 为仅记录错误
2 记录服务和非服务错误
缺省是 3，记录 db2 的错误和警告
4 是记录全部信息，包括成功执行的信息

一般情况下，请不要用 4，会造成 db2 的运行速度非常慢。

8.21.3 更改最大代理数

db2 update dbm cfg using MAXAGENTS 500

还有一个参数是最大并行代理数

db2 update dbm cfg using MAX_COORDAGENTS 500

最大代理数的设置要考虑一下内存问题，据马宏伟估计，如果同时开 1000 个代理，大概总的内存要使用 5、600M 内存，其中 AIX：DB2 的占用比率大概在 8:2 或 7:3。

此外，对 cpu 的压力也不小，用于切换时间对 6 个 cpu 要用掉 40%-50%(不知道是一个 cpu 的还是整个的)

8.22 db2 环境变量

看 db2 全部的环境变量

db2set -lr

文档在

db2_doc\db2d0\frame3.htm 查 registry 关键字

db2 重装后用如下方式设置 db2 的环境变量,以保证 sp 可编译
将 set_cpl 放到 AIX 上, chmod +x set_cpl, 再运行之

set_cpl 的内容

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="xlc_r -g \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
```

-bE:SQLROUTINE_FILENAME.exp -e SQLROUTINE_ENTRY \
-o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -lc -ldb2"

db2set DB2_SQLROUTINE_KEEP_FILES=1
db2set DB2_STRIPED_CONTAINERS=YES
db2set DB2_HASH_JOIN=Y

db2set DB2MEMMAXFREE=8000000
db2set DB2MEMDISCLAIM=yes
(上面两个参数是配对使用的，第一个参数是设置 Aix 系统管理直到一个进程有 8M 的空闲内存时，才真正释放内存，否则还是给该进程保留已经不用的内存空间，以备重用；沈刚说要不就设 8M 要不就不设，据说 IBM 实验室只测试了这个值，使用其他值不知道会有什么后果；
第二参数是个开关变量，设为 yes 第一个参数才有用)

db2set DB2_MMAP_WRITE=NO(和 EEE 版本有关)
db2set DB2_MMAP_READ=NO(和 EEE 版本有关)
db2set DB2_RR_TO_RS=ON
db2set DB2_FORCE_FCM_BP=NO(和 EEE 版本有关)
db2set DB2COMM=tcpip
db2set DB2CODEPAGE=1386
db2set DB2_PARALLEL_IO=* (指定 IO 对所有的 tablespace 都是并行处理，如果要指定 tablespace,将*改为 tablespace 的 id 即可,但只是对 RAID5 有效)
db2set DB2AUTOSTART=TRUE

DB2SET DB2_SQLROUTINE_PREPOPTS=CS|RR|RS|UR
Db2set DB2_SQLROUTINE_PREPOPTS="blocking all"
（设置编译的选项）

db2set DB2MAXFSCRSEARCH=10
(db2 数据库在缺省的情况下，只检查前 5 个每 500 个页面，如果前 2500 个页面都没有空间，则自动的变为 append 模式，这样即使中间有空间，只要最后的页面后没有空间了，系统也会报表空间满，删除数据也没用，这样中间的空间要到 reorg 后才能回收使用，这个参数就是 改为 检查 500*10 个页面的空间)

8.23 db2 命令环境设置

db2=>list command options
db2=>update command options using C off--或 on，只是临时改变
db2=>db2set db2options=+c --或-c，永久改变

8.24 改变隔离级别

交互环境更改 session 的隔离级别，

db2 change isolation to UR

请注意只有没有连接数据库时可以这样来改变隔离级别。

8.25 管理 db\instance 的参数

get db cfg for head(db)
get dbm cfg(instance)

8.26 升级后消除版本问题

db2 bind @db2ubind.lst
db2 bind @db2cli.lst

8.27 查看数据库表的死锁

再用命令中心查询数据时要注意,如果用了交互式查询数据,命令中心将会给所查的记录加了 s 锁.这时如果要 update 记录,由于 update 要使用 x 锁,排它锁,将会处于锁等待.

首先,将监视开关打开

db2 update dbm cfg using dft_mon_lock on

这是更改 instance 一级的参数，还可以使用 switch 只打开 session 一级。

db2 update monitor switches using lock on

快照

db2 get snapshot for Locks on gtjazb >snap.log
tables
bufferpools
tablespaces
database

然后再看 snap.log 中的内容即可。

对 Lock 可根据 Application handle（应用程序句柄）看每个应用程序的锁的情况。

监视完毕后，不要忘了将监视器关闭

db2 update dbm cfg using dft_mon_lock off

8.28 查看数据库的事件

先创建事件：

db2 "create event monitor whtmdeadlock for deadlocks write to file '/home/db2inst1/user/wht' "

再将事件监控打开：

db2 "set event monitor whtmdeadlock state=1"

然后在该目录下会有一个.evt 的文件。

然后使用 ibm 事件分析器可以看到各项内容。

或者用命令行形式：

db2evmon -path /home/db2inst1/user/wht > connect.log

8.29 数据库性能下降后做 runstats 及 rebind 包。

在表（大表）的记录的条数有 30%的变化以后，就应该做一次 runstats 从方便管理的角度，直接从 syscat.tables 中建 script。

db2 “select 'runstats on table db2inst1.' || tabname || ' and indexes all' from syscat.tables where tabschema='DB2INST1' and type='T'” >stats.sql

将 stats.sql 中的多余的信息删除后，运行该脚本。

db2 -tvf stats.sql

请注意在做 runstats 时，将其他应用全部断开。

为了提高速度，可以调整一下 database 的参数，将 applheapsz，sortheadp,stmtheap 先扩大 10 倍，但注意做完 rebind 后将参数恢复回来。

注意：在作完 runstats 后，相关的 sp 如果不做 rebind，性能不会有任何变化，不会使用新的统计数据来计算。

db2 “select 'rebind package '||pkgname from syscat.packages where pkgschema='DB2INST1’ “ >rebind.sql

将 stats.sql 中的多余的信息删除后，运行该脚本。

db2 -tvf rebind.sql

8.30 修复诊断数据库 db2dart 的使用

db2dart gtjazb
db2dart gtjazb /T 对一个表做诊断(从这里可以得到该表的 index 的 id 号)
db2dart gtjazb /MI 对一个表的 index 做修复

db2dart 还有一些功能，但 db2dart 是个比较深的用法，一般情况下不要使用。具体可以看帮助。

8.31 获取数据库的信息 db2support 的使用

db2support /gtja_emc/dream/ -d zbdb

8.32 分析 DB2diag.log 的方法

分析方法在 file:///C:/db2/db2_doc/db2p0/frame3.htm#index 中的 Part 3. Appendixes及相关链接中可以看到。
或者在信息中心/书籍/疑难解答/Troubleshooting Guide 的 Part 3 Appendixes 及相关链接中可以看到。

8.32.1 Obj={pool:2;obj:10;type:0}含义

The pool ID is 2：表明 table space ID 是 2。
The object ID is 10：表明 table ID 是 10。
The object type is 0：表明是数据 (Object type 1 是索引)

8.32.2 错误信息所在位置 1(errno)

2001-12-05-15.30.09.827998 Instance:gtjadb Node:000
PID:104076(db2agent (ZBDB)) Appid:0A64117B.057F.011205073006
oper_system_services sqloopenp Probe:36 Database:ZBDB

errno: 0000 001a
这个信息 /usr/include/errno.h 文件中，换为十进制后可以在这个文件中找得到。

8.32.3 错误信息所在位置 1(FFFF nnnn 或 nnnn FFFF)

file:///C:/db2/db2_doc/db2p0/frame3.htm#index中的Part 3. Appendixes

如果有 code 在这里没有的话，则可以和 db2 客户服务部联系。

如果是 FFFF nnnn 则可以直接用 nnnn 去看错误信息。
如果是 nnnn FFFF ，则将 nnnn 的前两位和后两位颠倒后再去查。

9 DB2 一般问题

9.1 有关锁的知识

在 db2 get snapshot for Locks on head >snap.log 时，经常会看到一些锁的类型，不是很明白：

S：share 锁，共享锁，加上后，其他应用程序可以读，但不可 update，每个占用 36 个字节。

X：exclusive，独占锁，加上后，其他程序除非使用 UR 隔离级，否则不可读。每个占用 72 个字节。

**** 表锁

IN （Intenet None）不需要行锁
IS （Intent Share） 需要行锁配合
IX (Intent eXclusive) 需要行锁配合
SIX (Share with Intent exclusive）需要行锁，共享排他锁
S （Share） 不需要行锁配合
U （Update） 不需要行锁配合
X （eXclusive） 不需要行锁配合
Z （Super Exclusive）不需要行锁配合

***** 行锁
S （Share）共享锁
U (Update) 更改锁
X (eXclusive) 排他锁
W (Weak eXclusive)弱排他锁
NS (Next Key Share) 下一行共享锁
NX (Next Key eXclusive)下一行排他锁
NW (Next Key Weak eXclusive)下一行弱排他锁

锁的叠加情况：

S 锁和 S 锁是可以多个程序对一个对象加。

X 锁和 S 锁不兼容。

？？

9.2 有关锁的对象知识

Object Type= Row (行锁)
Object Type= Table (表锁)
Object Type= Key Value
Object Type= Internal P Lock
Object Type= Internal V Lock
Object Type= Internal C Lock

还有个 end of table，不懂什么意思？

10 DB2 疑难问题

10.1 建 SP 时 drop 不掉怎么办

系统提示：

SQL0970N 系统试图写至只读文件

原因：

这是 DB2 的一个 Bug

解决办法：

打上 db2 fix package 3 即可。

10.2 C 的过程老是出现时间戳问题？

现象：

sqlcode 报-818 错误，存储过程不能运行

原因：

原因一：

一个 C 的过程在一个 instance 下的几个 database 下创建时，由于 db2 将全部的 C 的过程的 dll 文件是按照 instance 来管理，而不是按照 database 来管理。这样如果将一个 C 的过程在多个 database 上创建时，如果不换文件名，这时在一个 database 创建就会导致在其他 database 上的该过程无效。

原因二：

据沈刚介绍，C 的存储过程使用的表被 drop 后在重建，也会造成这种现象。

原因三：

C 过程的名称要注意保证前 8 位是唯一的。是由于 syscat.packages 的系统表中的 pkgname 这个字段只有 8 位长，而 C 的过程的名称就是作为 pkgname 的，所以 C 过程的前 8 位要保证是唯一的，否则会互相覆盖，只有后建的过程才可以用。

其他原因还不知道

解决办法：

原因一：

在 function 中按不同 database 来创建目录。

在写创建脚本时，注意将不同文件放在个 database 对应的目录下。

原因二：

rebind 该存储过程。

原因三：

保证前 8 位是唯一

10.3 FOR CURSOR 问题？

现象：1、pr_clear_create_deliver 单独调用时费用计算正确，但由总控来调用时只计算了第一个费用

2、有次在一个 sp 开始的地方删除了一部分数据后，使用了 commit,然后在使用 for 的 cursor，编译的时候通过，执行时 for 的循环执行了一次后在下一个循环后就报游标被关闭。后来使用分段注销后发现，只要去到 commit，程序即可执行成功。

原因：

不知道，我估计这是 DB2 的又一个 Bug。

解决办法：

1、将 For 定义的 cursor,改为显示的定义 Declare cursor 后计算正确。

2、不在前面用 commit,或将 for 改为 declare cursor

10.4 数据库启动资源冲突问题

现象：DB2start 时报 SQL1072C 错误，db2 管理资源不一致
原因：
据沈刚说，只有一种可能
直接用 kill 语句杀了进程，这个进程可能是 db2inst1 的管理进程，db2agent 的进程也可能会出现这种情况。
但 20010801 晚上是用的 db2stop 停的进程，当时就起不来，报资源正在释放，到 20010802 启动就出现这种情况。沈刚说这种情况不可能，但现实是出现了。

解决办法：
1、将机器重起。
2、使用 ipcrm 清除共享内存区。
使用 ipcs 看数据库占用的共享内存、信号灯及消息队列。
然后再使用 ipcrm 清除掉，好象共享内存不全部清除也可以。

10.5 DB2stop 不下去问题

现象：先 FORCE APPLICATIONS ALL
使用 db2stop force ,8 月 9 号晚上 19:03:40 秒开始，直到 20 时 30 分 30 秒还没有停下来。最后只好重起机器。
原因：
根据目前的看法，有可能是有比较大的事物，因为 db2 数据库对回滚的速度比较慢，比如有几个 G 的日志要回滚的话就会要回滚好几个小时。
不知道
处理方法：
在以前也出现过一次，当时问沈刚，结果是重起机器。
据马宏伟介绍,可以将数据库引擎程序 db2sysc 杀掉,但杀之前必须要看 cpu 和 io 的情况,如果 cpu 和 io 占用很大,则不能杀掉该进程。

10.6 数据库日志满问题

现象：大概在 8 月 9 号 18：30 分左右数据库 head 报日志满，这时操作都不能做。
等了大概 30 分钟，日志仍然满。
日志文件共有 7 个，每个 200M。使用
db2 get snapshot for applications on head 查看，将记录下来的全部在占用的日志空间，加起来占用了 13962 个 byte。1.4 个 G 的日志空间不知道被谁使用

原因：

不知道。

解决方法：

目前是 db2stop 后在 db2start。但 db2stop 常不能停成功。

10.7 Force Application 导致 instance 崩溃问题

现象：有时候由于程序编写有错误导致死循环，这时使用 force application 强制关掉该 handle，经常出现 instance 的崩溃。邱炜也碰到过。

原因：

不知道，据沈刚说，这种情况不可能。

解决方法：

没有，db2start 后继续工作。

据马宏伟说,碰到过这种情况,但也没什么办法。

10.8 存储过程名称和过程运行有关的问题

现象：一个存储过程，单独 call 执行没问题。但是如果通过一个应用调用就会产生一个 1131 的错误。

最后，将该存储过程换了一个名字后执行正常。张巍亲历。

原因：

不知道。

解决方法：

换个名字。

10.9 看 Db2diag.log 中的内容

如果是 ffff f1ca 的类似东西，后四位 ibm 是有个文档对应的，可以查。文档名是 sqlzrc.h。但好象是 ibm 的保密文件一样不肯给。

F1ca -→sql0980

F401 → sql0902

10.10 decimal 除法的问题，Db2 做 sum 时有 bug(实际上不是)

可以看例子：

```
drop table tmptestsum;
create table tmptestsum(
c1 decimal(19,4),
c2 decimal(19,4)
);

insert into tmptestsum
  values
  (12.5643,
  17.0);

insert into tmptestsum
  values
  (12.5643,
  17.0);

select sum(c1),c2,c2/sum(c1) from tmptestsum group by c2;

select sum(c1),c2,sum(c1)/c2 from tmptestsum group by c2;

select sum(c1),c2,sum(c1)/17.0 from tmptestsum group by c2;
```

--第二个查询出来的 sum(c1)/c2 出来的值怎么会变成 int?????!!!!!!

改为

```
select sum(c1),c2,decimal(sum(c1))/c2 from tmptestsum group by c2;
```

即可。

可以将下面的语句也执行一次：

```
db2 "select sum(c1),c2,sum(c1)/17.0 from tmptestsum group by c2"
db2 "select sum(c1),c2,sum(c1)/17.00 from tmptestsum group by c2"
db2 "select sum(c1),c2,sum(c1)/17.000 from tmptestsum group by c2"
db2 "select sum(c1),c2,sum(c1)/17.0000 from tmptestsum group by c2"
db2 "select sum(c1),c2,sum(c1)/17.00000 from tmptestsum group by c2"
```

可以看到结果，decimal 的除法要注意小数位的计算

被除数下标为 1　　除数下标为 2　　结果下标为 3

全部位数为 p,小数为 s

则
s3=31-p1+s1-s2
计算出来的 s3 不能为负数，否则会报错的。

Decimal(19,4) 在做 sum 时，sum 会自动升级为 decimal(31,4)
S3=31-31+4-4=0 (这样就小数位变成 0 了)

10.11 case 的问题

可以看例子：

```
echo build tmp_tb_currency;
drop table tmp_tb_currency;
create table tmp_tb_currency
(
   currency_code              char(1) not null,
   --   货币代码
   currency_name              char(20),
   --   货币名称
   currency_unit              char(10),
   --   货币单位
   currency_symbol            char(6),
   --   货币符号
     constraint pk_currency    primary key (currency_code)
   --  货币代码
);

echo tmp_testaa;
drop table tmp_testaa;
create table tmp_testaa
(
   currency_code              char(1) not null
);


drop FUNCTION test_get_name;
CREATE FUNCTION test_get_name
(c_code varchar(1))
 --货币代码
RETURNS varchar(20)
LANGUAGE SQL
READS SQL DATA
```

```
NO EXTERNAL ACTION
DETERMINISTIC
RETURN select currency_name from tmp_tb_currency where currency_code=c_code;


values(' 初始数据输入：货币代码表');
 delete from tmp_tb_currency ;
 insert into tmp_tb_currency (currency_code,currency_name,currency_unit,currency_symbol)
   values('0','人民币','元','￥'),
          ('1','美圆','元','$'),
          ('2','港币','元','HK$');

 delete from tmp_testaa;
 insert into  tmp_testaa(currency_code)
   values('0'),
          ('1'),
          ('2'),
        ('a');
```

--这个语句应该是出来四条记录,而只给了我 3 条,在打 fix package 4 前后结果不一样

--请注意 left join,不做 left join 都是对的

--而做了 left join 后就不对了

```
select case when c.sa = 'B' then '全部' else test_get_name(c.sa) end from
      (select case when currency_code = 'a' then 'B' else currency_code end as sa from tmp_testaa) as c
   left join tmp_tb_currency b on c.sa=b.currency_code ;
```

北京的王东明告诉我在 NT 平台上在打了 Fix Pack 4 后运行也是正确的。

改为这样后肯定是对的：

```
select case when tmp.sa = 'B' then '全部' else test_get_name(tmp.sa) end
from
table (select * from
          ((select case when currency_code = 'a' then 'B' else
currency_code end as sa from tmp_testaa) as c
      left join tmp_tb_currency b on c.sa=b.currency_code )) as tmp ;
```

10.12 一个较复杂 sql 语句错误

这个例子在 20011206 发给张巍
例子：

```
drop function test_getdatetime;

CREATE FUNCTION test_getdatetime (time_stamp char(26))
RETURNS char(16)
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
NOT DETERMINISTIC
RETURN                                          substr(time_stamp,1,4)
||substr(time_stamp,6,2)||substr(time_stamp,9,2)||

substr(time_stamp,12,2)||':'||substr(time_stamp,15,2)||':'||substr(time_stamp,18,2)
;



--
=============================================================
===
-- 8、无委托成交异常明细  TB_DEAL_NO_ORDER
--
=============================================================
===
values('test_tb_deal_no_order');
drop table test_tb_deal_no_order;
create table test_tb_deal_no_order
(
    CUR_NO              int                             not null
    generated always as identity(start with 1, increment by 1, no cache),
    --记录号
    OCCUR_DATETIME      CHAR(26)                        ,
    --异常发生时间
    BRANCH_CODE         CHAR(4)                         ,
    --营业部代码    2001.8.27 增加
    GROUP_CODE          CHAR(4)                         ,
    --群组代码
    CREATOR             VARCHAR(16)                     ,
    --创建者
    HANDLE_FLAG         CHAR(1)                         ,
```

```
        --处理标志
        HANDLE_PERSON          CHAR(16)                    ,
        --处理人员
        HANDLE_SUMMARY
VARCHAR(50)                    ,
        --处理备注
        MARKET_CODE            CHAR(1)                     ,
        --市场代码
        SEAT_CODE              CHAR(8)                     ,
        --席位号
        ALT_SERIAL_NO          INT                 ,
        --采集流水号
        ORDER_STOCKHOLDER
CHAR(15)                       ,
        --股东代码
        STOCK_CODE             CHAR(8)                     ,
        --证券代码
        STOCK_NAME             CHAR(12)                    ,
        --证券名称（证券简称）
        STOCK_TYPE             CHAR(1)                     ,
        --证券类别
        CONTRACT_NO            CHAR(10)                    ,
        --合同号
        MATCH_NO               CHAR(20)                    ,
        --成交编号
        BUSINESS_FLAG          CHAR(1)                     ,
        --买卖方向
        MATCH_QTY              INT                         ,
        --成交数量
        MATCH_PRICE            DECIMAL(12,4)               ,
        --成交价格
        MATCH_TIME             CHAR(8)                     ,
        --成交时间
        MARK                   CHAR(2)                     ,
        --标志
        PARA_CODE              INTEGER                     ,
        --参数代码
        MSGGRADE                                   CHAR(1)
default '1' ,
        --消息级别
        CREDITLEVEL                                CHAR(1)
default '1' ,
        --消息可信度
        ERROR_TYPE                                 CHAR(1)
```

```
default '0' ,
        --错误类型      （5.28 增加）
        ERROR_LEVEL                                     CHAR(1)
default '0' ,
        --错误级别
        RESULT_ID                                       CHAR(30)
default 'tb_deal_no_order',
        --结果 ID （专项审计专用）
        primary key (CUR_NO)
    );

    echo build test_tb_branch;
    drop table test_tb_branch;
    create table test_tb_branch
    (
      branch_code               char(4)         not null,
      --  营业部代码
      branch_abbrev             char(30),
      --  营业部简称
      branch_name                   char(60),
      --  营业部全称
      sub_company_code          char(4),
      --  所属分公司代码
      manager                   char(20),
      --  负责人
      addr                      char(60),
      --  地址
      tel                       char(30),
      --  电话号码
      post_zip                  char(10),
      --  邮政编码
      run_status                char(1)         default '0',
      --  营业部上线状态  字典  2007
      branch_version            char(1)         default '0',
      --  柜面版本
      unit_type                 char(1)         default '0',
      --单位类型   0  营业部  1  核算单位  9  其它
      --国泰君安对公司内的单位都是统一编号
      remarks                   char(200),
      --  备注
      constraint    pk_branch primary key(branch_code)
      --  营业部代码
    );
```

```
    --
==========================================================
===
    --      Table: TB_GROUP_MEMBER_JK
    --
==========================================================
===
    drop table test_TB_GROUP_MEMBER_JK;
    create table test_TB_GROUP_MEMBER_JK
    (
        CREATOR                 CHAR(16)                not null ,
        GROUP_CODE              CHAR(4)                 not null ,
        GROUP_TYPE              CHAR(4)                 not null ,
        GROUP_MEMBER_VALUE  CHAR(30)                    not null ,
        constraint PK_1125 primary key (CREATOR, GROUP_CODE,
GROUP_MEMBER_VALUE)
    )in para_tabspace;



    with tttt as ( select group_member_value from test_tb_group_member_jk )
    select *
      from test_tb_deal_no_order a
       left outer join test_tb_branch b
      on a.branch_code=b.branch_code where
      SUBSTR(test_GETDATETIME(          A.OCCUR_DATETIME),1,8)
>='20001206'                                                    and
SUBSTR(test_GETDATETIME(A.OCCUR_DATETIME),1,8) <= '20011206'
   and A.branch_code in (select * from tttt);
```

10.13 编译语句挂起的现象

这个例子在 20020415 发给张巍
例子：

```
    drop table tb_dbf_circulate_stk_all_ha;
    create table    tb_dbf_circulate_stk_all_ha
    (
      tradedate                 char(8)             not null,
      -- 交易所下发日期
      stockholder               char(15)            not null,
```

```
    --股东代码
    stock_code              char(8)          not null,
    --证券代码
    stock_qty               int,
    --证券余额
    stk_holder_status       char(1),
    --股东帐户状态
    seat_code               char(8)          not null,
    --席位号
    branch_code             char(5),
    --营业部代码
    end_date                char(8),
    --截止日期
    deal_flag               char(1)
    --处理标志
) ;


drop table tb_stkcode;
create table tb_stkcode
(
    market_code             char(1)      not null,
    --   市场代码
    stock_code              char(8)      not null,
    --   证券代码
    stock_type              char(1)       not null,

    --   证券类别
    stock_grade             char(1),
    --   证券级别(备用)
    currency_code             char(1),
    --   货币代码(备用)
    stock_abbrev            char(12),
    --   证券简称
    relative_code           char(8),
    --   相关证券代码
    frozen_code             char(8),
    --   冻结证券代码
    stock_name              char(20),
    --   证券全称
    english_abbrev            char(20),
    --   英文简称
    vocation_type             char(6),
    --   行业种类
```

```
par_value                   decimal(12,4),
--  每股面值
total_issue                 bigint,
--  总发行量
circulate_qty               bigint,
--  流通股
last_profit                 decimal(12,4),
--  上年每股利润
curr_profit                 decimal(12,4),
--  本年每股利润
issue_date                  char(8),
--  上市日期
deliv_date                  char(8),
--  到期交割日
buyback_days                int,
--  回购天数
limit_per_pieces            int,
--  每笔限量
buy_unit                    int,
--  买数量单位
sell_unit                   int,
--  卖数量单位
price_tick                  decimal(12,4),
--  价格档位
group_match_para            decimal(12,4),
--  集合竞价限价参数
cont_match_para             decimal(12,4),
--  连续竞价限价参数
upper_price                 decimal(12,4),
--  涨停价格
lower_price                 decimal(12,4),
--  跌停价格
national_debt_ratio         decimal(12,4),
--  国债折合比例
trade_status                char(1),
--  证券交易状态
pause_flag                  char(1),
--  停牌标志
rights_flag                 char(1),
--  除权除息标志
composition_flag            char(1),
--  成份股标志
  valid_date                    char(8)   not null with default
'99999999',
```

```
    -- 有效日期
  reserved_flag                    char(1),
  --   标志位
  reserved                         char(20),
  --   保留
    constraint pk_stk_code    primary key (market_code, stock_code)
  -- 市场代码+证券代码

);
```

select cast (count(*) as char(10)) from (select * from tb_dbf_circulate_stk_all_ha where seat_code = '70017' and tradedate = '20020411') as dbf left join (select * from tb_stkcode where market_code = '1')as stk on dbf.stock_code = stk.stock_code

这个语句包含在存储过程中就编译不过去，进程长期挂起。
后来我用 dynexpln 去解释，还是进程挂起，这个进程用 force application 也 force 不掉，用 db2stop force 也下不来，进程长期处于 compiling 状态。

(环境 db2v7.2.2，打了 fix4)
在 fix5 上没问题，在 fix4 上就有问题。

10.14 远程连接连不上去，报 tcp/ip 错误

多数情况下，是一个参数设置上出了问题。
Db2set 看 db2comm=tcpip 是否存在。

10.15 tabspce 实际上没有表，但还是报满

环境：db2v7.2.3，打了 fix5。
在公司的开发机上，bak_tabspace 分配了 200M 的空间，里面有两个表，每个表中只有 10000 条记录，大小远远不到 10M。
做了 runstats，显示就只有 10000 条记录，成功。
做 reorg 不成功，报 sql-289 错误，表空间不够。
查看了系统临时表空间，是 sms 的，而磁盘空闲还有 4G，不应该空间不够的。
后来 drop 掉了其中一个表，但表空间的空闲空间还是 0，没有释放。
将另一个表也 drop 掉，这时从 syscat.tables 中查询，这个表空间中已经没有表了，但空闲空间还是 0，不变。
没办法，重起数据库。
表空间的大小释放出来，db2diag.log 中显示以下内容：

SQL8017W 此机器上的处理器数超过了对产品 "DB2 企业版" 定义的许可数目
"1"。此机器上的处理器数为 "2"。您应该从 IBM
代表或特许经销商那里购买附加的处理器权利，并通过使用“许可证中心”或 db2licm
命令行实用程序来更新您的许可证。有关更新处理器许可证的详情，参考适用于您的平台
的“快速入门”手册。有关 db2licm 实用程序的详情，参考“DB2 命令参考”。

2002-04-19-16.09.48.744022 Instance:db2inst1 Node:000
PID:44200(db2agent (ZBHEAD)) Appid:*LOCAL.db2inst1.020419064105
buffer_pool_services sqlbinit Probe:180 Database:ZBHEAD

WARNING: estore is being used for multiple page sizes performance may not be optimal0000 4000 ..@.

2002-04-19-16.09.48.751370 Instance:db2inst1 Node:000
PID:44200(db2agent (ZBHEAD)) Appid:*LOCAL.db2inst1.020419064105
buffer_pool_services sqlbinit Probe:245 Database:ZBHEAD

WARNING: estore is OFF but bufferpools are configured to use it 0000 0000

....

11 DB2 编程教训

11.1 常被大家访问同一记录的表的修改

对有可能多个进程访问同一条记录的表中的记录修改要注意，一定要将锁定时间尽量做到最小，否则很容易发生锁定等待。

11.2 大表改小表

如果对编程的复杂度增加不是太大，建议将大表中的数据分为小表存放，访问时可提高速度，也可以避免锁定等待的问题。

11.3 查询表数据使用 ur 的隔离级别

查询表数据，如果不是要修改或对数据的准确度要求非常高的情况下，建议使用 ur 的隔离级别，不对库表中的记录加锁，从而不对别人有什么影响。

11.4 Delete,update 后及时 commit

因为这两种操作看起来语句很简单，但实际操作的数据却可能是个很大的数量。不 commit，第一可能占用很大的锁资源，第二占用很大的日志资源。

12 AIX 系统管理

12.1 查看磁盘使用情况

df
df –k .
-k 是指定以 kbyte 来表示。

有时候会碰到一些莫名其妙的错误,如果是要读写硬盘的操作,可以看一下硬盘的情况。

lsps -a
看 pgsp 的情况,包括配置和使用比率

lsdev -Cc disk
看有几个硬盘

lslv hd1
看逻辑卷信息。

12.2 看目录的文件占用硬盘情况

du –k

12.3 看 IO 情况

iostat 5
看某个盘的情况 iostat 5|grep hdiskpower33

12.4 查看 CPU 情况

sar –P ALL 1 100
(需要 root 权限)

12.5 查看系统资源总的使用情况

topas (root,system 组)

12.6 看正在运行的线程/进程

12.6.1 看正在运行的线程

ps –emo THREAD –p 87406

12.6.2 看按占 cpu 比例排序的进程

ps aux
在 Aix 中会看到以 root 身份起来的几个进程 kproc 会排在前面,这几个进程是为了让 cpu 活着的,每个 cpu 会对应一个这样的进程。

12.6.3 看按占内存比例排序的进程

ps –vg

12.7 查看内存使用情况

svmon(root,看虚拟内存)

看实际物理内存大小
rmss –p (root)

vmstat
单位是块，每块为 4k（不需 root）

unix 系统有一个区域是用于做内存交换用,经验值大概是实际物理内存的两倍大。
目前的 s80 这个区域是放在 pagespace 空间，是个裸设备。

Var 目录是用于放登陆信息及 reboot 信息和 vi 的临时交换空间等用途。

12.8 查看共享内存、消息队列等使用情况

看共享内存、消息队列等情况
ipcs
ipcs –b 可以看到各内存区所占用的内存的大小。
ipcrm 是清除该内存区。

12.9 根下不要建文件系统

强烈不建议在根/下建文件系统，好象是对系统的性能和管理都有影响。

12.10 文件操作

12.10.1 看文本文件自动新增长内容

tail -f out.log

12.10.2 将大文件拆分

split -b 54m connect.log
将文件按每个 54M 拆分。

12.10.3 文件打包

tar cvf newfile.tar *

12.10.4 文件压缩

compress newfile.tar

12.10.5 文件解压

uncompress newfile.tar.Z

12.10.6 bz2 文件处理

bzip2 –help
bzip2 –d filename.bz2 解压

12.11 看逻辑卷信息

lslv loglv
lslv -l loglv (看逻辑卷在哪个物理卷上)
lslv -p hdiskpower36 (可以看到这个物理卷的使用情况)
lspv -l hdiskpower36 (看这个物理券上有哪些逻辑卷)
lspv –p hdiskpower36 (看该物理券分配给逻辑卷的情况,包括大小\位置)

看有哪些进程访问目录
fuser /home

12.12 重启机器

shutdown -Fr

13 AIX 系统限制

13.1 Fork 太多会导致系统崩溃

据多伦多实验室实验，满配的 s80 在同时 fork 4000 个进程，会导致 s80 操作系统 crash。

同时 fork 进程过多时，最好 fork 几十个 sleep 一下，再 fork，这样比较安全。

13.2 对文件大小的限制

Aix 缺省对文件的最大限制为 1G。

可以通过修改配置，更改文件大小最大值。AIX Version 4.3 最大可到 64G。

所以对 SMS 的表空间，也有 64G 的限制。

而 DMS 的表空间，则可到 512G

13.3 磁带备份的速度

国泰君安电脑部测试，55M 数据入磁带用了半个小时。旧磁带，做 append 方式。

14 AIX 及 DB2 相关文档及网站

http://tpb.top263.net
http://aixdb2.myrice.com/
http://www.ibmusers.com

14.1 取 db2 最新补丁程序

您可以访问如下网址获取您所需要的补丁程序:
http://www-4.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/download.d2w/report

14.2 国际化的 DB2 用户组织

IDUG (International Database Users Group) 是一个国际化的DB2 用户组织, 您可以通过访问网址: http: //www.idug.org 获取更多的DB2 信息资源以及通过讨论组咨询问题等等. 另外, 您也可以通过 http://www.db2mag.com/ 查看DB2 的在线杂志.

14.3 错误信息所在位置 1(errno)

2001-12-05-15.30.09.827998 Instance:gtjadb Node:000
PID:104076(db2agent (ZBDB)) Appid:0A64117B.057F.011205073006
oper_system_services sqloopenp Probe:36 Database:ZBDB

errno: 0000 001a
这个信息 /usr/include/errno.h 文件中，换为十进制后可以在这个文件中找得到。

14.4 错误信息所在位置 1(FFFF nnnn 或 nnnn FFFF)

file:///C:/db2/db2_doc/db2p0/frame3.htm#index中的Part 3. Appendixes

如果有 code 在这里没有的话，则可以和 db2 客户服务部联系。

如果是 FFFF nnnn 则可以直接用 nnnn 去看错误信息。
如果是 nnnn FFFF ，则将 nnnn 的前两位和后两位颠倒后再去查。

15 DB2 和 oracle 的对比

15.1 用户管理不一样

1）DB2 的用户是和 os 用户捆绑的，认证由 os 进行，而 oracle 的用户是独立于 os

15.2 表空间使用不一样

DB2 采用 SMS 表空间时，一个表可以对应于一个文件，而 Oracle 没有

这种方式，oracle 里表空间和数据文件的关系相当于 DB2 里的 DMS 表空间

15.3 保证事务的一致性方式不一样

db2 用事务日志保证事务一致性
oracle 使用回滚段

16 oracle 上 sql 语句性能优化（DB2 也可以参考）

16.1 oracle 中索引问题

假设你设置了一个非常好的索引，任何傻瓜都知道应该使用它，但是 Oracle 却偏偏不用，那么，需要做的第一件事情，是审视你的 sql 语句。

Oracle 要使用一个索引，有一些最基本的条件：

1， where 子句中的这个字段，必须是复合索引的第一个字段；
2， where 子句中的这个字段，不应该参与任何形式的计算

具体来讲，假设一个索引是按 f1, f2, f3 的次序建立的，现在有一个 sql 语句, where 子句是 f2 = : var2, 则因为 f2 不是索引的第 1 个字段，无法使用该索引。

第 2 个问题，则在我们之中非常严重。以下是从 实际系统上面抓到的几个例子：

Select jobid from mytabs where isReq='0' and to_date (updatedate) >= to_Date ('2001-7-18', 'YYYY-MM-DD')；

.....

以上的例子能很容易地进行改进。请注意这样的语句每天都在我们的系统中运行，消耗我们有限的 cpu 和 内存资源。

除了 1，2 这两个我们必须牢记于心的原则外，还应尽量熟悉各种操作符对 Oracle 是否使用索引的影响。这里我只讲哪些操作或者操作符会显式（explicitly）地阻止 Oracle 使用索引。以下是一些基本规则：

1， 如果 f1 和 f2 是同一个表的两个字段，则 f1>f2, f1>=f2, f1
2， f1 is null, f1 is not null, f1 not in, f1 !=, f1 like ‘%pattern%’;
3， Not exist
4， 某些情况下，f1 in 也会不用索引

对于这些操作，别无办法，只有尽量避免。比如，如果发现你的 sql 中的 in 操作没有使用索引，也许可以将 in 操作改成 比较操作 + union all。笔者在实践中发现很多时候这很有效。

但是，Oracle 是否真正使用索引，使用索引是否真正有效，还是必须进行实地的测验。合理的做法是，对所写的复杂的 sql，在将它写入应用程序之前，先在产品数据库上做一次 explain . explain 会获得 Oracle 对该 sql 的解析（plan）,可以明确地看到 Oracle 是如何优化该 sql 的。

如果经常做 explain, 就会发现，喜爱写复杂的 sql 并不是个好习惯，因为过分复杂的 sql 其解析计划往往不尽如人意。事实上，将复杂的 sql 拆开，有时候会极大地提高效率，因为能获得很好的优化。当然这已经是题外话了。

16.2 oracle 中索引问题

需要对几个超过千万的表进行计算，JION，复杂查询等操作，觉得 SQL 语句的优化非常重要，把一些心得写得出，当然也不一定非常正确。

---与 SQL 优化(包括内存空间)有关一些 INIT 参数
OPTIMIZER_MODE
SORT_AREA
SHARED_POOL_SZIE

---一些不使用索引的情况
1，NOT IN
2，NOT BETWEEN
3，LIKE（第一个字符非%号除外，如 name like '李%'）
4，<>
5，IS NULL/IS NOT NULL
6，查询的字段加函数
7，在 8I 中，多字段的组合索引（A，B，C），select * from ** wher B='33',则索引也不会用。（按前缀式规则使用索引除外，如 A='33' and B='33' || A='33' || A='33' and C='33'）
注：9I 除外

---查询语句比较优化的写法：
1，加 HINT，改变其执行路径

2，可能使用 exsit 的地方就尽量不用 IN,可以使用 not exiist 的地方，尽量不要用 not in

3，两个表进行 JION 时，大表放在前面，JION 字段建索引

4，尽量用其它写法，取代 NOT IN，如 a,b 表同结构，数据量很大，则代替 select * from a where a.c not in (select c from b)

的语句有

a)select a.* from a, b where a.c = b.c + and b.c is null（据说速度比原写法提高 30 倍）

b)select * from a minus select a.* from a,b where a.c=b.c (速度其次)

c)select * from a where not exist(select a.* from a,b where a.c=b.c) (也不错)

5，动态 SQL 中，尽量多用 execute immediate,而少用 DBMS_SQL,前者综合效率优于后者

6，对于很复杂的查询语句，可以建立临时表进行缓冲（关于临时表的解释与使用，还希望同行告诉我在哪里有。。。）

7，COUNT（*）与 COUNT（某列）一样进行全表扫描 Fast Full Index Scan，速度差不多

8，经常同时存取多列，或经常使用 GROUP BY 的 SQL 语句，最好对表的 GROUP 字段建立组合索引。组合索引要尽量使关键查询形成索引覆盖，其前导列一定是使用最频繁的列。

9，对于字段取值单一（如性别字段只有男与女），而经常在性别上做查询，则建立位图索引。

注：BITMAP INDEX 通常用于 DSS,如果你的系统是 OLTP,DML 操作将 LOCK 整个 BITMAP SEGMENT,因此只在 DSS 下 考虑 BITMAP INDEX

主机重新启动后，表空间出现 OFFLINE 的状态。

原因很有可能在于裸设备的权限不是 DB2 实力的权限。通常做了 HA 的机器重新启动后会出现类似的情况，如果不是这样问题将非常严重。

解决方法：修改设备权限。chown db2inst1:db2iadm1 r*tbs*

对于表空间删除掉的设备，需要进行以下操作，才能释放

EXAMPLE：db2untag -f rparatbs1

分配和创建裸设备

mkvg -f -y'data1vg1' -s'32' hdiskpower4 hdiskpower11

mklv -y'paratbs1' data1vg1 1 hdiskpower4

mklv -y'paratbs2' data1vg1 1 hdiskpower11

RS6000 列出 CPU 相关信息

1、lscfg –vp |　grep -ip cpu
lscfg -vp|grep -ip processor

17　怎样判断 DB2 实例的版本号和修补级别？

环境	产品：DB2 平台：跨平台 版本：DB2 5.2 及以上版本
问题	怎样判断 DB2 实例的版本号和修补级别？
解答	用 db2level 命令。在 DB2 5.2 及以上版本中，在安装每个 DB2 实例时，即会装入 db2level 程序。db2level 命令的输出提供了有关 DB2 实例的版本及修补级别的详细信息。 命令输出如下所示： DB21085I Instance "<instance_name>" uses DB2 code release "<rel_ver_mod>" with level identifier "<level_id>" and informational tokens "<build_id1>", "<build_id2>" and "<build_id3>". 例如： DB21085I Instance "DB2" uses DB2 code release "SQL05020" with level identifier "02070103" and informational tokens "DB2 v5.2.0.30","c990717" and "WR21119". 下面解释以下这些信息： <instance_name> = DB2 DB2 的实例名 <rel_ver_mod> = SQL05020 Release 号 05，Version 号 02，Module 号 0 <level_id> = 02070103 内部使用的 DB2 版本号 <build_id1> = DB2 v5.2.0.30 实例的版本信息 <build_id2> = c990717 代码的级别信息 <build_id3> = WR2119 修补的级别信息 注：db2level 执行程序不能在不同的系统之间拷贝使用。 并且此程序只显示正式支持的修补级别信息。 对于 DB2 版本 5.0 和 2.0，可用如下方法获得版本信息： OS/2：syslevel 命令 NT：查询 regedit 变量：HKEY_LOCAL_MACHINE |

SOFTWARE | IBM | DB2 | DB2
universal database xx edition |
CurrentVersion

AIX：用 dump -H
$HOME/sqllib/lib/libdb2e.a

Solaris: cat 命令查看文件信息
/opt/IBMdb2/V5.0/cfg/bldlevel or
"ldd -s
$HOME/sqllib/lib/libdb2e.so |
grep engn|grep search|uniq"

HP: cat 命令查看文件信息
/opt/IBMdb2/V5.0/cfgbldlevel

在由备份恢复一个数据库时，遇到 SQL2542 错误，怎麼办？

文章编号:1307131000000

日　期:2001-01-09

在由备份恢复一个数据库时，遇到 SQL2542 错误，怎麼办？

环境	版本：(试用)DB2 V5.0,DB2 V6.1,DB2 V7.1 操作系统：(试用)Windows NT,AIX
问题	在由备份恢复一个数据库时，遇到 SQL2542 错误
解答	如果有几个数据库的备份，在做数据库恢复时，需要提供正确的路径和时间戳，如果是用 DB2 命令行来执行恢复操作，在 Windows NT 操作系统，可参照如下命令： RESTORE DATABASE SAMPLE FROM D:\backups TAKEN AT 19991117125141 此命令中要注意路径和时间戳。时间戳可以通过 list history 命令得到。可参照如下命令： LIST HISTORY BACKUP ALL FOR SAMPLE Op Obj Timestamp+sequence Type Dev Earliest log Current log Backup ID B D 19991117125141001 F D S0000000.LOG S0000000.LOG Contains 2 tablespace(s): 00001 SYSCATSPACE 00002 USERSPACE1 此命令的输出列出了备份的时间戳加上一个 3 位的数字序列： 时间戳+3 位的数字序列=19991117125141001 所以，可以在 restore 命令中使用时间戳：19991117125141。 如果你有多于一个备份，list history 命令将显示所有备份纪录的信息。 更多的信息可以参考“IBM DB2 Universal Database 命令手册：第三章 CLP 命令”。

DB2 中 转义符号的使用 可以用 ESCAPE 进行指定
D:\>db2 select * from test11 where name like '_%' ESCAPE '\'

NAME

_1111

D:\>db2 select * from test11 where name like '$_%' escape '$'

NAME

_1111

升级前锁定列表的最大百分比 (maxlocks) 的设置

下列更改涉及“升级前锁定列表的最大百分比 (maxlocks)”数据库配置参数的“建
议”部分。

建议：以下公式允许将 maxlocks 设置为允许应用程序保存锁定的平均数目的两倍：

maxlocks = 2 * 100 / maxappls

其中 2 用来获取对平均数目的翻倍， 而 100 表示允许的最大百分比值。如果只有几
个应用程序并行运行， 可将以下公式用作第一个公式的替代项：

maxlocks = 2 * 100 / （并行运行的应用程序的平均数目）

设置 maxlocks 时的注意事项之一就是将其与锁定列表 (locklist) 的大小配合使
用。在锁定升级之前， 应用程序可持有的锁定数目的实际限制为：

maxlocks * locklist * 4096 / (100 * 36)

其中 4096 是页面的字节数，100 是 maxlocks 所允许的最大百分比值，而 36 是
每个锁定的字节数。如果您知道每个应用程序需要 1000 个锁定，且不想要进行锁定
升级，则应选择此公式中 maxlocks 和 locklist 的值，以使结果大于 1000。（对 maxlocks
使用 10，而对 locklist 使用 100，此公式的结果就会大于所需的 1000。）

如果 maxlocks 设置得太低，当仍有足够的锁定空间可供其他并行应用程序使用时，
就会进行锁定升级。如果 maxlocks 设置得太高，少数几个应用程序可能会消耗大部
分的锁定空间， 而其他应用程序将不得不执行锁定升级。在此情况下的锁定升级需要
会导致并行度降低

数据库创建后出现代码页和环境不一致的解决办法。

对 DB2 环境变量进行设置。
把 DB2CODEPAGE 的内容和数据库中的配置相同

如：DB2SET DB2CODEPAGE = 1381

然后需要实例重新启动，才能生效

18 在 LINUX 上创建 DB2 裸设备方法

1. 将块设备映射成裸设备：

>
> raw /dev/raw/rawN blockdev
>
> 其中: /dev/raw/rawN 是裸设备名称，已经在 /dev/raw 创建好了。
> blockdev 可以是任何块随机存储设备，例如磁盘，磁盘分区，逻辑卷

> 等。
> 运行后，即完成映射。可以用 raw -qa 列出所有的映射。
>
> 例子:
>
> raw /dev/raw/raw1 /dev/hda
> raw /dev/raw/raw2 /dev/hdb4
> raw /dev/raw/raw3 /dev/vgdata/lv_usertb
>
> 2. 修改权限：
>
> 要将裸设备和相应的块设备赋予所需的访问权限。
> 例如：
>
> chmod 777 /dev/raw/raw1 /dev/hda
> chown db2inst1.db2iadm1 /dev/raw/raw2 /dev/hdb4
>
> 除此之外，/dev/rawctl 好像也要设置为可以读写。可以试一试。
>
> 3. 创建表空间：
>
> db2> create tablespace test managed by database using (device
> '/dev/raw/raw1' 8192

AIX ， 自动安装实例时失败
解决手工创建实例时远程客户不能连接的问题 ，

vi /etc/services
FIND
db2cdb2inst1 50000/tcp # Connection port for DB2 instance db2inst1

update dbm cfg using SVCENAME db2cdb2inst1

去掉 DB2 自动启动的设置
vi/etc/ inittab

19 DB2 relational conn 连接 SQLSERVER 的方法步骤

1、先通过安装 DB2 WAREHOUSE manager 来安装 ODBC DIRVER
2、在 AIX 主机中，用 DB2 用户登陆 配置.ODBC.INI 文件
文件在 DB2 的安装目录下一般为/home/db2inst1
ls –a 可以看到
vi .ODBC.INI
在文件底部追加,红字为关键部分。

```
[ODBC Data Sources]
pubs=MSSQL

[pubs]
Driver=/usr/lpp/db2_07_01/odbc/lib/ibmsss15.so
Address=172.28.145.250,1433
Database=pubs
UID=sa
PWD=
TDS=7.0
QuotedID=no
  AnsiNPW=yes
```

3、edit $insthome/sqllib/cfg/db2dj.ini set the following variables ;

```
ODBCINI=/home/db2eee/.odbc.ini
DJX_ODBC_LIBRARY_PATH=/usr/lpp/db2_07_01/odbc/lib
DB2EVNLIST=LIBPATH
```

4、设置 DB2 环境变量

```
DB2SET DB2_DJ_INI=/home/db2eee/sqllib/cfg/db2dj.ini
DB2SET DB2LIBPATH=/usr/lpp/db2_07_01/odbc/lib
 DB2SET DB2ENVLIST=LIBPATH
```

5、运行链接脚本

```
/usr/lpp/db2_07_01/bin/djxlink
```

6、设置环境变量

```
DB2SET DB2_DJ_COMM=libmssql3.a
```

检查 DB2SET 应该如下：

```
DB2_DJ_COMM=libmssql3.a
DB2_DJ_INI=/home/db2eee/sqllib/cfg/db2dj.ini
DB2LIBPATH=/usr/lpp/db2_07_01/odbc/lib
```

DB2ENVLIST=LIBPATH

7、重新启动数据库

DB2STOP ;DB2START

8、创建连接驱动

create wrapper "MSSQLODBC3" LIBRARY 'libmssql3.a' ;

9、创建连接服务

create server "PUBS" TYPE MSSQLSERVER VERSION 7.0 WRAPPER "MSSQLODBC3" OPTIONS(NODE 'pubs',DBNAME 'pubs')

10、创建用户映射

create user mapping for "DB2EEE" SERVER "PUBS" OPTIONS(REMOTE_AUTHID 'sa',REMOTE_PASSWORD '1234')

11、为数据库创建别名

create nickname "sysusers" for "PUBS"."dbo"."sysusers"

12、odbc samples 如下：

[ODBC Data Sources]
dBase=INTERSOLV 3.11 dBase Driver
Sybase11=INTERSOLV 3.11 Sybase 11 Driver
Oracle7=INTERSOLV 3.11 Oracle 7 Driver
Oracle8=INTERSOLV 3.11 Oracle 8 Driver
Informix9=INTERSOLV 3.11 Informix 9 Driver
OpenIngres=INTERSOLV 3.11 OpenIngres 1.2 Driver
OpenIngres20=INTERSOLV 3.11 OpenIngres 2.0 Driver
DB2=INTERSOLV 3.11 DB2 Driver
Text=INTERSOLV 3.11 Text Driver
OLAP=db2
SAMPLE=db2
pubs=MSSQL

[dBase]
Driver=/home/olap/server/dlls/ARdbf13.so
Description=dBase

[Sybase11]
Driver=/home/olap/server/dlls/ARsyb1113.so
Description=Sybase11
OptimizePrepare=2
SelectMethod=1

[Oracle7]
Driver=/home/olap/server/dlls/ARor713.so
Description=Oracle7

[Oracle8]

```
Driver=/home/olap/server/dlls/ARor813.so
Description=Oracle8

[Informix9]
Driver=/home/olap/server/dlls/ARinf913.so
Description=Informix9
[OLAP]
Driver=/home/db2inst1/sqllib/lib/db2_36.o
[SAMPLE]
Driver=/home/db2inst1/sqllib/lib/db2_36.o

[DB2]
Driver=/home/olap/server/dlls/ARdb213.so
Description=DB2

[OpenIngres]
Driver=/home/olap/server/dlls/AROing13.so
Description=OpenIngres1
Workarounds=1

[OpenIngres20]
Driver=/home/olap/server/dlls/AROi213.so
Description=OpenIngres2.0
Workarounds=1

[Text]
Driver=/home/olap/server/dlls/ARtxt13.so
Description=Text driver

[ODBC]
Trace=0
TraceFile=odbctrace.out
TraceDll=/home/olap/server/dlls/odbctrac.so
InstallDir=/home/olap/server/dlls

[pubs]
Driver=/usr/lpp/db2_07_01/odbc/lib/ibmsss15.so
Address=172.28.145.250,1433
Database=pubs
UID=sa
PWD=
TDS=7.0
QuotedID=no
AnsiNPW=yes
```

13 db2dj.ini example 如下：

ODBCINI=/home/db2eee/.odbc.ini
DJX_ODBC_LIBRARY_PATH=/usr/lpp/db2_07_01/odbc/lib
DB2EVNLIST=LIBPATH

复制表的方便方法：

1\
create table2 like table1 ;
alter table2 add column field1 int
alter table2 add column field2 char(5) ;
2\

create table2 as (select t1.*,current timestamp as field1 from table1 as t1) definition only ;

20 数据库配置参数摘要

环境	产品: DB2 UDB 平台: 跨平台 版本: V7
问题	数据库配置参数摘要
解答	下表列出数据库服务器的数据库管理程序配置文件中的参数。当更改数据库管理程序配置参数时，要考虑每个参数的详细信息。包括缺省值的特定操作环境信息是每个参数说明的一部分。 下表中的"性能影响"列指示每个参数影响系统性能的相对程度。 不可能将此列准确地应用于所有环境；您应该将此信息视为一般情况。 • 高－－指示该参数可以对性能有重要影响。应有意识地决定这些参数的值；在某些情况下，将意味着接受提供的缺省值。 • 中－－指示该参数可以对性能有某些影响。您的特定环境和需要将确定应对这些参数进行多大程度的调整。 • 低－－指示该参数对性能没有那么普遍或没有那么重要的影响。 • 无－－指示该参数对性能没有直接的影响。 当您不必因性能的原因调整这些参数时，它们对于您系统配置的其他方面（如启用通信支持）可能很重要。 表 54. 可配置的数据库配置参数

参数	性能影响	其他信息
app_ctl_heap_sz	中	应用程序控制堆大小 (app_ctl_heap_sz)

applheapsz	中	应用程序堆大小 (applheapsz)
audit_buf_sz	中	审查缓冲区大小 (audit_buf_sz)
autorestart	低	启用自动重新启动 (autorestart)
avg_appls	高	活动应用程序的平均数 (avg_appls)
buffpage	高	（活动时） 缓冲池大小 (buffpage)
catalogcache_sz	中	目录高速缓存大小 (catalogcache_sz)
chngpgs_thresh	高	更改页阈值 (chngpgs_thresh)
copyprotect	无	启用副本保护 (copyprotect)
dbheap	中	数据库堆 (dbheap)
dft_degree	高	缺省度 (dft_degree)
dft_extent_sz	中	表空间的缺省数据块大小 (dft_extent_sz)
dft_loadrec_ses	中	缺省的装入恢复对话数 (dft_loadrec_ses)
dft_prefetch_sz	中	缺省预读取大小 (dft_prefetch_sz)
dft_queryopt	中	缺省查询优化级别 (dft_queryopt)
dft_refresh_age	中	缺省刷新时限 (dft_refresh_age)
dft_sqlmathwarn	无	遇到算术异常继续 (dft_sqlmathwarn)
dir_obj_name	无	DCE 名称空间中的对象名 (dir_obj_name)
discover_db	中	发现数据库 (discover_db)
dlchktime	中	检查死锁的时间间隔 (dlchktime)
dl_expint	无	数据链路存取令牌到期时间间隔 (dl_expint)
dl_num_copies	无	数据链路副本数 (dl_num_copies)
dl_time_drop	无	卸下后的数据链路时间 (dl_time_drop)
dl_token	低	数据链路令牌算法 (dl_token)
dl_upper	无	大写的数据链路令牌 (dl_upper)
dyn_query_mgmt	低	动态 SQL 查询管理 (dyn_query_mgmt)
estore_seg_sz	中	扩充内存段大小 (estore_seg_sz)
indexrec	中	索引重建时间 (indexrec)
indexsort	低（参见***）	索引排序标志 (indexsort)
locklist	高（当它影响逐步升级时）	锁定列表的最大存储器 (locklist)
locktimeout	中	锁定超时 (locktimeout)
logbufsz	高	日志缓冲区大小 (logbufsz)
logfilsiz	中	日志文件的大小 (logfilsiz)
logprimary	中	主日志文件数 (logprimary)
logretain	低	启用日志保留 (logretain)
logsecond	中	辅助日志文件数 (logsecond)
maxappls	中	活动应用程序的最大数目 (maxappls)
maxfilop	中	每个应用程序可打开的数据库文件的最大数目 (maxfilop)
maxlocks	高（当它影响逐步升级时）	逐步升级前锁定列表的最大百分比 (maxlocks)
mincommit	高	对组的落实次数 (mincommit)
newlogpath	低	更改数据库日志路径 (newlogpath)

num_db_backups	无	数据库备份数 (num_db_backups)
num_estore_segs	中	扩充内存段数 (num_estore_segs)
num_freqvalues	低	保存的高频值数目 (num_freqvalues)
num_iocleaners	高	异步页清除器数 (num_iocleaners)
num_ioservers	高	I/O 服务器数目 (num_ioservers)
num_quantiles	低	列的分位数数目 (num_quantiles)
pckcachesz	高	程序包高速缓存大小 (pckcachesz)
rec_his_retentn	无	恢复历史保留期 (rec_his_retentn)
seqdetect	高	顺序检测标志 (seqdetect)
softmax	中	恢复范围和软检查点间隔 (softmax)
sortheap	高	排序堆大小 (sortheap)
stat_heap_sz	低	统计堆大小 (stat_heap_sz)
stmtheap	中	语句堆大小 (stmtheap)
tsm_mgmtclass	无	Tivoli 存储管理器 管理类 (tsm_mgmtclass)
tsm_nodename	无	Tivoli 存储管理器 节点名 (tsm_nodename)
tsm_owner	无	Tivoli 存储管理器 拥有者名 (tsm_owner)
tsm_password	无	Tivoli 存储管理器 口令 (tsm_password)
userexit	低	启用用户出口 (userexit)
util_heap_sz	低	实用程序堆大小 (util_heap_sz)

注意: 将 indexsort 参数更改为非缺省值的值， 可能对创建索引的性能带来负面影响。您应该始终尝试使用此参数的缺省值。

表 55. 参考性数据库配置参数

参数	其他信息
backup_pending	备份暂挂指示符 (backup_pending)
codepage	数据库的代码页 (codepage)
codeset	数据库的代码集 (codeset)
collate_info	整理信息 (collate_info)
country	数据库的国家代码 (country)
database_consistent	数据库一致 (database_consistent)
database_level	数据库发行版级别 (database_level)
log_retain_status	日志保留状态指示符 (log_retain_status)
loghead	第一个活动的日志文件 (loghead)
logpath	日志文件的位置 (logpath)
multipage_alloc	启用多页文件分配 (multipage_alloc)
numsegs	缺省 SMS 容器数 (numsegs)
release	配置文件发行版级别 (release)
restore_pending	复原暂挂 (restore_pending)
rollfwd_pending	前滚暂挂指示符 (rollfwd_pending)
territory	数据库的地区 (territory)
user_exit_status	用户出口状态指示符 (user_exit_status)

21 数据库管理程序配置参数摘要

环境	产品: DB2 UDB 平台: Windows, Unix 版本: V7
问题	数据库管理程序配置参数摘要
解答	下表列出数据库服务器的数据库管理程序配置文件中的参数。当更改数据库管理程序配置参数时，要考虑每个参数的详细信息。包括缺省值的特定操作环境信息是每个参数说明的一部分。 下表中的"性能影响"列指示每个参数影响系统性能的相对程度。 不可能将此列准确地应用于所有环境；您应该将此信息视为一般情况。 • 高－－指示该参数可以对性能有重要影响。应有意识地决定这些参数的值；在某些情况下，将意味着接受提供的缺省值。 • 中－－指示该参数可以对性能有某些影响。您的特定环境和需要将确定应对这些参数进行多大程度的调整。 • 低－－指示该参数对性能没有那么普遍或没有那么重要的影响。 • 无－－指示该参数对性能没有直接的影响。 当您不必因性能的原因调整这些参数时，它们对于您系统配置的其他方面（如启用通信支持）可能很重要。 表 52. 可配置的数据库管理程序配置参数

参数	性能影响	其他信息
agentpri	高	代理程序优先级 (agentpri)
agent_stack_sz	低	代理程序栈大小 (agent_stack_sz)
aslheapsz	高	应用程序支持层堆大小 (aslheapsz)
audit_buf_sz	高	审查缓冲区大小 (audit_buf_sz)
authentication	低	认证类型 (authentication)
backbufsz	中	缺省备份缓冲区大小 (backbufsz)
catalog_noauth	无	不需权限就允许编目 (catalog_noauth)
comm_bandwidth	中	通信带宽 (comm_bandwidth)
conn_elapse	中	连接经过时间 (conn_elapse)
cpuspeed	低	（参见注释） CPU 速度 (cpuspeed)
datalinks	低	启用数据链路支持 (datalinks)
dft_account_str	无	缺省交费帐户 (dft_account_str)
dft_client_adpt	无	缺省客户机适配器号 (dft_client_adpt)
dft_client_comm • dft_monswitches • dft_mon_bufpool • dft_mon_lock • dft_mon_sort • dft_mon_stmt	无	缺省客户机通信协议 (dft_client_comm)

• dft_mon_table		
dft_mon_uow	中	缺省数据库系统监控程序开关 (dft_monswitches)
dftdbpath	无	缺省数据库路径 (dftdbpath)
diaglevel	低	诊断错误捕捉级别 (diaglevel)
diagpath	无	诊断数据目录路径 (diagpath)
dir_cache	中	目录高速缓存支持 (dir_cache)
dir_obj_name	无	DCE 名称空间中的对象名 (dir_obj_name)
dir_path_name	无	DCE 名称空间中的目录路径名 (dir_path_name)
dir_type	无	目录服务类型 (dir_type)
discover	中	发现方式 (discover)
discover_comm	低	搜索发现通信协议 (discover_comm)
discover_inst	低	Discover 服务器实例 (discover_inst)
dos_rqrioblk	高	DOS 请求器 I/O 块大小 (dos_rqrioblk)
drda_heap_sz	低	DRDA 堆大小 (drda_heap_sz)
fcm_num_anchors	高	FCM 信息锚数 (fcm_num_anchors)
fcm_num_buffers	高	FCM 缓冲区数 (fcm_num_buffers)
fcm_num_connect	高	FCM 连接项数 (fcm_num_connect)
fcm_num_rqb	高	FCM 请求块数 (fcm_num_rqb)
federated	中	联合体数据库系统支持 (federated)
fileserver	无	IPX/SPX 文件服务器名 (fileserver)
indexrec	中	索引重建时间 (indexrec)
initdari_jvm	中	用 JVM 初始化 DARI 进程 (initdari_jvm)
intra_parallel	高	启用分区内并行性 (intra_parallel)
ipx_socket	无	IPX/SPX 套接字号 (ipx_socket)
java_heap_sz	高	最大 Java 解释程序堆大小 (java_heap_sz)
jdk11_path	无	Java Development Kit 1.1 安装路径 (jdk11_path)
keepdari	中	保存 DARI 进程指示符 (keepdari)
maxagents	中	代理程序的最大数目 (maxagents)
maxcagents	中	并行代理程序的最大数目 (maxcagents)
max_connretries	中	节点连接重试次数 (max_connretries)
max_coordagents	中	最大协调代理程序数 (max_coordagents)
maxdari	中	DARI 进程的最大数目 (maxdari)
max_logicagents	中	逻辑代理程序的最大数目 (max_logicagents)
max_querydegree	高	最大查询并行度 (max_querydegree)
max_time_diff	中	节点之间的最大时间差 (max_time_diff)
maxtotfilop	中	每个应用程序可打开的最大总文件数 (maxtotfilop)
min_priv_mem	中	落实的最小专用内存 (min_priv_mem)
mon_heap_sz	低	数据库系统监控程序堆大小 (mon_heap_sz)
Nname	无	NetBIOS 工作站名 (nname)
notifylevel	低	通知级别 (notifylevel)
Numdb	低	并行活动数据库的最大数目 (numdb)

num_initagents	中	池中初始代理程序数 (num_initagents)
num_initdaris	中	存储池中防护 DARI 进程的初始数目 (num_initdaris)
num_poolagents	高	代理程序池大小 (num_poolagents)
objectname	无	IPX/SPX DB2 服务器对象名 (objectname)
priv_mem_thresh	中	专用内存阈值 (priv_mem_thresh)
query_heap_sz	中	查询堆大小 (query_heap_sz)
restbufsz	中	缺省复原缓冲区大小 (restbufsz)
resync_interval	无	事务再同步间隔 (resync_interval)
route_obj_name	无	路径选择信息对象名 (route_obj_name)
Rqrioblk	高	客户机 I/O 块大小 (rqrioblk)
sheapthres	高	排序堆阈值 (sheapthres)
spm_log_file_sz	低	同步点管理程序日志文件大小 (spm_log_file_sz)
spm_log_path	中	同步点管理程序日志文件路径? (spm_log_path)
spm_max_resync	低	同步点管理程序再同步代理程序限制 (spm_max_resync)
spm_name	无	同步点管理程序名 (spm_name)
ss_logon	无	DB2START/DB2STOP 必需的注册 (ss_logon)
start_stop_time	低	启动和停止超时 (start_stop_time)
svcename	无	TCP/IP 服务名 (svcename)
Sysadm_group	无	系统管理权限组名 (sysadm_group)
sysctrl_group	无	系统控制权限组名 (sysctrl_group)
sysmaint_group	无	系统维护权限组名 (sysmaint_group)
tm_database	无	事务管理程序数据库名 (tm_database)
tp_mon_name	无	事务处理器监控程序名 (tp_mon_name)
Tpname	无	APPC 事务程序名 (tpname)
trust_allclnts	无	信任所有客户机 (trust_allclnts)
trust_clntauth	无	可信赖客户机认证 (trust_clntauth)
udf_mem_sz	低	UDF 共享内存集大小 (udf_mem_sz)

注意:
cpuspeed 参数可以对性能产生显著影响，除非在非常特定的情况下，否则应使用缺省值，如参数说明中所述。

表 53. 参考性数据库管理程序配置参数

参数	其他信息
nodetype	机器节点类型 (nodetype)
release	配置文件发行版级别 (release)

22 如何实施存储过程的发布

环境	[产品] DB2 UDB [平台] 跨平台 [版本] 7.2
问题	DB2 从 7.2 版本开始支持存储过程的发布，即可以将一个数据库上已编译好的存储过程抽取并安装到其它数据库上，目标数据库所在的服务器上不再需要 c 编译器。
解答	发布存储过程请按以下步骤： 1. 如果数据库是从 7.1 版本打补丁后升为 7.2 版本，请用 db2updv7 -d *数据库名* 使 DB2 可以抽取或安装已编译好的存储过程； 2. 在源数据库，编译好存储过程； 3. 在源数据库，运行 db2 "get routine into *文件名* from procedure 存储过程名" 抽取存储过程； 4. 上传文件至目标服务器； 5. 在目标服务器端，运行 db2 "put routine from *文件名*" 安装已编译好的存储过程。

23 表空间重定向

环境	[产品] DB2 UDB [平台] Windows NT/2000, Unix, Linux [版本] 5.x/6.x/7.x
问题	恢复数据库时，如果数据库表空间使用的容器（container）被别的数据库占用，那么在恢复时须要做表空间重定向。

解答

下面是一个表空间重定向的例子，数据库的别名为 MYDB：

1. 使用 restore database 加 redirect 参数：
   db2 restore db mydb replace existing redirect

   在第一步后，第三步前，数据库恢复可以用下面命令取消：
   db2 restore db mydb abort
2. 用 set tablespace containers 命令重定义容器
   db2 set tablespace containers for 5 using (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)

   第二步中须要重定义所以需要做表空间重定向的容器。
3. 成功运行第一步和第二步后，使用：
   db2 restore db mydb continue 完成恢复工程。如果第三步失败，重定向须要从第一步做起。

不可以把 DMS 表空间重定向为 SMS 表空间，反之亦然。

24 如何设置 RAID 环境下的 DB2 表空间

环境 DB2 V7.2
问题 如何设置 RAID 环境下的 DB2 表空间
解答 如果要把 DB2 的数据存储在 RAID 设备上，通常情况下用户需要对表空间进行如下的一些设置：

为每个 RAID 设备上等表空间设置唯一的一个容器；
设置表空间的 EXTENTSIZE 大小为 RAID 条带(Stripe)大小的整数倍；
设置 DB2_PARALLEL_IO 注册变量确保对表空间的并发访问
设置 DB2_STRIPED_CONTAINERS 注册变量以协调表空间 extent 和 RAID 设备 stripe 之间的数量上的一致性

。

25 DB2 的安全管理机制有哪些?

环境 产品: DB2
平台: Crossplatform
软件版本: v7.2
问题 DB2 的安全管理机制有哪些?

解答 DB2 UDB 数据库产品具有 Server 端认证、Client 端认证、DCE 认证、DCS 认证、DRDA 认证等多种认证

方式，用户名和密码可以以明文或加密方式在网络上传输。对实例及数据库，不同的用户组可以通过数据库

系统管理员、数据库系统控制员、数据库系统维护员以及数据库管理员的角色进行不同层次的数据库系统管

理和维护。对于每一个数据库，可以为每一个用户或用户组分别对数据库、表空间、表、索引、应用程序、

模式(schema)、视图等进行安全权限控制。从 DB2 的操作历史记录中可以获取所有对数据库的关键操作，实

现追踪审计。另外 DB2 UDB 支持数据加密，可以定义数据库中的某个表或表中的某个字段以加密方式存储

26 DB2V7.1 在 RedHatV7.2 下的安装说明

内容

提要 DB2 从 6 版本起支持 LINUX 操作系统。本文简单地描述一下在 RedHat 版本 7.2 上安装 DB2 版本 7 的经验

。

正文 一、安装前：

把 DB2 安装盘 mount 上后可以选择用 db2_install 命令或着用 db2setup 命令安装。

用 db2_install 只是简单地把 DB2 产品 RPM 包安装到操作系统中，创建实例和管理实例的工作要等到

db2_install 运行完后再手工创建。

用 db2setup 命令会出现一个类似 AIX 下 smitty 的字符仿图形的安装界面。推荐用户选择用 db2setup 安装 DB2

1. 如果安装 RedHatV7.2 时默认安装没有选中所有的包，在

RedHatV7.2 下要使用 db2setup 命令，要先安装下

面的几个包：

1)从 RedHat 产品盘 2 中安装 pdksh-5.2.14-13.i386.rpm，例如以 root 帐户登录，mount 光驱至/mnt/cdrom，运

行下列命令：
rpm -ivh /mnt/cdrom/RedHat/RPMS/pdksh-5.2.14-13.i386.rpm

2)从 RedHat 产品盘 2 中安装 compat-egcs-c++-6.2-1.1.2.16.i386.rpm：

rpm -ivh /mnt/cdrom/RedHat/RPMS/compat-egcs-c++-6.2-1.1.2.16.i386.rpm

但是安装 compat-egcs-c++-6.2-1.1.2.16.i386.rpm 又有先决条件，归纳下来，要先顺序执行下面命令：

rpm -ivh /mnt/cdrom/RedHat/RPMS/binutils-2.11.90.0.8-9.i386.rpm (产品盘 2 中)
rpm -ivh /mnt/cdrom/RedHat/RPMS/kernel-headers-2.4.7-10.i386.rpm (产品盘 1 中)
rpm -ivh /mnt/cdrom/RedHat/RPMS/glibc-devel-2.2.4-13.i386.rpm (产品盘 2 中)
rpm -ivh /mnt/cdrom/RedHat/RPMS/compat-egcs-6.2-1.1.2.16.i386.rpm (产品盘 2 中)

3)从 RedHat 产品盘 2 中安装 compat-libs-6.2-3.i386.rpm：

rpm -ivh /mnt/cdrom/RedHat/RPMS/compat-libs-6.2-3.i386.rpm

2. db2setup 命令会查找/usr/lib/libncurses.so.4 文件，对于 RedHatV7.2 需要建立链接文件，使该文件指向
/usr/i386-glibc21-linux/lib/libncurses.so.4.0，命令如下：

ln -sf /usr/i386-glibc21-linux/lib/libncurses.so.4.0 /usr/lib/libncurses.so.4

3. 默认地 msgmni 内核设置至容许两个并发连接连到 DB2，在 /etc/sysctl.conf 文件下加入如下行增大该设置：

# Sets maximum number of message queues to 128
# Set this to 1024 or higher on production systems
kernel.msgmni = 128

二、安装和安装 DB2 产品补丁：

以上设置完成后运行 db2setup 安装 DB2V7.1。除了默认选项之外，如果要使用控制中心，安装时请选择

Admin Client 中的控制中心组件。

安装时选择安装完创建 DB2 实例和 DB2 管理实例并完成相关的设置，可以选择默认设置。

DB2V7.1 的最新补丁为 5，安装 DB2 后建议马上安装 DB2V7.1 的补丁。

请到 ftp://ftp.software.ibm.com
/ps/products/db2/fixes/english-us/db2linuxv7/FP3_U475381 下载 DB2 补丁 3
/ps/products/db2/fixes/english-us/db2linuxv7/FP5_U480366 下载 DB2 补丁 5

如果您安装的是 DB27.1，要安装 DB2 补丁 5，需要先安装 DB2 补丁 3（安装完 DB2 补丁 3 后 7.1 升级为 7.2）。

如果您安装的是 DB2 7.2，则可以直接安装 DB2 补丁 5。（DB2 补丁当前情况请参考

http://www-4.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/download.d2w/report）。

注意安装完补丁后请用 db2iupdt 更新 DB2 实例，用 dasiupdt 更新 db2as 实例。

三、安装后：

安装 DB2 补丁 3 后控制中心用到 JDK1.3，请到 http://www.ibm.com/java/jdk/linux130 下下载 IBM Developer Kit

for Java, version 1.3 RPM 包。

运行 rpm -ivh IBMJava2-SDK-1.3-10.0.i386.rpm 安装包。

之后编辑/etc/profile 文件，加上下面一行：

export PATH=$PATH:/opt/IBMJava2-13/bin

为所有用户添加 JAVA 运行环境的路径。

安装完毕运行 java -fullversion 测试当前 JAVA 版本信息（如果没重起操作系统，请运行 . /etc/profile 使上面的

设置生效）。

IBM JDK 1.3 废除了 jre 命令（即使您安装了 IBM JRE 1.3）。但是启动控制中心的脚本 db2cc 中仍然用到 jre 命

令，一个简单办法就是用 root 用户建立一个链接，使 /opt/IBMJava2-13/bin 目录下的 jre 命令指向

/opt/IBMJava2-13/jre/bin 目录下的 java 命令。

ln -sf /opt/IBMJava2-13/jre/bin/java /opt/IBMJava2-13/bin/jre

同时脚本 db2cc 调用 jre 命令时加上了-nojit 选项，也是 IBM JDK 1.3 不支持的。

编辑 实例安装目录/sqllib/bin/db2cc 脚本，更改

JRE_OPTIONS="-nojit -ss256k -mx128m -Ddb2path=$DB2PATH

一列为

JRE_OPTIONS='-ss16m -mx128m -Ddb2path=$DB2PATH" 。

上面 ss 是指 maximum native stack size，原始设置大小为 256k，从安装经验来看，这个设置有可能会小一些

，可以适当调大，例如调整到 16m。

注意：如安装完 DB2 最新的补丁，并更新实例后，该脚步会被重置，请重新编辑该脚本，确保

JRE_OPTIONS 的设置。

在 UNIX 环境下 su 至一个用户时，用 db2cc 命令起控制中心可能会碰到下面错误：

Xlib: connection to "localhost:0.0" refused by server
这时请用 root 用户运行

xhost hostname

或简单地运行

xhost + 。

安装 DB2 补丁 4 之后，在/usr/IBMdb2/V7.1/bin/下有一条命令叫 db2icons，用该条命令可以将控制中心，命令

行处理器等的快捷键安装到程序菜单和桌面上。

参考资料:
有关 DB2 在其它一些 LINUX 平台上安装的说明请参考 http://www.linux.com/howto/DB2-HOWTO/index.html。

27 如何在 Red Hat Linux 7.1 上安装 DB2 EEE(扩展企业版)

内容
提要 本文给出了在 Red Hat Linux 上安装 DB2 EEE 的主要步骤
正文 清注意所有步骤都应以 root 用户登录, 除非特殊说明

安装步骤概述

1.在服务器上安装 Red Hat Linux 7.1
2.配置网络硬件
3.装载 NFS 文件系统

4.创建数据库的用户和组
5.激活 rsh
6.安装和配置 DB2
7.配置多个节点(数据库的多个分区)
8.配置数据库管理服务器
9.配置操作系统内核
10.创建数据库

安装步骤详述:

步骤 1. 安装 Red Hat 7.1

典型的 Red Hat 7.1 的安装并没有包含安装和运行 DB2 EEE 的所需要的所有软件包, 请根据需要安装下列软件

包, 另外请选择不要安装防火墙软件.
X - 如果要运行基与 Java 的 DB2 Control Center(控制中心)
Xinetd - DB2 通讯时需要这个软件包
rsh - 为 DB2 EEE 所用(ssh 将不工作)
pdksh - 安装 EEE 时需要
Nfs-utils - NFS 文件系统装载时需要

步骤 2. 配置网络硬件

DB2 EEE 要求分区间的通讯, 所以我们推荐 EEE 节点间的通讯最好用专有的网络, 下面这个例子通过在每台机

器上安装两个网卡实现了公有网络和 DB2 EEE 的私有网络的分离

机器 1
主机名: DB2lab1
网卡 1 - IP 地址 (公共): 9.19.156.33
- 子网掩码 : 255.255.252.0
网卡 2 - IP 地址(DB2 EEE 专用): 10.10.10.9
- 子网掩码: 255.255.255.0

机器 2
主机名: DB2lab2
网卡 1 - IP 地址 (公共): 9.19.156.34
- 子网掩码: 255.255.252.0
网卡 2 - IP 地址(DB2 EEE 专用): 10.10.10.10
- 子网掩码: 255.255.255.0

步骤 3.配置 NFS 文件系统

DB2 EEE 的可伸缩性来源于对硬件资源的最大可能的并行应用. 有一些配置文件是所有 DB2 分区都需要共享

的, 通过装载(Mount)NFS 文件系统这些文件才得以共享.
创建和测试 NFS Mount(装载)的步骤如下:

1 在 DB2 集群的所有节点上创建 /db2home 目录

2 在 DB2 集群的第一个节点上(该节点是实例所有者)导出 NFS 文件系统
(1) 在 DB2lab1 上创建/etc/exports 文件
(2) 在 /etc/exports 文件中加入下列入口: /db2home db2lab*.local.domain(rw)
(3) 在 DB2lab1 上重起 NFS 服务, 命令如下:
./etc/init.d/nfs restart
(4) 验正文件系统导出是可工作的, 用下面的语句
showmount --exports

3 在集群的所有其它机器上装载该文件系统
(1) 在/etc/fstab 文件中插入下面语句: 'DB2lab2:/db2home /db2home nfs rw 0 0'
(2) 在集群的所有机器上装载被导出的文件系统, 命令如下
mount /db2home

步骤 4 创建用户和组

DB2 安装程序能创建 EEE 所需的用户和组. 下面是手工创建所需用户和组的步骤, 手工创建是为了保证所有机

器的一致性

1. 在所有机器上创建所需的组, 命令如下:
groupadd -g 550 db2iadm
groupadd -g 551 db2fadm
groupadd -g 552 db2as

2. 在所有机器上创建所需的用户, 命令如下:
useradd -u 550 -g 550 -d /db2home/db2inst1 db2inst1
useradd -u 551 -g 551 -d /db2home/db2fenc1 db2fenc1
useradd -u 552 -g 552 -d /db2home/db2as db2as

3. 为所有机器上的用户设置密码，命令如下:
passwd db2inst1
passwd db2fenc1
passwd db2as
用户 db2inst1 将作为实例的所有者. 存储过程将会运行在 db2fenc1 用户下

步骤 5 安装 rsh

DB2 EEE 用 rsh 来远程地执行命令. 有两种方法可以激活 rsh. 第一种方法是在实例所有者的根目录下提供安全

文件 .rhosts. 第二种方法是为集群中的每一台机器提供安全文件 /etc/hosts.equiv. 这两种方法都会在文件中列

出允许发远程命令的用户及执行该命令的机器名. 在 Red Hat 7.1 中, root 用户是不能使用 rsh 的.

方法 1：
1. 创建/db2home/db2inst1/.rhosts.equiv 文件
2. 在.rhosts.equiv 文件中加入下列语句：
DB2lab1 db2inst1
DB2lab2 db2inst1
3. 在所有机器上重起 xinetd 服务，命令如下：
/etc/init.d/xinetd restart

方法 2：
1. 在集群的每一台机器上创建/etc/hosts.equiv 文件
2. 在 hosts.equiv 文件中加入下列语句：
DB2lab1 db2inst1
DB2lab2 db2inst1
3. 在所有机器上重起 xinetd 服务，命令如下：
/etc/init.d/xinetd restart

步骤 6 安装和配置 DB2 EEE

有两个脚本对于安装和配置 DB2 很重要， 一个是 db2_install, 一个是 db2setup. 前者只能安装 DB2 的库文件，

不能用来创建 EEE 的实例. 后者却两件事都可以做. 下面的例子用 db2setup 程序来完成安装.

1. 在每一个节点上运行 db2setup 程序，db2 将会被装到 /usr/IBMdb2/v7.1/目录下. 需要注意的是 db2setup 脚本

要求 libncurses.so.4 库能被正确查看，但 Red Hat 7.1 的缺省安装并没有这个库. 变通的方法是用下面的命令

来创建和 libncurses.so.5 的象征连接
ln -sf /usr/lib/libncurses.so.5 /usr/lib/libncurses.so.4

2.在第一台机器创建 DB2 实例(db2lab1),命令如下:(请注意只在第一台机器创建实例)
/usr/IBMdb2/v7.1/install/db2setup

步骤 7： 配置 EEE 的多个节点

在你安装和配置完 DB2 EEE 后， 你将需要创建 EEE 的新的分区. /db2home/db2inst1/sqllib/db2nodes.cfg 文件

定义了 DB2 EEE 中都存在哪些分区. 添加分区可以用下列步骤:

1. 在 db2nodes.cfg 文件中为新分区加一行记录，例如：
1 DB2lab2 0

2. 在创建实例的机器上打开/etc/services 文件并检查为 FCM 通讯保留的监听端口已存在，该入口示例如下：
db2inst1 60000/tcp(db2inst1 是实例名)

3. 登录集群中的其它机器并修改／etc／services 文件，加入同样的入口.

4. 编辑/db2home/db2inst1/sqllib/db2nodes.cfg 文件并指定 FCM 通讯经过的网络地址，样例如下：
0 db2lab1 0 10.10.10.9
1 db2lab2 0 10.10.10.10

步骤 8 配置管理服务器

管理服务器用来管理 DB2 实例，通过下列步骤来创建管理服务器：
1. 执行/usr/IBMdb2/V7.1／install/db2setup
2. 选择创建管理服务器选项

步骤 9 配置内核

为了提高 DB2 性能和可同时存在的数据库连接的个数，你需要修改 ipc 内核参数， 步骤如下：

1. 配置 msgmni 参数
sysctl -w kernel.msgmni=128

2. 为了系统启动时该参数配置就生效，需要在/etc/sysctl.conf 文件中加入下列入口：
kernel.msgmni=128

步骤 10 创建数据库

在缺省情况下， 数据库会被建在实例所有者的/home 目录下， 这不是我们所需要的， 因为实例所有者的

/home 目录位于 NFS 文件系统下. 所以我们需要在发出创建数据库命令的时候为数据库指定位置. 数据库系统

表只位于第一个数据库分区里，用户表会分布在各个分区里. 在创建数据库时指定的路径在集群中的每一台

机器上都应该存在而且实例所有者对该路径应该具有读和写权限，最好的选择是把该路径的所有权赋予实例

所有者. 具体操作步骤如下：

1. 分别在 db2lab1 和 db2lab2 上以 root 用户的身份创建一个目录
mkdir /testdbdir

2. 把对该目录的所有权赋予实例所有者
chown /testdbdir db2inst1

3. 以实例所有者的身份创建数据库
db2 "create database test on /testdbdir"

安装补丁程序后启动 DB2 管理实例时遇到 SQL1652N 和启动 DB2 实例时遇到 SQL5043N 错误

环境 [产品] DB2 UDB
[平台] UNIX
[版本] 5.x/6.1/7.x
问题 安装补丁程序后启动 DB2 管理实例时和 DB2 实例时遇到错误：
SQL1652N 启动 DB2 管理实例时

SQL5043N 启动 DB2 实例时
但是安装补丁前确没有这种情况。
解答 上述情况可能是由于安装 DB2 补丁后没有及时更新实例。

可以用 dasiupdt 更新 DB2 管理实例，用 db2iupdt 更新 DB2 实例。请参考 DB2 命令手册查看上述命令的使用

方法，或直接到 DB2 安装目录下的 instance 目录中不加参数运行上述命令查看联机帮助。

注意：在 UNIX 平台上安装 DB2 补丁一定要按照随补丁一起提供的安装 DB2 补丁的说明文件。除了更新实例

和管理实例外，安装补丁后还要对已有的数据库重新联编相关程序。

28 db2diag.log 中大量出现关于 TCP/IP 协议的 DIA3208E 错误

环境 [产品] DB2 UDB
[平台] AIX(多 CPU)
[版本] 7.x
问题 在 db2diag.log 中反复出现下面错误：
DIA3208E error encountered in TCP/IP protocol support. TCP/IP function "accept". Socket was "48". Errno

was "76".
出了使 db2diag.log 文件大小增长，该错误不影响 DB2 的使用
解答 这个问题已由 AIX 的补丁解决。相应的 AIX 补丁 APAR 号为 IY17704。
如果不能马上得到 AIX 的补丁，又希望解决此问题，可以用
db2set DB2TCPCONNMGRS=1
来避免此类日志的生成。

注意：
命令执行完还要运行:
db2 terminate
db2stop 停止实例

db2start 从新启动实例使命令生效

29 如何在命令行用 FETCH 命令查看用 DECLARE CURSOR(游标)指定的结果集

环境 [产品] DB2 UDB
[平台] 跨平台
[版本] 5.x/6.x/7.x
问题 如何在命令行用 FETCH 命令查看用 DECLARE CURSOR(游标)指定的结果集
解答 下面是一个简单的例子
定义 CURSOR（WITH HOLD 很重要，替代地，可以用 UPDATE COMMAND OPTION USING C OFF

将 AUTOCOMMIT 关掉避免 CURSOR 打开后自动关闭）:
DECLARE c1 CURSOR WITH HOLD FOR select * from staff

打开 CURSOR:
OPEN c1

查看结果集:
FETCH c1 for n rows or
FETCH c1 for all rows

关闭 CURSOR:
CLOSE c1

如何配置 ODBC.INI?

环境 产品: DB2 UDB
平台: Windows, Unix
版本: 6.x, 7.x

30 问题 如何配置 ODBC.INI?

解答 Microsoft 的 16 位“ODBC 驱动程序管理器”和所有非 Microsoft“ODBC 驱动程序管理器”都使用 odbc.ini

文件来记录关于可用驱动程序和数据源的信息。UNIX 平台上的“ODBC 驱动程序管理器” 还使用 odbcinst.ini

文件。尽管在大多数平台中必要的文件都由工具自动更新，但是在 UNIX 平台上的 ODBC 用户还是要人工编

辑这些文件。文件 odbc.ini（以及 odbcinst.ini，若需要的话）位于：

UNIX

运行 ODBC 应用程序的用户 ID 的主目录（在 UNIX 上， odbc.ini 文件名的前面有一个点：.odbc.ini ） 也可

以人工修改此文件。不要更改该文件中任何现存项目。要人工编辑此文件，执行下列步骤：

使用 ASCII 编辑器来编辑 odbc.ini 文件。
以下是一个样本 odbc.ini 文件：

```
[ODBC Data Sources]
MS Access Databases=Access Data (*.mdb)

[MS Access Databases]
Driver=D:\WINDOWS\SYSTEM\simba.dll
FileType=RedISAM
SingleUser=False
UseSystemDB=False
```

[ODBC Data Sources] 节列出每个可用数据源的名称和关联驱动程序的说明。
对于每个在 [ODBC Data Sources] 节中列出的数据源，都有一节列出该数据源的其他信息。这些节称为数据

源说明节。

在 [ODBC DATA SOURCE] 项下，添加下列行：

database_alias=IBM DB2 ODBC DRIVER

其中， database_alias 是在数据库目录中编目的数据库别名（由命令行处理器的 CONNECT TO 语句使用的

数据库名）。

将一个新项添加至“数据源说明”节中 ，以使该数据源与驱动程序相关联：

[database_alias]
Driver=x:\windows\system\db2cliw.dll

其中：

database_alias 是在数据库目录中编目的数据库的别名，在“数据源说明”节中列出。

x: 是安装 Windows 操作系统的驱动器。
添加了 IBM 数据源各项的示例文件如下所示：

[ODBC Data Sources]
MS Access Databases=Access Data (*.mdb)
SAMPLE=IBM DB2 ODBC DRIVER

[MS Access Databases]
Driver=D:\WINDOWS\SYSTEM\simba.dll
FileType=RedISAM
SingleUser=False
UseSystemDB=False

[SAMPLE]
Driver=D:\WINDOWS\SYSTEM\db2cliw.dll
Description=Sample DB2 Client/Server database