Best Practices for Using Oracle Transparent Gateway for DB2 and Oracle Transparent Gateway for DRDA

An Oracle White Paper
October 2005

ORACLE

Best Practices for Using Oracle Transparent Gateway for DB2 and Oracle Transparent Gateway for DRDA

Executive Overview 3
Introduction 3
DB2 Locking Behavior 3
SQL Dialects 4
Oracle Transparent Gateway - Concepts 4
Examples 5
Best Coding Practices 6
Use of SQL Functions 6
Transparent Gateway for DB2 - SQL-Function processing 7
Transparent Gateway for DRDA SQL-Function processing 8
Tuning Queries accessing a DB2 for z/OS database via the Oracle Transparent Gateways 10
Oracle SQL Tuning 10
How can post–processing be determined? 10
How can post-processing be circumvented? 12
Tuning Distributed Queries 12
DB2-SQL Tuning 14
DB2 Predicates 14
Dealing with the DB2 Locking Model 16
Miscellaneous 18
Problem Determination 20
Dictionary Views 20
Tracing 22
SQL Trace 22
Gateway Trace 23
Gateway Setup suggestions 24
Transparent Gateway for DB2 24
Transparent Gateway for DRDA 25
Conclusion 25

Best Practices for Using Oracle Transparent Gateway for DB2 and Oracle Transparent Gateway for DRDA

EXECUTIVE OVERVIEW

Oracle offers two solutions to integrate DB2 for z/OS databases into your distributed environment—the Oracle Transparent Gateway for IBM DB2 and the Oracle Transparent Gateway for DRDA. Both Gateways enable Oracle users to access mainframe DB2 data without knowing its DB2 characteristics. In fact, users may actually be unaware that the data resides on DB2. In the majority of cases, applications accessing DB2 for z/OS data have very good performance; however, the DB2 locking behavior and the DB2 SQL-dialect can affect the throughput of Oracle applications using Transparent Gateways to access DB2 data. This paper gives an overview of coding and design techniques for efficient use of the Oracle Transparent Gateway for DB2/DRDA.

INTRODUCTION

Oracle Transparent Gateways provide transparent access to non-Oracle datastores. Gateways enable users to access data without knowing whether that data actually resides in an Oracle database. In general, there is very little performance impact when accessing data through a gateway. When accessing DB2 for z/OS databases, however, the differences in the DB2 locking behavior and SQL dialect can affect the throughput of Oracle applications accessing this data using the Oracle Transparent Gateway for IBM DB2 or the Oracle Transparent Gateway for DRDA.

This paper first examines these differences between IBM DB2 and the Oracle database, and then provides a general overview of gateway concepts. The following sections of this paper then look in detail at how to tune Oracle queries to accommodate these differences, as well as how to tune the DB2 database to deal with the DB2 locking model. The remaining sections include information on monitoring gateway operations, as well as suggested gateway configurations for DB2 and DRDA.

DB2 Locking Behavior

Oracle introduced Multi-version Concurrency Control to provide read consistency. DB2 does not have such functionality, but instead, uses the following Isolation Levels to control concurrency:

- CS - Cursor Stability
- RR - Repeatable Read
- RS - Read Stability
- UR - Uncommitted Read

In most cases the Isolation Level CS is used, because this Isolation Level allows the maximum concurrency with data integrity. The DB2-Isolation Levels CS, RR and RS acquire "Shared Read Locks" to provide data integrity. The disadvantage of "Shared Read Locks" is that writers block readers and for that reason a lot of DB2 sites report concurrency problems. "Shared Read Locks" have to be explicitly released by a Commit or a Rollback statement – even in read only applications.

Many DB2 sites use the Isolation Level UR (Uncommitted Read, Dirty Read) to circumvent those concurrency problems. With this Isolation Level, the application acquires no page or row locks and can run concurrently with most other operations. However, the application is in danger of reading data that was changed by another operation but not yet committed.

SQL Dialects

The Oracle Transparent Gateways compensate for differences in SQL-dialects between Oracle and DB2. However, there are a couple of performance impacts when the DB2-Optimizer behavior is not considered in the SQL Queries that are passed to DB2. It is quite important to have knowledge of the DB2 data model and the DB2 Predicate Processing when SQL queries are executed against DB2.

ORACLE TRANSPARENT GATEWAY - CONCEPTS

The Oracle Transparent Gateway for DB2 and the Oracle Transparent Gateway for DRDA provide transparent SQL Access to DB2 for z/OS databases. Transparent means that the Oracle application developer can execute "Oracle-SQL-Queries" against DB2. This implies that the following transformations have to be performed by the Oracle Transparent Gateway and the Oracle server:

- Datatype Mapping between Oracle and DB2
- Pre-processing of the query before preparing/executing this query in DB2
  There are a couple of differences between the Oracle- and the DB2-SQL language. In the Pre-processing phase, Oracle specific language elements are removed from the query. After that, the query is executed in DB2.
- Post-processing
  If Oracle specific SQL-functionalities were removed from the query during the pre-processing phase, those SQL-functionalities have to be applied to a temporary result set in the post-processing phase. The Oracle Server performs the post-processing. Post-processing can have a

negative impact on the query performance, especially if scan operations on large DB2 objects are performed. Depending on the SQL clause, a pipe mechanism or temporary tables are used to carry out the post-processing

Examples

Example 1: Execution of a query with a similar syntax in Oracle and DB2

	Oracle Server	**Action performed by the Oracle server and the Transparent Gateway for DB2/DRDA**	**DB2**
Step 1:	Select max(salary) from tab@DB2 where name = 'SMITH'		
Step 2		Execute query in DB2	Select max(salary) from tab@DB2 where name = 'SMITH'
Step 3		Transfer Result Set to Oracle	

This figure shows the interaction between Oracle and DB2 when compatible SQL is used.

Example 2: Execution of a query with a different Syntax in Oracle and DB2. The function used in the Oracle SQL query is available in DB2, but has a different syntax.

	Oracle Server	**Action performed by the Oracle server and the Transparent Gateway for DB2/DRDA**	**DB2**
Step 1	Select Name,age from tab1@DB2 Where NVL(Age,18)= 18		
Step 2		**Pre-processing:** Select Name,age from tab1WhereValue(Age,18) = 18	
Step 3		Execute query in DB2	Select Name,age from tab1Where Value(Age,18)= 18
Step 4		Transfer Result Set to Oracle	

This figure shows the interaction between Oracle and DB2 when SQL functions that have the same meaning but different names are used. The Oracle-SQL dialect is translated to the DB2-SQL dialect.

Example 3: Execution of a query with an Oracle-SQL function that does not exist in DB2.

	Oracle Server	**Action performed by the Oracle server and the Transparent Gateway for DB2/DRDA**	**DB2**
Step 1	Select * from tab1@DB2 Where Soundex(Name)= 'Scott'		
Step 2		**Pre-processing:** Select * from tab1@DB2	
Step 3		Execute query in DB2	Select * from tab1@DB2
Step 4		Transfer of the result set to the Oracle Server	
Step 5	Select * from temptab Where Soundex(name)= 'Scott'	**Post-processing** Apply the missing SQL function	

This figure shows the interaction between Oracle and DB2 when Oracle-SQL functions that do not exist in DB2 are used.

BEST CODING PRACTICES

A lot of customers use the Oracle Transparent Gateway for DB2 and the Oracle Transparent Gateway for DRDA in an OLTP environment. One of the key-requirements of OLTP environments is "good" response times. In cases of heterogeneous access to DB2 Subsystems, "good" response times can only be achieved if the DB2 characteristics are considered. This paper gives an overview of coding and design techniques for efficient use of the Oracle Transparent Gateway for DB2/DRDA.

Use of SQL Functions

One key element of efficient SQL is the use of SQL-functions in WHERE clauses. The use of functions in WHERE clauses, has a bearing on the overall query performance.

If DB2 data is accessed by a Transparent Gateway for DB2/DRDA, the application developer should check if the function used in the WHERE clause will

result in post-processing. Post-processing has no performance impact if the result set of a query executed in DB2 consists of a few rows only. But there is an influence on the query performance if post-processing is performed, a sub-optimal access path to DB2 data is used, and a large number of rows are returned to Oracle.

Another important point is the use of functions on columns in the WHERE clause. Because DB2 does not support "Function Based Indexes", DB2 is not able to support the access by an index (DB2 Term: Predicate is not Indexable). In order to avoid a sub-optimal access path to DB2 data, the query that is sent to DB2 should be explained in DB2. This Technique is described in the section "Use of DB2 Explain".

The next section gives an overview of the Gateway SQL-Function processing.

Transparent Gateway for DB2 - SQL-Function processing

The SQL Functions used in queries that access DB2 tables are categorized in three areas:

- Compatible
  SQL functions with the same meaning and result on both the Oracle database server and DB2.
  - AVG
  - CONCAT
  - COUNT (*)
  - MAX
  - MIN
  - SUM
- Translated
  SQL Functions that provide the same functionality, but use a different name or syntax.
  - NVL
  - TO_NUMBER
  - TO_DATE
- Compensated
  Oracle-SQL functions that cannot fully be represented by DB2-SQL functions. These functions are compensated for by the post-processing, which was described earlier.
  - ABS, ACOS, ADD_MONTHS, ASIN,ASCII, ATAN, ATAN2, CEIL,

- CHAR_TO_ROWID, CHR, CONVERT, COS, COSH,DECODE, DUMP, EXP,
- FLOOR, GREATEST, HEXTORAW, INITCAP, INSTR,INSTRB, LAST_DAY,
- LEAST, LENGTH, LENGTHB, LN, LOG, LOWER, LPAD,LTRIM, MOD,
- MONTHS_BETWEEN, NEW_TIME, NEXT_DAY,
- NLS_INITCAP,NLS_LOWER, NLS_UPPER, NLSSORT, POWER,
- RAWTOHEX, REPLACE,ROUND, ROWIDTOCHAR, RPAD, RTRIM, SIGN,
- SIN, SINH, SOUNDEX, SQRT, STDDEV, SUBSTR, SUBSTRB, SYSDATE,
- TAN, TANH, TO_CHAR, TO_DATE, TO_LABEL, TO_MULTI_BYTE,
- TO_NUMBER, TO_SINGLE_BYTE, TRANSLATE, TRUNC, UID, UPPER,
- USER, USERENV, VARIANCE , VSIZE.

Transparent Gateway for DRDA SQL-Function processing

In the case of the Transparent Gateway for DRDA, the SQL Functions used in queries can be categorized in four areas:

- Compatible
  SQL functions with the same meaning and result on both the Oracle database server and DB2.
  - AVG
  - CONCAT
  - COUNT (*)
  - MAX
  - MIN
  - SUM
- Translated
  SQL Functions that provide the same functionality, but use a different name or syntax.
  - NVL
  - TO_NUMBER

- TRIM
- TO_DATE

- Compensated
  Oracle-SQL functions that cannot fully be represented by DB2-Functions.
  - ABS, ACOS, ADD_MONTHS, ASIN, ASCII, ATAN, ATAN2, CEIL,
  - CHAR_TO_ROWID, CHR, CONVERT, COS COSH, DECODE, DUMP, EXP,
  - FLOOR, GREATEST, HEXTORAW, INITCAP, INSTR, INSTRB, LAST_DAY,
  - LEAST, LENGTH, LENGTHB, LN, LOG, LOWER, LPAD, LTRIM, MOD,
  - MONTHS_BETWEEN, NEW_TIME, NEXT_DAY, NLS_INITCAP,
  - NLS_LOWER, NLS_UPPER, NLSSORT, POWER, RAWTOHEX, REPLACE,
  - ROUND, ROWIDTOCHAR, RPAD, RTRIM, SIGN, SIN, SINH,
  - SOUNDEX, SQRT, STDDEV, SUBSTR, SUBSTRB, SYSDATE, TAN, TANH,
  - TO_CHAR, TO_DATE, TO_LABEL, TO_MULTI_BYTE, TO_NUMBER,
  - TO_SINGLE_BYTE, TRANSLATE, TRUNC, UID, UPPER, USER, USERENV,
  - VARIANCE, VSIZE
- Native Semantics
  The Transparent Gateway for DRDA allows the native use of SQL functions. No special pre-processing or post-processing is done for SQL functions that are native-semantics-enabled. The functions are passed through to DB2 unmodified. This implies that a DB2 error will raise if the function, which is enabled for native semantics, is called with parameters that are not compatible to DB2. The Transparent Gateway for DRDA parameter "DRDA_CAPABILITY" can be used to enable functions for native Semantics. If SQL-transparency is not needed, native semantics are an easy way to improve performance.
  - ABS, ACOS, ASCII, ASIN, ATAN, ATAN2, CAST, CEIL, CHR, CONVERT, COS,

- COSH, DECODE, DUMP, EXP, FLOOR, GREATEST, HEXTORAW, INITCAP,
- INSTR INSTRB, LEAST, LENGTH, LENGTHB, LN, LOG, LOWER, LPAD,
- LTRIM, MOD, NLS_INITCAP, NLS_UPPER, NLS_LOWER, NLSSORT, NVL2,
- POWER, RAWTOHEX, REPLACE, REVERSE, ROUND, RPAD, RTRIM, SIGN,
- SIN, SINH, SQRT, STDDEV, SUBSTR, SUBSTRB, TAN, TANH, TO_NUMBER,
- TRANSLATE, TRIM, TRUNC, UPPER, VARIANCE, VSIZE

TUNING QUERIES ACCESSING A DB2 FOR Z/OS DATABASE VIA THE ORACLE TRANSPARENT GATEWAYS

If queries that access DB2 via the Transparent Gateway for DB2/DRDA do not perform well, there are two areas with tuning potential:

- Oracle SQL Tuning
  This area consists of an Oracle query explain and various Oracle traces
- DB2 SQL Tuning
  This area consists of a DB2 query explain and DB2 traces

Both the Oracle-SQL tuning and the DB2-SQL tuning help to answer questions about why the performance of a query is poor. There are a couple of reasons for poor query performance. One reason for poor query performance could be gateway post-processing.

Oracle SQL Tuning

How can post–processing be determined?

Gateway post-processing is an additional phase in the execution of a heterogeneous query. The application developer and the Oracle administrator are able to monitor post processing with one of the following methods:

- Explain Plan
  The Explain Plan output shows the query that is sent to DB2. If parts of the WHERE clause are missing, or joins are split, then the query is post-processed
- SQL Trace
  The SQL Trace can be used to monitor the execution behavior of a query. The advantage of the SQL Trace is that it can be switched on dynamically. The Oracle administrator is able to monitor a specific session if performance problems are reported. The remote query that is sent to DB2 can be visualized, if the EXPLAIN option of the TKPROF utility is used.

- Gateway Trace
  Changing gateway initialization parameters activates Gateway Trace.

The following example demonstrates how post-processing can be detected:

```
select * from scott.emp@DB2_for_ZOS where soundex(ename)
= 'SMITH' ;
```

The Oracle-SQL function SOUNDEX is not supported in DB2. Gateway post processing is performed, because of that function. The post-processing can be visualized by one of the previously defined methods. Explain plan is used in this example.

Explain Plan

The SQL Query can be explained as follows:

```
explain plan for
select * from scott.emp@DB2_for_ZOS where soundex(ename)
= 'SMITH';
```

The result of that operation is that the access path is written to the *USERID.PLAN_TABLE*. The content of the PLAN_TABLE can be read and formatted with the following query:

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY('plan_table',
null, 'ALL'));
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------
| Id  | Operation           |  Name       | Rows  | Bytes | Cost  | Inst   |IN-OUT|
------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |             |    20 |  1160 |    52 |        |      |
|*  1 |  FILTER             |             |       |       |       |        |      |
|   2 |   REMOTE            |             |       |       |       | DB2_F~ | R->S |
------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SOUNDEX("A1"."ENAME")='SMITH')

Remote SQL Information (identified by operation id):
----------------------------------------------------

   2 - SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO"
            FROM "SCOTT"."EMP" (accessing 'DB2_FOR_ZOS.US.ORACLE.COM' )
```

Post-processing can be recognized by the check of the “Remote SQL Information” (in that example with ID=2). “Remote SQL Information” contains the query that will be executed in DB2. Post-processing will be performed, because the WHERE clause, which contains a function that does not exist in DB2, is omitted.

Explain Plan is a proactive method to analyze the query performance. The SQL- and Gateway Trace are available after the execution of the Query. The setup and content of those traces is explained in the section “Problem Determination”.

How can post-processing be circumvented?

Post processing can be circumvented by not using nonexistent DB2-SQL functions in the WHERE clause. The following example shows how compensated functions can be removed from the WHERE clauses of a query.

The example-query uses the compensated Oracle-SQL functions TO_CHAR and SYSDATE in the WHERE clause. For that reason the Gateway pre-and post-processing is performed.

```
select a.c1 from
      tab1@DB2_FOR_ZOS a, tab2@DB2_FOR_ZOS b
   where
      a.c3  between
            to_char(sysdate-1,'YYYY-MM-DD')||'-00.00.00.000000' and
            to_char(sysdate,'YYYY-MM-DD')||'-00.00.00.000000' and
            a.c2=b.c2
```

If the TO_DATE and SYSDATE functionality is removed from the WHERE clause, and bind variables are used in the query instead, no post-processing is performed.

Two additional steps to determine the results of those functions are necessary before executing the query with bind variables.

```
DECLARE
date_minus_1 char(26);
date_curr char(26);
c1        char(10);
BEGIN
   select to_char(sysdate-1,'YYYY-MM-DD')||'-00.00.00.000000'
          into date_minus_1
          from dual;
   select to_char(sysdate,'YYYY-MM-DD')||'-00.00.00.000000'
          into date_curr
          from dual;
   Select a.c1
          into c1
          from
             tab1@DB2_FOR_ZOS a, tab2@DB2_FOR_ZOS b
                    where
                      a.c3 between date_minus_1 and date_curr
                      and a.c2=b.c2;
end;
```

Tuning Distributed Queries

The cost based optimizer uses Object statistics to determine the access path to data. Distributed queries are also optimized based on statistic values of the distributed database. Both the Transparent Gateway for DB2 and the Transparent Gateway for DRDA allow DB2 catalog statistics to be passed to the Oracle optimizer (explained in the section "Miscellaneous"). With that functionality, the Oracle optimizer can determine an accurate access path, even if the distributed queries access DB2 databases. Due to missing or inaccurate DB2 catalog statistics, it sometimes may be possible that the access path to the DB2 data is not optimal. In those cases, distributed joins can sometimes be split into single queries in DB2. The join is then performed on the Oracle server. A rewrite of the SQL-Query can resolve such behavior. The join condition has to be placed into a separate query block to pass it to DB2 unchanged.

The next example shows how to resolve such optimizer behavior; a local table and two remote DB2 tables are accessed:

```
select a.ename from
                    local_user.emp a,
                    tniewel.emp@db2_for_zos b,
                    scott.emp@db2_for_zos c
      where
           a.ename=b.ename and
           b.ename=c.ename;
```

The query was explained to visualize the execution plan. The access to DB2 is done by two single queries. The join operation of those tables is performed on the Oracle server.

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

-------------------------------------------------------------------------------
| Id  | Operation            |  Name       | Rows  | Bytes | Cost  | Inst   |IN-OUT|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             | 10933 |   224K|   108 |        |      |
|*  1 |  HASH JOIN           |             | 10933 |   224K|   108 |        |      |
|*  2 |   HASH JOIN          |             |    82 |  1148 |    55 |        |      |
|   3 |    REMOTE            |             |    15 |   105 |    52 | DB2_F~ | R->S |
|   4 |    TABLE ACCESS FULL | EMP         |    82 |   574 |     2 |        |      |
|   5 |   REMOTE             |             |  2000 | 14000 |    52 | DB2_F~ | R->S |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("B"."ENAME"="C"."ENAME")
   2 - access("A"."ENAME"="B"."ENAME")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   3 - SELECT "ENAME" FROM "TNIEWEL"."EMP" (accessing 'DB2_FOR_ZOS.US.ORACLE.COM' )

   5 - SELECT "ENAME" FROM "SCOTT"."EMP" (accessing 'DB2_FOR_ZOS.US.ORACLE.COM' )
```

If the two DB2 tables accessed in that example are placed into a separate query block, then the join is carried out in DB2. The NO_MERGE hint is necessary and causes Oracle not to merge the mergeable inline view that accesses DB2.

```
select a.ename from
       local_user.emp a ,
             (select /*+ NO_MERGE */ b.ename from
                              tniewel.emp@db2_for_zos b,
                                 scott.emp@db2_for_zos c
                           where b.ename=c.ename) bc
      where a.ename=bc.ename;
```

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------
| Id  | Operation            |  Name       | Rows  | Bytes | Cost  | Inst   |IN-OUT|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |    82 |  1148 |     3 |        |      |
|*  1 |  HASH JOIN           |             |    82 |  1148 |     3 |        |      |
|   2 |   TABLE ACCESS FULL  | EMP         |    82 |   574 |     2 |        |      |
|   3 |   VIEW               |             |   100 |   700 |     5 |        |      |
|   4 |    REMOTE            |             |       |       |       | DB2_F~ | R->S |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."ENAME"="BC"."ENAME")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   4 - SELECT A2."ENAME" FROM "TNIEWEL"."EMP" A2, "SCOTT"."EMP" A1 WHERE
             A2."ENAME"=A1."ENAME" (accessing 'DB2_FOR_ZOS.US.ORACLE.COM' )
```

Another method of forcing joins to be executed in DB2 is the creation of views in DB2. This method can be used, if remote joins are performed often and a split of remote joins is observed, because of SQL clauses or inaccurate DB2 catalog statistics. The application needs to access those DB2 views instead of the DB2 base tables. The disadvantage of this method is additional overhead in DB2. Depending on the DB2 view complexity, the view may show up with poor performance, because of a sub-optimal access path to the DB2 data. A recommendation is to explain and optimize those views in DB2 before using them in the Oracle application.

DB2-SQL Tuning

DB2 Predicates

To retrieve data, DB2 uses a two-stage architecture. The Data Manager, called Stage 1 applies simple SQL predicates and can use indexes to retrieve the data. If the Data Manager can't handle the predicate, it is passed to Relational Data System (RDS) as a Stage 2 predicate. For that reason, DB2 predicates in a WHERE condition are classified into the following categories:

- Stage 1
  These predicates offer the best DB2 performance.
- Stage 2
  These predicates cause additional CPU overhead, because of additional processing done by the RDS.
- Indexable
  Indexable predicate types can match index entries.

As described above, the application developer should make sure that Stage 1 predicates and indexable predicates are used in queries sent to DB2. But be careful – the DB2 predicate attributes are different in the various DB2 Versions.

A summary of DB2 predicate types for DB2 Version 7 can be found in the DB2 Version 7 Administration Guide (Document Number: SC26-9931-05).

http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsnagh15/5.8.3.2?SHELF=&DT=20050721100022

The next picture shows an extract of the DB2 predicate table.

Predicate Type	Index-able?	Stage 1?	Notes
COL = *value*	Y	Y	13
COL = *noncol expr*	Y	Y	9, 11, 12
COL IS NULL	Y	Y	
COL *op value*	Y	Y	
COL *op noncol expr*	Y	Y	9, 11
COL BETWEEN *value1* AND *value2*	Y	Y	
COL BETWEEN *noncol expr1* AND *noncol expr2*	Y	Y	9, 11
value BETWEEN COL1 AND COL2	N	N	
COL BETWEEN COL1 AND COL2	N	N	10
COL BETWEEN *expression1* AND *expression2*	N	N	7

Use of the DB2 Explain

Queries sent to DB2 can be extracted by the Oracle explain, the Oracle SQL Trace or the Gateway Trace. The optimization of the queries sent to DB2 can be done with the DB2 Explain. The DB2 Explain command is similar to the Oracle Explain. The access path to data can be visualized with the DB2 tool "Visual Explain" or with a simple query. The next example shows a DB2 Explain Plan executed in SPUFI.

The Join operation performed is a "Merge Scan Join" (Method=2), and the Access Method used is a "Tablespace Scan" (Accesstype=R). The detailed format of the DB2 PLAN_TABLE is also explained in the DB2 Administration Guide (Chapter 33: Using Explain to improve the query performance).

```
---------+---------+---------+---------+---------+---------+---------+---
    EXPLAIN PLAN FOR
    SELECT A2."ENAME" FROM "TNIEWEL"."EMP" A2, "SCOTT"."EMP" A1 WHERE
           A2."ENAME"=A1."ENAME" ;
---------+---------+---------+---------+---------+---------+---------+---
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0

---------+---------+---------+---------+---------+---------+---------+-------
    SELECT *
    FROM PLAN_TABLE WHERE TIMESTAMP=
              (SELECT MAX(TIMESTAMP) FROM PLAN_TABLE)
         ORDER BY QUERYNO, QBLOCKNO, PLANNO;
---------+---------+---------+---------+---------+---------+---------+-------
PLANNO  METHOD  TNAME               TABNO  ACCESSTYPE  MATCHCOLS  ACCESSNAME
---------+---------+---------+---------+---------+---------+---------+-------
     1       0  EMP                   1  R                   0
     2       2  EMP                   2  R                   0
```

Dealing with the DB2 Locking Model

As described in the introduction, DB2 uses "Shared Read Locks" to control concurrency. DB2 sites report the problem that nightly batches abnormally end, because of shared read locks, which are not released. Another problem area in DB2 is Lock Escalation. If a the number of locks on a table is beyond a threshold, the DB2 locking strategy switches to the next higher level, which is a table lock.

The DB2 Administration Guide gives hints on how to resolve the DB2 concurrency problems.

> ***Duration of page and row locks:*** If a page or row is locked, DB2 acquires the lock only when it is needed. When the lock is released depends on many factors, but it is rarely held beyond the next commit point
> ……
>
> ***Commit work as soon as is practical:*** To avoid unnecessary lock contentions, issue a COMMIT statement as soon as possible after reaching a point of consistency, even in read-only applications. To prevent unsuccessful SQL statements (such as PREPARE) from holding locks, issue a ROLLBACK statement after a failure. Statements issued through SPUFI can be committed immediately by the SPUFI autocommit feature.

As a consequence of that behavior, the Oracle application developer has to commit after Cursor operations. A close cursor leaves the last DB2-lock acquired active – the lock will be released after an explicit commit statement. But the application developer has to be careful – all open cursors will be closed implicitly by that commit statement.

Gateway Setup to Compensate for the DB2 Locking Behavior

Concurrency problems have a long history in DB2. For that reason, dirty reads were introduced with Version 4 of DB2 for MVS. One way to enable "dirty reads" is to BIND DB2 application plans (or packages) with the isolation level UR.

This isolation level does not acquire shared read locks. Commit´s after read operations are not necessary with that isolation level. The disadvantage of that isolation level is that there is a chance of reading logically inconsistent data.

The risks of the isolation level UR are also described in the DB2 Administration Guide

> However, if you use uncommitted read (UR) isolation, you can violate that basic principle of locking. Uncommitted read (UR) isolation lets users to see uncommitted data. Although the data is physically consistent, a number of logical inconsistencies can occur, or the data could be wrong.

A lot of sites using Oracle Transparent Gateway for DB2/DRDA decided to set up two instances of the Transparent Gateway for DB2/DRDA. One Gateway instance acquiring shared read locks and another Gateway instance with the "dirty read functionality".

Developers accessing DB2 data via an Oracle Transparent Gateway have to clarify which SQL operations do not use consistency. Those SQL operations can be processed via the gateway instance using dirty reads. Concurrency problems in DB2 can be reduced with that setup.

Dirty Reads - Setup of the Transparent Gateway for DB2

The DB2 isolation level is controlled by DB2 Bind parameters or "isolation-overriding" in SQL Statements. The Transparent Gateway for DB2 allows users to control the DB2 isolation level by binding the Transparent Gateway for DB2 PLAN with the isolation level UR. Another PLAN name should be chosen for that additional "Dirty Read" Gateway.

The next picture shows the modified BIND Plan statement of the JCL supplied in the G4DB2BPL member (part of the *db_hlq*.INSTLIB library).

```
BIND PLAN(G4DB2UR) -
   MEMBER(G4DB2IX, G4DB2PRC, G4DB2V51, G4DB2V61, G4DB2PIN) -
   LIBRARY('ORACLE.TG4DB292.G4DB2.SRCLIB') -
   QUALIFIER(OTGDB2)                -
   ACQUIRE(USE)                       -
   PKLIST(V92011A.G4DB2PLN,           -
          V92011B.G4DB2PLN,           -
          V92011C.G4DB2PLN,           -
          V92011D.G4DB2PLN,           -
          V92011E.G4DB2PLN,           -
          V92011F.G4DB2PLN,           -
          V92011G.G4DB2PLN,           -
          V92011H.G4DB2PLN,           -
          V92011I.G4DB2PLN,           -
          V92011J.G4DB2PLN,           -
          *.*)                        -
   ACTION(REPLACE) ISOLATION(UR)      -
   RELEASE(COMMIT)
```

Dirty Reads - Setup of the Transparent Gateway for DRDA

The Transparent Gateway for DRDA also allows influencing the DB2 isolation level during the DB2 BIND process. To add an additional Gateway instance for uncommitted read processing, the Transparent Gateway for DRDA intitialization file has to be changed. The parameter DRDA_ISOLATION_LEVEL has to be set to the value NC and another DRDA_PACKAGE_NAME has to be chosen. After that, a remote Bind of the Gateway package can be performed by executing the package GTW$_BIND_PKG@dblink. The next picture shows a modified Gateway initialization file.

```
#
# HS specific parameters
#
FDS_CLASS=TG4DRDA_DB2MVS
#TRACE_LEVEL=255
#LOG_DESTINATION=DB2.log
#ORACLE_DRDA_TCTL=debug.tctl
HS_COMMIT_POINT_STRENGTH=255
HS_NLS_DATE_FORMAT=YYYY-MM-DD
HS_LANGUAGE=AMERICAN_AMERICA.WE8ISO8859P1
HS_RPC_FETCH_REBLOCKING=off
HS_RPC_FETCH_SIZE=32767
HS_FDS_FETCH_ROWS=20
#
```

```
# DRDA specific parameters
#
DRDA_CONNECT_PARM=DRDACON1
DRDA_REMOTE_DB_NAME=DB2V7R1
DRDA_PACKAGE_COLLID=ORACLE
DRDA_PACKAGE_NAME=G2DRUR
DRDA_PACKAGE_CONSTOKEN=A92617CB3FE54701
DRDA_RECOVERY_USERID=ORADRDA
DRDA_RECOVERY_PASSWORD=ORADRDA
DRDA_ISOLATION_LEVEL=NC
```

Miscellaneous

z/OS Workload Manager support

Remote clients that access the Oracle Transparent Gateway for DB2 through the Oracle Net service are dispatched on a lightweight unit of work called an enclave SRB within the Net address space. The performance characteristics of such work can be effectively managed when used with WLM in goal mode. Enclave transactions are managed separately from one another as well as from the Oracle Net address space in which they run. WLM Service Classes determine the performance characteristics of enclaves.

The z/OS workload manager has an influence on the overall system performance in case of high CPU-usage periods (>80%). It is quite important that the work initiated by the Transparent Gateway for DB2 or the Transparent Gateway for DRDA is assigned to a non-discretionary and non-default service class. If the Service Class SYSOTHER is used, the performance of the Transparent Gateways will be unpredictable in high CPU-usage periods. Enclaves can be monitored with SDSF enclave display. The next example shows the SDSF enclave display

```
Display Filter View Print Options Help
-------------------------------------------------------------------------------
SDSF ENCLAVE DISPLAY  MVS08    ALL                      LINE 1-2 (2)
COMMAND INPUT ===>,                                      SCROLL ===> HALF
NP   TOKEN            SSType Status   SrvClass     CPU-Time OwnerSys OwnerJob
     3000000932       DDF    INACTIVE SYSOTHER         0.07 MVS08    D71SDIST
     2400000A5A       OSDI   INACTIVE ORACLEH          0.00 MVS08    NET8TNI
```

The example of an SDSF enclave display shows two enclaves. The enclave with the SSTYPE=DDF is originated by the Transparent Gateway for DRDA. The Transparent Gateway for DB2 owns the enclave with the SSTYPE=OSDI.

In that example, the enclave, which is owned by the Transparent Gateway for DRDA, uses the Service Class SYSOTHER. Performance problems may be the result of this assignment.

Transparent Gateway for DB2 Version 8

With the Transparent Gateway for DB2 Version 8, all work in DB2 was processed with the Gateway priority. No enclaves were used.

Transparent Gateway for DB2 Version 9.2

With Version 9 of the Transparent Gateway for DB2 workload from the network runs in an enclave until it gets to the Transparent Gateway for DB2 address space. The network enclave has to be classified (recommendation: Velocity Goal with high importance); otherwise the performance of the Transparent Gateway for DB2 is unpredictable. Starting with the Transparent Gateway for DB2 Version 9.2.0.6, work, which was processed in a TCB, is joined to enclaves. But not all work is running in enclaves. WLM Response time goals are possible with the latest patches of Transparent Grateway for DB2 Version 9.2.

Transparent Gateway for DB2 Version 10.1.0.3

With the Transparent Gateway for DB2 Version 10g, there were some architectural changes. In previous versions of the Transparent Gateway for DB2, CAF was used to connect to DB2. With the Transparent Gateway for DB2 Version 10g RRSAF is used instead. With that new interface to DB2 all work, initiated by a Transparent Gateway for DB2, runs in an enclave. WLM Response Time Goals are recommended with Version 10g of the Transparent Gateway for DB2.

Transparent Gateway for DRDA

The Distributed Data Facility (DDF) handles the DB2-requests initiated by the Transparent Gateway for DRDA. Inbound network tasks are assigned to an enclave SRB, which is classified by WLM. WLM Response Time Goals are recommended with the Transparent Gateway for DRDA.

Accounting of CPU Time

Transparent Gateway for DB2

Sometimes DB2-Adminitrators think that the Oracle Transparent Gateway for DB2 uses a lot of CPU internally, because the Gateway CPU usage shown by tools such as SDSF is excessive. The reason for this behavior is that the CPU accounted to the gateway and Net address space includes both - the CPU utilization of the gateway and the CPU time required for DB2 SQL processing. To monitor the amount of CPU the Gateway used for internal and SQL processing the enclave CPU field (ECPU field in the SDSF DA display) should be used.

Transparent Gateway for DRDA

The Oracle Transparent Gateway for DRDA is implemented as a DRDA-Application Requestor. The DB2 XXXXDIST address space does the communication to DB2. The CPU used by SQL executed via the Transparent Gateway for DRDA is accounted to that address space. The enclave CPU (ECPU in field in the SDSF DA display) should be used to monitor SQL-CPU usage.

Usage of DB2 Catalog Statistics

Both, the Transparent Gateway for DB2 and the Transparent Gateway for DRDA offer functionality that allows passing DB2 catalog statistics to the Oracle optimizer. This functionality is controlled by Gateway initialization parameters.

Usage of DB2 Catalog Statistics with the Transparent Gateway for DB2

The parameter DB2STATS enables the Transparent Gateway for DB2 to pass DB2 statistic values to the Oracle optimizer:

- DB2STATS=YES
  The DB2 catalog tables SYSIBM.SYSTABLES, SYSIBM.SYSKEYS, SYSIBM.SYSINDEXES are read once for every DB2 object accessed. The DB2 quantity structures are cached for later processing.
- DB2STATS=NO
  A default column cardinality of 2000 and an average row length of 100 is passed to the Oracle optimizer.

Usage of DB2 Catalog Statistics with the Transparent Gateway for DRDA

The parameter DRDA_OPTIMIZE_QUERY enables the Transparent Gateway for DRDA to pass DB2 statistic values to the Oracle optimizer:

- DRDA_OPTIMIZE_QUERY=TRUE
  The DB2 catalog tables SYSIBM.SYSTBLES, SYSIBM.SYSKEYS, and SYSIBM.SYSINDEXES are read once for every DB2 object accessed. The DB2 quantity structures are cached for later processing.
- DRDA_OPTIMIZE_QUERY=FALSE
  A default column cardinality of 2000 and an average row length of 100 is passed to the Oracle optimizer.

PROBLEM DETERMINATION

Dictionary Views

The Oracle Transparent Gateway for DB2 and the Transparent Gateway for DRDA are parts of the Oracle Heterogeneous Service architecture. The Gateways are implemented as Heterogeneous Service Agents. The Heterogeneous Service architecture offers a number of dynamic dictionary views, which help to monitor the Gateway operation. A description of those views is documented in the Oracle "Heterogeneous Connectivity Administrator's Guide" (Part No: B10764-01). The views V$HS_SESSION, V$HS_AGENT and V$HS_PARAMETER are useful for Gateway monitoring.

- V$HS_SESSION
  Lists the sessions for each agent, specifying the database link used

- V$HS_AGENT
  Identifies the set of Heterogeneous Services agents currently running on a given host, using one row for each agent process.
- V$HS_PARAMETER
  Lists Heterogeneous Services parameters and values registered in the Oracle database server.

The next example shows how to identify users who access a DB2 database using an Oracle Transparent Gateway.

```
set linesize 132
col username format a15
col machine format a15
col db_link format a20
col program format a15

select a.username,
      c.machine,
      c.program,
      b.db_link,
      c.process
 from
      v$session a,
      v$hs_session b,
      v$hs_agent c
 where
      a.sid=b.sid and
      c.agent_id=b.agent_id


USERNAME        MACHINE         PROGRAM         DB_LINK              PROCESS
--------------- --------------- --------------- -------------------- ---------
TNIEWEL         MVS08           G4DB2SRV        EPG1.US.ORACLE.COM   00030001
SCOTT           MVS08           G4DB2SRV        EPG1.US.ORACLE.COM   000D0000
```

DB2 administrators sometimes need to know the originator of a query that needs an excessive amount of CPU. The Oracle Transparent Gateway for DRDA allows administrators to detect the Oracle user with help of the DB2 command "Display Thread".

The next example shows how to determine an Oracle user starting from a DB2 Display Thread command.

```
DSNV401I  <D71F DISPLAY THREAD REPORT FOLLOWS -,
DSNV402I  <D71F ACTIVE THREADS -,
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN,
SERVER   RA *   578 G4002232     TNIEWEL  DISTSERV 0088  2078,
 V445-O2239471.OAE6.BDA3560EAC32=2078 ACCESSING DATA FOR,
      130.35.148.113,
SERVER   RA *   578 G4002230     TNIEWEL  DISTSERV 0088  2077,
 V445-O2239471.OAE5.BDA355E69277=2077 ACCESSING DATA FOR,
      130.35.148.113,
TSO      T  *     3 TNIEWEL      TNIEWEL           009C  2079,
DISPLAY ACTIVE REPORT COMPLETE,
DSN9022I  <D71F DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION,
***,
```

The output of the DB2 "-display thread(*)" command gives an overview of the active threads. The column "ID" contains a value that is concatenated. G4 indicates that the thread is a Transparent Gateway for DRDA thread (starting with Transparent Gateway for DRDA Version `9.0.1.0.1`). The number following that G4 indicates the Oracle Gateway process ID in a hexadecimal representation. In this example, there are two Gateway processes working, one with the process-id 8752(x'2230') and the other with the process-id 8754(x'2232'). The output of the query that shows how to identify users who access a DB2 database via an Oracle Transparent Gateway is as follows:

```
USERNAME        MACHINE         PROGRAM         DB_LINK             PROCESS
--------------- --------------- --------------- ------------------- ---------
TNIEWEL         bay.us.oracle.c g4drsrvd10t-db2 TDRDA               8752
                om              -7-pac@bay.us.o
                                racle.com

SCOTT           bay.us.oracle.c g4drsrvd10t-db2 TDRDA               8754
                om              -7-pac@bay.us.o
                                racle.com
```

This example shows that the Oracle-username can be identified by the process-id.

Tracing

Various Trace methods allow monitoring the SQL operations even if DB2 Subsystems are accessed.

SQL Trace

The SQL Trace lets administrators monitor SQL operations in specific sessions. The SQL Trace of a session can be enabled as follows:

- Alter session set SQL_trace=true
  Enables the trace of a current session.
- dbms_system.set_sql_trace_in_session(sid, serial,TRUE)
  This procedure can be used to enable the trace of a specific session.
- Enterprise Manager
  Oracle Enterprise Manager can be used to enable the trace of a specific session.

The Trace output has to be formatted with the TKPROF utility.

The next example shows a SQL Trace of a distributed query that is accessing a DB2 database via the Transparent Gateway for DB2. TKPROF was used with the explain option.

```
select a.ename
from
 local_user.emp a , (select /*+ NO_MERGE */b.ename from
  tniewel.emp@db2_for_zos b, scott.emp@db2_for_zos c where b.ename=c.ename)
  bc where a.ename=bc.ename


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          1          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.17          0          3          0          13
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.17          0          4          0          13

Misses in library cache during parse: 1
Optimizer goal: CHOOSE
Parsing user id: 62  (TNIEWEL)

Rows     Row Source Operation
-------  ---------------------------------------------------
     13  HASH JOIN
     15   TABLE ACCESS FULL EMP
     13   VIEW
     13    REMOTE


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
     13   HASH JOIN
     15    TABLE ACCESS (FULL) OF 'EMP'
     13    VIEW
     13     REMOTE [DB2_FOR_ZOS.US.ORACLE.COM]
               SELECT A2."ENAME" FROM "TNIEWEL"."EMP" A2, "SCOTT"."EMP" A1
                 WHERE A2."ENAME"=A1."ENAME"
```

Gateway Trace

The Gateway Trace lets administrators diagnose the Transparent Gateway for DB2/DRDA. Various trace levels can be set for debugging purposes. The 0Transparent Gateway for DB2/DRDA offers a Trace level that monitors SQL only. Application developers can use this trace to visualize SQL statements sent to DB2.

Transparent Gateway for DB2 – Gateway Trace

Setting the Gateway initialization parameter TRACELEVEL to 4 enables the Transparent Gateway for DB2 Gateway Trace. This TRACELEVEL reports only SQL statements. SQL statements sent to DB2 can be identified by the tag fdsapi.

Note that the Gateway Trace can only be enabled/disabled by restarting the gateway. The next figure shows an example of the Transparent Gateway for DB2 Gateway Trace with a TRACELEVEL=4.

```
////////////////////////////////////////////////////////////////////////
/   Oracle for OS/390 Diagnostic Trace  --  2005/01/15  07:19:07.357      /
/ System MVS08    Subsystem EPO3 Service EPG1     Session  16 PID 00020010/
////////////////////////////////////////////////////////////////////////
*** hoapars SQL : SELECT "NAME" FROM "SYSIBM"."SYSTABLES"
fdsapi:SQL statement follows.  Length=39
SELECT "NAME" FROM "SYSIBM"."SYSTABLES"
*** hoapars SQL : SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM",
"DEPTNO" FROM "TNIEWEL"."EMP"
fdsapi:SQL statement follows.  Length=95
SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO" FROM
"TNIEWEL"."EMP"
```

Transparent Gateway for DRDA – Gateway Trace

Tracing for diagnostic purposes is not enabled in the Transparent Gateway for DRDA for performance reasons. If diagnostic Traces are necessary, a debugging Gateway executable has to be specified in the listener.ora file. However, if only SQL should be monitored, it is not necessary to change the listener.ora file. To enable the Transparent Gateway for DRDA Gateway trace for SQL monitoring, the parameter TRACE_LEVEL has to be set to 4. The incoming SQL Statement is marked by the tag HGAPARS. The statement sent to DB2 is labeled by the tag HGAXMSQL. The next figure shows an example of a Transparent Gateway for DRDA Gateway-Trace output.

```
2005/04/22 15:34:53 hgapars: SQL Statement IN is:
SELECT "IN_INSTNUMMER", "GR_NUMMER", "VON_DATUM", "BIS_DATUM", "PLZ" FROM
"QUAL000"."DB2X500" WHERE ?<="BIS_DATUM" AND "VON_DATUM"<=? AND "IN_INSTNUMMER"=?
2005/04/22 15:34:53:*** HGAXMSQL ***
2005/04/22 15:34:53:hgapars: SQL Statement OUT is:
SELECT "IN_INSTNUMMER", "GR_NUMMER", "VON_DATUM", "BIS_DATUM", "PLZ" FROM
"QUAL000"."DB2X500" WHERE ?<="BIS_DATUM" AND "VON_DATUM"<=? AND "IN_INSTNUMMER"=?
```

GATEWAY SETUP SUGGESTIONS

There are a few parameters, which have an influence on the overall performance. The next section annotates some of the Gateway initialization parameters:

Transparent Gateway for DB2

- CURRDEGREE
    - CURRDEGREE=1
      With this setting, all queries that are passed to DB2 through the Transparent Gateway are not executed with DB2-parallelism.
    - CURRDEGREE=ANY
      With this setting, all queries that are passed to DB2 through the Transparent Gateway are executed with DB2-parallelism.

Depending on the DB2 Version, the control of parallelism in DB2 is very poor. Some users experienced that the complete z/OS-CPU-resources were used by only a few queries if the DB2 "parallel query" feature is used. The recommendation is to set the parameter CURRDEGREE=1 in case of OLTP applications. CURRDEGREE=ANY can be used in data warehousing environments, but keep in mind, that the influence of the parallel degree used by a query is atomic in DB2.

The maximum degree of query-parallelism can only be specified in the DSNZPARM parameter dataset.

- HS_RPC_FETCH_REBLOCKING=ON and
  HS_RPC_FETCH_SIZE=40000
  These settings influence the array size for SELECT statements. The HS_RPC_FETCH_SIZE parameter defines the number of bytes sent with each buffer from the gateway to the Oracle database server. Customers were able to improve the performance of SELECT statements with large result sets.

Transparent Gateway for DRDA

- HS_RPC_FETCH_REBLOCKING=ON and
  HS_RPC_FETCH_SIZE=32767 and
  HS_FDS_FETCH_ROWS=20
  These settings influence the array size for SELECT statements. The HS_RPC_FETCH_SIZE parameter defines the number of bytes sent with each buffer from the gateway to the Oracle database server. HS_FDS_FETCH_ROWS specifies the number of rows to fetch at one time. Customers were able to improve the performance of SELECT statements with large result sets by using those settings.

CONCLUSION

The Oracle Transparent Gateway for IBM DB2 and the Oracle Transparent Gateway for DRDA enable Oracle users to access mainframe DB2 data without knowing its DB2 characteristics. Although in the majority of cases, applications accessing DB2 for z/OS data have very good performance; by applying the coding and design techniques described in this paper, the effects of DB2 locking behavior and differences in the DB2 SQL-dialect can be further minimized.

Best Practices for Using Oracle Transparent Gateway for DB2 and Oracle Transparent Gateway for DRDA
October 2005
Contributing Authors: Thoma Niewel

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com