Oracle Database 10*g* SQL Enhancements

An introduction to the SQL enhancements provided with Oracle10g.

Presented by Dave Anderson

March 9, 2006

NYOUG

SKILLBUILDERS

What's New with 10g…

- Regular Expressions
- Case Insensitive Sort and Search
- New technique for coding quotes in literals
- MERGE statement enhancements
- ORA_ROWSCN pseudocolumn
- OLAP / DSS / Warehouse features
  - Partition Outer Join
  - SQL MODEL clause

…What's New with 10g

- New Functions, New Datatypes
- LOB Enhancements
- Enhanced CONNECT BY
- Nested Table Enhancements
- Temporary Table Enhancements
- Aggregates in RETURNING
- XQuery Support

Regular Expressions…

- Like UNIX regular expressions
- Powerful text pattern matching
  - Much more powerful than LIKE
- Compliant with POSIX and Unicode standards
  - Common metacharacters, matching lists, repeating sequences, subexpressions, backreferences, more…
- Release 2 adds Perl expression support

…Regular Expressions

- New condition
  - REGEXP_LIKE
- New functions
  - REGEXP_INSTR, REGEXP_REPLACE, REGEXP_SUBSTR

REGEXP_LIKE

```
DAVE@linux3> select street from customer
  2  where lower(street) like '%apt%';

STREET
------------------------------
738 Marci St. Apt 3A
388 Park Ave Apt 62

DAVE@linux3> select street from customer
  2  where regexp_like(street, 'apt', 'i');

STREET
------------------------------
738 Marci St. Apt 3A
388 Park Ave Apt 62
```

LIKE is TRUE if pattern matches entire string

Must code leading / trailing %

"i" for case insensitive search

REGEXP_LIKE is TRUE if pattern is within the string

Common Metasymbols

Dot matches any character except NULL

Asterisk operates on the dot. Dot matches zero or more characters

```
DAVE@linux3> select cust_no, lastname from customer
  2  where regexp_like(lastname, '.*son$','i');

   CUST_NO LASTNAME
---------- --------------------
         1 Son
         3 Anderson
```

Dollar sign anchors to end of line

```
DAVE@linux3> select cust_no, lastname from customer
  2  where regexp_like(lastname, '.+son$','i');

   CUST_NO LASTNAME
---------- --------------------
         3 Anderson
```

Plus sign operates on the dot. Dot must match 1 or more chars

Perl Expressions

- R2 adds support for Perl expressions

```
DAVE@linux3> select cust_no, lastname
  2  from customer
  3  where regexp_like(lastname, '\s')
  4  /

   CUST_NO LASTNAME
---------- --------------------
         1 Van Doren
```

Matching Sets and Repeaters

```
DAVE> select c1, c2  from t
  2  where regexp_like(c2, '[0-9]{3}-[0-9]{4}-[0-9]{3}');

---- ----------------------------------------
         1 My social security number is 111-1111-1111.
         3 My phone number is 111-1111 and my social secu
         4 My social security number is 333-3333-333
```

One of the characters in brackets must match string

Repeat match set 3 times

```
DAVE> select c1, regexp_substr(c2, '[0-9]{3}-[0-9]{4}-[0-9]{3}')
  2  from t
  3  where regexp_like(c2, '[0-9]{3}-[0-9]{4}-[0-9]{3}');

        C1 SSN
---------- --------------------
         1 111-1111-111
         3 222-2222-222
         4 333-3333-333
```

REGEXP_SUBSTR extracts the pattern from the string

Backreferences

REGEXP_REPLACE function replaces 2nd pattern with the 3rd pattern

Subexpression is enclosed in parenthesis

```
DAVE@lir    > update customer
  2    se  lastname =
  3    regexp_replace(lastname, '(.+)son$', '\1sen',1,1,'i')
  4    where regexp_like(lastname, '.+son$','i');

1 row updated.
```

"\1" refers to the 1st subexpression

Continuation of Notes

- This is a full page of notes.
- The slide is hidden.

Summary: Regular Expressions…

- Powerful text pattern search and update tool
  - Much more than what's presented here
- Supports common industry standard syntax

…Summary: Regular Expressions

- Supported datatypes
  - CHAR, VARCHAR2, CLOB
- But performance
  - LIKE and Intermedia Text support indexes
- Recommend reading:
  - Mastering Oracle SQL,
    - Sanjay Mishra and Alan Beaulieu
  - Metalink Note 263140.1
  - Oracle10g SQL Reference

Case-Insensitive Sort…

- Default case sensitivity controlled with NLS_SORT parameter
  - Session level and statement level control
- Default is usually ‘BINARY’
  - Append “_ci” for case-insensitive sort
  - Append “_ai” for case-insensitive *and* accent-insensitive sort

…Case-Insensitive Sort…

```
LINUX3> select * from nls_database_parameters
  2  where parameter='NLS_SORT';

PARAMETER                      VALUE
------------------------------ ----------------
NLS_SORT                       BINARY
```

```
LINUX3> select * from t order by c1;

C
-
A
A
B
a
a
b

6 rows selected.
```

```
LINUX3> alter session set nls_sort=binary_ci;

Session altered.
LINUX3> select * from t order by c1;

C
-
a
A
A
a
b
B

6 rows selected.
```

Lower case “a” is equivalent to upper case “A”

Continuation of Notes

- This is a full page of notes.
- The slide is hidden.

…Case-Insensitive Sort

```
  1  select * from t
  2* order by nlssort(c1, 'NLS_SORT=binary_ci')
LINUX3> /

C
-
a
A
A
a
b
B

6 rows selected.
```

Can also request in ORDER BY clause

Case-Insensitive Search…

```
SQL> alter session set nls_sort=binary_ci nls_comp=ansi;

Session altered.

SQL> select * from t where c1 = 'a';

C
-
a
A
A
a
```

Set NLS_COMP to ANSI

```
SQL> select * from t
  2  where c1 in ('a', 'b');

C
-
a
A
A
a
b
B
```

- =, !=, <, >, <=, >=
- [NOT] IN
- [NOT] BETWEEN
- CASE
- ORDER BY
- HAVING
- START WITH

…Case-Insensitive Search

- Use NLSSORT function for statement-level control

```
SQL> select * from t
  2  where nlssort(c1,'nls_sort=binary_ci')
  3        = nlssort('a','nls_sort=binary_ci');

C
-
a
A
A
a
```

Accent-Insensitive Sort and Search

```
SQL> select * from t where letter = 'a';

LETTER
----------
a

SQL> select * from t
  2  where nlssort(letter, 'nls_sort=binary_ai') =
  3        nlssort('a', 'nls_sort=binary_ai');

LETTER
----------
ä
a
A
```

Effect on Index Use…

- "Normal" indexes are binary indexes
  - "built according to a binary order of keys"
- Not used for case-insensitive searches
- Solution:
  - Create function-based index on NLSSORT

…Effect on Index Use

Create function-based index on NLSSORT

```
DAVE@linux3> create index ti_function on t
             (nlssort(c1,'nls_sort=binary_ci')) nologging;

Index created.
 . . . . . .
DAVE@linux3> alter session set nls_sort=binary_ci nls_comp=ansi;

Session altered.

DAVE@linux3> set autotrace traceonly
DAVE@linux3> select c1 from t where c1 = 'a';

154 rows selected.


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2 Card=173 B
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'T' (TABLE) (Cost=2
   2    1     INDEX (RANGE SCAN) OF 'TI_FUNCTION' (INDEX) (Cost=
```

Function-based index used to access table

Text Literal Quote Character

```
SQL> insert into ord
  2  (order_no, cust_no, order_date, payment_method,
  3              gift_message)
  4  values
  5  (order_seq.nextval, 1, sysdate, 'CA',
  6              q'[Peg's birthday present]');

1 row created.

SQL>
SQL> select order_no from ord
  2  where gift_message = q'+Peg's birthday present+';

  ORDER_NO
----------
        34
```

[is the opening quote delimiter

] is the closing quote delimiter – it identifies the close quote

+ is the opening and closing quote delimiter

MERGE Enhancements (1)

- Four enhancements to MERGE

1. Optional WHERE clause
   - WHEN MATCHED THEN UPDATE clause
     - Test source or target row
     - Perform update if only if condition is true
   - WHEN NOT MATCHED THEN INSERT clause
     - Test source rows
     - Insert only if condition is true
2. UPDATE clause has DELETE WHERE option
   - Test result of UPDATE
   - Delete row if meets condition

MERGE Example (1)

```
  4  begin
  5     merge /*+append*/ into target
  6        using (select * from source) source
  7        on    (target.c1 = source.c1)
  8     when matched then
  9          update set target.c2 = target.c2 + source.c2
 10          where source.c2 is not null
 11          delete where target.c2 > 100000
 12     when not matched then
 13          insert (target.c1, target.c2)
 14          values (source.c1, source.c2)
 15          where source.c2 < 50000;
 16
 17     dbms_output.put_line(sql%rowcount ||' rows merged');
. . . . .
24290 rows merged
.84 elapsed seconds
.39 cpu seconds
```

Conditional update

Conditional delete

Conditional insert; can use subquery

MERGE Enhancements (2)

3. Constant Filter Predicate
   - Code 0=1 to skip join in ON clause
4. Not required to specify both INSERT and UPDATE clauses

MERGE Example (2)

```
  4      v1                number := 0;
  5      v2                number := 1;
  6  begin
  7      merge /*+append*/ into target1
  8         using (select * from source) source
  9         on    (v1 = v2)
 10      when matched then
 11           update set target1.c2 = target1.c2 + source.c2
 12      when not matched then
 13           insert (target1.c1, target1.c2)
 14           values (source.c1, source.c2);
 15
 16      dbms_output.put_line(sql%rowcount ||' rows merged');
. . . . . .
48437 rows merged
1.6 elapsed seconds
.45 cpu seconds
```

Force != condition

Join logic bypassed

Only insert logic executes

Partition Outer Join…

- Purpose
  - Create dense reports
  - "Densify" data
- Dense data good for analytic windowing functions
  - Oracle8i feature
- "Easy" to code
  - Compared to manual
- Faster*

```
PROD_ID SALE_DATE  S
------- --------- --
      2 06-OCT-04 25
      2 09-OCT-04 25
      2 10-OCT-04 25
      2 11-OCT-04 25
      3 06-OCT-04 12
```

Sparse report

```
_ID DATE_COL   T
--- --------- --
  2 06-OCT-04 25
  2 07-OCT-04  0
  2 08-OCT-04  0
  2 09-OCT-04 25
  2 10-OCT-04 25
  2 11-OCT-04 25
  3 06-OCT-04 12
  3 07-OCT-04  0
  3 08-OCT-04  0
  3 09-OCT-04  0
  3 10-OCT-04  0
  3 11-OCT-04  0
```

Dense report

…Partition Outer Join

- Create partitions from results of left side
- Outer join each partition to right side result
- Union the results together

```
DAVE@linux3> select s.prod_id, d.date_col, nvl(total_sales,0)
  2  from (
  3        select prod_id, trunc(sale_date) sale_date,
  4               sum(amount) total_sales
  5        from sales
  6        group by prod_id, trunc(sale_date)
  7        ) s
  8  PARTITION BY (s.prod_id)
  9  right outer join
 10        (
 11         select trunc(date_col) date_col from c
 12        ) d
 13  on s.sale_date = d.date_col;
```

PID	DATE_COL	TOT
1	30-DEC-04	10
2	25-DEC-04	25
2	26-DEC-04	0
2	27-DEC-04	0
2	28-DEC-04	25
2	29-DEC-04	25
2	30-DEC-04	25
3	25-DEC-04	12

Continuation of Notes

- This is a full page of notes.
- The slide is hidden.

MODEL Clause Concepts

- Query result viewed as multi-dimensional array
- Spreadsheet-like capability from the database
  - Reduce need to dump data out of DB
- Integrating more analytic capability into the database
- Inter-row references
  - Without self-joins or unions
  - Should provide better performance
- Define calculations on “cells”
- “Upsert” capability allows projections
  - Update the array
  - Insert into the array

MODEL Components

- Partition
  - Defines array of data to be worked on
  - Formulas view each partition independently
- Dimension
  - Uniquely identifies cell in array
- Measure
  - Like a spreadsheet cell
- Rules
  - Formulas

MODEL: Example 1…

Q. What are projected Oracle-related website hits delivered by Google in 2005?

Dimension must uniquely identify measure (cell)

```
LINUX> select year, interest, tech_subject, hits
  2  from webhits
  3  where referer = 'GOOGLE' and tech_subject = 'ORACLE'
  4  model
  5  partition by (referer)
  6  dimension by (year, interest, tech_subject)
  7  measures (hits)
  8  rules (
  9   hits [2005, 'COURSE-OUTLINES', 'ORACLE'] =
 10         hits[2004, 'COURSE-OUTLINES', 'ORACLE'] * 1.10
 11  ,hits [2005, 'TECH-ARTICLES', 'ORACLE'] =
 12         hits[2004, 'TECH-ARTICLES', 'ORACLE'] * 1.10
 13         )
 14  order by year, interest;
```

Assume 10% growth

Positional reference to cell – DIMENSION BY (year, interest, tech_subject)

See output next page

…MODEL: Example 1

Array output from query on previous page

YEAR	INTEREST	TECH_SUBJECT	HITS
2002	COURSE-OUTLINES	ORACLE	5072
2003	COURSE-OUTLINES	ORACLE	5004
2003	TECH-ARTICLES	ORACLE	5947
2004	COURSE-OUTLINES	ORACLE	5025
2004	INTRANET	ORACLE	5023
2004	TECH-ARTICLES	ORACLE	9023
2005	COURSE-OUTLINES	ORACLE	5528
2005	TECH-ARTICLES	ORACLE	9925

Note since there is no rule for 'INTRANET' hits, there's no 2005 cell for 'INTRANET'

Projected 10% increase in hits in 2005

MODEL: Example 2

```
LINUX> select year, interest, tech_subject, hits
  2  from webhits
  3  where referer = 'GOOGLE' and tech_subject = 'ORACLE'
  4  model return updated rows
  5  partition by (referer)
  6  dimension by (year, interest, tech_subject)
  7  measures (hits)
  8  rules
  9   (hits [2005, 'COURSE-OUTLINES', 'ORACLE'] =
 10         AVG(hits)[year between 2002 and 2004,
 11                   'COURSE-OUTLINES', 'ORACLE'] * 1.10
 12   ,hits [2005, 'TECH-ARTICLES', 'ORACLE'] =
 13         AVG(hits)[year between 2002 and 2004,
 14                   'TECH-ARTICLES', 'ORACLE'] * 1.10   );

YEAR INTEREST             TECH_SUBJECT             HITS
---- -------------------- -------------------- -------
2005 TECH-ARTICLES        ORACLE                  8234
2005 COURSE-OUTLINES      ORACLE                  5537
```

Show only rows updated in or inserted into array

Use the average hits

ORA_ROWSCN…

- Approximate SCN of most recent change to row
  - By default, ORA_ROWSCN returns *block-level* SCN

```
LINUX> select c1, ora_rowscn from test;

        C1 ORA_ROWSCN
---------- ----------
         1    1375539

LINUX> update test set c1 = 2;

1 row updated.

LINUX> commit;

Commit complete.
```

Current SCN

```
LINUX> select c1, ora_rowscn from test;

        C1 ORA_ROWSCN
---------- ----------
         2    1459673
```

New SCN

…ORA_ROWSCN

- Can request precise SCN at table-level

```
LINUX> create table test (c1 number) rowdependencies;
Table created.
[ load table ]
LINUX> select c1,
  2> ora_rowscn from te

        C1 ORA_ROWSCN
---------- ----------
         1    1469023
        10    1469023
       100    1469023
```

```
LINUX> update test set c1=2 where c1=1;
1 row updated.

LINUX> commit;
Commit complete.

LINUX> select c1, ora_rowscn from test;

        C1 ORA_ROWSCN
---------- ----------
         2    1469051
        10    1469023
       100    1469023
```

New SCN

Original SCN's

ORA_ROWSCN Use

- Can be used to prevent lost updates
- New, easier optimistic locking technique

```
LINUX> select c1, ora_rowscn from test where c1=10;

        C1 ORA_ROWSCN
---------- ----------
        10    1469023

<application code to display data retrieved>

LINUX> update test set c1=1000
  2    where c1=10 and ora_rowscn=1469023;

1 row updated.
```

Capture SCN

Update only if row has not changed while viewing

New Datatypes

- BINARY_FLOAT
  - 4 bytes + 1 length byte
  - 32 bit single-precision floating point
- BINARY_DOUBLE
  - 8 bytes + 1 byte length field
  - 64 bit single-precision floating point
- IEEE754 compatible
  - Calculations can perform faster
- Compare to NUMBER
  - Variable length 1 to 22 bytes
  - Maximum length number 38 digits
  - More precise

LOB Enhancements

- Up to 8 TB or 128 TB
  - Depends on DB blocksize
- DML performance
- Remote DML support
- Parallel LONG to LOB conversion
- Transportable tablespace support
- Regular expression support
- NCLOB / CLOB implicit conversion
- IOT support for LOBs

New Functions

- CARDINALITY
- COLLECT
- CORR_S
- CORR_K
- CV
- ITERATION_NUMBER
- LNNVL
- MEDIAN
- NANVL
- ORA_HASH
- POWERMULTISET
- PRESENTNNV
- PRESENTV
- PREVIOUS
- REGEXP_INSTR
- REGEXP_REPLACE
- REGEXP_SUBSTR
- REMAINDER
- SCN_TO_TIMESTAMP
- SET
- STATS_*
- TIMESTAMP_TO_SCN
- TO_BINARY_DOUBLE
- TO_BINARY_FLOAT

RETURNING Aggregates

- Aggregates in RETURNING clause

```
DAVE@linux3> var x number
DAVE@linux3> update t set c1=c1-2
  2          returning sum(c1) into :x;

1 row updated.
```

Returns sum of updated values

CONNECT_BY_ROOT

- Operator provides access to root row

```
DAVE@linux3> select lastname as name,
  2          connect_by_root lastname as root,
  3          connect_by_root emp_no   as mgr_no
  4   from employee
  5   start with emp_no = 1
  6   connect by prior emp_no = mgr;

NAME          ROOT             MGR_NO
------------- ------------ ----------
Gardinia      Gardinia              1
Anderson      Gardinia              1
Somers        Gardinia              1
```

CONNECT BY: NOCYCLE

- NOCYCLE keyword
  - Use when child is also a parent
  - Suppresses ORA-01436 error

```
DAVE@linux3> select lastname as name,
  2          sys_connect_by_path(lastname, '/') AS path
  3   from employee
  4   connect by prior emp_no = mgr;
from employee
     *
ERROR at line 3:
ORA-01436: CONNECT BY loop in user data

DAVE@linux3> select lastname as name,
  2          sys_connect_by_path(lastname, '/') AS path
  3   from employee
  4   connect by nocycle prior emp_no = mgr;
```

CONNECT BY: Pseudocolumns

- CONNECT_BY_ISCYCLE
  - Returns 1 if row is part of a cycle (has a child that is also an ancestor)

```
DAVE@linux3> select lastname as name,
  2          connect_by_iscycle
  3   from employee
  4   where connect_by_iscycle = 1
  5   connect by nocycle prior emp_no = mgr;

NAME          CONNECT_BY_ISCYCLE
------------ ------------------
Gardinia                      1
```

- CONNECT_BY_ISLEAF
  - Returns 1 if current row is a leaf

COMMIT Enhancements

- Can COMMIT w/o waiting for log write
  - Performance increase
  - BATCH groups redo streams for write
- Can lose transaction
- Might consider for data loads with frequent commits

```
SYSTEM@orcl> commit write batch nowait;

Commit complete.
```

```
SYSTEM@orcl> show parameter commit_write

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
commit_write                         string
```

Miscellaneous SQL Features

- Nested Table Enhancements
  - Compare two nested tables with
    - =, !=, IN and NOT IN
  - New conditions
    - SUBMULTISET, IS A SET
  - New “multiset” operators
    - MULTISET EXCEPT, MULTISET UNION, MULTISET INTERSECT
- Temporary tables now support VARRAY columns

Introduction to XQuery

- XML query language
  - SQL is for relational tables
  - XQuery is for XML data
- Based on XPath
- In development by W3C
  - Not finalized yet
  - Oracle support might change
    - Caution building apps on this release
- Support from all major vendors
  - Oracle, Altova, Microsoft, Sun, IBM, many more

Oracle XQuery Support

- Database includes XQuery engine
  - Native XQuery execution
  - Index support
- Oracle Application Server too
  - Use to combine XML from many sources

XQuery Examples

R2

```
DAVE@linux3> select XMLQuery('for $i in ora:view
("xml_test") return $i' returning content) x
  2   from dual;

X
------------------------------
<ROW><ID>1</ID><DOC><ROWSET>
  <ROW>
    <TABLE_T>
      <VERS_MAJOR>1</VERS_MAJ
```

```
DAVE@linux3> xquery for $i in ora:view ("xml_test")
 2  return $i
 3  /

Result Sequence
--------------------------------
<ROW><ID>1</ID><DOC><ROWSET>
  <ROW>
    <TABLE_T>
```

Summary…

- Regular Expressions
  - Much more powerful than LIKE
  - R2 adds Perl expression support
- Case Insensitive Sort and Search
  - Convenient and possibly efficient technique for negating case
- New technique for coding quotes in literals
  - Simplify coding of literal quote

…Summary…

- MERGE Enhancements
  - More functionality for this 9i invention
- Row-level dependency tracking with ORA_ROWSCN
  - New optimistic locking technique
- Analytic / DSS / Warehouse features
  - Partition Outer Join
    - Densify data
  - SQL MODEL clause
    - Spreadsheet-like reports

…Summary…

- LOB Support Enhancements
  - Terabyte size LOBs and much more
- BINARY_DOUBLE and BINARY_FLOAT
  - Provide better performance
  - But not same precision
- CONNECT BY
  - CONNECT_BY_ROOT, NOCYCLE and pseudo-columns

…Summary

- Aggregates in RETURNING
  - returning sum(c1) into :x
- Nested Table Enhancements
  - Compare nested tables
  - Condition and Operators on Nested Tables
- VARRAY columns in temporary tables
- XQuery support in Release 2

Oracle 10*g* Classes

- Oracle 10*g* Administration
  - March 20 – 24, NYC
  - May 1 – 5, NYC
- Oracle 10*g* New Features for Administrators
  - May 9 – 12, Richmond, VA
  - May 15 – 18, NYC
- Oracle 10*g* New Features for Developers
  - May 9 – 11, NYC

To Register:

Online at skillbuilders.com

Or Call 888-803-5607

Or, call us to teach a class at your site:

888-803-5607

The End

- Thanks for listening!
  - Dave Anderson
  - dave@skillbuilders.com
  - www.skillbuilders.com

FREE Tech papers and Presentations

FREE Oracle Tutorials

FREE Practice Exams