Insert Information Protection Policy Classification from Slide 8
Five things you *probably* didn’t know about SQL

Thomas Kyte
http://asktom.oracle.com/
Who am I

- Been with Oracle since 1993
- User of Oracle since 1987
- The “Tom” behind AskTom in Oracle Magazine
  www.oracle.com/oramag
- Expert Oracle Database Architecture
- Effective Oracle by Design
- Expert One on One Oracle
- Beginning Oracle
Five things you probably didn’t know about SQL

- SQLNet Compression
- NULLS and Indexes and Cardinality
- You are being watched!
- Scalar Subquery Caching
- Calling statement level non-deterministic functions
SQLNet Compression
SQLNet Compression

• How you retrieve the data matters
• Not all result sets are the same – even if they have the same data
SQLNet Compression

ops$tkyte%ORA11GR2> create table t
    2   as
    3   select *
    4   from all_objects;
Table created.

ops$tkyte%ORA11GR2> begin
    2       dbms_stats.gather_table_stats( user, 'T' );
    3   end;
    4   /
PL/SQL procedure successfully completed.
SQLNet Compression

ops$tkyte%ORA11GR2> set arraysize 15

ops$tkyte%ORA11GR2> set autotrace traceonly statistics
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t;
72228 rows selected.

Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>consistent gets</td>
<td>5794</td>
</tr>
<tr>
<td>bytes sent via SQL*Net to client</td>
<td>8015033</td>
</tr>
<tr>
<td>bytes received via SQL*Net from client</td>
<td>53385</td>
</tr>
<tr>
<td>SQL*Net roundtrips to/from client</td>
<td>4817</td>
</tr>
<tr>
<td>rows processed</td>
<td>72228</td>
</tr>
</tbody>
</table>
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t order by timestamp;
72228 rows selected.

Statistics

1031  consistent gets
3427630 bytes sent via SQL*Net to client
53385  bytes received via SQL*Net from client
 4817  SQL*Net roundtrips to/from client
 72228  rows processed
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t order by timestamp, object_type, owner;
72228 rows selected.

Statistics

1031 consistent gets
3280011 bytes sent via SQL*Net to client
53385 bytes received via SQL*Net from client
4817 SQL*Net roundtrips to/from client
72228 rows processed
SQLNet Compression

ops$tkyte%ORA11GR2> set arraysize 100
ops$tkyte%ORA11GR2> set autotrace traceonly statistics
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t;
72228 rows selected.

Statistics

- 1842 consistent gets
- 7482943 bytes sent via SQL*Net to client
- 8362 bytes received via SQL*Net from client
- 724 SQL*Net roundtrips to/from client
- 72228 rows processed
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t order by timestamp;
72228 rows selected.

Statistics

1031 consistent gets
2907819 bytes sent via SQL*Net to client
8362 bytes received via SQL*Net from client
724 SQL*Net roundtrips to/from client
72228 rows processed
SQLNet Compression

ops$tkyte%ORA11GR2> select * from t order by timestamp, object_type, owner;
72228 rows selected.

Statistics

<table>
<thead>
<tr>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1031</td>
<td>consistent gets</td>
</tr>
<tr>
<td>2760200</td>
<td>bytes sent via SQL*Net to client</td>
</tr>
<tr>
<td>8362</td>
<td>bytes received via SQL*Net from client</td>
</tr>
<tr>
<td>724</td>
<td>SQL*Net roundtrips to/from client</td>
</tr>
<tr>
<td>72228</td>
<td>rows processed</td>
</tr>
</tbody>
</table>
## SQLNet Compression

<table>
<thead>
<tr>
<th>No Order 15</th>
<th>Some Order 15</th>
<th>Very Ordered 15</th>
<th>No Order 100</th>
<th>Some Order 100</th>
<th>Very Ordered 100</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bytes Sent</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.01 m</td>
<td>3.42 m</td>
<td>3.28 m</td>
<td>7.48 m</td>
<td>2.90 m</td>
<td>2.76 m</td>
</tr>
<tr>
<td><strong>% of original</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>43%</td>
<td>41%</td>
<td>93%</td>
<td>36%</td>
<td>34%</td>
</tr>
<tr>
<td><strong>Consistent Gets</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5832</td>
<td>1033</td>
<td>1033</td>
<td>1741</td>
<td>1033</td>
<td>1033</td>
</tr>
</tbody>
</table>

```sql
ops$tkyte%ORA11GR2> select round(1033*8/1024,2) from dual;
ROUND (1033*8/1024,2)
---------------------
     8.07
```
## SQLNet Compression

<table>
<thead>
<tr>
<th></th>
<th>No Order</th>
<th>Some Order</th>
<th>Very Ordered</th>
<th>No Order</th>
<th>Some Order</th>
<th>Very Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>Bytes Sent</strong></td>
<td>7.39 m</td>
<td>2.82 m</td>
<td>2.67 m</td>
<td>7.48 m</td>
<td>2.90 m</td>
<td>2.76 m</td>
</tr>
<tr>
<td><strong>% of original</strong></td>
<td>92%</td>
<td>35%</td>
<td>33%</td>
<td>93%</td>
<td>36%</td>
<td>34%</td>
</tr>
<tr>
<td><strong>Consistent Gets</strong></td>
<td>1105</td>
<td>1033</td>
<td>1033</td>
<td>1741</td>
<td>1033</td>
<td>1033</td>
</tr>
</tbody>
</table>

```sql
ops$tkyte%ORA11GR2> select round(1033*8/1024,2) from dual;

ROUND (1033*8/1024,2)
-----------------------
8.07
```
NULLS and Indexes and Cardinality
“Wrong cardinality = Wrong Plan”
NULLs and Cardinality

```sql
ops$tkyte%ORA11GR2> create table t
  2  pctfree 20
  3  as
  4  select a.*,
  5    case when mod(rownum,100) <= 50
  6        then last_ddl_time
  7    end end_date
  8  from all_objects a;
```

Table created.
NULLs and Cardinality

```
ops$tkyte%ORA11GR2> create index t_idx
    2                       on t(end_date);

Index created.
```
NULLs and Cardinality

ops$tkyte%ORA11GR2> select count(*)
   2 from t
   3 where end_date
   4  between to_date('01-sep-2010', 'dd-mon-yyyy')
   5      and to_date('30-sep-2010', 'dd-mon-yyyy');

    COUNT (*)
    ----------
       36267
NULLs and Cardinality

```
opstkte%ORA11GR2> begin
  2       dbms_stats.gather_table_stats(user, 'T');
  3     end;
  4   /

PL/SQL procedure successfully completed.
```
### NULLs and Cardinality

```sql
ops$tkyte%ORA11GR2> select count(*),
    2     count(distinct end_date),
    3     count(end_date),
    4     min(end_date),
    5     max(end_date)
    6  from t;

<table>
<thead>
<tr>
<th>CNT</th>
<th>CNTD</th>
<th>CNT2</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>72228</td>
<td>703</td>
<td>36850</td>
<td>01-OCT-02</td>
<td>30-SEP-11</td>
</tr>
</tbody>
</table>
```
NULLs and Cardinality

```sql
ops$tkyte%ORA11GR2> set autotrace traceonly explain
ops$tkyte%ORA11GR2> select *
   2   from t
   3   where end_date
   4       between to_date( '01-sep-2010', 'dd-mon-yyyy' )
   5       and to_date( '30-sep-2010', 'dd-mon-yyyy' );
```

**Execution Plan**

Plan hash value: 1601196873
NULLs and Cardinality

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>36024</td>
<td>3588K</td>
<td>339 (1)</td>
<td>00:00:05</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL</td>
<td>T</td>
<td>36024</td>
<td>3588K</td>
<td>339 (1)</td>
<td>00:00:05</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter("END_DATE"<=TO_DATE('2010-09-30 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "END_DATE">=TO_DATE('2010-09-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
NULLs and Cardinality

```sql
ops$tkyte%ORA11GR2> update t
   2   set end_date =
   3       to_date( '01-jan-9999','dd-mon-yyyy' )
   4   where end_date is null;

35378 rows updated.

ops$tkyte%ORA11GR2> commit;
Commit complete.
```
NULLs and Cardinality

begin
    dbms_stats.gather_table_stats(user, 'T');
end;
/

PL/SQL procedure successfully completed.
NULLs and Cardinality

```sql
ops$tkyte%ORA11GR2> select * 2    from t 3   where end_date 4 between to_date('01-sep-2010', 'dd-mon-yyyy') 5 and to_date('30-sep-2010', 'dd-mon-yyyy');
```

Execution Plan
```
Plan hash value: 470836197
```
NULLs and Cardinality

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>175</td>
<td>18375</td>
<td>10 (0)</td>
</tr>
<tr>
<td>1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T</td>
<td>175</td>
<td>18375</td>
<td>10 (0)</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX RANGE SCAN</td>
<td>T_IDX</td>
<td>175</td>
<td></td>
<td>2 (0)</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter("END_DATE"<=TO_DATE(' 2010-09-30 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "END_DATE">=TO_DATE(' 2010-09-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
“Wrong cardinality = Wrong Plan”
Nulls and Indexes

- There is a pervasive myth that indexes and NULLs are like matter and anti-matter
- There is the thought that “where column is null” cannot use an index
- There is a thought that NULLs are not indexed
- None of that is true…
NULLs and Indexes

```sql
ops$tkyte%ORA11GR2> create table t
    2   as
    3   select a.*,
    4       case when mod(rownum,100) > 1
    5       then object_type
    6       end otype
    7   from all_objects a;

Table created.
```
NULLs and Indexes

ops$tkyte%ORA11GR2> select count(*) from t where otype is null;

    COUNT(*)
-------
   1445
NULLs and Indexes

ops$tkyte%ORA11GR2> begin
  2   dbms_stats.gather_table_stats( user, 'T' );
  3 end;
  4 /

PL/SQL procedure successfully completed.
NULLs and Indexes

ops$tkyte%ORA11GR2> create index t_idx
     on t(otype,owner);

Index created.
NULLs and Indexes

ops$tkyte%ORA11GR2> set autotrace traceonly explain
ops$tkyte%ORA11GR2> select * from t where otype is null;

Execution Plan

Plan hash value: 470836197
NULLs and Indexes

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1445</td>
<td>149K</td>
<td>96</td>
<td>00:00:02</td>
</tr>
<tr>
<td>1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T</td>
<td>1445</td>
<td>149K</td>
<td>96</td>
<td>00:00:02</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX RANGE SCAN</td>
<td>T_IDX</td>
<td>1445</td>
<td></td>
<td>7</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 - access("OTYPE" IS NULL)
NULLs and Indexes

ops$tkyte%ORA11GR2> drop index t_idx;

Index dropped.

ops$tkyte%ORA11GR2> create index t_idx
    on t(otype,0);

Index created.
### NULLs and Indexes

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1445</td>
<td>149K</td>
<td>96</td>
<td>00:00:02</td>
</tr>
<tr>
<td>1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T</td>
<td>1445</td>
<td>149K</td>
<td>96</td>
<td>00:00:02</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX RANGE SCAN</td>
<td>T_IDX</td>
<td>1445</td>
<td></td>
<td>7</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 - access("OTYPE" IS NULL)
Nulls and Indexes

• What is true is that entirely NULL key entries are not made in B*Tree indexes

• Therefore, an index on just OTYPE cannot be used to find NULLs

• But – what about B*Tree cluster indexes and Bitmap indexes?
You are being WATCHED!
You are being WATCHED!

• 9i and before – V$ tables
• 10g – ASH and AWR are obvious
• But there is more
  – We watch what you ask for and change how statistics are gathered based on that.
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> create table t
  2   as
  3   select a.*,
  4       case when rownum < 500
  5       then 1
  6       else 99
  7       end some_status
  8   from all_objects a
  9 /
Table created.
```
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> begin
  2  dbms_stats.gather_table_stats(user,'T');
  3  end;
  4  /

PL/SQL procedure successfully completed.
```
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select histogram
     2   from user_tab_cols
     3   where table_name = 'T'
     4   and column_name = 'SOME_STATUS';

HISTOGRAM
----------
NONE
```
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> create index t_idx
               on t(some_status);
```

Index created.
You are being WATCHED!

ops$tkyte%ORA11GR2> set autotrace traceonly explain
ops$tkyte%ORA11GR2> select * from t where some_status = 1;

Execution Plan
-------------------------------------------------------------
Plan hash value: 1601196873

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>36115</td>
<td>3526K</td>
<td>300 (1)</td>
<td>00:00:04</td>
</tr>
<tr>
<td>1</td>
<td>TABLE ACCESS FULL</td>
<td>T</td>
<td>36115</td>
<td>3526K</td>
<td>300 (1)</td>
<td>00:00:04</td>
</tr>
</tbody>
</table>
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> select * from t where some_status = 99;
```

Execution Plan

```
Plan hash value: 1601196873

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>36115</td>
<td>3526K</td>
<td>300</td>
<td>00:00:04</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL</td>
<td>T</td>
<td>36115</td>
<td>3526K</td>
<td>300</td>
<td>00:00:04</td>
</tr>
</tbody>
</table>
```
You are being WATCHED!

ops$tkyte%ORA11GR2> begin
  2 dbms_stats.gather_table_stats( user, 'T' );
  3 end;
  4 /

PL/SQL procedure successfully completed.
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> select histogram
    2    from user_tab_cols
    3    where table_name = 'T'
    4    and column_name = 'SOME_STATUS';

HISTOGRAM
            -------------
FREQUENCY
```
You are being WATCHED!

ops$tkyte%ORA11GR2> select * from t where some_status = 1;

Execution Plan

Plan hash value: 470836197

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>539</td>
<td>53900</td>
<td>10</td>
<td>00:00</td>
</tr>
<tr>
<td>1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T</td>
<td>539</td>
<td>53900</td>
<td>10</td>
<td>00:00</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX RANGE SCAN</td>
<td>T_IDX</td>
<td>539</td>
<td></td>
<td>2</td>
<td>00:00</td>
</tr>
</tbody>
</table>
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select * from t where some_status = 99;

Execution Plan

Plan hash value: 1601196873

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>71683</td>
<td>7000K</td>
<td>300</td>
<td>00:00:04</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL</td>
<td>T</td>
<td>71683</td>
<td>7000K</td>
<td>300</td>
<td>00:00:04</td>
</tr>
</tbody>
</table>
```
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> select *  
  2   from  
  3 (  
  4   select *  
  5   from sys.col_usage$  
  6   where obj# = (select object_id  
  7     from dba_objects  
  8     where object_name = 'T'  
  9     and owner = 'OPS$TKYTE' )  
 10 )  
11   unpivot (value for x in  
12     ( EQUALITY_PREDS, EQUIJOIN_PREDS, NONEQUIJOIN_PREDS,  
13       RANGE_PREDS, LIKE_PREDS, NULL_PREDS ) )  
14 /  
```
You are being WATCHED!

<table>
<thead>
<tr>
<th>OBJ#</th>
<th>INTCOL#</th>
<th>TIMESTAMP</th>
<th>X</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>EQUALITY_PRED</td>
<td>1</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>EQUIJOIN_PRED</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>NONEQUIJOIN_PRED</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>RANGE_PRED</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>LIKE_PRED</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>NULL_PRED</td>
<td>0</td>
</tr>
</tbody>
</table>

6 rows selected.
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select * from t where some_status > 100;
no rows selected

ops$tkyte%ORA11GR2> begin
  2    dbms_stats.gather_table_stats( user, 'T' );
  3 end;
  4 /
PL/SQL procedure successfully completed.
```
You are being WATCHED!

<table>
<thead>
<tr>
<th>OBJ#</th>
<th>INTCOL#</th>
<th>TIMESTAMP</th>
<th>X</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>EQUALITY_PREDS</td>
<td>2</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>EQUIJOIN_PREDS</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>NONEQUIJOIN_PREDS</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>RANGE_PREDS</td>
<td>1</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>LIKE_PREDS</td>
<td>0</td>
</tr>
<tr>
<td>98040</td>
<td>16</td>
<td>30-SEP-11</td>
<td>NULL_PREDS</td>
<td>0</td>
</tr>
</tbody>
</table>

6 rows selected.
You are being WATCHED!

- You can ‘seed’ column stats pre-emptively
- Adds more “watching”
- Suggests possible extended statistics as well
You are being WATCHED!

```sql
ops$tkyte%ORA11GR2> begin
  2   dbms_stats.seed_col_usage( null, null, 10 );
  3   end;
  4   /

PL/SQL procedure successfully completed.
```
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select * 
    2    from t 
    3    where owner = 'SYS' 
    4     and object_type = 'DIMENSION';

no rows selected
```
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select dbms_stats.report_col_usage( user, 'T' )
    2   from dual;
```

<table>
<thead>
<tr>
<th>Legend</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>Used in single table EQality predicate</td>
</tr>
<tr>
<td>RANGE</td>
<td>Used in single table RANGE predicate</td>
</tr>
<tr>
<td>LIKE</td>
<td>Used in single table LIKE predicate</td>
</tr>
<tr>
<td>NULL</td>
<td>Used in single table is (not) NULL predicate</td>
</tr>
<tr>
<td>EQ_JOIN</td>
<td>Used in EQality JOIN predicate</td>
</tr>
<tr>
<td>NONEQ_JOIN</td>
<td>Used in NON EQality JOIN predicate</td>
</tr>
<tr>
<td>FILTER</td>
<td>Used in single table FILTER predicate</td>
</tr>
<tr>
<td>JOIN</td>
<td>Used in JOIN predicate</td>
</tr>
<tr>
<td>GROUP_BY</td>
<td>Used in GROUP BY expression</td>
</tr>
</tbody>
</table>
You are being WATCHED!

```
ops$tkyte%ORA11GR2> select dbms_stats.report_col_usage( user, 'T' )
    2   from dual;

...  

COLUMN USAGE REPORT FOR OPS$TKYTE.T  
.................................

1. OBJECT_TYPE            : EQ
2. OWNER                 : EQ
3. SOME_STATUS           : EQ RANGE
4. (OWNER, OBJECT_TYPE)  : FILTER

..................................................  

```
Scalar Subquery Caching
Scalar Subquery Caching

• A scalar subquery is a query that returns zero or one rows and a single column

• Can be used anywhere an expression can be used

• Is executed conceptually once for each row it is processed against

• For example:
Scalar Subquery Caching

Select dname, (select count(*)
    from emp
    where emp.deptno = dept.deptno)
from dept;

Is a lot like....
Scalar Subquery Caching

Begin
    for x in (select dname, deptno from dept)
    loop
        select count(*) into cnt
        from emp
        where deptno = X.DEPTNO;

        dbms_output.put_line
        ( x.dname || '' || x.cnt );
    end loop;
End;
Scalar Subquery Caching

- Conceptually it is like that…
- In reality there is caching going on
- Up to 255 entries can be saved
- Only for the duration of the query! Not across queries
Scalar Subquery Caching

ops$tkyte%ORA11GR2> create table t
  2  as
  3  select *
  4   from all_objects;

Table created.
Scalar Subquery Caching

```sql
ops$tkyte%ORA11GR2> begin
    2   dbms_stats.gather_table_stats( user,'T' );
    3  end;
    4   /

PL/SQL procedure successfully completed.
```
Scalar Subquery Caching

```sql
ops$tkyte%ORA11GR2> create or replace
  function f( x in varchar2 )
2    return number
3      as
4      begin
5        dbms_application_info.set_client_info
6        ( to_number(userenv('client_info'))+1 );
7      
8      return length(x);
9    end;
10   /
Function created.
```
Scalar Subquery Caching

```
ops$tkyte%ORA11GR2> variable startcpu number;
ops$tkyte%ORA11GR2> begin
  2  dbms_application_info.set_client_info(0);
  3  :startcpu := dbms_utility.get_cpu_time;
  4  end;
  5  /
```

PL/SQL procedure successfully completed.

This is run before every subsequent query...
Scalar Subquery Caching

```
ops$tkyte%ORA11GR2> select owner, f(owner) from t;

72233 rows selected.

ops$tkyte%ORA11GR2> select userenv('client_info') ci, 
2             dbms_utility.get_cpu_time:-:startcpu cpu 
3       from dual;

CI          CPU
----------    --------
72233       115
```
Scalar Subquery Caching

```
ops$tkyte%ORA11GR2> select owner, 
    (select f(owner) from dual) from t;
72233 rows selected.
```

```
ops$tkyte%ORA11GR2> select userenv('client_info') ci, 
    2       dbms_utility.get_cpu_time-:startcpu cpu 
    3 from dual;

<table>
<thead>
<tr>
<th>CI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>25</td>
</tr>
</tbody>
</table>
```
Scalar Subquery Caching

ops$tkyte%ORA11GR2> create or replace
  function f( x in varchar2 )
  
  2  return number
  3  DETERMINISTIC
  4  as

  ...
  10  end;
  11  /

  Function created.
Scalar Subquery Caching

```sql
ops$tkyte%ORA11GR2> select owner, f(owner) from t;

72233 rows selected.

ops$tkyte%ORA11GR2> select userenv('client_info') ci, dbms_utility.get_cpu_time-:startcpu cpu from dual;

<table>
<thead>
<tr>
<th>CI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>8278</td>
<td>74</td>
</tr>
</tbody>
</table>
```
Scalar Subquery Caching

ops$tkyte%ORA11GR2> create or replace
  function f( x in varchar2 )
    return number
  RESULT_CACHE
    as
  begin
    ...
  end;
  /
  Function created.
Scalar Subquery Caching

```sql
ops$tkyte%ORA11GR2> select owner, f(OWNER) from t;

72233 rows selected.

ops$tkyte%ORA11GR2> select userenv('client_info') ci,
2         dbms_utility.get_cpu_time-:startcpu cpu
3     from dual;

<table>
<thead>
<tr>
<th>CI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>60</td>
</tr>
</tbody>
</table>
Scalar Subquery Caching

ops$tkyte%ORA11GR2> select owner, f(owner) from t;

72233 rows selected.

ops$tkyte%ORA11GR2> select userenv('client_info') ci, dbms_utility.get_cpu_time-:startcpu cpu from dual;

<table>
<thead>
<tr>
<th>CI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>62</td>
</tr>
</tbody>
</table>
Scalar Subquery Caching

```sql
ops$tkyte%ORA11GR2> select owner,  
    (select f(owner) from dual) from t;
72233 rows selected.

ops$tkyte%ORA11GR2> select userenv('client_info') ci,  
    dbms_utility.get_cpu_time:-:startcpu cpu  
    from dual;

<table>
<thead>
<tr>
<th>CI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>22</td>
</tr>
</tbody>
</table>
Scalar Subquery Caching

How many times will $g($‘scott’$)$ be invoked?

```
Select * from T where owner = g('scott');
```

It depends of course...
Scalar Subquery Caching

Now How many times will \( g('scott') \) be invoked?

\[
\text{Select } * \text{ from } T \text{ where } \text{owner} = (\text{select } g('scott') \text{ from dual});
\]

It won’t depend this time…
Statement Level

non-Deterministic Functions
Statement level non-deterministic functions

- What is a deterministic function?
- What is a statement level deterministic function?
- Why do we care?
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> create table t
  2   as
  3   select *
  4    from all_users
  5    where rownum <= 5;

Table created.
```
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> create or replace function f
   2  return number
   3  as
   4       pragma autonomous_transaction;
   5       l_cnt number;
   6  begin
   7       select count(*) into l_cnt from t;
   8
   9       insert into t (username, user_id, created )
  10          values ( 'hello', 123, sysdate );
  11       commit;
  12
  13       return l_cnt;
  14  end;
  15 /
```
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> select count(*) over () cnt1,
   2         (select count(*) from t) cnt2,
   3         f() cnt3,
   4         (select f() from dual) cnt4
   5         from t;
```
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> select count(*) over () cnt1,
        2          (select count(*) from t) cnt2,
        3          f() cnt3,
        4          (select f() from dual) cnt4
        5          from t;
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> create or replace
    function f(p_scn in number)
 2  return number
 3  as
 4    pragma autonomous_transaction;
 5    l_cnt number;
 6  begin
 7    select count(*) into l_cnt from t
 8    as of scn p_scn;
 9
10    insert into t (username, user_id, created )
11    values ( 'hello', 123, sysdate ) ;
12    commit;
13
14    return l_cnt;
15  end;
16 /
```
Statement level non-deterministic functions

ops$tkyte%ORA11GR2> variable scn number
ops$tkyte%ORA11GR2> exec :scn :=
    dbms_flashback.get_system_change_number

PL/SQL procedure successfully completed.
Statement level non-deterministic functions

ops$tkyte%ORA11GR2> select count(*) over () cnt1, 
              2 (select count(*) from t) cnt2, 
              3 f(:scn) cnt3, 
              4 (select f(:scn) from dual) cnt4 
              5 from t;

<table>
<thead>
<tr>
<th>CNT1</th>
<th>CNT2</th>
<th>CNT3</th>
<th>CNT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> insert into t
    (username, user_id, created) values
    ('x', 1, sysdate);

1 row created.

ops$tkyte%ORA11GR2> exec :scn :=
    dbms_flashback.get_system_change_number

PL/SQL procedure successfully completed.
```
Statement level non-deterministic functions

```sql
ops$tkyte%ORA11GR2> select count(*) over () cnt1,
    2   (select count(*) from t) cnt2,
    3   f(:scn) cnt3,
    4   (select f(:scn) from dual) cnt4
    5   from t;
```

<table>
<thead>
<tr>
<th>CNT1</th>
<th>CNT2</th>
<th>CNT3</th>
<th>CNT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

6 rows selected.
Five things you probably didn’t know about SQL

• SQLNet Compression
• NULLS and Indexes and Cardinality
• You are being watched!
• Scalar Subquery Caching
• Calling statement level non-deterministic functions
Q&A
Hardware and Software

Engineered to Work Together