

# Indexes

- ❖ Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

*-[http://en.wikipedia.org/wiki/Database\\_index](http://en.wikipedia.org/wiki/Database_index)*

```
select index_owner, index_name, table_owner, table_name,  
       column_name  
from all_ind_columns  
where table_owner = 'PROD_SATURN'  
      and table_name = 'SSBSECT'  
;
```

# Select Statements

- ❖ List out desired columns in your select statement instead of selecting all columns.

- ❖ Instead of:

```
select *  
from prod_saturn.ssbsect  
;
```

- ❖ Use:

```
select ssbsect_term_code, ssbsect_crn, ssbsect_subj_code,  
ssbsect_crse_num  
from prod_saturn.ssbsect  
;
```

# Limit Returned Rows

- ❖ If you are simply exploring values, limit the number of rows returned. Your DBA will thank you.

- ❖ Use:

```
select *  
from prod_saturn.ssbsect  
where rownum < 100  
;
```

# Having clause

- ❖ The `having` clause filters after all rows have been selected. Only use `having` when filtering transformation such as `group by`.

- ❖ Use:

```
select ssbsect_term_code, ssbsect_subj_code,  
count(ssbsect_crn)  
from prod_saturn.ssbsect  
group by ssbsect_term_code, ssbsect_subj_code  
having count(ssbsect_crn) > 100
```

# Minimize subquery blocks

- ❖ Subqueries process the outer query before processing the inner query. Use sparingly.

- ❖ Instead of:

```
SELECT name
FROM employee
WHERE salary = (SELECT MAX(salary) FROM employee_details)
AND age = (SELECT MAX(age) FROM employee_details)
AND dept = 'Psychology';
```

- ❖ Use:

```
SELECT name
FROM employee
WHERE (salary, age ) = (SELECT MAX (salary), MAX (age)
FROM employee_details)
AND dept = 'Psychology';
```

# EXISTS and IN

- ❖ Usually IN has the slowest performance.
- ❖ IN is efficient when most of the filter criteria is in the sub-query.
- ❖ EXISTS is efficient when most of the filter criteria is in the main query.

- ❖ Instead of:

```
select szrcbm1_pidm, szrcbm1_first_name  
from prod_txcnmgr.szrcbm1  
where szrcbm1_pidm in(select szrcbm8_pidm from  
prod_txcnmgr.szrcbm8);
```

- ❖ Use:

```
select szrcbm1_pidm, szrcbm1_first_name  
from prod_txcnmgr.szrcbm1  
where exists (select *  
              from prod_txcnmgr.szrcbm8  
              where szrcbm1_pidm = szrcbm8_pidm);
```

# UNION and UNION ALL

❖ Instead of:

```
select program, courses
from txir.cat_programs
join dw_prog conc
  on bnr_conc = conc.dap_block_value
  and conc.dap_block_type = 'CONC'
union
select *
from txir.cat_programs
join dw_prog cert
  on bnr_cert = cert.dap_block_value
  and cert.dap_block_type = 'SPEC'
```

❖ Use:

```
select program, courses
from txir.cat_programs
join dw_prog conc
  on bnr_conc = conc.dap_block_value
  and conc.dap_block_type = 'CONC'
union all
select *
from txir.cat_programs
join dw_prog cert
  on bnr_cert = cert.dap_block_value
  and cert.dap_block_type = 'SPEC'
```

# Avoid NOT

- ❖ When possible, use logic operators as opposed to NOT

- ❖ Instead of:

```
select first_name, last_name  
from person  
where age != 18
```

- ❖ Use:

```
select first_name, last_name  
from person  
where age < 18
```



# Substrings instead of LIKE

- ❖ LIKE requires the processor to look through the entire string. Use substrings instead.

- ❖ Instead of:

```
select first_name, last_name  
from person  
where last_name like 'Smi%'
```

- ❖ Use:

```
select first_name, last_name  
from person  
where substr(last_name,1,3) ='Smi'
```

# Use views efficiently

- ❖ Check your select statement to see if variables truly need to be pulled from a view.
- ❖ If all variables in the select exist in a single table, use the table instead of the view.

# Check for unintentional Cartesian products

- ❖ Cartesian products or cross-joins return all rows in all tables listed in the query. They are usually a result of no relationship being defined between tables.

```
select ID, classification
from person, student
```

<u>ID</u>	<u>classification</u>
A1	FR
A1	SO
A1	JR
A1	SR
A2	FR
A2	SO
A2	JR
A2	SR

# Avoid transformed columns in the WHERE clause

- ❖ Instead of:

```
where  
to_number(substring(ssbsect_term_code,instr(ssbsect_term_c  
ode,2)) =  
to_number(substring(szrcbm4_term_code,instr(ssbsect_term_c  
ode,2))
```

- ❖ Use:

```
where ssbsect_term_code = szrcbm4_term_code
```

# Use binary logic instead of BETWEEN

- ❖ Instead of:  
where age between(18 and 24)
- ❖ Use:  
where age >=18 AND <=24

# SQL OPTIMIZATION CHEAT SHEET

## Define Select Statements

Instead of:  
SELECT \*  
FROM schema.table;

Use:  
SELECT col1, col2, col3  
FROM schema.table;

## Avoid IN

Instead of:  
SELECT name  
FROM student  
WHERE classification in('FR', 'SO');

Use:  
SELECT name  
FROM student  
WHERE (classification = 'FR'  
OR classification = 'SO');

## Avoid NOT

When filtering on a column with few options,  
list out the desired variables instead using NOT.

Instead of:  
SELECT name  
FROM student  
WHERE level != 'UG';

Use:  
SELECT name  
FROM student  
WHERE level = 'GR' OR level = 'DR'

## Avoid BETWEEN

Instead of:  
SELECT name  
FROM person  
WHERE age BETWEEN(18 AND 24);

Use:  
SELECT name  
FROM person  
WHERE age >=18 AND age <=24;

## SUBSTR instead of LIKE

Instead of:  
SELECT first\_name, last\_name  
FROM person  
WHERE last\_name like 'Smi%';

Use:  
SELECT first\_name, last\_name  
FROM person  
WHERE substr(last\_name,1,3) = 'Smi';

## Limit Returned Rows

When exploring data, limit the rows returned.

SELECT \*  
FROM schema.table  
WHERE rownum <= 100;

## Use UNION ALL

UNION will analyze data for duplicates.

UNION ALL simply appends the additional rows.

SELECT name  
FROM student  
UNION ALL  
SELECT name  
FROM employee;

## Use JOIN to avoid Cartesian Products

Instead of:  
SELECT first\_name, last\_name, major  
FROM person, student  
WHERE person.id = student.id;

Use:  
SELECT first\_name, last\_name, major  
FROM person  
JOIN student  
ON person.id = student.id;

## Minimize Subquery Blocks

Use sparingly and pull all variables in a single subquery, if possible.

Instead of:  
SELECT name  
FROM person  
WHERE sch = (SELECT SUM(sch) FROM student)  
AND gpa = (SELECT MAX(gpa) FROM student);

Use:  
SELECT name  
FROM person  
WHERE (sch,gpa) = (SELECT SUM(sch),MAX(gpa)  
FROM student);

## Use Indexes

Filtering on indexes first leads to quicker results. Find indexes with:

SELECT index\_owner, index\_name, table\_owner,  
table\_name, column\_name  
FROM all\_ind\_columns  
WHERE table\_owner = <insert schema name>  
AND table\_name = <insert table name>;

## Use VIEWS efficiently

If all variables in the select exist in a single table,  
use the table instead of the view.

To see the SQL behind a view:

SELECT text  
FROM all\_views  
WHERE view\_name = '<view\_name>';

Created by: Kate Amorella Proff  
Research Analyst  
Texas State University  
[kproff@txstate.edu](mailto:kproff@txstate.edu)  
512-245-2386

March 12, 2015