CONFIO SOFTWARE

# Looney Tuner? No, there IS a method to my madness!

Janis Griffin

Senior DBA

# Who Am I?

- Senior DBA for Confio Software
  - JanisGriffin@confio.com
  - Twitter - @DoBoutAnything
  - Current – 25 Years in Oracle, SQL Server
  - DBA and Developer
- Specialize in Performance Tuning
- Review Database Performance for Customers and Prospects
- Common Thread – Paralyzed by Tuning

CONFIO SOFTWARE

# Agenda

- Challenges Of Tuning
  - Who should tune
  - Which SQLs to tune
- Utilize Response Time Analysis (RTA)
- Gather Details about SQL
  - Example Query Execution Plans
- Use SQL or Query Diagramming
  - Who registered yesterday for Tuning Class
  - Lookup paycheck information
  - Check order status
- Monitor – Make sure it stays tuned

CONFIO SOFTWARE

# Challenges Of Tuning

- SQL Tuning is Hard
- Requires Expertise in Many Areas
  - Technical – Plan, Data Access, SQL Design
  - Business – What is the Purpose of SQL?
- Tuning Takes Time
  - Large Number of SQL Statements
  - Each Statement is Different
- Low Priority in Some Companies
  - Vendor Applications
  - Focus on Hardware or System Issues
- Never Ending

CONFIO SOFTWARE

# Who Should Tune

- Developers?
  - Developing applications is very difficult
  - Typically focused on functionality
  - Not much time left to tune SQL
  - Don't get enough practice or simply don't know
  - SQL runs differently in Production versus Dev/Test
- DBA?
  - Do not know the code like developers do
  - Focus on "Keep the Lights On"
  - Very complex environment
- Need a team approach  (DevOps)

CONFIO SOFTWARE

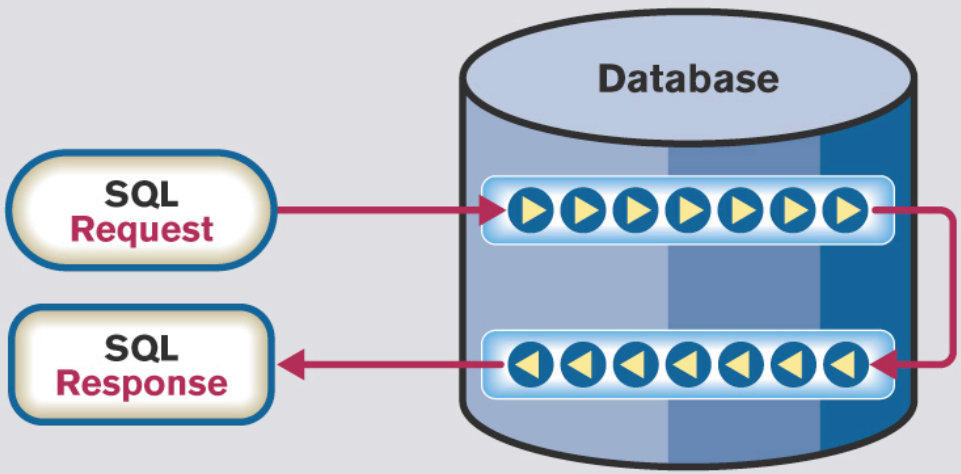
# Which SQL to Tune

- User / Batch Job Complaints
  - Known Poorly Performing SQL
  - Trace Session/Process
- Queries Performing Most I/O (LIO, PIO)
  - Table or Index Scans
- Queries Consuming CPU
- Highest Response Times (Ignite8)

CONFIO SOFTWARE

# Response Time Analysis (RTA)

## Focus on Response Time



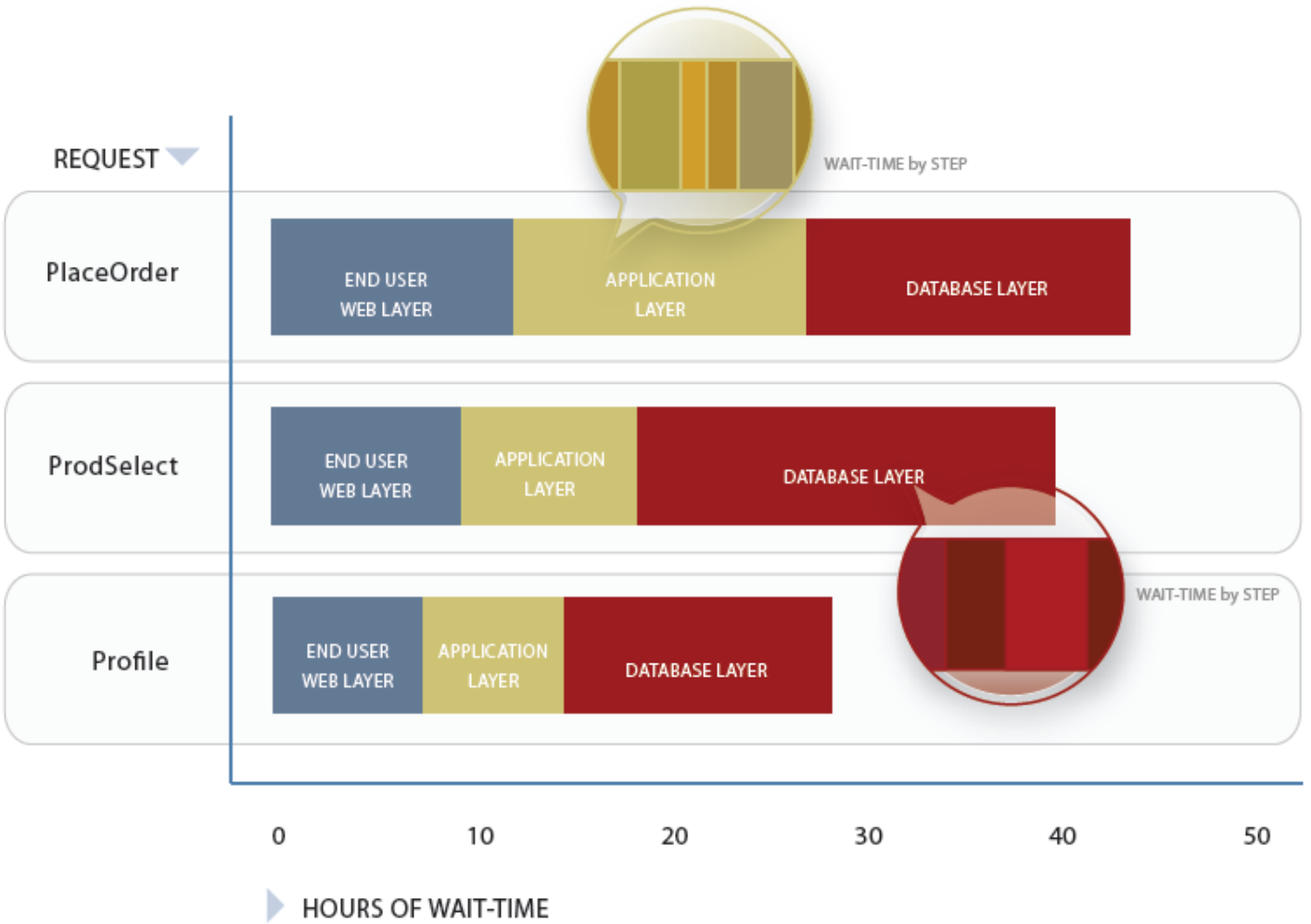

**Identify Wait-Time at every step and rank bottlenecks by user impact.**

- Understand the total time a Query spends in Database
- Measure time while Query executes
- Oracle helps by providing Wait Events

CONFIO SOFTWARE

# Identify End-to-End Time

## Accurate End-to-End Response Time Analysis



# Wait Event Information

**V$SESSION**

SID
USERNAME
SQL_ID
PROGRAM
MODULE
ACTION
PLAN_HASH_VALUE

**V$SESSION_WAIT**

SID
EVENT
P1, P1RAW, P2, P2RAW, P3, P3RAW
STATE (WAITING, WAITED…)

- Oracle 10g added this info to V$SESSION

**V$SQL_PLAN**

SQL_ID

**V$SQL**

SQL_ID
SQL_FULLTEXT

**V$SQLAREA**

SQL_ID
EXECUTIONS
PARSE_CALLS

**V$SQL_BIND_CAPTURE**

SQL_ID
NAME

**DBA_OBJECTS**

OBJECT_ID
OBJECT_NAME
OBJECT_TYPE

CONFIO SOFTWARE

# Base Query

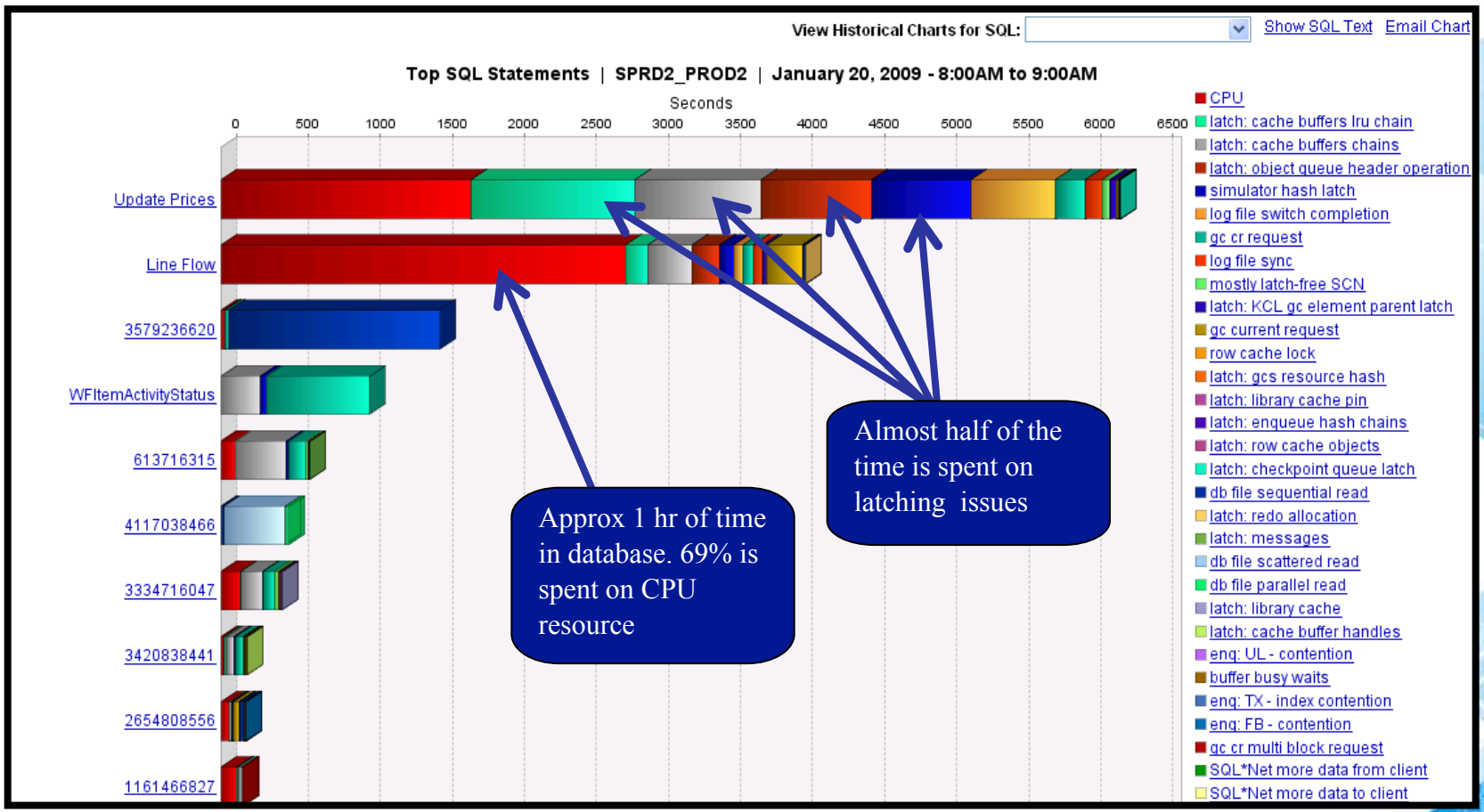
```
SELECT
   sid, username, program, module, action,
   machine, osuser, …,
   sql_id, plan_hash_value,
   decode(state, 'WAITING', event, 'CPU') event,
   p1, p1raw, p2, …,
   SYSDATE
FROM V$SESSION s
WHERE s.status = 'ACTIVE'
AND event NOT IN (<idle wait events>);
-- (AND wait_class != 'Idle')
```

CONFIO SOFTWARE

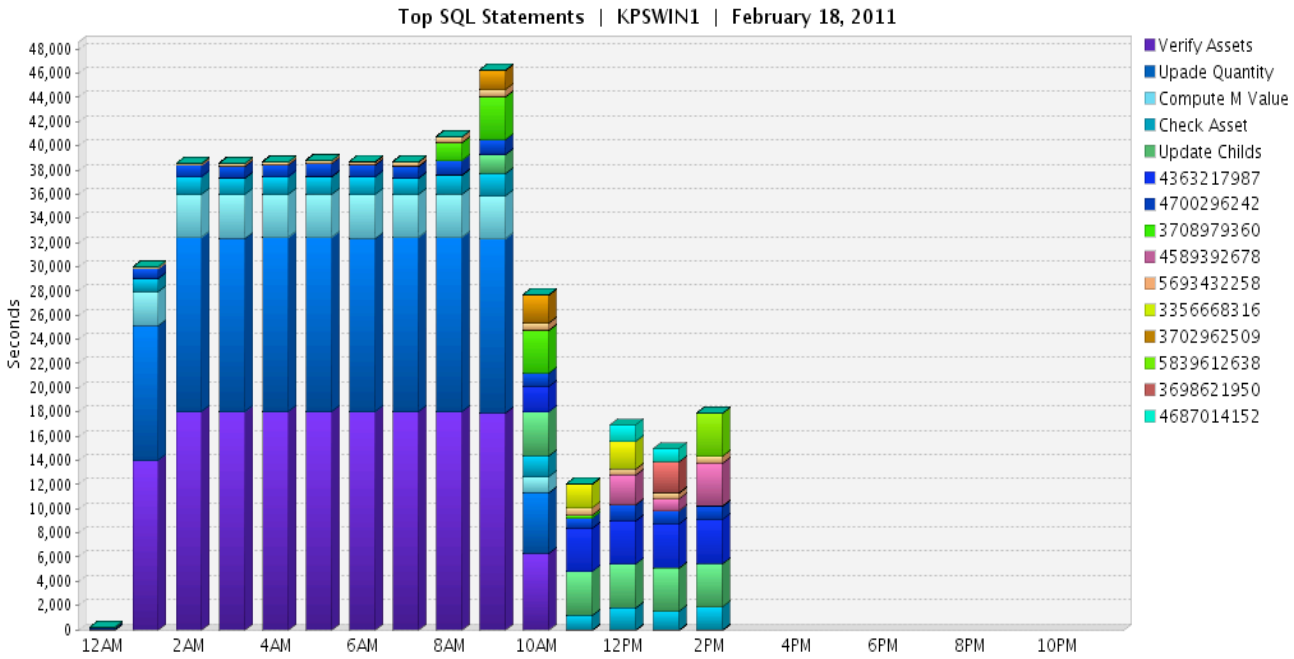
# Active Session History (ASH)

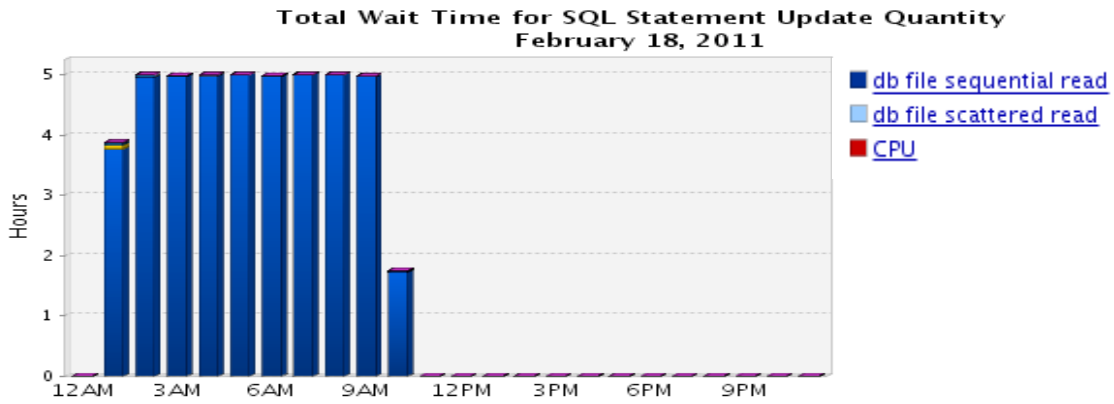
- V$ACTIVE_SESSION_HISTORY
  - Data warehouse for session statistics
  - Oracle 10g and higher
  - Data is sampled every second
  - Holds at least one hour of history
  - Never bigger than:
    - 2% of SGA_TARGET
    - 5% of SHARED_POOL (if automatic sga sizing is turned off)
- WRH$_ACTIVE_SESSION_HISTORY
  - Above table gets flushed to this table

# RTA - Wait Time & Events



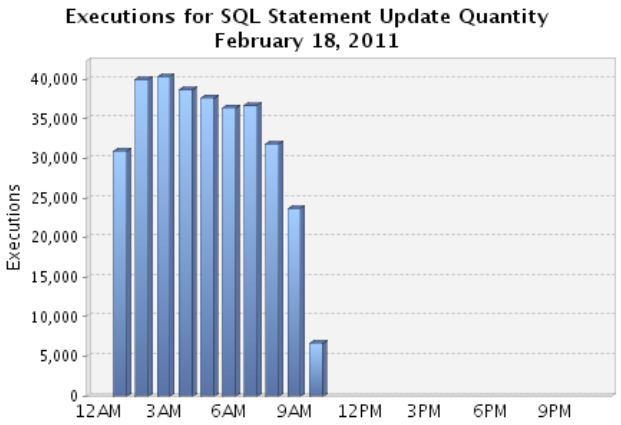
# RTA Tuning Data

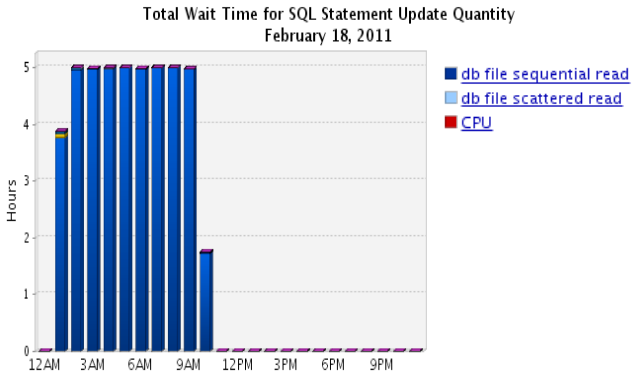


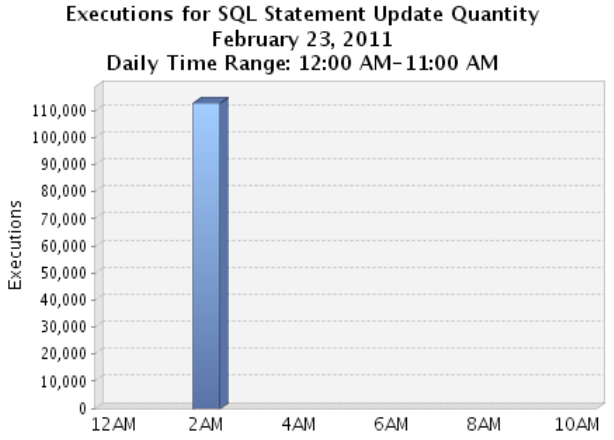
# RTA Benefits

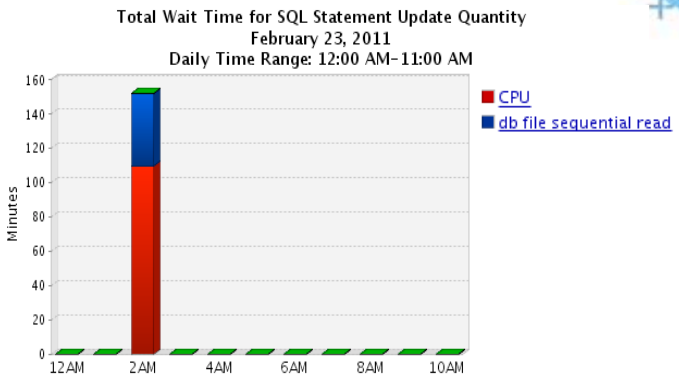
## Before




## After



CONFIO SOFTWARE

# Gather Metrics

- Get baseline metrics
  - How long does it take now
  - What is acceptable (10 sec, 2 min, 1 hour)
- Collect Wait Event Information
  - Locking / Blocking (enq)
  - I/O problem (db file sequential read)
  - Latch contention (latch)
  - Network slowdown (SQL*Net)
  - May be multiple issues
  - All have different resolutions

CONFIO SOFTWARE

# Get Execution Plan

- EXPLAIN PLAN
  - Estimated plan - can be wrong for many reasons
    - Best Guess, Blind to Bind Variables or Data types
    - Explain Plan For … sql statement
    - Set autotrace (on | trace | exp | stat | off)
- V$SQL_PLAN (Oracle 9i+)
  - Actual execution plan
  - Use DBMS_XPLAN for display
- Tracing (all versions) / TKPROF
  - Get all sorts of good information
  - Works when you know a problem will occur
- Historical Plans – AWR, Confio Ignite

CONFIO SOFTWARE

# DBMS_XPLAN – 11g

## New functions in 11g

| | |
|---|---|
| BUILD_PLAN_XML | Return the last plan, or a named plan, explained as XML |
| **DISPLAY** | Shows the last plan explained – EXPLAIN PLAN ** Only FUNCTION in Oracle 9i |
| **DISPLAY_AWR** | Format & display the plan of a stored SQL statement in AWR |
| **DISPLAY_CURSOR** | Format & display the execution plan of any loaded cursor |
| **DISPLAY_PLAN** | Return the last plan, or a named plan, explained as a CLOB |
| **DISPLAY_SQLSET** | Format & display the execution plan of statements stored in a SQL tuning set |
| **DISPLAY_SQL_PLAN_BASELINE** | Displays one or more plans for the specified SQL statement |
| FORMAT_NUMBER | Returns a number as a string |
| FORMAT_NUMBER2 | Returns a number as a string formatted with a leading space (CHR(32)) |
| FORMAT_SIZE | Undocumented |
| FORMAT_SIZE2 | Undocumented |
| FORMAT_TIME_S | Undocumented |
| PREPARE_PLAN_XML_QUERY | - function to build the XML version of a select query that is run before the display function to retrieve and display the execution plan of a SQL |
| PREPARE_RECORDS | Used Internally |
| VALIDATE_FORMAT | Used Internally |

CONFIO SOFTWARE

# View Using Display Plan

```
EXPLAIN PLAN
SET STATEMENT_ID = 'inventory' FOR
        SELECT PRODUCTS.PRODUCT_ID, PRODUCT_NAME, PRODUCT_DESCRIPTION,CATEGORY_ID,
        WEIGHT_CLASS,
             WARRANTY_PERIOD,  SUPPLIER_ID, PRODUCT_STATUS,
             LIST_PRICE,MIN_PRICE, CATALOG_URL, QUANTITY_ON_HAND
        FROM PRODUCTS,
               INVENTORIES
        WHERE INVENTORIES.PRODUCT_ID = PRODUCTS.PRODUCT_ID
        AND PRODUCTS.CATEGORY_ID = :B3
        AND INVENTORIES.WAREHOUSE_ID = :B2
        AND ROWNUM < :B1;

set pages 0 head off
set linesize 132
set long 1000000
col xplan format a100

spool inventory.html

SELECT dbms_xplan.display_plan(statement_id => 'inventory',type=>'HTML') AS XPLAN
FROM dual;
```

# View Using Display Plan


SQL Explain Plan Report

file:///c:/users/owner/inventory.html

**Plan Hash Value** : 1842583762

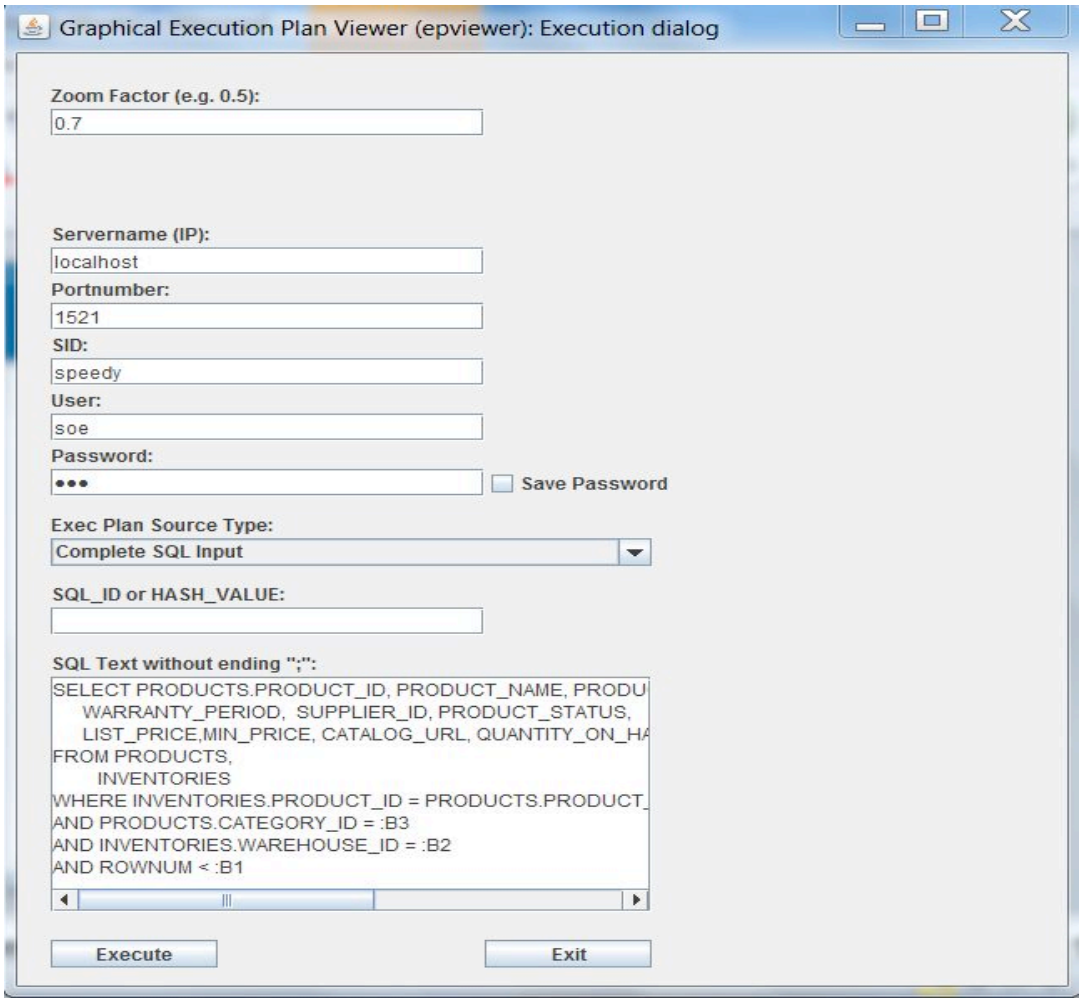
| Id | Operation | Name | Rows | Bytes | Cost | Time |
|---|---|---|---|---|---|---|
| 0 | SELECT STATEMENT | | 6001 | 1260210 | 1090 | 00:00:14 |
| * 1 | COUNT STOPKEY | | | | | |
| * 2 | HASH JOIN RIGHT OUTER | | 6001 | 1260210 | 1090 | 00:00:14 |
| * 3 | TABLE ACCESS FULL | PRODUCT_DESCRIPTIONS | 10 | 180 | 18 | 00:00:01 |
| * 4 | HASH JOIN | | 6001 | 1152192 | 1071 | 00:00:13 |
| 5 | TABLE ACCESS BY INDEX ROWID | INVENTORIES | 896 | 12544 | 876 | 00:00:11 |
| * 6 | INDEX RANGE SCAN | INVENTORIES_IX1 | 896 | | 4 | 00:00:01 |
| 7 | TABLE ACCESS BY INDEX ROWID | PRODUCT_INFORMATION | 6692 | 1191176 | 194 | 00:00:03 |
| * 8 | INDEX RANGE SCAN | PROD_CATEGORY_IX | 6692 | | 17 | 00:00:01 |

Predicate Information (identified by operation id):

- 1 - filter(ROWNUM<TO_NUMBER(:B1))
- 2 - access("D"."PRODUCT_ID"(+)="I"."PRODUCT_ID")
- 3 - filter("D"."LANGUAGE_ID"(+)=SYS_CONTEXT('USERENV','LANG'))
- 4 - access("INVENTORIES"."PRODUCT_ID"="I"."PRODUCT_ID")
- 6 - access("INVENTORIES"."WAREHOUSE_ID"=TO_NUMBER(:B2))
- 8 - access("I"."CATEGORY_ID"=TO_NUMBER(:B3))

# "Free" Graphical Plans

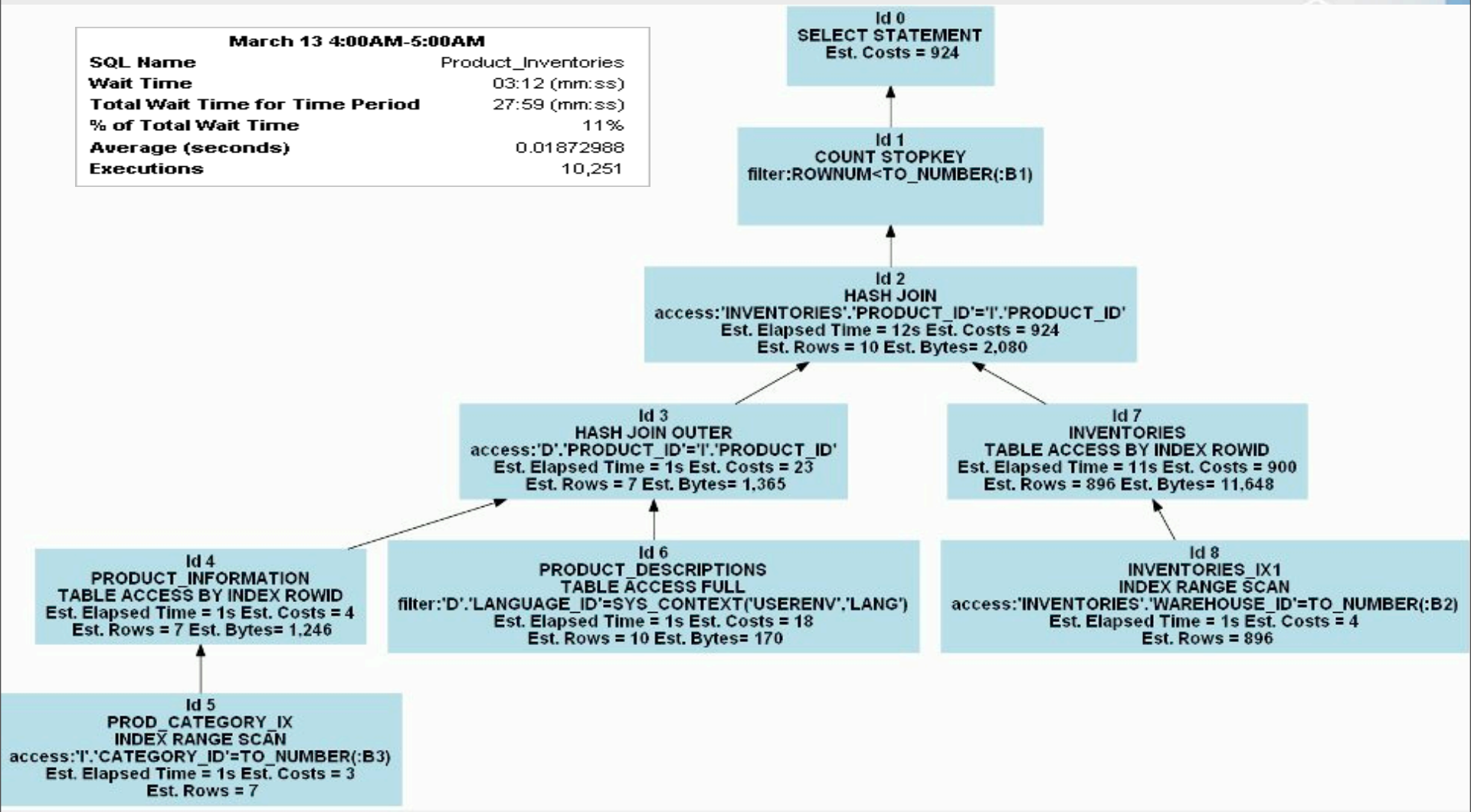
http://www.epviewer.bplaced.net/downloads



CONFIO SOFTWARE

# "Free" Graphical Plans

| March 13 4:00AM-5:00AM | |
|---|---|
| SQL Name | Product_Inventories |
| Wait Time | 03:12 (mm:ss) |
| Total Wait Time for Time Period | 27:59 (mm:ss) |
| % of Total Wait Time | 11% |
| Average (seconds) | 0.01872988 |
| Executions | 10,251 |



# Execution Plan Details

SELECT e.empno EID, e.ename "Employee_name,  d.dname "Department", e.hiredate "Date_Hired"
FROM  emp e,  dept d  WHERE  d.deptno =  :P1 AND e.deptno = d.deptno;

Actual Plan:  V$SQL_PLAN using dbms_xplan.display_cursor

```
SQL>
SQL> select * from table(dbms_xplan.display_cursor('bbh4gphampy33',0));

SQL_ID  bbh4gphampy33, child number 0
-------------------------------------
SELECT e.empno EID, e.ename "Employee_name",  d.dname "Department",
e.hiredate "Date_Hired" FROM  emp e,  dept d WHERE  d.deptno =  :P1 AND
e.deptno = d.deptno

Plan hash value: 568005898

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |       |       |    15 (100)|          |
|   1 |  NESTED LOOPS                |         |  3958 |   139K|    15   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |     1 |    11 |     2   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN         | PK_DEPT |     1 |       |     1   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL          | EMP     |  3958 | 98950 |    13   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   3 - access("D"."DEPTNO"=TO_NUMBER(:P1))
   4 - filter("E"."DEPTNO"=TO_NUMBER(:P1))
```

CONFIO SOFTWARE

# Need Table & Index Info

- Understand objects in execution plans
  - Table & Segment sizes
  - Number of Rows
  - Indexes & their column order
  - Column data types
  - Cardinality of columns / Data Skew
  - Statistic Gathering
  - Histograms?
- Use TuningStats.sql
  - http://support.confio.com/kb/1534
- Run it for expensive data access targets

CONFIO SOFTWARE

# Case Studies

- SQL Diagramming
  - Who registered yesterday for Tuning Class
  - Lookup paycheck information
  - Inventory lookup for new orders by customer

CONFIO SOFTWARE

# SQL Statement 1

- Who registered yesterday for SQL Tuning

```
SELECT s.fname, s.lname, r.signup_date
FROM   student s
   INNER JOIN registration r ON s.student_id = r.student_id
   INNER JOIN class c ON r.class_id = c.class_id
WHERE  c.name  = 'SQL TUNING'
AND   r.signup_date BETWEEN :beg_date AND :end_date
AND   r.cancelled = 'N'
```

- Execution Stats – 21,829 Buffer Gets
- Execution Time – 22 seconds to execute
- Wait Events – Waits 90% direct path read

# Execution Plan

| January 27 2:00PM-2:30PM | |
|---|---|
| SQL ID | 008x4scyck1tn |
| Wait Time | 29:43 (mm:ss) |
| Total Wait Time for Time Period | 49:15 (mm:ss) |
| % of Total Wait Time | 60% |
| Average (seconds) | 22.2875 |
| Executions | 80 |

**SQL Text**

SELECT s.fname, s.lname, r.signup_date FROM student s INNER JOIN registration r ON s.student_id = r.student_id INNER JOIN class c ON r.class_id = c.class_id WHERE c.name = 'SQL TUNING' AND r.signup_date BETWEEN :beg_date and :end_date AND r.cancelled = 'N'

```
SQL_ID  008x4scyck1tn, child number 0
-------------------------------------
SELECT s.fname, s.lname, r.signup_date FROM   student s     INNER
registration r ON s.student_id = r.student_id     INNER JOIN class
r.class_id = c.class_id WHERE  c.name  = 'SQL TUNING' AND
r.signup_date BETWEEN :beg_date and :end_date AND   r.cancelled =

Plan hash value: 1244828764
-----------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT             |              |       |       |  5584 (100)|          |
|*  1 |  FILTER                      |              |       |       |            |          |
|   2 |   NESTED LOOPS               |              |       |       |            |          |
|   3 |    NESTED LOOPS              |              |    70 |  8190 |  5584   (1)| 00:01:08 |
|*  4 |     HASH JOIN                |              |    70 |  5810 |  5514   (1)| 00:01:07 |
|*  5 |      TABLE ACCESS FULL       | CLASS        |     1 |    65 |    34   (0)| 00:00:01 |
|*  6 |      TABLE ACCESS FULL       | REGISTRATION | 88570 |  1556K|  5479   (1)| 00:01:06 |
|*  7 |     INDEX UNIQUE SCAN        | PK_STUDENT   |     1 |       |     0   (0)|          |
|   8 |    TABLE ACCESS BY INDEX ROWID| STUDENT      |     1 |    34 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------

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

1 - filter(TO_DATE(:BEG_DATE)<=TO_DATE(:END_DATE))
4 - access("R"."CLASS_ID"="C"."CLASS_ID")
5 - filter("C"."NAME"='SQL TUNING')
6 - filter(("R"."SIGNUP_DATE">=:BEG_DATE AND "R"."SIGNUP_DATE"<=:END_DATE AND
        "R"."CANCELLED"='N'))
7 - access("R"."STUDENT_ID"="S"."STUDENT_ID")
```

# Tuning Advisor

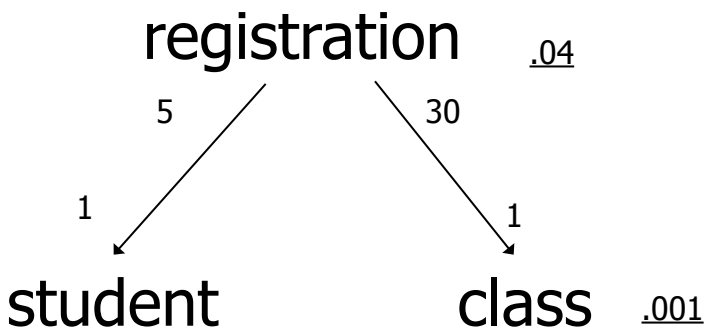
- Recommends – 3 new indexes

```
DECLARE
  l_sql_tune_task_id  VARCHAR2(100);
BEGIN
  l_sql_tune_task_id := DBMS_SQLTUNE.create_tuning_task ( sql_id => '&sql_id',
   scope => DBMS_SQLTUNE.scope_comprehensive, time_limit  => 60,
   task_name => '&sql_id', description => 'Tuning task for class registration query');
  DBMS_OUTPUT.put_line('l_sql_tune_task_id: ' || l_sql_tune_task_id);
END;
/

EXEC DBMS_SQLTUNE.execute_tuning_task(task_name => '&sql_id');
```

```
SELECT DBMS_SQLTUNE.report_tuning_task('OO8x4scyck1tn') AS recommendations FROM dual

RECOMMENDATIONS
1- Index Finding (see explain plans section below)
--------------------------------------------------
The execution plan of this statement can be improved by creating one or more
indices.
Recommendation (estimated benefit: 84.79%)
------------------------------------------
create index CSU.IDX$$_102CB0001 on CSU.CLASS("NAME");

create index CSU.IDX$$_102CB0002 on CSU.REGISTRATION("CLASS_ID");

create index CSU.IDX$$_102CB0003 on CSU.REGISTRATION("CANCELLED","SIGNUP_DATE");
```

# SQL Diagramming

- Great Book "SQL Tuning" by Dan Tow
  - Great book that teaches SQL Diagramming
  - http://www.singingsql.com

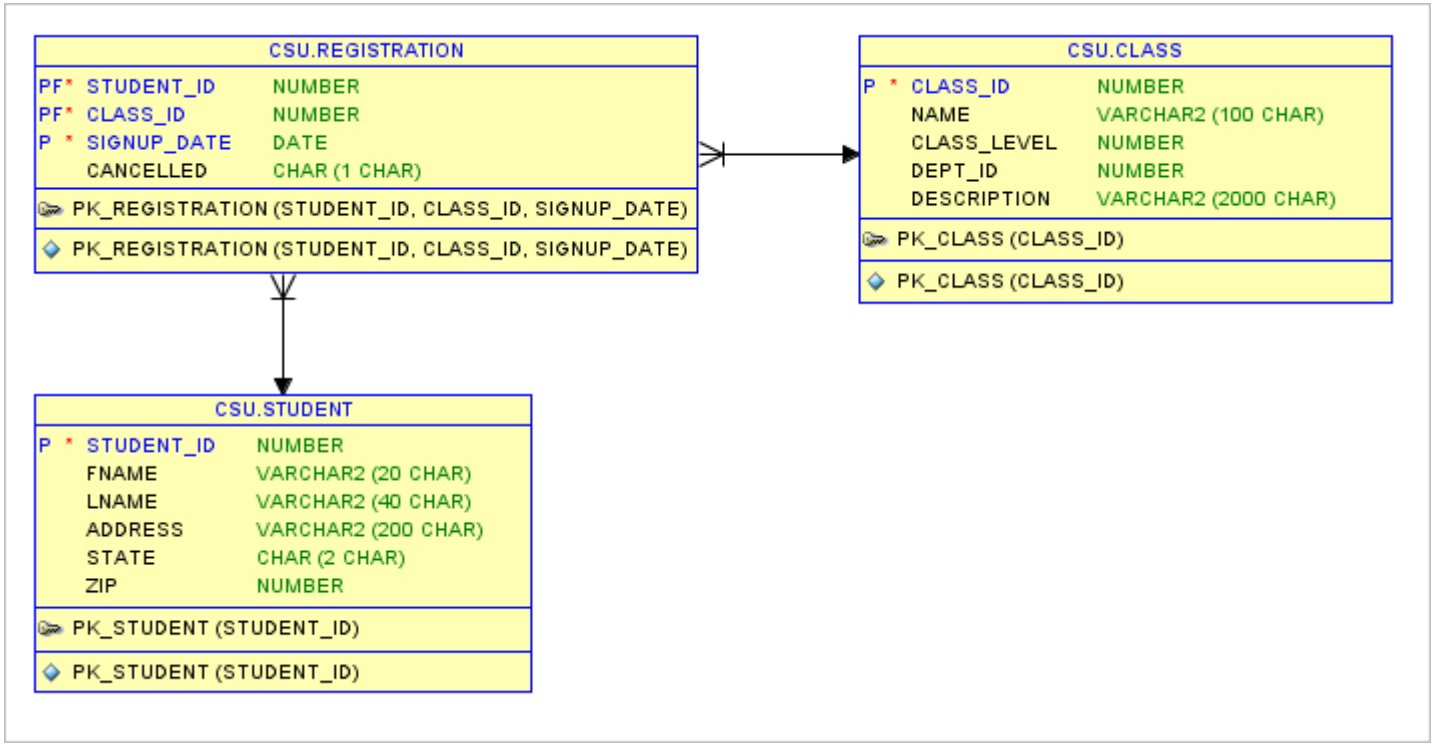



```
select count(1) from registration where cancelled = 'N'
and signup_date between '2012-03-23 00:00' and '2012-03-24 00:00'

    64112 / 1783066 = .035956044


select count(1) from class where name = 'SQL TUNING'

    2 / 1,267 = .001
```

# Relationship Diagram

- FREE - Oracle SQL Developer Data Modeler

http://www.oracle.com/technetwork/developer-tools/datamodeler/sqldevdm31ea-download-515132.html



# New Execution Plan

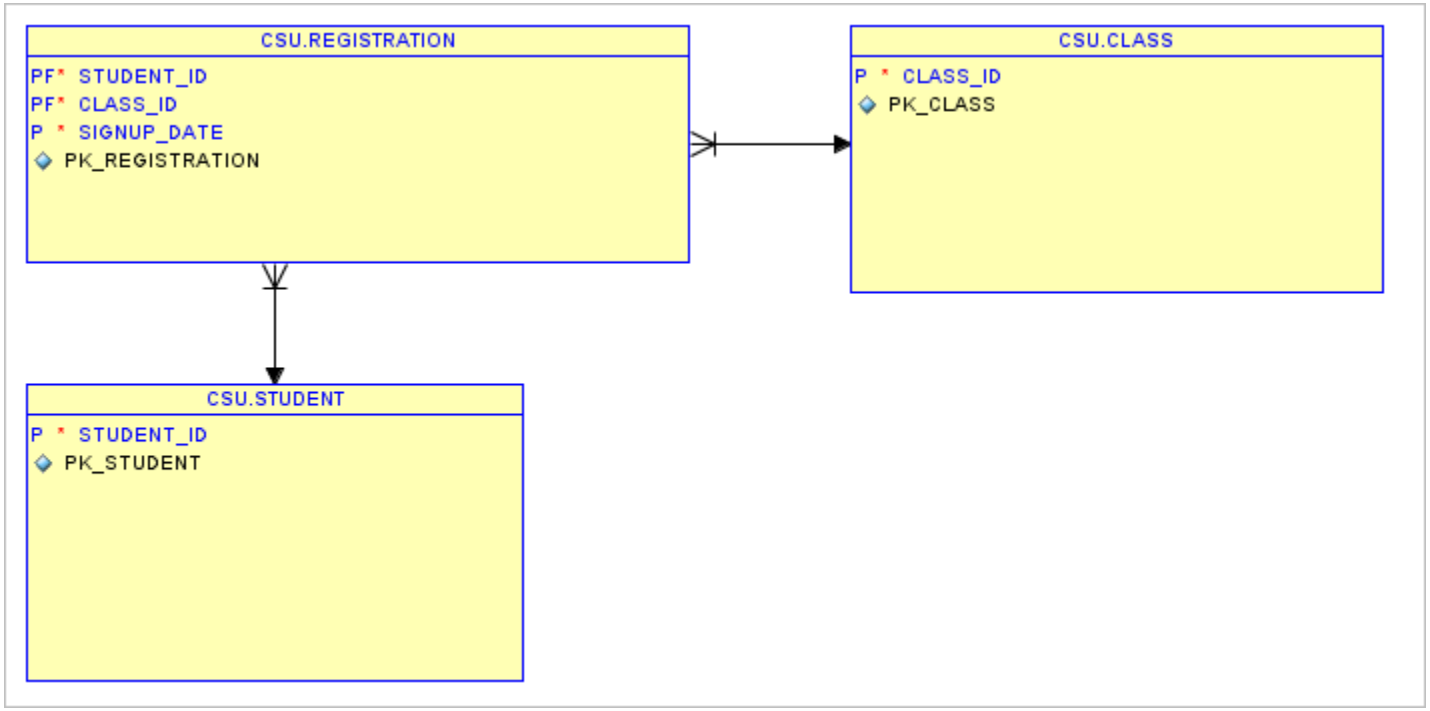
CREATE INDEX cl_name ON class(name);

```
select * from table (dbms_xplan.display_cursor('008x4scyck1tn','0'))
----------------------------------------------------------------------------------------------------
SQL_ID  008x4scyck1tn, child number 0
-------------------------------------
SELECT s.fname, s.lname, r.signup_date FROM   student s     INNER JOIN
registration r ON s.student_id = r.student_id     INNER JOIN class c ON
r.class_id = c.class_id WHERE  c.name  = 'SQL TUNING' AND
r.signup_date BETWEEN :beg_date and :end_date AND   r.cancelled = 'N'

Plan hash value: 2038084866

-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |       |       |  5569 (100)|          |
|*  1 |  FILTER                             |              |       |       |            |          |
|   2 |   NESTED LOOPS                      |              |       |       |            |          |
|   3 |    NESTED LOOPS                     |              |    77 |  9009 |  5569   (1)| 00:01:07 |
|*  4 |     HASH JOIN                       |              |    77 |  6391 |  5492   (1)| 00:01:06 |
|   5 |      TABLE ACCESS BY INDEX ROWID    | CLASS        |     1 |    65 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN              | CL_NAME      |     1 |       |     1   (0)| 00:00:01 |
|*  7 |      TABLE ACCESS FULL              | REGISTRATION | 97637 |  1716K|  5489   (1)| 00:01:06 |
|*  8 |     INDEX UNIQUE SCAN               | PK_STUDENT   |     1 |       |     0   (0)|          |
|   9 |    TABLE ACCESS BY INDEX ROWID      | STUDENT      |     1 |    34 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

1 - filter(TO_DATE(:BEG_DATE)<=TO_DATE(:END_DATE))
4 - access("R"."CLASS_ID"="C"."CLASS_ID")
6 - access("C"."NAME"='SQL TUNING')
7 - filter(("R"."SIGNUP_DATE"<=:END_DATE AND "R"."SIGNUP_DATE">=:BEG_DATE AND "R"."CANCELLED"='N'))
8 - access("R"."STUDENT_ID"="S"."STUDENT_ID")
```

- Execution Stats – 20,348 buffer gets
- Why is a full table scan still occurring on REGISTRATION?

# Review Index Order

- CLASS_ID not left leading in index




- Execution Stats – 20,348 buffer gets
- Twice the work to use Primary Key Index on REGISTRATION

# New Execution Plan

CREATE INDEX reg_alt ON registration(class_id);

```
select * from table (dbms_xplan.display_cursor('008x4scyck1tn','0'))

SQL_ID  008x4scyck1tn, child number 0
-------------------------------------
SELECT s.fname, s.lname, r.signup_date FROM   student s     INNER JOIN
registration r ON s.student_id = r.student_id     INNER JOIN class c ON
r.class_id = c.class_id WHERE  c.name  = 'SQL TUNING' AND
r.signup_date BETWEEN :beg_date and :end_date AND   r.cancelled = 'N'

Plan hash value: 3574817656

-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |       |       |  1470 (100)|          |
|*  1 |  FILTER                             |              |       |       |            |          |
|   2 |   NESTED LOOPS                      |              |       |       |            |          |
|   3 |    NESTED LOOPS                     |              |    66 |  7722 |  1470   (0)| 00:00:18 |
|   4 |     NESTED LOOPS                    |              |    66 |  5478 |  1404   (0)| 00:00:17 |
|   5 |      TABLE ACCESS BY INDEX ROWID    | CLASS        |     1 |    65 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN              | CL_NAME      |     1 |       |     1   (0)| 00:00:01 |
|*  7 |      TABLE ACCESS BY INDEX ROWID    | REGISTRATION |    66 |  1188 |  1402   (0)| 00:00:17 |
|*  8 |       INDEX RANGE SCAN              | REG_ALT      |  1407 |       |     3   (0)| 00:00:01 |
|*  9 |     INDEX UNIQUE SCAN               | PK_STUDENT   |     1 |       |     0   (0)|          |
|  10 |    TABLE ACCESS BY INDEX ROWID      | STUDENT      |     1 |    34 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

1 - filter(TO_DATE(:BEG_DATE)<=TO_DATE(:END_DATE))
6 - access("C"."NAME"='SQL TUNING')
7 - filter(("R"."SIGNUP_DATE">=:BEG_DATE AND "R"."SIGNUP_DATE"<=:END_DATE AND
        "R"."CANCELLED"='N'))
8 - access("R"."CLASS_ID"="C"."CLASS_ID")
9 - access("R"."STUDENT_ID"="S"."STUDENT_ID")
```

- Execution Stats – 3000 Buffer Gets / Average Execs - .008 Secs

# New Execution Plan

CREATE INDEX reg_cancel_signup ON registration(cancelled,signup_date);

```
select * from table (dbms_xplan.display_cursor('008x4scyck1tn','0'))

SQL_ID  008x4scyck1tn, child number 0
-------------------------------------
SELECT s.fname, s.lname, r.signup_date FROM   student s     INNER JOIN
registration r ON s.student_id = r.student_id     INNER JOIN class c ON
r.class_id = c.class_id WHERE  c.name  = 'SQL TUNING' AND
r.signup_date BETWEEN :beg_date and :end_date AND   r.cancelled = 'N'

Plan hash value: 1103429630

-----------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                   |       |       |   106 (100)|          |
|*  1 |  FILTER                                |                   |       |       |            |          |
|   2 |   NESTED LOOPS                         |                   |       |       |            |          |
|   3 |    NESTED LOOPS                        |                   |    70 |  8190 |   106   (1)| 00:00:02 |
|   4 |     NESTED LOOPS                       |                   |    70 |  5810 |    36   (3)| 00:00:01 |
|   5 |      TABLE ACCESS BY INDEX ROWID       | CLASS             |     1 |    65 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN                 | CL_NAME           |     1 |       |     1   (0)| 00:00:01 |
|   7 |      TABLE ACCESS BY INDEX ROWID       | REGISTRATION      |    70 |  1260 |    36   (3)| 00:00:01 |
|   8 |       BITMAP CONVERSION TO ROWIDS      |                   |       |       |            |          |
|   9 |        BITMAP AND                      |                   |       |       |            |          |
|  10 |         BITMAP CONVERSION FROM ROWIDS  |                   |       |       |            |          |
|* 11 |          INDEX RANGE SCAN              | REG_ALT           |  7971 |       |     3   (0)| 00:00:01 |
|  12 |         BITMAP CONVERSION FROM ROWIDS  |                   |       |       |            |          |
|  13 |          SORT ORDER BY                 |                   |       |       |            |          |
|* 14 |           INDEX RANGE SCAN             | REG_CANCEL_SIGNUP |  7971 |       |    25   (0)| 00:00:01 |
|* 15 |     INDEX UNIQUE SCAN                  | PK_STUDENT        |     1 |       |     0   (0)|          |
|  16 |    TABLE ACCESS BY INDEX ROWID         | STUDENT           |     1 |    34 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

1 - filter(TO_DATE(:BEG_DATE)<=TO_DATE(:END_DATE))
6 - access("C"."NAME"='SQL TUNING')
11 - access("R"."CLASS_ID"="C"."CLASS_ID")
14 - access("R"."CANCELLED"='N' AND "R"."SIGNUP_DATE">=:BEG_DATE AND "R"."SIGNUP_DATE"<=:END_DATE)
     filter(("R"."SIGNUP_DATE"<=:END_DATE AND "R"."SIGNUP_DATE">=:BEG_DATE AND "R"."CANCELLED"='N'))
15 - access("R"."STUDENT_ID"="S"."STUDENT_ID")
```

- Execution Stats – 1107 Buffer Gets /  Average Exec – 0.14 Secs

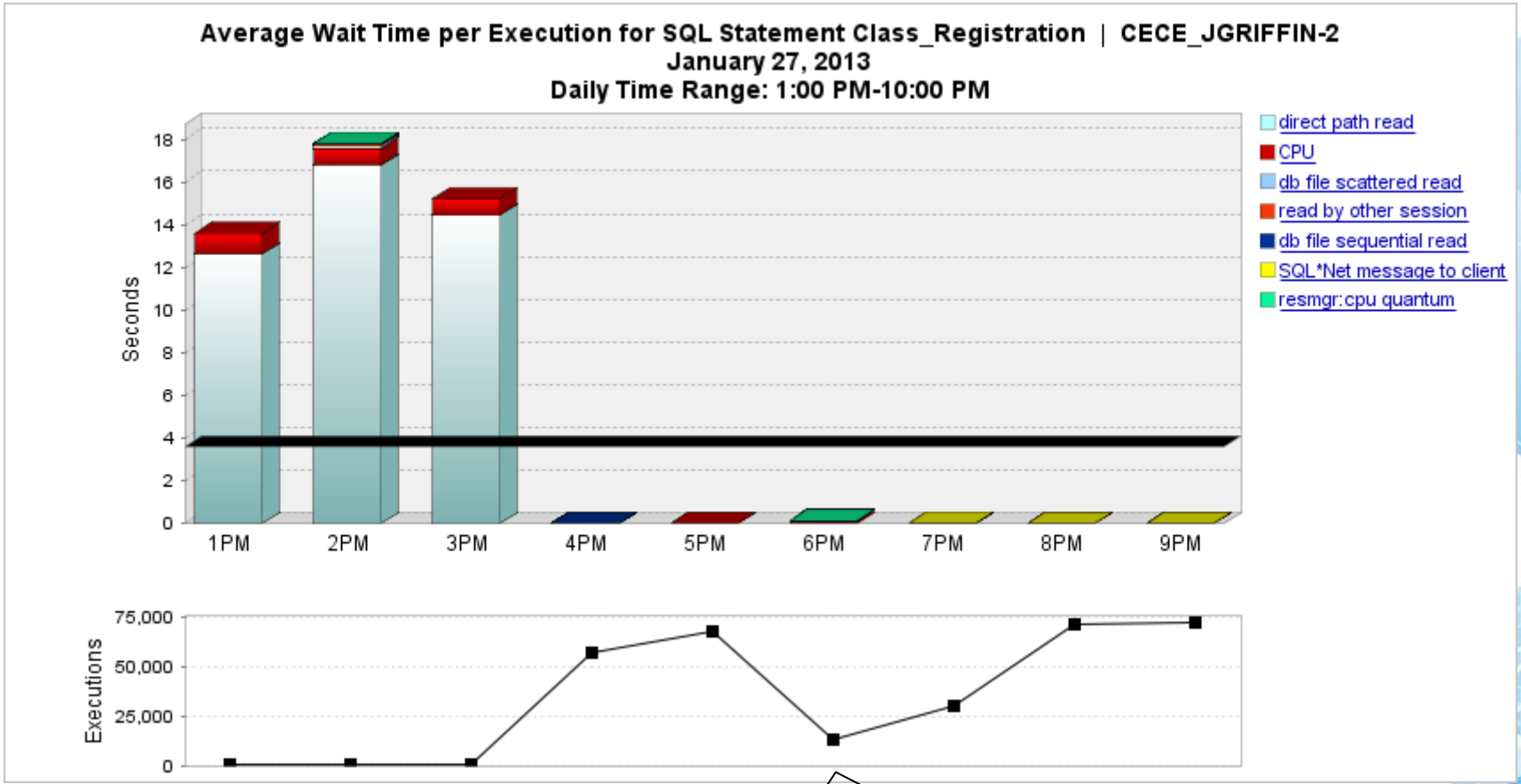
# Better Execution Plan

CREATE INDEX reg_alt ON registration(class_id,signup_date, cancelled);

```
select * from table (dbms_xplan.display_cursor('008x4scyck1tn','1'));

SQL_ID  008x4scyck1tn, child number 1
-------------------------------------
SELECT s.fname, s.lname, r.signup_date FROM   student s     INNER JOIN
registration r ON s.student_id = r.student_id     INNER JOIN class c ON
r.class_id = c.class_id WHERE  c.name  = 'SQL TUNING' AND
r.signup_date BETWEEN :beg_date and :end_date AND   r.cancelled = 'N'

Plan hash value: 3574817656

-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |       |       |   186 (100)|          |
|*  1 |  FILTER                             |              |       |       |            |          |
|   2 |   NESTED LOOPS                      |              |       |       |            |          |
|   3 |    NESTED LOOPS                     |              |    91 | 10647 |   186   (0)| 00:00:03 |
|   4 |     NESTED LOOPS                    |              |    91 |  7553 |    95   (0)| 00:00:02 |
|   5 |      TABLE ACCESS BY INDEX ROWID    | CLASS        |     1 |    65 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN              | CL_NAME      |     1 |       |     1   (0)| 00:00:01 |
|   7 |      TABLE ACCESS BY INDEX ROWID    | REGISTRATION |    91 |  1638 |    93   (0)| 00:00:02 |
|*  8 |       INDEX RANGE SCAN              | REG_ALT      |    91 |       |     2   (0)| 00:00:01 |
|*  9 |     INDEX UNIQUE SCAN               | PK_STUDENT   |     1 |       |     0   (0)|          |
|  10 |    TABLE ACCESS BY INDEX ROWID      | STUDENT      |     1 |    34 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

1 - filter(TO_DATE(:BEG_DATE)<=TO_DATE(:END_DATE))
6 - access("C"."NAME"='SQL TUNING')
8 - access("R"."CLASS_ID"="C"."CLASS_ID" AND "R"."SIGNUP_DATE">=:BEG_DATE AND
        "R"."CANCELLED"='N' AND "R"."SIGNUP_DATE"<=:END_DATE)
filter("R"."CANCELLED"='N')
9 - access("R"."STUDENT_ID"="S"."STUDENT_ID")
```

- Execution Stats – 445 Buffer Gets / Average Execs - .002 Secs

# Performance Improved?




canceled_signup index

CONFIO SOFTWARE

# SQL Statement 2

- Current paychecks for specific employees

```
SELECT e.first_name, e.last_name, l.region_name
FROM emp e
   INNER JOIN dept d ON e.department_id =
  d.department_id
   INNER JOIN loc l on l.location_id = d.location_id
WHERE (e.last_name like :b1)
AND EXISTS (
   SELECT 1
   FROM wage_pmt w
   WHERE w.employee_id = e.employee_id
   AND w.pay_date>= sysdate-31);
```

- Execution Stats - 3,890 Buffer Gets
- Average Execution - .31 seconds

CONFIO SOFTWARE

# Execution Plan

```
select * from table (dbms_xplan.display_cursor('2g7vydk4ng7an','0'))

SQL_ID  2g7vydk4ng7an, child number 0
-------------------------------------
SELECT e.first_name, e.last_name, l.region_name FROM emp e    INNER
JOIN dept d ON e.department_id = d.department_id    INNER JOIN loc l on
l.location_id = d.location_id WHERE (e.last_name like :b1) AND EXISTS (
SELECT 1    FROM wage_pmt w    WHERE w.employee_id = e.employee_id
AND w.pay_date>= sysdate-31)

Plan hash value: 1262318565

-----------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |          |       |       |       | 1806 (100)|          |
|*  1 |  HASH JOIN                          |          |  4537 |   239K|       | 1806   (2)| 00:00:22 |
|   2 |   TABLE ACCESS FULL                 | LOC      |    23 |   253 |       |    3   (0)| 00:00:01 |
|*  3 |   HASH JOIN                         |          |  4537 |   190K|       | 1803   (2)| 00:00:22 |
|   4 |    TABLE ACCESS FULL                | DEPT     |    27 |   189 |       |    3   (0)| 00:00:01 |
|   5 |    MERGE JOIN SEMI                  |          |  4579 |   160K|       | 1799   (2)| 00:00:22 |
|*  6 |     TABLE ACCESS BY INDEX ROWID     | EMP      |  4579 |   102K|       |  753   (1)| 00:00:10 |
|   7 |      INDEX FULL SCAN                | PK_EMP   | 54784 |       |       |  116   (0)| 00:00:02 |
|*  8 |     SORT UNIQUE                     |          | 50763 |   644K|  2408K| 1046   (2)| 00:00:13 |
|*  9 |      TABLE ACCESS FULL              | WAGE_PMT | 50763 |   644K|       |  802   (3)| 00:00:10 |
-----------------------------------------------------------------------------------------------------

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

   1 - access("L"."LOCATION_ID"="D"."LOCATION_ID")
   3 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
   6 - filter("E"."LAST_NAME" LIKE :B1)
   8 - access("W"."EMPLOYEE_ID"="E"."EMPLOYEE_ID")
       filter("W"."EMPLOYEE_ID"="E"."EMPLOYEE_ID")
   9 - filter("W"."PAY_DATE">=SYSDATE@!-31)
```

CONFIO SOFTWARE

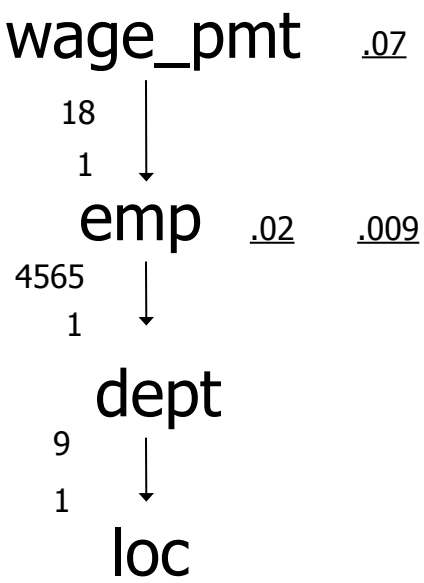
# Tuning Advisor

- **No recommendations?**

```
SQL_ID
-------------
2g7vydk4ng7an

RECOMMENDATIONS
--------------------------------------------------------------------------------
GENERAL INFORMATION SECTION
--------------------------------------------------------------------------------
Tuning Task Name    : 2g7vydk4ng7an
Tuning Task Owner   : HR
Workload Type       : Single SQL Statement
Scope               : COMPREHENSIVE
Time Limit(seconds): 60
Completion Status   : COMPLETED
Started at          : 01/31/2013 18:54:55
Completed at        : 01/31/2013 18:55:26

--------------------------------------------------------------------------------
Schema Name: HR
SQL ID      : 2g7vydk4ng7an
SQL Text    : SELECT e.first_name, e.last_name, l.region_name
              FROM emp e
                 INNER JOIN dept d ON e.department_id = d.department_id
                 INNER JOIN loc l on l.location_id = d.location_id
              WHERE (e.last_name like :b1)
              AND EXISTS (
                 SELECT 1
                 FROM wage_pmt w
                 WHERE w.employee_id = e.employee_id
                 AND w.pay_date>= sysdate-31)

--------------------------------------------------------------------------------
There are no recommendations to improve the statement.

--------------------------------------------------------------------------------
```

CONFIO SOFTWARE

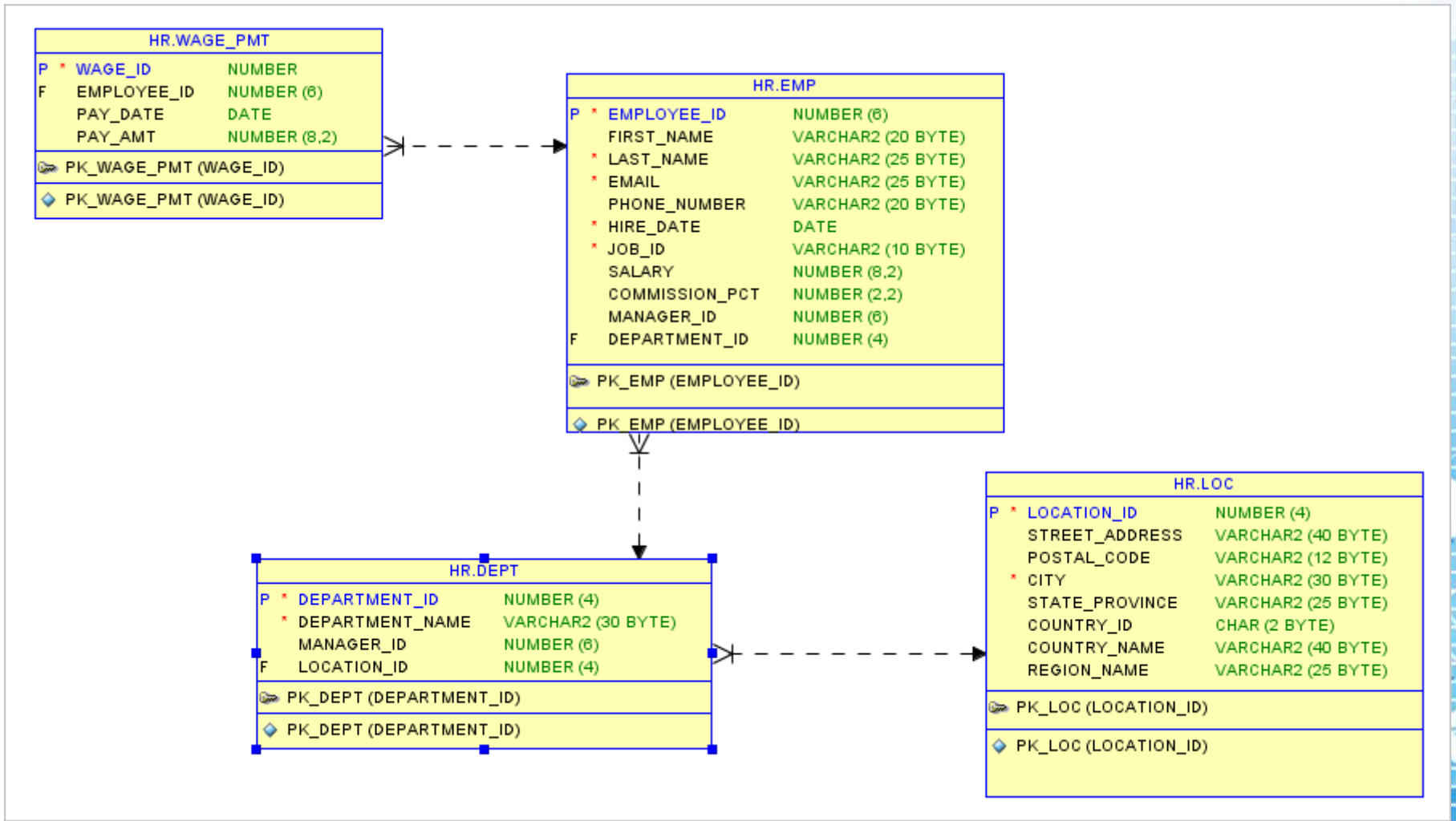
# SQL Diagramming




```
select count(1) from wage_pmt
where pay_date >= sysdate - 31

54,784 / 821,760 = .066

select max(cnt), min(cnt)
from (select last_name, count(1) cnt from emp group by last_name)

1,024 / 54,784 = .018 - max
512  /  54,784 = .009 - min
```

CONFIO SOFTWARE

# Relationship Diagram



# New Execution Plan

CREATE INDEX ix_last_name ON emp(last_name);

```
SQL_ID  2g7vydk4ng7an, child number 0
-------------------------------------
SELECT e.first_name, e.last_name, l.region_name FROM emp e    INNER
JOIN dept d ON e.department_id = d.department_id    INNER JOIN loc l on
l.location_id = d.location_id WHERE (e.last_name like :b1) AND EXISTS (
SELECT 1    FROM wage_pmt w    WHERE w.employee_id = e.employee_id
AND w.pay_date>= sysdate-31)

Plan hash value: 3027319603
-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |       |       |  2070 (100)|          |
|*  1 |  HASH JOIN SEMI               |               |  1427 | 77058 |  2070   (1)| 00:00:25 |
|*  2 |   HASH JOIN                   |               |  1427 | 58507 |  1268   (1)| 00:00:16 |
|   3 |    MERGE JOIN                 |               |    27 |   486 |     6  (17)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| LOC           |    23 |   253 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | PK_LOC        |    23 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                 |               |    27 |   189 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL        | DEPT          |    27 |   189 |     3   (0)| 00:00:01 |
|   8 |    TABLE ACCESS BY INDEX ROWID | EMP           |  1440 | 33120 |  1261   (0)| 00:00:16 |
|*  9 |     INDEX RANGE SCAN          | IX_LAST_NAME  |  1440 |       |     5   (0)| 00:00:01 |
|* 10 |   TABLE ACCESS FULL           | WAGE_PMT      | 50763 |   644K|   802   (3)| 00:00:10 |
-----------------------------------------------------------------------------------------------

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

1 - access("W"."EMPLOYEE_ID"="E"."EMPLOYEE_ID")
2 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
6 - access("L"."LOCATION_ID"="D"."LOCATION_ID")
filter("L"."LOCATION_ID"="D"."LOCATION_ID")
9 - access("E"."LAST_NAME" LIKE :B1)
filter("E"."LAST_NAME" LIKE :B1)
10 - filter("W"."PAY_DATE">=SYSDATE@!-31)
```

- Execution Stats – 1105 Buffer Gets / Average Execs - .06 Secs

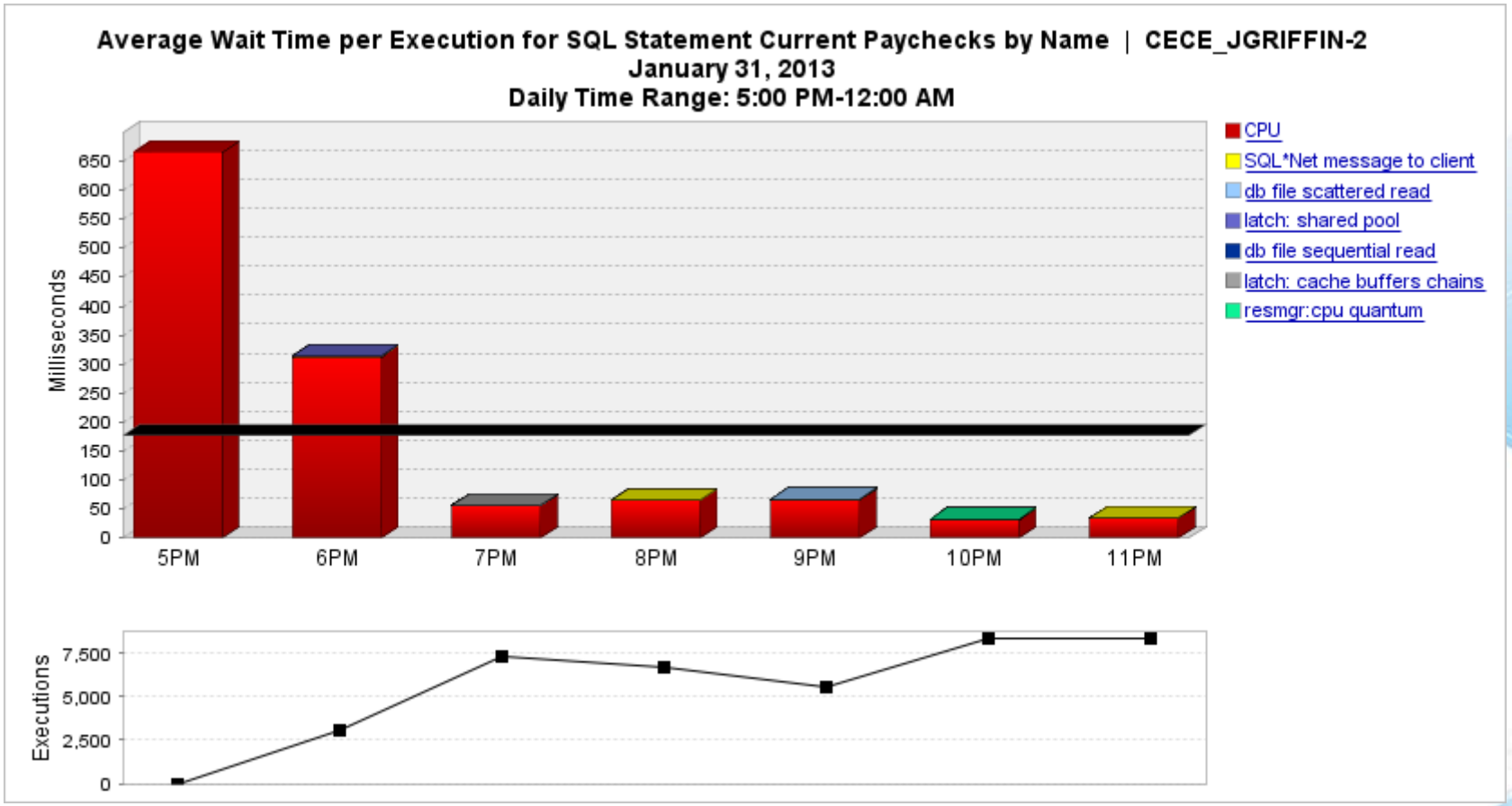
# New Execution Plan

CREATE INDEX wp_pd_emp ON wage_pmt(pay_date,employee_id);

```
SQL_ID  2g7vydk4ng7an, child number 0
-------------------------------------
SELECT e.first_name, e.last_name, l.region_name FROM emp e    INNER
JOIN dept d ON e.department_id = d.department_id    INNER JOIN loc l on
l.location_id = d.location_id WHERE (e.last_name like :b1) AND EXISTS (
SELECT 1    FROM wage_pmt w    WHERE w.employee_id = e.employee_id
AND w.pay_date>= sysdate-31)

Plan hash value: 3085468589

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |       |       |  1884 (100)|          |
|*  1 |  HASH JOIN SEMI               |              |  1929 |   101K|  1884   (1)| 00:00:23 |
|*  2 |   HASH JOIN                   |              |  1929 | 79089 |  1711   (1)| 00:00:21 |
|   3 |    MERGE JOIN                 |              |    27 |   486 |     6  (17)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| LOC          |    23 |   253 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | PK_LOC       |    23 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                 |              |    27 |   189 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL        | DEPT         |    27 |   189 |     3   (0)| 00:00:01 |
|   8 |    TABLE ACCESS BY INDEX ROWID | EMP          |  1947 | 44781 |  1704   (0)| 00:00:21 |
|*  9 |     INDEX RANGE SCAN          | IX_LAST_NAME |  1947 |       |     6   (0)| 00:00:01 |
|* 10 |   INDEX RANGE SCAN            | WAGE_PD_EMP  | 50763 |   644K|   172   (0)| 00:00:03 |
-----------------------------------------------------------------------------------------------

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

1 - access("W"."EMPLOYEE_ID"="E"."EMPLOYEE_ID")
2 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
6 - access("L"."LOCATION_ID"="D"."LOCATION_ID")
filter("L"."LOCATION_ID"="D"."LOCATION_ID")
9 - access("E"."LAST_NAME" LIKE :B1)
filter("E"."LAST_NAME" LIKE :B1)
10 - access("W"."PAY_DATE">=SYSDATE@!-31 AND "W"."PAY_DATE" IS NOT NULL)
```

- Execution Stats – 695 Buffer Gets / Average Execs - .03 Secs

# Improved Performance?




- Execution Stats – 695 Buffer Gets / Average Execs - .03 Secs

CONFIO SOFTWARE

# SQL Statement 3

- Inventory lookup for New Orders by Customer

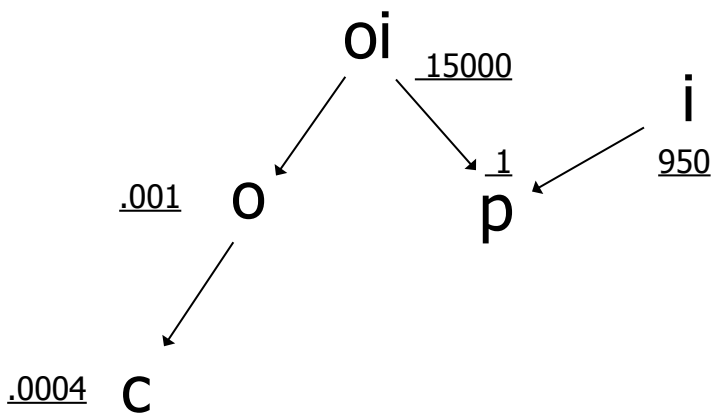
```
SELECT c.cust_first_name, c.cust_last_name,
   o.order_date, o.order_status, o.order_mode,
   i.line_item_id, p.product_Description, i.unit_price * i.quantity
   total_price, quantity quantity_ordered, ip.total_on_hand
FROM orders o, order_Items i,
   customers c, product p,
   (SELECT product_id, sum(quantity_on_hand) total_on_hand
    FROM inventories
    GROUP BY product_id) ip
WHERE i.order_id = o.order_id
AND c.customer_id = o.customer_id
AND p.product_id = i.product_id
AND p.product_id = ip.product_id
AND c.cust_last_name = :B1
AND o.order_date between to_date(:BEG_DATE,'mm/dd/yyyy') and
```

CONFIO SOFTWARE

# Execution Plan

```
Plan hash value: 2485762199

-----------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                     |       |       | 13392 (100)|          |
|*  1 |  HASH JOIN                          |                     |   183 | 53619 | 13392   (1)| 00:02:41 |
|   2 |   VIEW                              |                     |  1000 | 26000 |  3013   (2)| 00:00:37 |
|   3 |    HASH GROUP BY                    |                     |  1000 | 10000 |  3013   (2)| 00:00:37 |
|*  4 |     FILTER                          |                     |       |       |            |          |
|   5 |      TABLE ACCESS FULL              | INVENTORIES         |   894K|  8738K|  2988   (1)| 00:00:36 |
|   6 |   NESTED LOOPS                      |                     |       |       |            |          |
|   7 |    NESTED LOOPS                     |                     |   183 | 48861 | 10378   (1)| 00:02:05 |
|   8 |     NESTED LOOPS                    |                     |   183 | 13359 | 10195   (1)| 00:02:03 |
|   9 |      NESTED LOOPS                   |                     |    65 |  3510 | 10035   (1)| 00:02:01 |
|* 10 |       TABLE ACCESS BY INDEX ROWID   | ORDERS              |   240 |  7920 |  9555   (1)| 00:01:55 |
|* 11 |        INDEX RANGE SCAN             | ORD_ORDER_DATE_IX   | 10699 |       |    55   (0)| 00:00:01 |
|* 12 |       TABLE ACCESS BY INDEX ROWID   | CUSTOMERS           |     1 |    21 |     2   (0)| 00:00:01 |
|* 13 |        INDEX UNIQUE SCAN            | CUSTOMERS_PK        |     1 |       |     1   (0)| 00:00:01 |
|  14 |      TABLE ACCESS BY INDEX ROWID    | ORDER_ITEMS         |     3 |    57 |     3   (0)| 00:00:01 |
|* 15 |       INDEX RANGE SCAN              | ORDER_ITEMS_IX      |     3 |       |     2   (0)| 00:00:01 |
|* 16 |     INDEX UNIQUE SCAN               | PK_PRODUCT          |     1 |       |     0   (0)|          |
|  17 |    TABLE ACCESS BY INDEX ROWID      | PRODUCT             |     1 |   194 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

1 - access("P"."PRODUCT_ID"="IP"."PRODUCT_ID")
4 - filter(TO_DATE(:BEG_DATE,'mm/dd/yyyy')<=TO_DATE(:END_DATE,'mm/dd/yyyy'))
10 - filter("O"."ORDER_STATUS"=0)
11 - access("O"."ORDER_DATE">=TO_DATE(:BEG_DATE,'mm/dd/yyyy') AND
          "O"."ORDER_DATE"<=TO_DATE(:END_DATE,'mm/dd/yyyy'))
12 - filter("C"."CUST_LAST_NAME"=:B1)
13 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
15 - access("I"."ORDER_ID"="O"."ORDER_ID")
16 - access("P"."PRODUCT_ID"="I"."PRODUCT_ID")
```

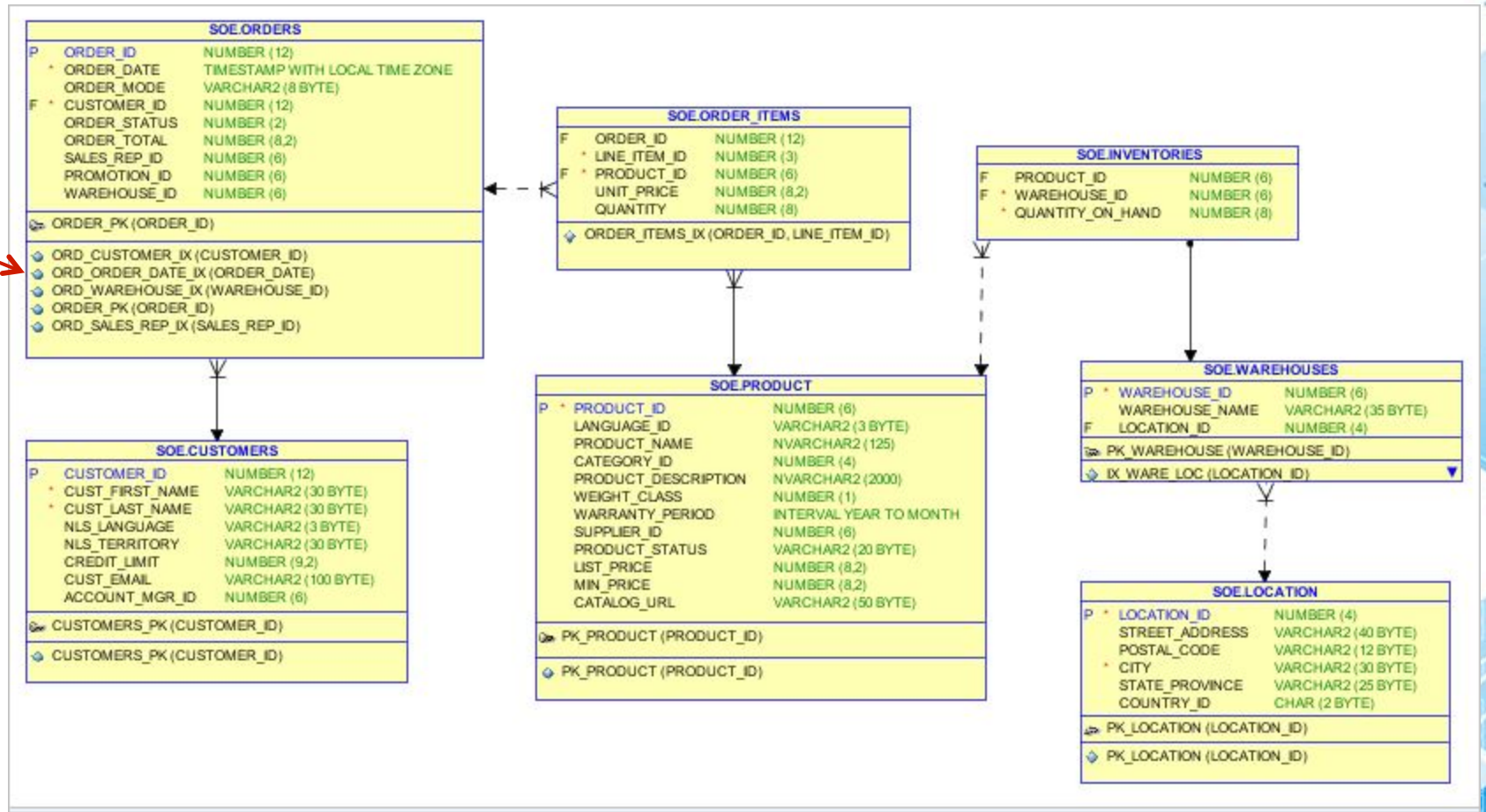
# SQL Diagramming




```
SELECT COUNT(1) FROM customer WHERE cust_last_name LIKE 'SMI%'

2054 / 5812142 =.00035


SELECT COUNT(1) FROM orders
WHERE order_status = 0
AND order_date BETWEEN TO_DATE(:BEG_DATE,'mm/dd/yyyy')
AND TO_DATE(:END_DATE,'mm/dd/yyyy`

8767 / 7399600 = .0011
```

CONFIO SOFTWARE

# Database Diagram



# New Execution Plan

CREATE INDEX ix_cust_last_name ON customers (cust_last_name);

```
Plan hash value: 1275669193
-------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                     |       |       |  3662 (100)|          |
|*  1 |  HASH JOIN                          |                     |   183 | 53619 |  3662   (1)| 00:00:44 |
|   2 |   VIEW                              |                     |  1000 | 26000 |  3013   (2)| 00:00:37 |
|   3 |    HASH GROUP BY                    |                     |  1000 | 10000 |  3013   (2)| 00:00:37 |
|*  4 |     FILTER                          |                     |       |       |            |          |
|   5 |      TABLE ACCESS FULL              | INVENTORIES         |   894K|  8738K|  2988   (1)| 00:00:36 |
|   6 |   NESTED LOOPS                      |                     |       |       |            |          |
|   7 |    NESTED LOOPS                     |                     |   183 | 48861 |   649   (1)| 00:00:08 |
|   8 |     NESTED LOOPS                    |                     |   183 | 13359 |   465   (0)| 00:00:06 |
|   9 |      NESTED LOOPS                   |                     |    65 |  3510 |   306   (0)| 00:00:04 |
|  10 |       TABLE ACCESS BY INDEX ROWID   | CUSTOMERS           |    65 |  1365 |    63   (0)| 00:00:01 |
|* 11 |        INDEX RANGE SCAN             | IX_CUST_LAST_NAME   |    65 |       |     3   (0)| 00:00:01 |
|* 12 |       TABLE ACCESS BY INDEX ROWID   | ORDERS              |     1 |    33 |     5   (0)| 00:00:01 |
|* 13 |        INDEX RANGE SCAN             | ORD_CUSTOMER_IX     |     2 |       |     2   (0)| 00:00:01 |
|  14 |      TABLE ACCESS BY INDEX ROWID    | ORDER_ITEMS         |     3 |    57 |     3   (0)| 00:00:01 |
|* 15 |       INDEX RANGE SCAN              | ORDER_ITEMS_IX      |     3 |       |     2   (0)| 00:00:01 |
|* 16 |     INDEX UNIQUE SCAN               | PK_PRODUCT          |     1 |       |     0   (0)|          |
|  17 |    TABLE ACCESS BY INDEX ROWID      | PRODUCT             |     1 |   194 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
1 - access("P"."PRODUCT_ID"="IP"."PRODUCT_ID")
4 - filter(TO_DATE(:BEG_DATE,'mm/dd/yyyy')<=TO_DATE(:END_DATE,'mm/dd/yyyy'))
11 - access("C"."CUST_LAST_NAME"=:B1)
12 - filter(("O"."ORDER_STATUS"=0 AND "O"."ORDER_DATE">=TO_DATE(:BEG_DATE,'mm/dd/yyyy') AND
        "O"."ORDER_DATE"<=TO_DATE(:END_DATE,'mm/dd/yyyy')))
13 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
15 - access("I"."ORDER_ID"="O"."ORDER_ID")
16 - access("P"."PRODUCT_ID"="I"."PRODUCT_ID")
```

- Execution Stats – 11,182 Buffer Gets

# Best Execution Plan

CREATE INDEX ix_product ON inventories (inventories);

```
Plan hash value: 3266027157

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |       |       |  3579 (100)|          |
|   1 |  NESTED LOOPS                 |                    |   183 | 51972 |  3579   (1)| 00:00:43 |
|   2 |   NESTED LOOPS                |                    |   183 | 49593 |   649   (1)| 00:00:08 |
|   3 |    NESTED LOOPS               |                    |   183 | 13359 |   465   (0)| 00:00:06 |
|   4 |     NESTED LOOPS              |                    |    65 |  3510 |   306   (0)| 00:00:04 |
|   5 |      TABLE ACCESS BY INDEX ROWID| CUSTOMERS          |    65 |  1365 |    63   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN        | IX_CUST_LAST_NAME  |    65 |       |     3   (0)| 00:00:01 |
|*  7 |      TABLE ACCESS BY INDEX ROWID| ORDERS           |     1 |    33 |     5   (0)| 00:00:01 |
|*  8 |       INDEX RANGE SCAN        | ORD_CUSTOMER_IX    |     2 |       |     2   (0)| 00:00:01 |
|   9 |     TABLE ACCESS BY INDEX ROWID | ORDER_ITEMS      |     3 |    57 |     3   (0)| 00:00:01 |
|* 10 |      INDEX RANGE SCAN         | ORDER_ITEMS_IX     |     3 |       |     2   (0)| 00:00:01 |
|  11 |    TABLE ACCESS BY INDEX ROWID | PRODUCT           |     1 |   198 |     1   (0)| 00:00:01 |
|* 12 |     INDEX UNIQUE SCAN         | PK_PRODUCT         |     1 |       |     0   (0)|          |
|  13 |   VIEW PUSHED PREDICATE       |                    |     1 |    13 |    16   (0)| 00:00:01 |
|* 14 |    FILTER                     |                    |       |       |            |          |
|  15 |     SORT AGGREGATE            |                    |     1 |    10 |            |          |
|  16 |      TABLE ACCESS BY INDEX ROWID| INVENTORIES       |   895 |  8950 |    16   (0)| 00:00:01 |
|* 17 |       INDEX RANGE SCAN        | IX_PRODUCT         |   895 |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

6 - access("C"."CUST_LAST_NAME"=:B1)
7 - filter(("O"."ORDER_STATUS"=0 AND "O"."ORDER_DATE">=TO_DATE(:BEG_DATE,'mm/dd/yyyy')
        AND "O"."ORDER_DATE"<=TO_DATE(:END_DATE,'mm/dd/yyyy')))
8 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
10 - access("I"."ORDER_ID"="O"."ORDER_ID")
12 - access("P"."PRODUCT_ID"="I"."PRODUCT_ID")
14 - filter((COUNT(*)>0 AND TO_DATE(:BEG_DATE,'mm/dd/yyyy')<=TO_DATE(:END_DATE,'mm/dd/yyyy'
       )))
17 - access("PRODUCT_ID"="P"."PRODUCT_ID")
```

- Execution Stats – 262 Buffer Gets

CONFIO SOFTWARE

# Data Skew Problems

```
SELECT order_status, COUNT(1)
FROM orders
GROUP BY order_status
```

| ORDER_STATUS | COUNT(1) |
|---|---|
| 0 | 165308 |
| 1 | 472961 |
| 2 | 175 |
| 3 | 25 |
| 4 | 399013 |
| 5 | 844311 |
| 6 | 847388 |
| 7 | 845958 |
| 8 | 846991 |
| 9 | 2130175 |

- Only 2% of rows are equal 0

CONFIO SOFTWARE

# Monitor

- Monitor the improvement
  - Be able to prove that tuning made a difference
  - Take new metric measurements
  - Compare them to initial readings
  - Brag about the improvements – no one else will
- Monitor for next tuning opportunity
  - Tuning is iterative
  - There is always room for improvement
  - Make sure you tune things that make a difference
- Shameless Product Pitch - Ignite

CONFIO SOFTWARE

# Takeaway Points

- Tuning Queries gives more "bang for the buck"
- Make sure you are tuning the right query
- Use Wait Events & Response Time Analysis
  - Locking problems may not be a Query Tuning issue
  - Wait Events tell you where to start
- Use "Real Execution Plans"
  - Get SQL_ID & Child_Number in V$SQL
  - Display using DBMS_XPLAN.display_cursor
- SQL Diagramming
  - Query Tuner Tools can mislead you
- Monitor For Next Tuning Opportunity

# Confio Software

- Wait-Based Performance Tools
- Ignite8 / IgniteVM
  - Ignite for Oracle, SQL Server, DB2, Sybase
- Helps show which SQL to tune
- Based in Colorado, worldwide customers
- Free trial at www.confio.com
- http://www.ignitefree.com – Free Current View