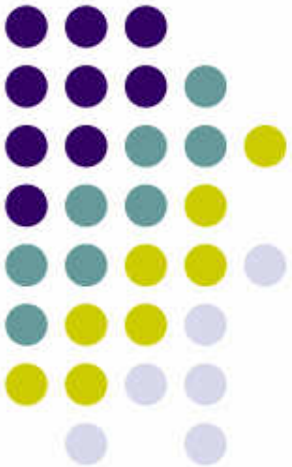
11g Tech Briefing: Performance

Part 1 of 2



Presenter

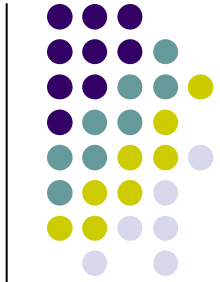
JEREMY SCHNEIDER

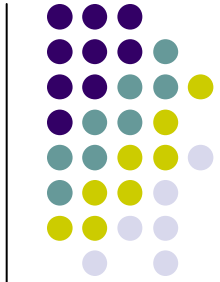
jeremy.schneider@ardentperf.com

Senior Consultant,
ITC Technology Services
OCP, RAC since 2002, Systems Admin
and Developer in previous lives
Blogger - http://www.ardentperf.com

11g Performance Agenda

- New Caches
  - Server Cache (Query Blocks, PL/SQL Functions)
  - Client Cache (OCI)
- Optimizer Evolution
  - Self-Learning (Auto-Tuning and Plan Management)
  - *Statistics Improvements (Extending, Gathering, Publishing)*
  - *Invisible Indexes*
- *Misc Performance Improvements*
  - *Compression, Default Values, PL/SQL, SecureFiles*
  - *JDBC, Streams, Data Guard, NativeNFS*

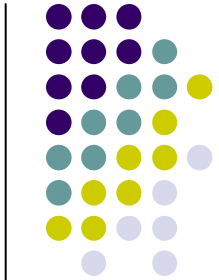




11g New Caches

- Server Result Cache
  - SQL Query Result Cache
  - PL/SQL Function Result Cache
- Client Result Cache
  - OCI Consistent Client Cache

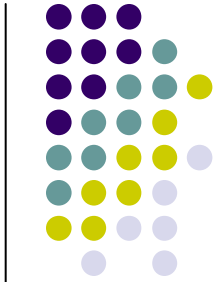
Server Result Cache



- Caches results of query blocks and PL/SQL functions
  - Parameterized: results cached by bind variable value or function parameter; only match with same values
- Flashback queries can be cached

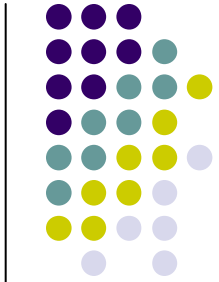
- Result NOT cached if:
  - Query non-current version of data (read consistency)
  - Current session has outstanding transaction on dependant table
- Changes to dependant objects automatically invalidate cache
- Primitive memory management with first release
  - Grows to maximum size, does not automatically free memory
  - DBMS_RESULT_CACHE.FLUSH frees memory

Server Result Cache



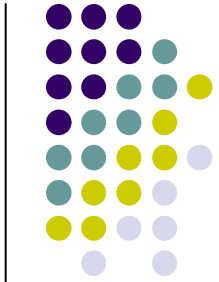
- Memory allocated from Shared Pool
  - cross-session, instance-specific
- Setup - Init Parameters
  - RESULT_CACHE_MAX_SIZE
  - RESULT_CACHE_MAX_RESULT
- Monitoring – Dynamic Performance Views
  - [G]V$RESULT_CACHE_STATISTICS
  - [G]V$RESULT_CACHE_OBJECTS
  - [G]V$RESULT_CACHE_DEPENDENCY
  - [G]V$RESULT_CACHE_MEMORY
- Reporting/Management - PL/SQL Package
  - DBMS_RESULT_CACHE *(BYPASS, FLUSH, INVALIDATE, INVALIDATE_OBJECT, MEMORY_REPORT, STATUS)*

Server Result Cache



```
SQL> set serveroutput on
SQL> execute dbms_result_cache.memory_report

Result Cache Memory Report
[Parameters]
Block Size = 1024 bytes
Maximum Cache Size = 950272 bytes (928 blocks)
Maximum Result Size = 47104 bytes (46 blocks)
[Memory]
Total Memory = 46340 bytes [0.048% of the Shared Pool]
... Fixed Memory = 10696 bytes [0.011% of the Shared Pool]
... State Object Pool = 2852 bytes [0.003% of the Shared Pool]
... Cache Memory = 32792 bytes (32 blocks) [0.034% of the Shared Pool]
....... Unused Memory = 30 blocks
....... Used Memory = 2 blocks
........... Dependencies = 1 blocks
........... Results = 1 blocks
............... SQL = 1 blocks
```

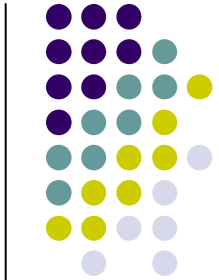
Server Result Cache



- A system that makes good use of the Server Result Cache should show relatively low values for Create Count Failure and Delete Count Valid, while showing relatively high values for Find Count. (11g Perf Tuning Guide)

```
SQL> column name format a20
SQL> select name, value from v$result_cache_statistics;

NAME                            VALUE
------------------------------ ----------
Block Size (Bytes)                1024
Block Count Maximum               3136
Block Count Current                 32
Result Size Maximum (Blocks)       156
Create Count Success                 2
Create Count Failure                 0
Find Count                           0
Invalidation Count                   0
Delete Count Invalid                 0
Delete Count Valid                   0
```

SQL Query Result Cache

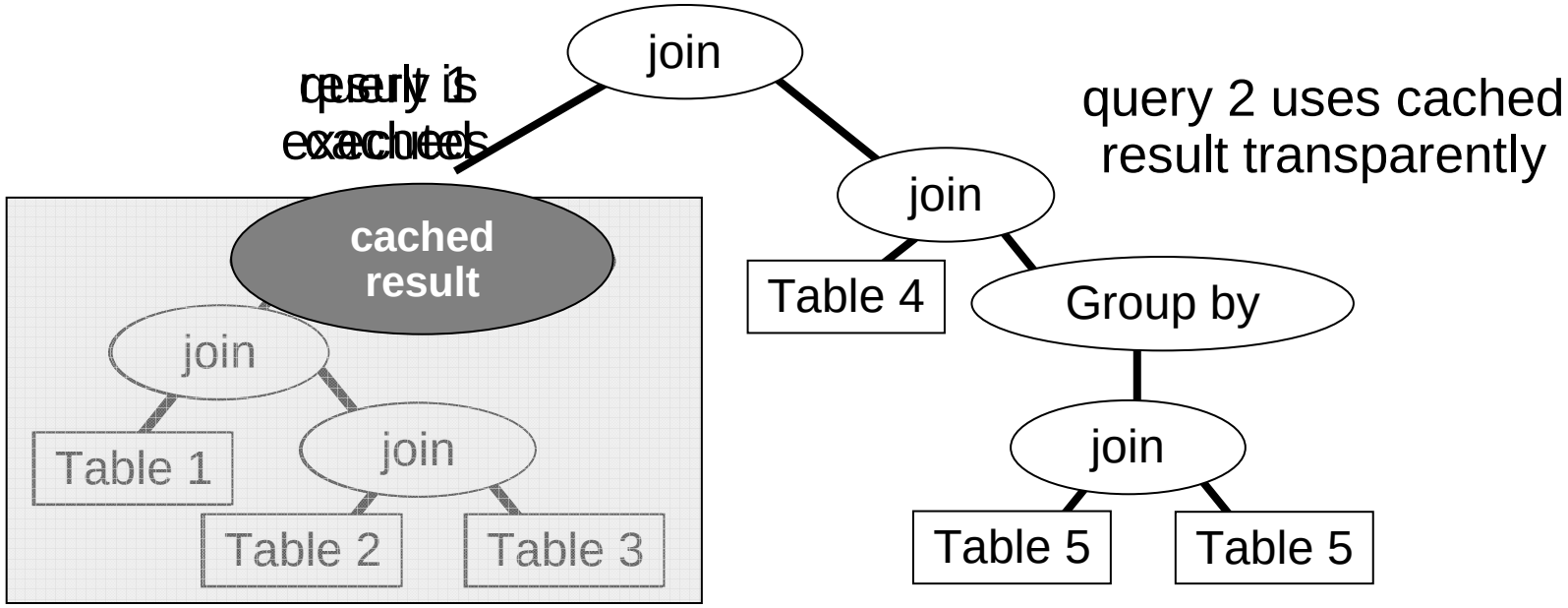


- Results of query BLOCKS are candidates for caching
- Enabled globally with RESULT_CACHE_MODE init param
  - MANUAL (default) | AUTO | FORCE
- Override on a single query with RESULT_CACHE and NO_RESULT_CACHE hints

```
select /*+ RESULT_CACHE */ p.prod_category
,      sum(s.amount_sold) revenue
from products p
,    sales    s
where s.prod_id = p.prod_id
and   s.time_id
  between to_date('01-JAN-2006','dd-MON-yyyy')↰
  and     to_date('31-DEC-2006','dd-MON-yyyy')↰
group by rollup (p.prod_category)↰
```

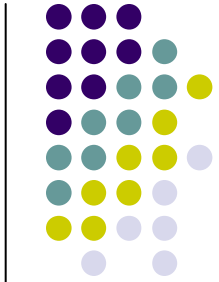
SQL Query Result Cache

- How it works:



Animation from Oracle

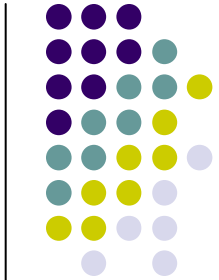
SQL Query Result Cache



- “Name” field displayed in execution plan corresponds to **CacheId** field in **V$RESULT_CACHE_OBJECTS**

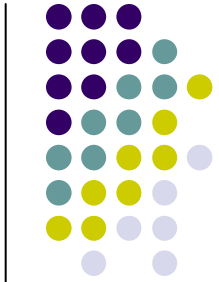
```
-----------------------------------------------------------------------
| Id  | Operation                      | Name                         |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                              |
|   1 |  RESULT CACHE                  | fz6cm4jbpcwh48wcyk60m7qypu   |
|   2 |   SORT GROUP BY ROLLUP         |                              |
|*  3 |    HASH JOIN                   |                              |
|   4 |     PARTITION RANGE ITERATOR   |                              |
|*  5 |      TABLE ACCESS FULL         | SALES                        |
|   6 |     VIEW                       | index$_join$_001             |
|*  7 |      HASH JOIN                 |                              |
|   8 |       INDEX FAST FULL SCAN     | PRODUCTS_PK                  |
|   9 |       INDEX FAST FULL SCAN     | PRODUCTS_PROD_CAT_IX         |
-----------------------------------------------------------------------
```

SQL Query Result Cache



- Prevents some optimizations on initial execution (subsequent executions use cache)
  - View Merging
  - Predicate push-down
  - Column projection
- Some queries are ineligible
  - Temp or dict tables
  - Non-deterministic PL/SQL functions
  - Sequences
  - Distributed Queries by default – can be enabled with RESULT_CACHE_REMOTE_EXPIRATION to non-zero

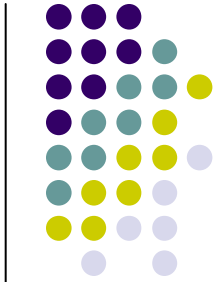
PL/SQL Function Result Cache



- Enabled with RESULT_CACHE clause in PL/SQL function definition
- Dependant tables/views specified with RELIES_ON clause

```
CREATE OR REPLACE PACKAGE department_pks IS
FUNCTION get_dept_info (dept_id NUMBER) RETURN dept_info_record
        RESULT_CACHE;
...

CREATE OR REPLACE PACKAGE BODY department_pks AS
FUNCTION get_dept_info (dept_id NUMBER) RETURN dept_info_record
        RESULT_CACHE RELIES_ON (EMPLOYEES);
...
```

PL/SQL Function Result Cache

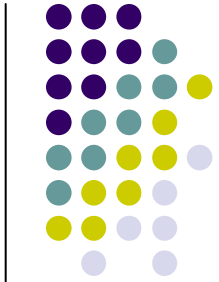


```
SQL> create or replace function test_result_cache( p_in in number )
return number
2 as
3 begin
4 sys.dbms_lock.sleep(10);
5 return( p_in );
6 end;
7 /

Function created.

Elapsed: 00:00:00.17
```

```
SQL> select test_result_cache(10) from dual;

TEST_RESULT_CACHE(10)
--------------------
                  10

Elapsed: 00:00:10.39
SQL> select test_result_cache(10) from dual;

TEST_RESULT_CACHE(10)
--------------------
                  10

Elapsed: 00:00:10.10
```

Demo from AMIS Technology Blog

PL/SQL Function Result Cache

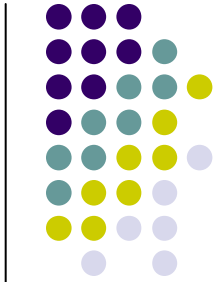


```
SQL> create or replace function test_result_cache( p_in in number )
return number result_cache
2 as
3 begin
4 sys.dbms_lock.sleep(10);
5 return( p_in );
6 end;
7 /

Function created.

Elapsed: 00:00:00.07
```

```
SQL> select test_result_cache(10) from dual;

TEST_RESULT_CACHE(10)
---------------------
                   10

Elapsed: 00:00:10.35
SQL> select test_result_cache(10) from dual;

TEST_RESULT_CACHE(10)
---------------------
                   10

Elapsed: 00:00:00.00
```

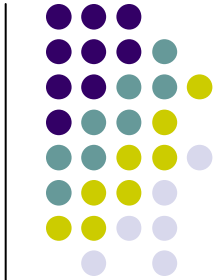
Demo from AMIS Technology Blog

PL/SQL Function Result Cache



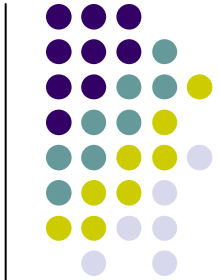
- Some functions are ineligible:
  - Invoker's rights modules or functions in anonymous blocks
  - Pipelined functions
  - IN parameter or return value of an unsupported type
    - BLOB
    - CLOB
    - NCLOB
    - REF CURSOR
    - collections,
    - Objects
    - records
  - OUT or IN OUT parameters

Client Result Cache



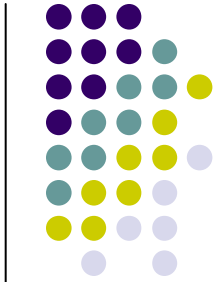
- EXTENDS Server Result Cache
  - Includes query blocks and PL/SQL function results
  - Enabled with same hint (RESULT_CACHE) or init param (RESULT_CACHE_MODE)
  - Can be enabled/disabled independently of Server Cache
- Available on all OCI-based clients
  - JDBC OCI *(not available for thin JDBC driver)*
  - OCCI
  - ODP.NET
  - PHP
  - ODBC
- Requires 11g client and 11g server (naturally)

OCI Consistent Client Cache



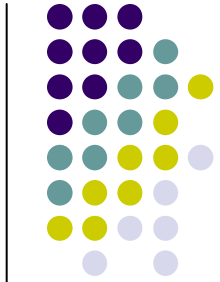
- Memory allocated client-side by OCI library
  - cross-session, process-specific
- Setup – Init Parameter
  - CLIENT_RESULT_CACHE_SIZE
- Setup – sqlnet.ora (overrides init param)
  - OCI_RESULT_CACHE_MAX_SIZE
  - OCI_RESULT_CACHE_MAX_RSET_SIZE
  - OCI_RESULT_CACHE_MAX_RSET_ROWS
- Monitoring
  - client_result_cache_stats$

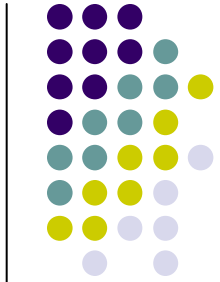
OCI Consistent Client Cache



Consistency maintained automatically; server invalidates client cache when dependant objects change

- IN-BAND notifications: invalidations piggyback on existing round-trip messages
- If client is idle (no calls to server) for specified timeout then it will explicitly check with the server for invalidations
  - Timeout default is 5 seconds, configurable through init param CLIENT_RESULT_CACHE_LAG





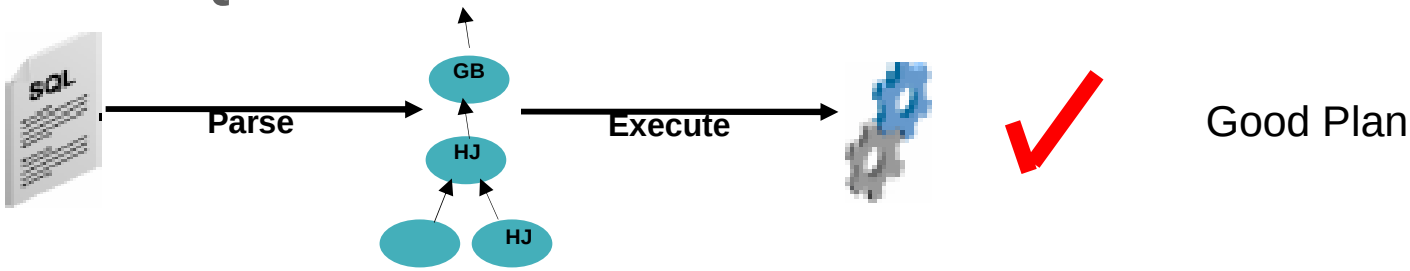
11g Optimizer Evolution

- Self-Learning
  - Auto-Tuning
  - Plan Management
- Statistics Improvements
  - Extending Statistics
  - Gathering Statistics
  - Publishing Statistics
- Invisible Indexes

Self-Learning

- SQL Tuning in Oracle 10g

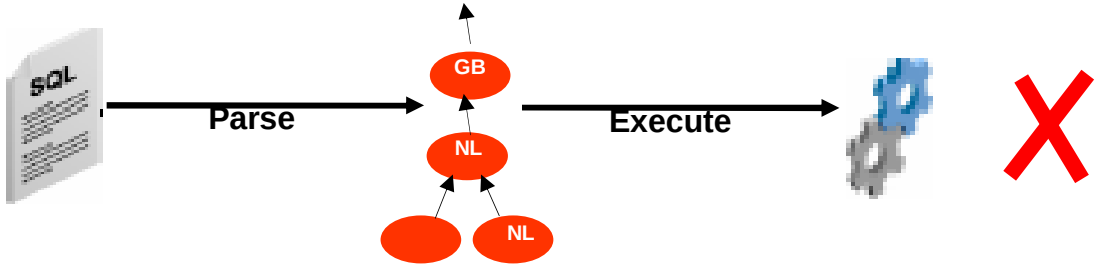
1. First SQL Execution: Hard Parse



Good Plan

2. Environmental Change: stats job, smaller UGA, etc

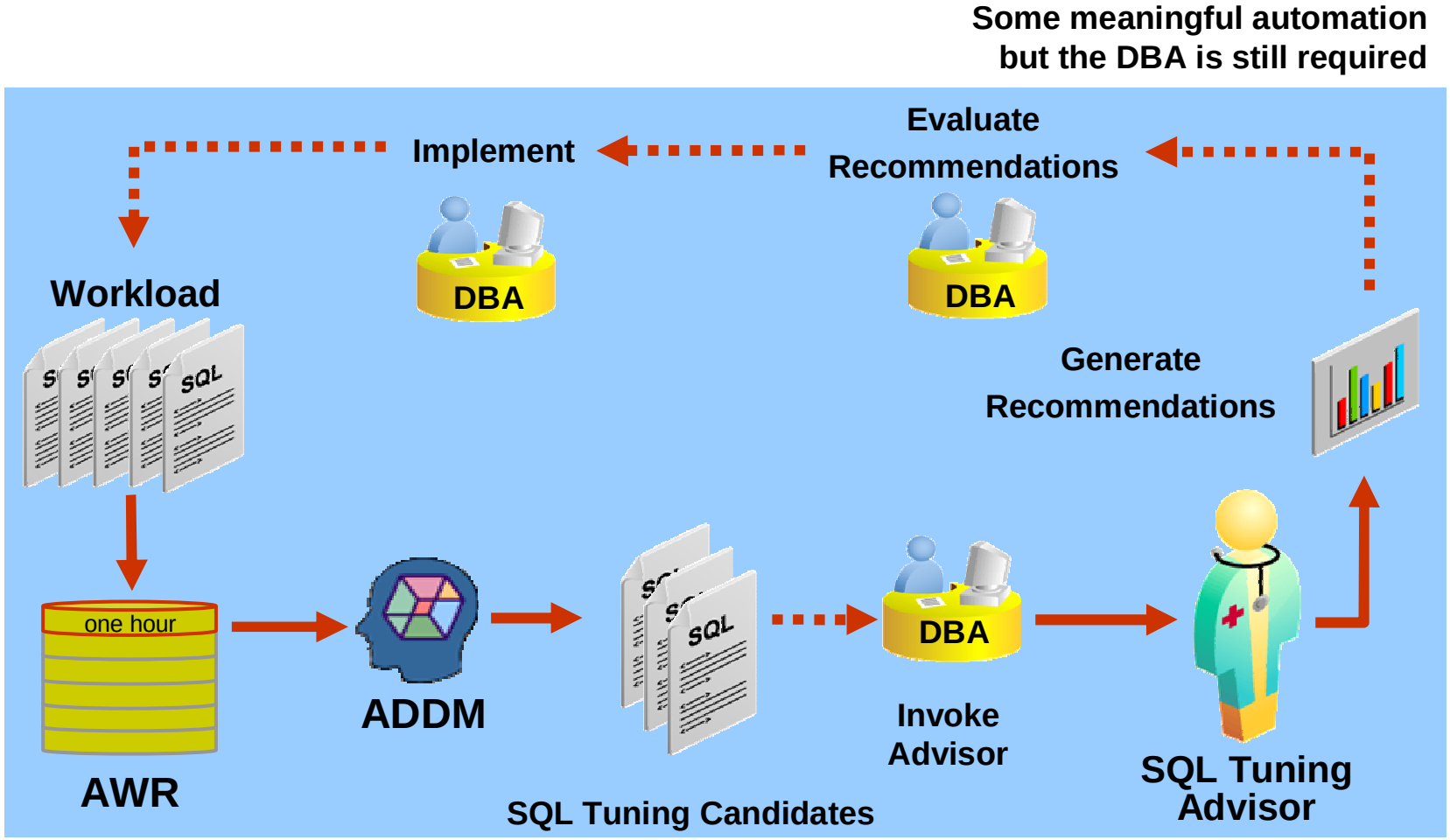
3. Plan Invalidated: Hard Parse results in new plan



Bad Plan!

Self-Learning

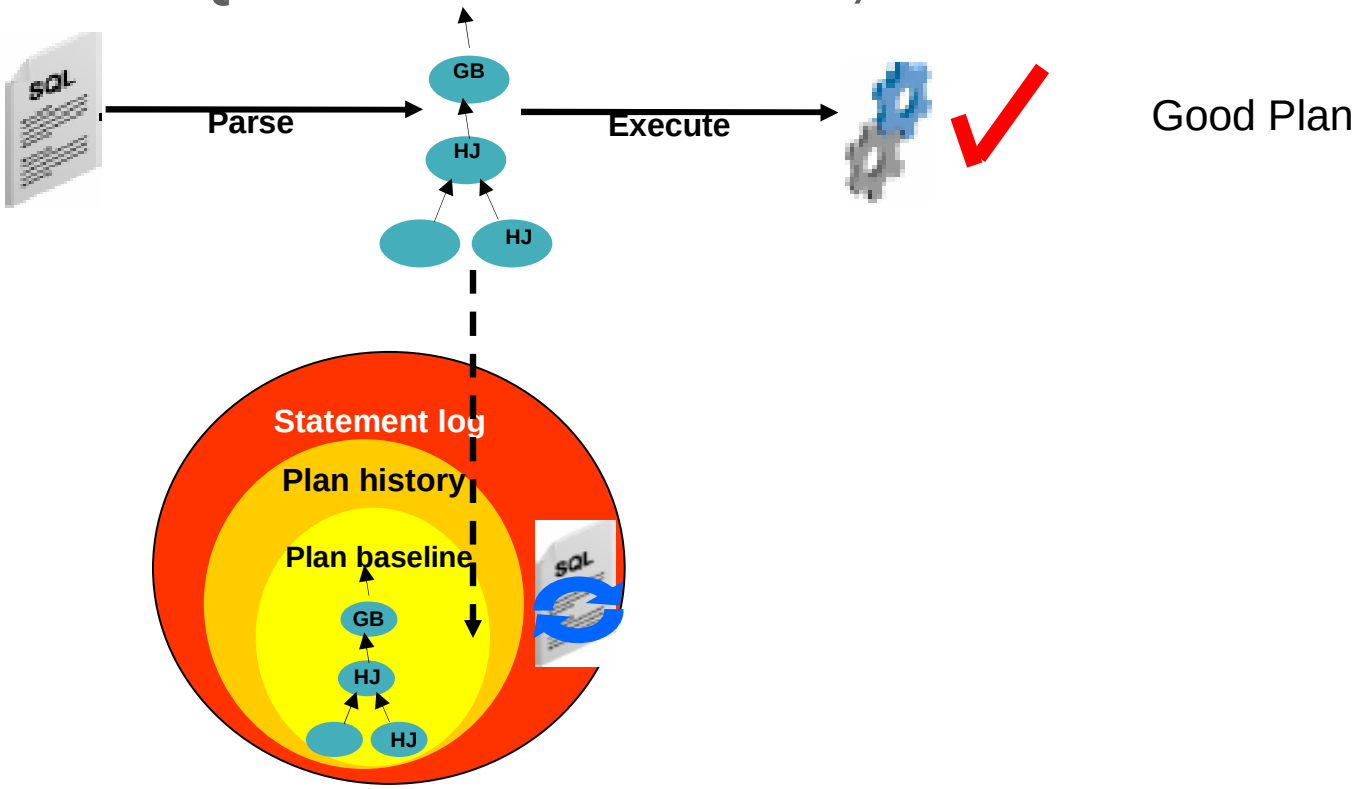
- SQL Tuning in Oracle 10g



Self-Learning

- SQL Tuning in **Oracle 11g**

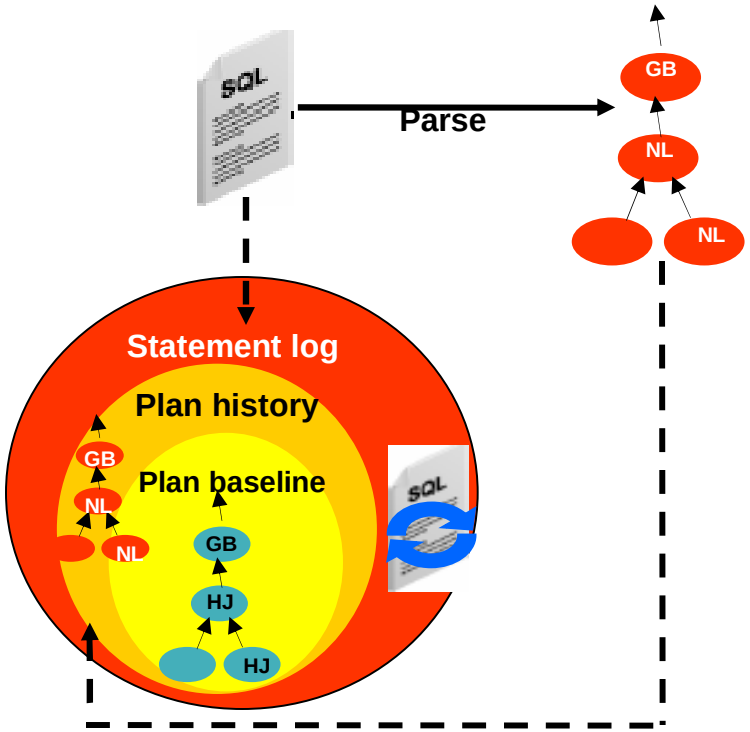
1. **First SQL Execution: Hard Parse, *STORE "BASELINE"***



Self-Learning

- SQL Tuning in **Oracle 11g**

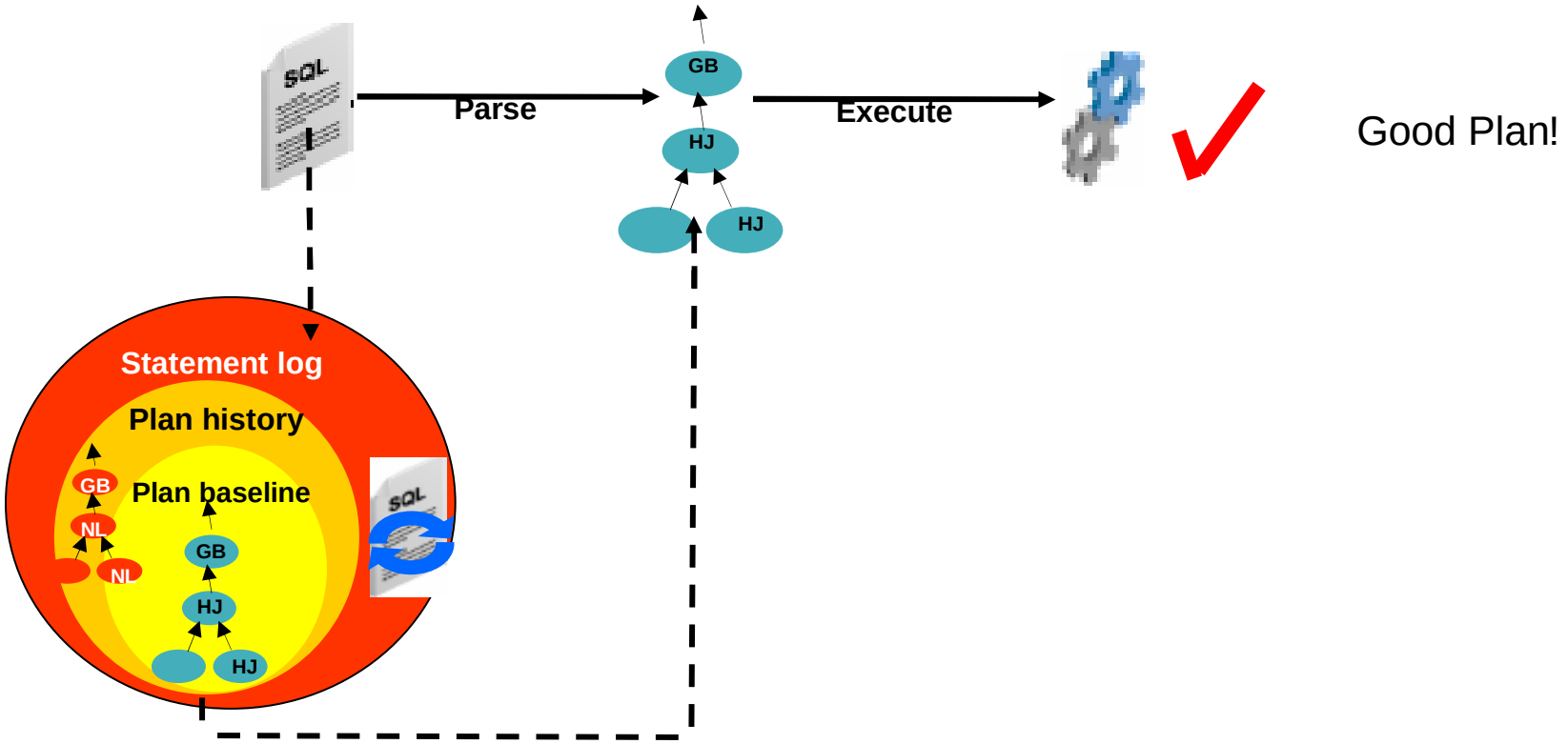
2. **Environmental Change, Plan Invalidated, Hard Parse – NEW PLAN IS NOT EXECUTED BUT MARKED FOR VERIFICATION**



Self-Learning

- SQL Tuning in **Oracle 11g**

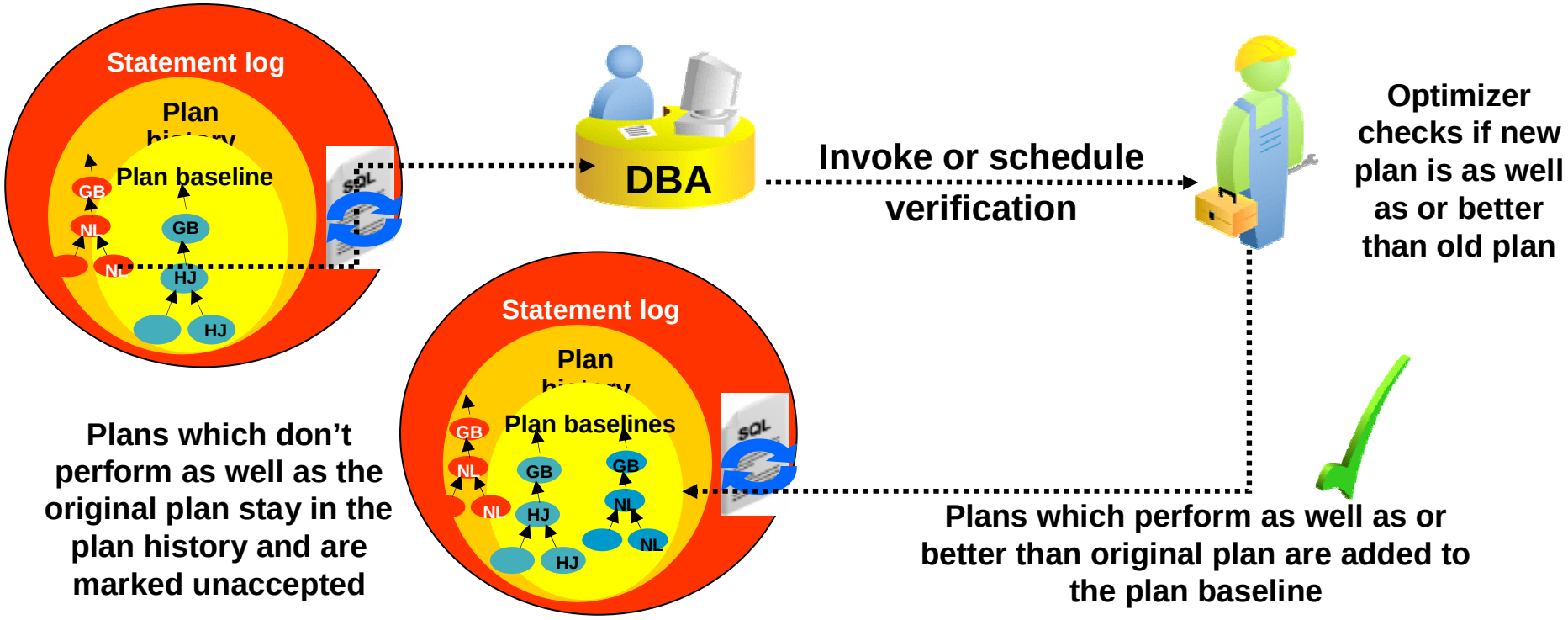
3. BASELINE (ORIGINAL) PLAN IS EXECUTED



Self-Learning

- SQL Tuning in **Oracle 11g**

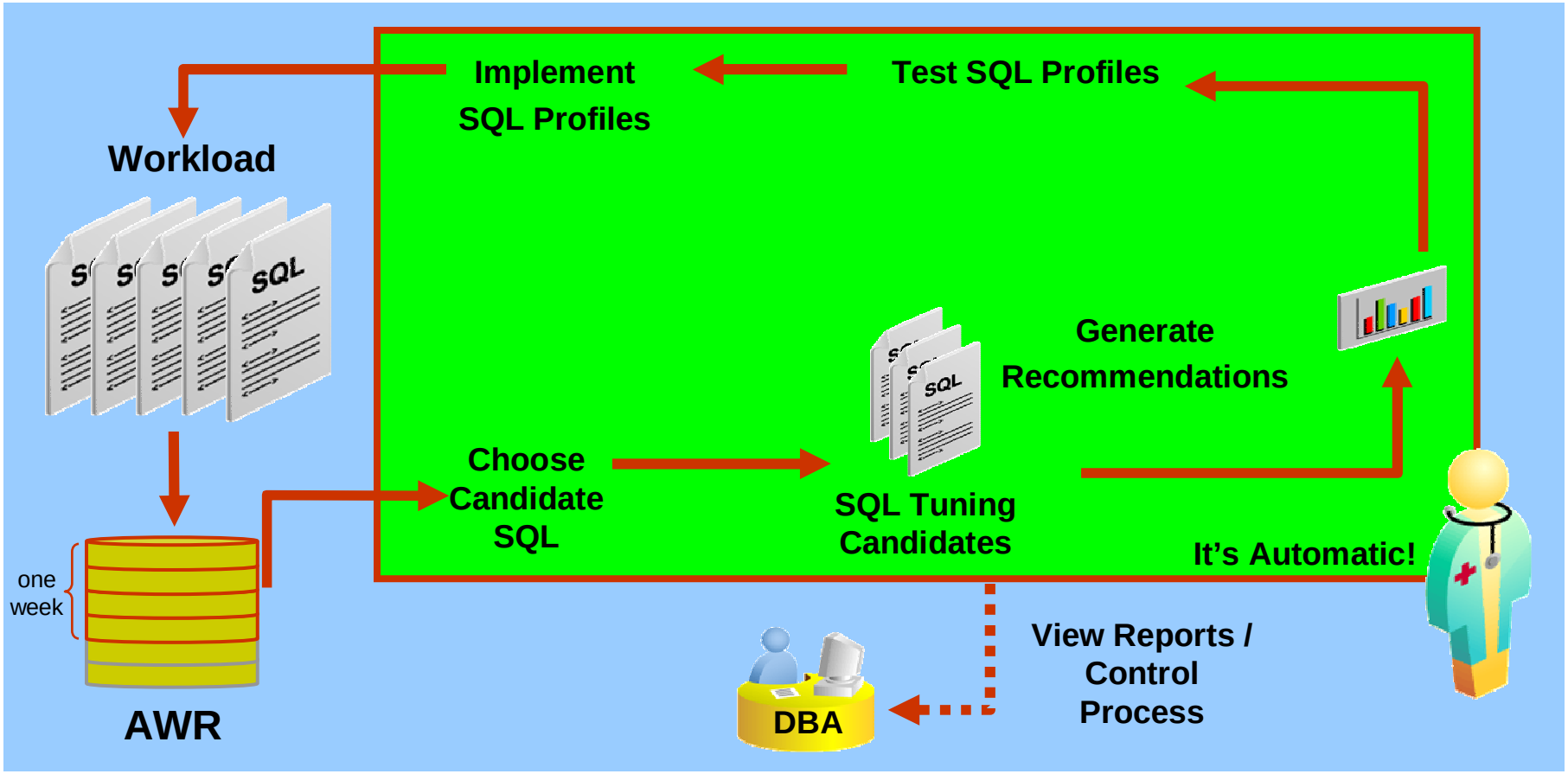
4. AFTER LATER VERIFICATION, PLANS THAT IMPROVE PERFORMANCE AUTOMATICALLY ADDED TO BASELINE



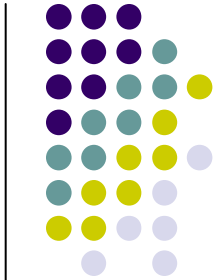
Self-Learning

- SQL Tuning in **Oracle 11g**

Animation from Oracle

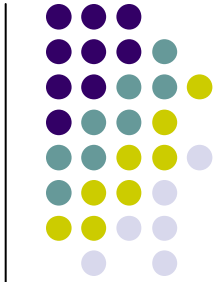


Self-Learning



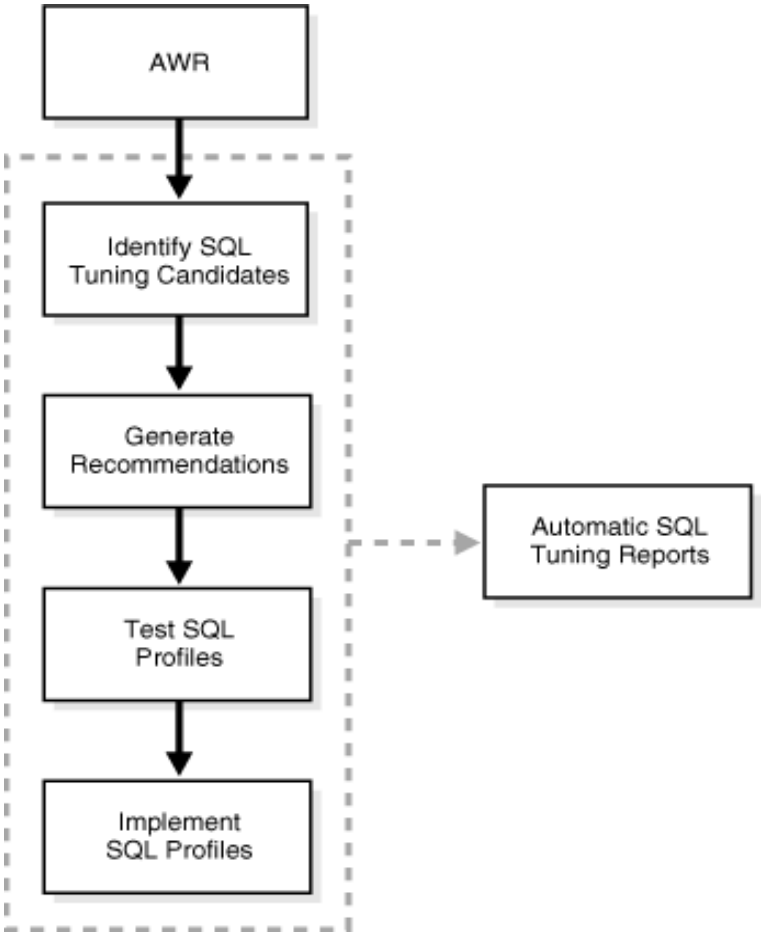
- Built on features available in 10g
  - ADDM
  - SQL Profiles
  - SQL Tuning Advisor
  - Maintenance Window
- ***Automatically* solves only problems related to cardinality/selectivity estimates or optimizer goal**
  - Does not fix poorly written SQL or poorly architected schemas
  - Does not create indexes
  - Does not gather statistics
- Requires Tuning Pack (extra licensing cost)

Automatic SQL Tuning Advisor



- Runs nightly
  - Scheduling handled by Automated Maintenance Task (AUTOTASK) framework
    - Uses scheduler's Maintenance Window
    - Uses DEFAULT_MAINTENANCE_PLAN (25% CPU)
    - DBA_AUTOTASK_* views and DBMS_AUTO_TASK_ADMIN package
  - By default job is enabled on new installs, disabled on upgrades
- Configurable
  - DBA_ADVISOR_* views and DBMS_SQLTUNE package
  - ACCEPT_SQL_PROFILE, MAX_SQL_PROFILES_PER_EXEC, MAX_AUTO_SQL_PROFILES, EXECUTION_DAYS_TO_EXPIRE, TIME_LIMIT, LOCAL_TIME_LIMIT, TEST_EXECUTE, etc.
- Can automatically implement plans with 3x improvement

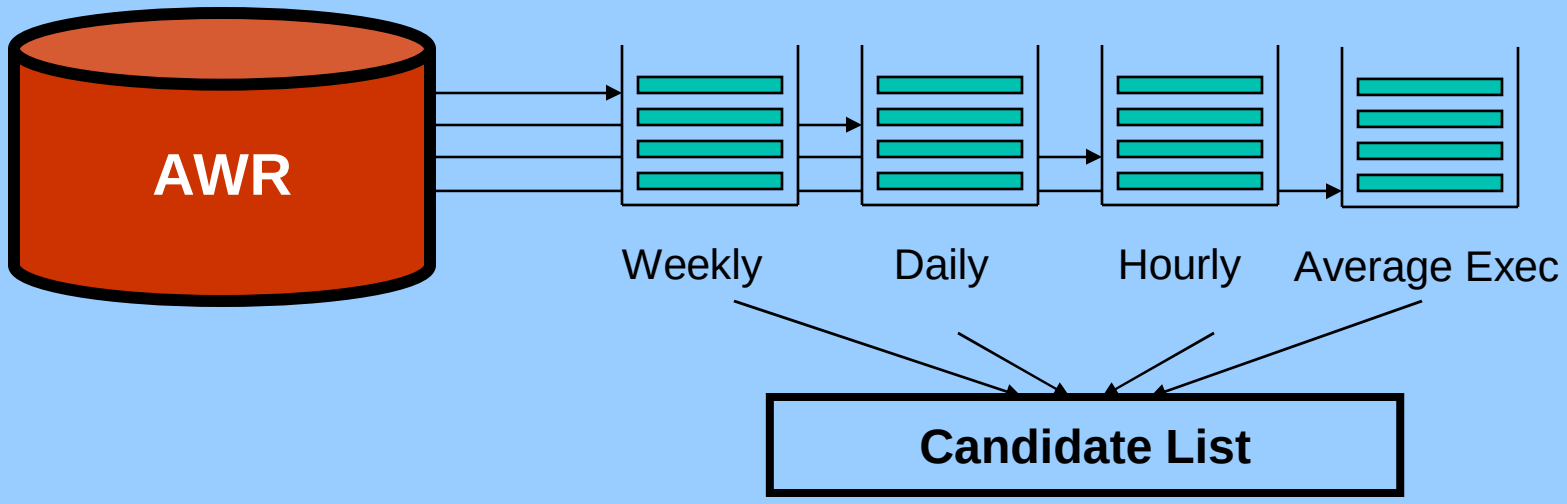
Automatic SQL Tuning Advisor



1. Identify candidates for SQL Tuning
2. Tune each statement individually by calling the SQL Tuning Advisor
3. Test SQL Profiles by executing the SQL statement
4. Optionally, automatically implement SQL Profiles with 3x improvement

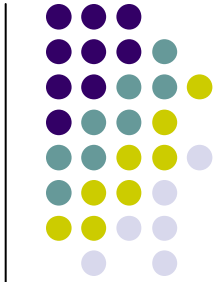
Automatic SQL Tuning Advisor

- Picking candidate SQL



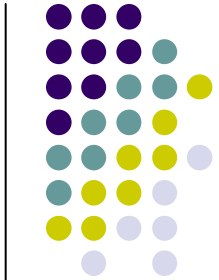
1. **Pull the top queries from the past week into four buckets:**
   - **Top for the past week**
   - **Top for any day in the past week**
   - **Top in any hour (single snapshot)**
   - **Top by average single execution**
2. **Combine four buckets into one, assigning weights**
3. **Cap at 150 queries per bucket**

Automatic SQL Tuning Advisor



- Some SQL is ineligible for **automatic** tuning
  (they can still be manually submitted to the SQL Tuning Advisor)
  - Parallel queries
  - Ad-hoc/rarely repeated queries (not repeated within a week)
  - Long-running queries
  - Recursive SQL
  - DML (insert/update) or DDL (create table as select)
  - Statements that were recently processed (within the past month)

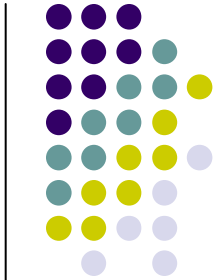
SQL Plan Management



- Next generation of Stored Outlines *(Outlines are officially deprecated in 11g but can be converted to baselines)*
  - CONTROLS plan evolution
  - GURANTEES plan stability
- Works hand-in-hand with Automatic SQL Tuning Advisor

- Optimizer remembers SQL Plans
  - Only known and verified plans are used
  - Plan changes can be tested and verified automatically or manually
  - Actually runs the statement to verify execution – evaluates real-world performance
  - Plans can be transported between databases (e.g. QA -> Prod)

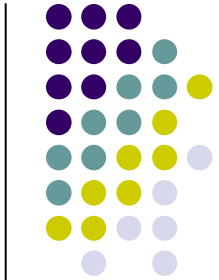
SQL Plan Management



Protects against execution plan changes in many situations

- Database Upgrades
  - Use OPTIMIZER_FEATURES_ENABLE
- System and Data Changes
  - Object or System Statistics
  - Session or System Parameters
  - Schema Changes (e.g. add index)
- Deployment of new application module
  - Can import plans that were pre-verified on a test system

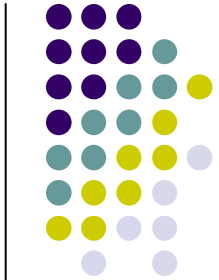
SQL Plan Management



SQL Management Base (SMB)

- Part of data dictionary, resides in SYSAUX
- Stores SQL-related data
  - Statement Log
  - Plan Histories
  - SQL Plan Baselines *(Note: outlines are moved to SYSAUX on upgrade)*
  - SQL Profiles
- Purge task runs weekly during maintenance window
  - Disk Space Quota: *default 10% of SYSAUX, can be 1-50%*
  - Plan Retention: *default 53 weeks since last use*
- Configurable
  - DBA_SQL_MANAGEMENT_CONFIG view and DBMS_SPM package

SQL Plan Management



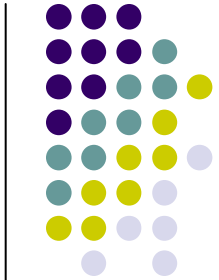
- OPTIMIZER_USE_SQL_PLAN_BASELINES = TRUE|FALSE
  - When enabled, new plans are *not used before verification.*
  - Automatic plan verification requires SQL Tuning Pack License!
  - Enabled by default! (Sites w/o Tuning Pack may want to disable)

- Baselines can be “Fixed”
  - Even with automatic verification, these plans cannot change without DBA review

- Displayed with DBMS_XPLAN package

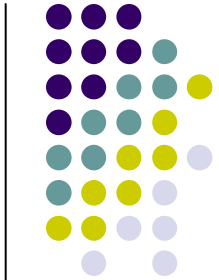
```
select * from table(dbms_xplan.display_sql_plan_baseline(
      sql_handle=>'SYS_SQL_209d10fabbedc741',format=>'basic‘
      ));
```

SQL Plan Management



- Automatic Plan Capture
  - OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
    (like CREATE_STORED_OUTLINES but w/o category)
  - Can be enabled at System or Session level
  - Optimizer stores plan history
    - SQL text
    - Outline
    - Bind Variables
    - Compilation Environment
  - Initial plan is always “accepted” but subsequent plans must be verified
- Manual Plan Loading
  - Plans can be manually loaded into the SMB from SQL Tuning Sets, the AWR, or the Cursor Cache with DBMS_SPM package

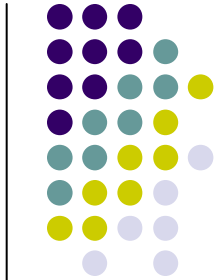
SQL Plan Management



- Evolving Plans Manually
  - When loading plans manually they can be automatically marked as verified
- Evolving Plans Automatically
  - Can verify and evolve one plan or several plans or all queued plans

```
SET SERVEROUTPUT ON
SET LONG 10000
DECLARE
    report clob;
BEGIN
    report := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
        sql_handle => 'SYS_SQL_593bc74fca8e6738');
    DBMS_OUTPUT.PUT_LINE(report);
END;
/
```

SQL Plan Management



```
-------------------------------------------------------------------------------
                         Evolve SQL Plan Baseline Report
-------------------------------------------------------------------------------
Inputs:
-------
SQL_HANDLE = SYS_SQL_593bc74fca8e6738
PLAN_NAME  =
TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
VERIFY     = YES
COMMIT     = YES

Plan: SYS_SQL_PLAN_ca8e6738a57b5fc2
-----------------------------------
Plan was verified: Time used .07 seconds.
Passed performance criterion: Compound improvement ratio >= 7.32.
Plan was changed to an accepted plan.
                  Baseline Plan Test Plan Improv. Ratio
                  ------------- --------- -------------
Execution Status:      COMPLETE  COMPLETE
Rows Processed:              40        40
Elapsed Time(ms):            23         8          2.88
CPU Time(ms):                23         8          2.88
Buffer Gets:                450        61          7.38
Disk Reads:                   0         0
Direct Writes:                0         0
Fetches:                      0         0
Executions:                   1         1
-------------------------------------------------------------------------------
Report Summary
-------------------------------------------------------------------------------
Number of SQL plan baselines verified: 1.
Number of SQL plan baselines evolved: 1.
```

Q&A

Questions, comments, suggestions?