The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remain at the sole discretion of Oracle.
Agenda

- Oracle 11g Database Overview
- Top New Features
- Summary
- Q & A
Overview
Enterprise Grid Computing

- SMP Dominance
- RAC Clusters for Availability
- Grids of low cost hardware and storage
Database Background Processes

• In Version 6 we had 5 Basic Background Processes
  • PMON
  • SMON
  • DBWR
  • LGWR
  • ARCH

• In Oracle 11g we have > 50 background processes!
Binary Sizes

Oracle Executable Binary Sizes For 32-bit Linux

- 8.1.7.4 26M
- 9.0.1 37M
- 9.2.0.4 49M
- 10.2.0.3 91M
- 11.1.0.4 124M
Self Managing Database

Auto-Tuning

Advisory

Instrumentation

Storage  Backup  Memory  Apps/SQL  Schema  RAC  Recovery  Replication
Self Managing Database

Auto-Tuning

Advisory

Instrumentation

Storage  Backup  Memory  Apps/SQL  Schema  RAC  Recovery  Replication

NYOUG 2007 - Daniel T. Liu
Automatic Memory Management
Automatic Memory Management

```
MEMORY_MAX_SIZE
MEMORY_TARGET

SGA_MAX_SIZE

SGA_TARGET

PGA_AGGREGATE_TARGET

SHARED_POOL_SIZE
DB_CACHE_SIZE
LARGE_POOL_SIZE
JAVA_POOL_SIZE
STREAMS_POOL_SIZE

DB_KEEP_CACHE_SIZE
DB_RECYCLE_CACHE_SIZE
DB_Nk_CACHE_SIZE

LOG_BUFFER
RESULT_CACHE_SIZE

OTHERS

MECHANICAL_KEEP_SIZE
MECHANICAL_RECYCLE_SIZE
MECHANICAL_Nk_SIZE

MECHANICAL_KEEP_SIZE
MECHANICAL_RECYCLE_SIZE
MECHANICAL_Nk_SIZE
```
Automatic Diagnostic Repository
Automatic Diagnostic Repository

- Automatic Diagnostic Repository (ADR) is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more.
- The default location for all trace information is defined by DIAGNOSTIC_DIST
ADR Repository File Structure

- In Oracle Database 11g, the directory structure as follow:

/u01/app/oracle/diag/rdbms/DatabaseName/
InstanceName/trace
   /alert
   /cdump
   /incident
   /incpkg
   /hm
   /metadata
   /(others)
ADR Repository File Structure

- The **TRACE** directory contains text alert log and background/foreground process trace files.
- The **ALERT** directory contains an XML version of the alert log.
- The **CDUMP** directory contains all core dump files.
- The **INCIDENT** directory contains multiple subdirectories, where each subdirectory is named for a particular incident, and where each contains dumps pertaining only to that incident.
- The **INCPKG** directory contains a collection of metadata to be ready to upload the data to Oracle Support Services.
- The **HM** directory contains the checker run reports generated by the Health Monitor.
- The **METADATA** directory contains important files for the repository itself.
Invisible Index
Invisible Index

• An invisible index is an index that is ignored by the optimizer unless you explicitly set the `OPTIMIZER_USE_INVISIBLE_INDEXES` initialization parameter to `TRUE` at the session or system level. The default value for this parameter is `FALSE`.

• Making an index invisible is an alternative to making it unusable or dropping it. Using invisible indexes, you can do the following:
  • Test the removal of an index before dropping it.
  • Use temporary index structures for certain operations or modules of an application without affecting the overall application.
Invisible Index

• Here are a few examples:

```
SQL> alter index emp_id_idx invisible;
SQL> alter index emp_id_idx visible;
SQL> create index emp_id_idx on emp (emp_id) invisible;
```
SQL Query Result Cache
Data Warehouse Workload

- Analyze data across large data sets
  - reporting
  - forecasting – trend analysis
  - data mining
- Use parallel execution for good performance
- Result
  - very IO intensive workload – direct reads from disk
  - memory is less important
    - mostly execution memory
Data Warehouse Query Example

```sql
select 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)
```

• accesses very many rows
• returns few rows
Data Warehouse Configuration Sizing

- Critical success factors
  - IO throughput
    - number of physical disks
    - number of channels to disks
  - CPU power
- Everything else follows
  - Storage capacity (500GB – 1TB common)
    - use surplus for high availability and ILM
  - Memory capacity (4GB/CPU is “standard”)
    - use surplus for... RESULT CACHE
**SQL Query Result Cache Benefits**

- Caches results of queries, query blocks, or pl/sql function calls
- Read consistency is enforced
  - DML/DDL against dependent database objects invalidates cache
- Bind variables parameterize cached result with variable values
SQL Query Result Cache

Enabling

- `result_cache_mode` initialization parameter
  - MANUAL, use hints to populate and use
  - FORCE, queries will use cache without hint

- `result_cache_max_size` initialization parameter
  - default is dependent on other memory settings
    - (0.25% of `memory_target` or 0.5% of `sga_target` or 1% of `shared_pool_size`)
  - 0 disables result cache
  - never >75% of shared pool (built-in restriction)

- `/*+ RESULT_CACHE */` hint in queries
SQL Query Result Cache

Example

• Use RESULT_CACHE hint

```sql
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-yyy')
   and to_date('31-DEC-2006','dd-MON-yyy')
group by rollup (p.prod_category)
```
### SQL Query Result Cache Example

- **Execution plan fragment**

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>RESULT CACHE</td>
<td>fz6cm4jbpcwh48wcyk60m7qypu</td>
</tr>
<tr>
<td>2</td>
<td>SORT GROUP BY ROLLUP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PARTITION RANGE ITERATOR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>SALES</td>
</tr>
<tr>
<td>6</td>
<td>VIEW</td>
<td>index$_join$_001</td>
</tr>
<tr>
<td>7</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>INDEX FAST FULL SCAN</td>
<td>PRODUCTS_PK</td>
</tr>
<tr>
<td>9</td>
<td>INDEX FAST FULL SCAN</td>
<td>PRODUCTS_PROD_CAT_IX</td>
</tr>
</tbody>
</table>
SQL Query Result Cache Opportunity

- Depends... based on
  - query repetitiveness
  - query execution times
  - DML activity (cache invalidation frequency)
- Remember data warehouse workload
  - query may run 30 minutes
  - query may return 5 rows
  - query served from result cache would take split second
Oracle Database 11g – Flashback Data Archive
Data History and Retention

- Data retention and change control requirements are growing
  - Regulatory oversight and Compliance
    - Sarbanes-Oxley, HIPAA, Basel-II, Internal Audit
  - Business needs
    - Extract “temporal” dimension of data
    - Understand past behavior and manage customer relationships profitably
- Failure to maintain appropriate history & retention is expensive
  - Legal risks
  - Loss of Reputation
- Current approaches to manage historical data are inefficient and often ineffective
Data History and Retention - Requirements

- Historical data needs to be secure and tamper proof
  - Unauthorized users should not be able to access historical data
  - No one should be able to update historical data
- Easily accessible from existing applications
  - Seamless access
  - Should not require special interfaces or application changes
- Minimal performance overhead
- Optimal Storage footprint
  - Historical data volume can easily grow into hundreds of terabytes
- Easy to set up historical data capture and configure retention policies
Managing Data History – Current Approaches

• Application or mid-tier level
  • Combines business logic and archive policies
  • Increases complexity
  • No centralized management
  • Data integrity issues if underlying data is updated directly
• Database level
  • Enabled using Triggers
  • Significant performance and maintenance overhead
• External or Third-party
  • Mine redo logs
  • History stored in separate database
  • Cannot seamlessly query OLTP and history data
• None of the above approaches meet all customer requirements
  • Customers are therefore forced to make significant compromises
Introducing Flashback Data Archive

- New feature in Oracle Database 11g
- Transparently tracks historical changes to all Oracle data in a highly secure and efficient manner
  - Historical data is stored in the database and can be retained for as long as you want
  - Special kernel optimizations to minimize performance overhead of capturing historical data
  - Historical data is stored in compressed form to minimize storage requirements
  - Automatically prevents end users from changing historical data
- Seamless access to archived historical data
  - Using “AS OF” SQL construct

```
select * from product_information AS OF TIMESTAMP '02-MAY-05 12.00 AM' where product_id = 3060
```
Introducing Flashback Data Archive

- Extremely easy to set up
  - Enable history capture in minutes!
- Completely transparent to applications
- Centralized and automatic management
  - Policy-based
  - Multiple tables can share same Retention and Purge policies
  - Automatic purge of aged history

Automatically Purge Data based on Retention policy
How Does Flashback Data Archive Work?

- Primary source for history is the undo data
- History is stored in automatically created history tables inside the archive
- Transactions and its undo records on tracked tables marked for archival
  - Undo records not recycled until history is archived
- History is captured asynchronously by new background process (fbda)
  - Default capture interval is 5 minutes
  - Capture interval is self-tuned based on system activities
  - Process tries to maximize undo data reads from buffer cache for better performance
  - INSERTs do not generate history records
Flashback Data Archive And DDLs

- Possible to add columns to tracked tables
- Automatically disallows any other DDL that invalidates history
  - Dropping and truncating a tables
  - Dropping or modifying a column
- Must disable archiving before performing any major changes
  - Disabling archiving discards already collected history
- Flashback Data Archive guarantees historical data capture and maintenance
  - Any operations that invalidates history or prevents historical capture will be disallowed
Historical Data Storage

• A new database object, flashback data archive, is a logical container for storing historical information
• Consists of one or more tablespaces
  - ‘QUOTA’ determines max amount of space a flashback data archive can use in each tablespace (default is Unlimited)
• Specify duration for retaining historical changes using ‘RETENTION’ parameter
• Tracks history for one or more tables
  - Tables should share the archiving characteristics
• Automatically purges aged-out historical data based on retention policy
• Create as many flashback data archives as needed
  - Group related tables by desired retention period
    - HIPAA requires all health transactions be maintained for 6 years
Creating Flashback Data Archive &
Enable History Tracking

1. Create tablespace (ASSM is required)
2. Create a flashback data archive
   - Set the retention period
     CREATE FLASHBACK ARCHIVE fda1 TABLESPACE tbs1 RETENTION 5 YEAR;
3. Enable archiving on desired tables
   ALTER TABLE EMPLOYEES FLASHBACK ARCHIVE fda1;
Managing Flashback Data Archive

- Static data dictionary views
  - *
    FLASHBACK_ARCHIVE - Displays information about Flashback Data Archives.
  - *
    FLASHBACK_ARCHIVE_TS - Displays tablespaces of Flashback Data Archives.
  - *
    FLASHBACK_ARCHIVE_TABLES - Displays information about tables that are enabled for flashback archiving.
- Alerts generated when flashback data archive is 90% full
- Automatically purges historical data after expiration of specified retention period
- Supports ad-hoc purge by administrators (privileged operation)
  - ALTER FLASHBACK ARCHIVE fla1 PURGE BEFORE
    TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' DAY);
Managing Flashback Data Archive

- SYS_FBA_HIST_* - Internal History Table
  - Replica of tracked table with additional timestamp columns
  - Partitioned for faster performance
  - No modifications allowed to internal partitions
  - Compression reduces disk space required
  - No out-of-box indexes
  - Support for copying primary key indexes from tracked table in later releases (TBD)

- Applications don’t need to access internal tables directly
  - Use ‘AS OF’ to seamlessly query history
Oracle Database 11g
Advanced Compression Option
Challenges

• Explosion in data volume managed by Enterprises
  • Government regulations (Sarbanes-Oxley, HIPPA, etc)
  • User generated content (Web 2.0)
• IT managers must support larger volumes of data with limited technology budgets
  • Need to optimize storage consumption
  • Also maintain acceptable application performance
• Intelligent and efficient compression technology can help address these challenges
Introducing Advanced Compression Option

- Oracle Database 11g introduces a comprehensive set of compression capabilities
  - Structured/Relational data compression
  - Unstructured data compression
  - Compression for backup data
  - Network transport compression
- Reduces resource requirements and costs
  - Storage System
  - Network Bandwidth
  - Memory Usage
Table Compression

- Introduced in Oracle9i Release 2
  - Supports compression during bulk load operations (Direct Load, CTAS)
  - Data modified using conventional DML not compressed
- Optimized compression algorithm for relational data
- Improved performance for queries accessing large amounts of data
  - Fewer IOs
  - Buffer Cache efficiency
- Data is compressed at the database block level
  - Each block contains own compression metadata – improves IO efficiency
  - Local symbol table dynamically adapts to data changes
- Compression can be specified at either the table or partition levels
- Completely transparent to applications
- Noticeable impact on write performance
OLTP Table Compression

• Oracle Database 11g extends compression for OLTP data
  • Support for conventional DML Operations (INSERT, UPDATE, DELETE)

• New algorithm significantly reduces write overhead
  • Batched compression ensures no impact for most OLTP transactions

• No impact on reads
  • Reads may actually see improved performance due to fewer IOs and enhanced memory efficiency
OLTP Table Compression

- Adaptable, continuous compression
- Compression automatically triggered when block usage reaches PCTFREE
- Compression eliminates holes created due to deletions and maximizes contiguous free space in block
OLTP Table Compression

Employee Table

<table>
<thead>
<tr>
<th>ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Doe</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>Doe</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>4</td>
<td>Jane</td>
<td>Doe</td>
</tr>
</tbody>
</table>

Initially Uncompressed Block

<table>
<thead>
<tr>
<th>Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>1・John・Doe 2・Jane・Doe 3・John・Smith 4・Jane・Doe</td>
</tr>
</tbody>
</table>

Free Space

```
INSERT INTO EMPLOYEE
VALUES (5, 'Jack', 'Smith');
COMMIT;
```
## OLTP Table Compression

### Employee Table

<table>
<thead>
<tr>
<th>ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Doe</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>Doe</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>4</td>
<td>Jane</td>
<td>Doe</td>
</tr>
<tr>
<td>5</td>
<td>Jack</td>
<td>Smith</td>
</tr>
</tbody>
</table>

### Compressed Block

**Header**

- John=0 | Doe=1 | Jane=2 | Smith=3

**Free Space**

- 1 5

### Local Symbol Table

- John
- Doe
- Jane
- Smith

NYOUG 2007 - Daniel T. Liu
OLTP Table Compression

Uncompressed Block

<table>
<thead>
<tr>
<th>Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>1•John•Doe 2•Jane•Doe 3•John•Smith 4•Jane•Doe 5•Jack•Smith</td>
</tr>
<tr>
<td>Free Space</td>
</tr>
</tbody>
</table>

Compressed Block

<table>
<thead>
<tr>
<th>Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>John=0</td>
</tr>
<tr>
<td>1•0•1 2•2•1 3•0•3 4•2 5•Jack•3</td>
</tr>
<tr>
<td>Free Space</td>
</tr>
</tbody>
</table>

Local Symbol Table

More Data Per Block
Using OLTP Table Compression

- Requires database compatibility level at 11.1 or greater
- New Syntax extends the ‘COMPRESS’ keyword
  - COMPRESS [FOR {ALL | DIRECT_LOAD} OPERATIONS]
  - DIRECT_LOAD (DEFAULT)
    - Refers to Bulk load operations from 10g and prior releases
    - ALL
      - OLTP + Direct loads
- Enable compression for new tables
  create table t1 compress for all operations
- Enable only direct load compression on existing table
  alter table t2 compress
  - Only new rows are compressed, existing rows are uncompressed
Data Pump Compression

- Metadata compression available since Oracle Database 10g
- Oracle Database 11g extends compression to table data during exports
  - No need to decompress before import
- Single step compression of both data and metadata
  - Compressed data directly hits disk resulting in reduced disk space requirements
  - 75% reduction in dump file size on export of sample OE and SH schemas
- Compression factor comparable to GNU gzip utility
- Application transparent
  - Complete Data Pump functionality available on compressed files
Backup data and Network transport Compression

• Fast RMAN Compression
  • Compresses the backup set contents before writing them to disk or tape
  • No extra decompression steps are required during recovery when you use RMAN compression.
  • High performance, industry standard compression algorithm
    • 40% faster backup compression versus Oracle Database 10g
  • Suitable for fast, incremental daily backups
  • Reduces network usage

• Data Guard Network Compression
  • Compression of redo traffic over the network
  • Improves redo transport performance
    • Gap resolution is up to 2x faster
Oracle Database 11g
Real Application Testing
Real Application Testing

- **Value**
  - Rapid technology adoption
  - Higher testing quality
- **Business Benefit**
  - Lower cost
  - Lower risk

**Solution for the Agile Business**
Database Replay
The Need for Database Replay

- Businesses want to adopt new technology that adds value
- Extensive testing and validation is expensive in time and cost
- Despite expensive testing success rate low
  - Many issues go undetected
  - System availability and performance negatively impacted
- Cause of low success rate
  - Current tools provide inadequate testing
    - Simulate synthetic workload instead of replaying actual production workload
    - Provide partial workflow coverage

Database Replay makes real-world testing possible
Database Replay

• Replay actual production database workload in test environment
• Identify, analyze and fix potential instabilities before making changes to production

• Capture Workload in Production
  • Capture full production workload with real load, timing & concurrency characteristics
  • Move the captured workload to test system

• Replay Workload in Test
  • Make the desired changes in test system
  • Replay workload with full production characteristics
  • Honor commit ordering

• Analyze & Report
  • Errors
  • Data divergence
  • Performance divergence

Analysis & Reporting
Comparison of LoadRunner & DB Replay
Testing e-Business Suite

<table>
<thead>
<tr>
<th>Step</th>
<th>LoadRunner</th>
<th>DB Replay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install &amp; Setup</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Understand Application Usage</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Identify Key Transactions</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Generate Workload</td>
<td>80</td>
<td>5</td>
</tr>
<tr>
<td>Run Test</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Total Testing Time**
- DB Replay: 2 weeks
- LoadRunner: 30 weeks
Why DB Replay?

From:
- Artificial workloads
- Partial workflows
- Months of development
- Manual intensive
- High risk

To:
- Production workloads
- Complete workflows
- Days of development
- Automated
- Low risk

150 Days → 10 Days
Step 1: Workload Capture

- All external client requests captured in binary files
- System background, internal activity excluded
- Minimal performance overhead for capture
- For RAC, shared and local file system supported
- Specify interesting time period for capture, e.g., peak workload, month-end processing, etc.
- Can capture on 10.2.0.4 and replay on 11g
Capture Options

- Workload can be filtered to customize what is captured
  - Filter Types
    - Inclusion Filters: Specifies which sessions should be captured
    - Exclusion Filters: Specifies which sessions should NOT be captured
  - Filter Attributes: Workload capture can be filtered using any of the following session attributes
    - User
    - Program
    - Module
    - Action
    - Service
    - Session ID
- Workload capture can be run on-demand or scheduled to run at later time
Step 2: Process Workload Files

- Setup test system
  - Test DB is at same point in time as before production capture
  - Use RMAN to physically restore production db from backup
  - Use Snapshot standby
  - Use imp/exp, Data Pump, etc.
- Processing transforms captured data into replayable format
- Once processed, workload can be replayed many times
- For RAC copy all capture files to single location for processing
Step 3: Replay Workload

- Replays workload preserving timing, concurrency and dependencies of the capture system

- Replay Driver is a special client program that consumes processed workload and sends requests to the replay system

- Replay Driver consists of one or more clients. For workloads with high concurrency, it may be necessary to start multiple clients to drive workload
Replay Options

• **Synchronized Replay (Default)**
  • Workload is replayed in full synchronized mode
  • Exact same concurrency and timing as production workload
  • Transaction commit order is honored
  • Ensures minimal data divergence

• **Unsynchronized Replay**
  • Workload can be replayed in unsynchronized mode
  • Useful for load/stress testing
  • High data divergence
  • Three (3) parameters provided to control degree of synchronization
    • Think time synchronization
    • Connect (logon) time synchronization
    • Commit order synchronization
Analysis & Reporting

• Comprehensive reports are provided for analysis purposes
• There (3) types of divergences are reported
  • Data Divergence: Number of rows returned by each call are compared and divergences reported
  • Error Divergence: For each call error divergence is reported
    • New: Error encountered during replay not seen during capture
    • Not Found: Error encountered during capture not seen during replay
    • Mutated: Different error produced in replay than during capture
  • Performance Divergence
    • Capture and Replay Report: Provides high-level performance information
    • ADDM Report: Provides in-depth performance analysis
    • AWR, ASH Report: Facilitates comparative or skew analysis
Database Replay Summary Report

View Workload Replay: REPLAY3

Elapsed Time Comparison

Assessing the Replay

The Elapsed Time Comparison chart shows how much time the replayed workload has taken to accomplish the same amount of work as captured.

When the replay bar is shorter than the capture bar then the replay environment is processing the workload faster than the capture environment.

The divergence table gives information about both the data and error discrepancies between the replay and capture environments, which can be used as a measure of the replay quality.

NYOUG 2007 - Daniel T. Liu
Performance Page – Database Replay

A: Original Workload
B: Capture Workload
C: Replay Workload
(B,C): Include automatic Snapshots/AWR export at the end of capture/replay
Top Activity Page: Database Replay

A: Original Workload
B: Capture Workload
C: Replay Workload
(B, C): Include automatic Snapshots/AWR export at end of capture/replay
Best Practices

• Capture
  • Provide adequate disk space for captured workload (binary files)
  • Database restart (Optional): Recommended to minimize divergence
  • For RAC, use shared file system

• Test System Setup
  • Ensure data in test is identical to production as of capture start time to minimize data divergence during replay
  • Use RMAN backup/restore or Snapshot Standby to setup test system
  • For performance analysis test system capacity should be similar to production
  • Reset system clock to same time as production if application logic involves SYSDATE usage

• Process Workload
  • Processing workload has performance overhead and can possibly take a long time
  • Process workload on test system instead of production

• Replay
  • Use Client Calibration Advisor to identify number of replay clients needed to replay workload properly
SQL Performance Analyzer (SPA)
The Need for SQL Performance Analyzer (SPA)

- Businesses want systems that are performant and meet SLA’s
- SQL performance regressions are #1 cause of poor system performance
- Solution for proactively detecting all SQL regressions resulting from changes not available
- DBA’s use ineffective and time-consuming manual scripts to identify problems

SPA identifies all changes in SQL performance before impacting users
Why SQL Performance Analyzer?

From:
- Manual workload creation
- Synthetic workload
- Months of manual analysis
- Partial workload
- High risk

To:
- Automated workload capture
- Production workload
- Automated analysis in minutes
- Complete workload
- Low risk
SQL Performance Analyzer

- Test impact of change on SQL query performance
- Capture SQL workload in production including statistics & bind variables
- Re-execute SQL queries in test environment
- Analyze performance changes – improvements and regressions
SPA Benefits

• Enables identification of SQL performance regressions before end-users can be impacted

• SPA can help with any change that impacts SQL execution plan
  • DB upgrades
  • Optimizer statistics refresh
  • New indexes, Materialized Views, Partitions, etc.

• Automates SQL performance tracking of hundreds of thousands of SQL statements – impossible to do manually

• Captures SQL workload with low overhead

• Integrated with SQL Tuning Advisor and SQL Plan Baselines for regression remediation
SQL Performance Analyzer Workflow

Production (10.2)

Clients

Mid-Tier

Storage

Test (11.1)

Storage

Capture SQL

Transport SQL

Execute SQL Pre-change

Execute SQL Post-change

Compare Perf

NYOUG 2007 - Daniel T. Liu
Step 1: Capture SQL Workload

- SQL Tuning Set (STS) used to store SQL workload
- STS includes:
  - SQL Text
  - Bind variables
  - Execution plans
  - Execution statistics
- Incremental capture used to populate STS from cursor cache over a time period
- SQL tuning set’s filtering and ranking capabilities filters out undesirable SQL
- SQL workload captured in 10.2.0.1 and higher can be used for SPA tasks in 11g
Step 2: Move SQL Workload to Test System

- Copy SQL tuning set to staging table (“pack”)
- Transport staging table to test system (datapump, db link, etc.)
- Copy SQL tuning set from staging table (“unpack”)

Production Database

Export/Import

Test Database
Step 3: Execute SQL Before Making Change

- Establishes SQL workload performance baseline
- SQL execution plan and statistics captured
- SQL executed serially (no concurrency)
- Each SQL executed only once
- DDL/DML skipped
- Option to do Explain Plan only analysis
Step 4: Execute SQL After Making Change

- **Manually implement the planned change**
  - Database upgrade, patches
  - Optimizer statistics refresh
  - Schema changes
  - Database parameter changes
  - Tuning actions, e.g., SQL Profile creation

- **Re-execute SQL after change**
  - Gathers new SQL execution plans and statistics
Step 5: Compare & Analyze Performance

- Compare performance using different metrics, e.g.,
  - Elapsed Time
  - CPU Time
  - Optimizer Cost
  - Buffer Gets

- SPA Report shows impact of change for each SQL
  - Improved SQL
  - Regressed SQL
  - Unchanged SQL

- Fix regressed SQL using SQL Tuning Advisor or SQL Plan Baselines
**SPA Report**

**SQL Performance Analyzer Task Result: SYS.SYSTEMCHANGES1**

- **Task Name**: SYSTEMCHANGES1
- **Task Owner**: SYS
- **SQL Tuning Set Name**: HR_WORKLOAD
- **STS Owner**: APPS
- **Replay Trial 1**: Before
- **Replay Trial 2**: After
- **Total SQL Statements**: 50
- **SQL Statements With Errors**: 0

**Global Statistics**

**Projected Workload Execute Elapsed Time**

- Before: 80
- After: 20

**Change in Execute Elapsed Time**

- Improvement Impact: 69%↑
- Regression Impact: -5%↓
- Overall Impact: 85%↑

**Recommendations**

- Run SQL Tuning Advisor to tune regressed SQL statements.
- Schedule SQL Tuning Advisor.

*NYOUG 2007 - Daniel T. Liu*
SPA Report – Regressed SQL Statements

Regressed Plan Changed SQL Statements

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>Executions</th>
<th>Net Impact on Workload (%)</th>
<th>Execute Elapsed Time</th>
<th>Net Impact on SQL (%)</th>
<th>% of Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2wtxbj6u2by</td>
<td>1</td>
<td>-4.390</td>
<td>2.004</td>
<td>4.697</td>
<td>-134.380</td>
</tr>
<tr>
<td>654x8xs5wpz</td>
<td>1</td>
<td>-0.020</td>
<td>0.000</td>
<td>0.015</td>
<td>-1,500.000</td>
</tr>
</tbody>
</table>

SQL Details: 2wtxbj6u2by

Execution Frequency: 1

Schedule SQL Timing Advisor

Problem Findings
The performance of this SQL has regressed.

Symptom Findings
The structure of the SQL execution plan has changed.

Information Findings

Plan Comparison

Before_change
Plan Hash Value: 1511732424

After_change
Plan Hash Value: 3517011486

ORACLE
Best Practices

- **Workload Capture**
  - Use incremental STS capture
  - Peak or representative workloads can be more accurately captured with this mechanism

- **Test system**
  - Run SPA on test system instead of production as analysis can be resource intensive
  - Ensure test system has similar configuration and comparable optimizer statistics as production

- **Performance comparison**
  - Use several different metrics, e.g., elapsed time, CPU time, etc., to compare pre- and post-change performance for reliable results

- **Regression remediation**
  - Use SQL Tuning Advisor and SQL Plan Baselines to fix regressions
Oracle Partitioning in Oracle Database 11g
## Oracle Partitioning

### Ten Years of Development

<table>
<thead>
<tr>
<th></th>
<th>Core functionality</th>
<th>Performance</th>
<th>Manageability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle8</td>
<td>Range partitioning</td>
<td>“Static” partition pruning</td>
<td>Basic maintenance operations: add, drop, exchange</td>
</tr>
<tr>
<td></td>
<td>Global range indexes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle8i</td>
<td>Hash and composite range-hash partitioning</td>
<td>Partition-wise joins “Dynamic” pruning</td>
<td>Merge operation</td>
</tr>
<tr>
<td>Oracle9i</td>
<td>List partitioning</td>
<td></td>
<td>Global index maintenance</td>
</tr>
<tr>
<td>Oracle9i R2</td>
<td>Composite range-list partitioning</td>
<td>Fast partition split</td>
<td></td>
</tr>
<tr>
<td>Oracle10g</td>
<td>Global hash indexes</td>
<td></td>
<td>Local Index maintenance</td>
</tr>
<tr>
<td>Oracle10g R2</td>
<td>1M partitions per table</td>
<td>“Multi-dimensional” pruning</td>
<td>Fast drop table</td>
</tr>
<tr>
<td>Oracle Database 11g</td>
<td>More composite choices REF Partitioning Virtual Column Partitioning</td>
<td></td>
<td>Interval Partitioning Partition Advisor</td>
</tr>
</tbody>
</table>
Oracle Partitioning Enhancements

• Complete the basic partitioning strategies defines *HOW data is going to be partitioned*
  • new composite partitioning methods

• Introduce partitioning extensions defines *WHAT controls the data placement*
  • enhance the manageability and automation
  • virtual column based partitioning
  • REF partitioning
  • interval partitioning
  • partition advisor
Composite Partitioning in Oracle Database 11g
Extended Composite Partitioning Strategies

- Concept of composite partitioning
  - Data is partitioned along two dimensions (A,B)
  - A distinct value pair for the two dimensions uniquely determines the target partitioning
- Composite partitioning is complementary to multi-column range partitioning
- Extensions in Oracle Database 11g

<table>
<thead>
<tr>
<th>New 11g Strategy</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>List – Range</td>
<td>Geography - Time</td>
</tr>
<tr>
<td>Range - Range</td>
<td>ShipDate - OrderDate</td>
</tr>
<tr>
<td>List - Hash</td>
<td>Geography - OrderID</td>
</tr>
<tr>
<td>List - List</td>
<td>Geography - Product</td>
</tr>
</tbody>
</table>
Composite Partitioning - Concept

Table SALES
RANGE(order_date)-RANGE(ship_date)

ship_date

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Composite Partitioning - Concept

Table SALES
RANGE(order_date)-RANGE(ship_date)

• All records with order_date in March 2006
Composite Partitioning - Concept

Table SALES
RANGE(order_date)-RANGE(ship_date)

- All records with ship_date in May 2006

Jan 2006
Feb 2006
May 2006


order_date
Composite Partitioning - Concept

Table SALES
RANGE(order_date)-RANGE(ship_date)

- All records with order_date in March 2006 AND ship_date in May 2006
Virtual Column based Partitioning
Virtual Columns

Business Problem

• Extended Schema attributes are fully derived and dependent on existing common data
• Redundant storage or extended view definitions are solving this problem today
  • requires additional maintenance and creates overhead

Solution

• Oracle Database 11g introduces virtual columns
  • purely virtual, meta-data only
• Treated as real columns except no DML
  • can have statistics
  • eligible as partitioning key
• Enhanced performance and manageability
Virtual Columns - Example

• Base table with all attributes ...

```
CREATE TABLE accounts
(acc_no number(10) not null,
acc_name varchar2(50) not null, ...)
```

<table>
<thead>
<tr>
<th>acc_no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>12500</td>
<td>Adams</td>
</tr>
<tr>
<td>12507</td>
<td>Blake</td>
</tr>
<tr>
<td>12666</td>
<td>King</td>
</tr>
<tr>
<td>12875</td>
<td>Smith</td>
</tr>
</tbody>
</table>
Virtual Columns - Example

- Base table with all attributes ...
  - ... is extended with the virtual (derived) column

```
CREATE TABLE accounts
  (acc_no number(10) not null,
   acc_name varchar2(50) not null, ...
   acc_branch number(2) generated always as
     (to_number(substr(to_char(acc_no),1,2)))
```

<table>
<thead>
<tr>
<th>acc_no</th>
<th>acc_name</th>
<th>acc_branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>12500</td>
<td>Adams</td>
<td>12</td>
</tr>
<tr>
<td>12507</td>
<td>Blake</td>
<td>12</td>
</tr>
<tr>
<td>12666</td>
<td>King</td>
<td>12</td>
</tr>
<tr>
<td>12875</td>
<td>Smith</td>
<td>12</td>
</tr>
</tbody>
</table>
Virtual Columns - Example

- Base table with all attributes ...
  - ... is extended with the virtual (derived) column
  - ... and the virtual column is used as partitioning key

```
CREATE TABLE accounts
(acc_no number(10) not null,
 acc_name varchar2(50) not null, ...
acc_branch number(2) generated always as
 (to_number(substr(to_char(acc_no),1,2)))
partition by list (acc_branch) ...
```

```
12500  Adams  12
12507  Blake  12
12666  King   12
12875  Smith  12
...
32320  Jones  32
32407  Clark  32
32758  Hurd  32
32980  Phillips  32
```
Interval Partitioning
Interval Partitioning

- Partitioning is key-enabling functionality for managing large volumes of data
  - one logical object for application transparency
  - multiple physical segments for administration

but
- Physical segmentation requires additional data management overhead
  - new partitions must be created on-time for new data

Automate the partition management
Interval Partitioning

• Interval Partitioning
  • extension to range partitioning
  • full automation for equi-sized range partitions
• Partitions are created as metadata information only
  • start partition is made persistent
• Segments are allocated as soon as new data arrives
  • no need to create new partitions
  • local indexes are created and maintained as well

No need for any partition management
Interval Partitioning

*How it works*

```
CREATE TABLE sales (order_date DATE, ...)
PARTITION BY RANGE (order_date)
INTERVAL(NUMTOYMINTERVAL(1,'month'))
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');
```

Table SALES

```
| Jan 2006 | Feb 2006 | Mar 2006 | ... | Jan 2007 | Oct 2009 | Nov 2009 | ...
```

First segment is created
Interval Partitioning

*How it works*

```
CREATE TABLE sales (order_date DATE, ...)
PARTITION BY RANGE (order_date)
INTERVAL (NUMTOYMINTERVAL(1, 'month'))
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');
```

Table SALES

|----------|----------|----------|----------|----------|----------|

Other partitions only exist in metadata
Interval Partitioning

*How it works*

CREATE TABLE sales (order_date DATE, ...)
PARTITION BY RANGE (order_date)
INTERVAL (NUMTOYMINTERVAL(1,'month'))
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');

Table SALES

|----------|----------|----------|----------|----------|----------|

New segment is automatically allocated

INSERT INTO sales (order_date DATE, ...)
VALUES ('04-MAR-2006',...);
Interval Partitioning

*How it works*

CREATE TABLE sales (order_date DATE, ...)
PARTITON BY RANGE (order_date)
INTERVAL (NUMTOYMINTERVAL(1,'month'))
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');

Table SALES

|----------|----------|----------|----------|----------|----------|

... whenever data for a new partition arrives

INSERT INTO sales (order_date DATE, ...)
VALUES ('17-OCT-2009',...);
Interval Partitioning

*How it works*

- Interval partitioned table can have classical range and automated interval section
  - Automated new partition management plus full partition maintenance capabilities: "Best of both worlds"

<table>
<thead>
<tr>
<th>Table SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 2006</td>
</tr>
</tbody>
</table>

*ORACLE*
Interval Partitioning

*How it works*

- Interval partitioned table can have classical range and automated interval section
  - Automated new partition management plus full partition maintenance capabilities: *“Best of both worlds”*

**Table SALES**

2006

Range partition section

MERGE and move old partitions for ILM

Interval Partitioning

*How it works*

- Interval partitioned table can have classical range and automated interval section
  - Automated new partition management plus full partition maintenance capabilities: *“Best of both worlds”*

Table **SALES**

- **Range partition section**
- **Interval partition section**
- **MERGE and move old partitions for ILM**
- **Insert new data**
  - **Automatic segment creation**
  
  ```sql
  INSERT INTO sales (order_date DATE, ...) VALUES ('13-NOV-2009',...);
  ```
REF Partitioning
REF Partitioning

Business Problem
- Related tables benefit from same partitioning strategy
  - e.g. order – lineitem
- Redundant storage of the same information solves this problem
  - data overhead
  - maintenance overhead

Solution
- Oracle Database 11g introduces REF Partitioning
  - child table inherits the partitioning strategy of parent table through PK-FK relationship
  - intuitive modelling
- Enhanced Performance and Manageability
Before REF Partitioning

Table ORDERS

- RANGE(order_date)
- Primary key order_id
- Redundant storage of order_date
- Redundant maintenance

Table LINEITEMS

- RANGE(order_date)
- Foreign key order_id
REF Partitioning

Table ORDERS

- RANGE(order_date)
- Primary key order_id

Table LINEITEMS

- RANGE(order_date)
- Foreign key order_id

PARTITION BY REFERENCE
- Partitioning key inherited through PK-FK relationship
Partitioning Advisor

- Considers entire query workload to improve query performance
- Advises on partitioning methods
  - Range (equal-interval), range key and interval
  - Hash, hash key
- Integrated, non-conflicting advice with Indexes, MVs

Well-tuned SQL & Schema
Thanks For Coming !!

Daniel Liu Contact Information
Phone: (714) 376-8416
Email: daniel.liu@oracle.com
Email: daniel_t_liu@yahoo.com

Company Web Site:
http://www.oracle.com