Optimizing Composer Multi-Platform Oracle Applications

Session 750

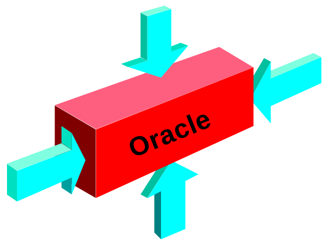
Rebecca Lawson
Texas Instruments

Overview

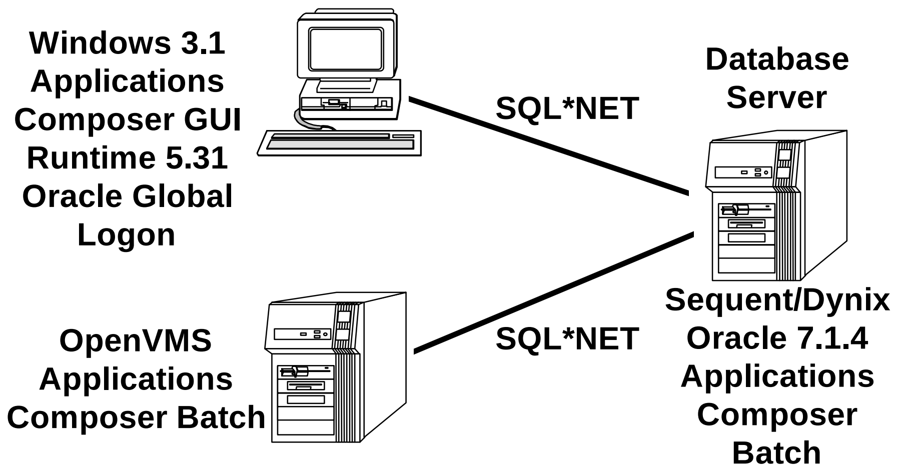
- Objective
- Technical environment
- Oracle tuning approach
- Summary

Objective

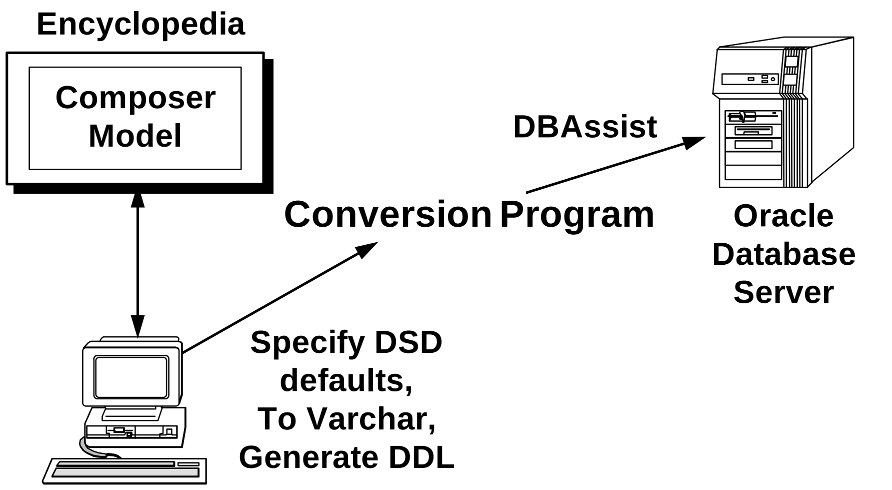
- To enable application developers to improve performance of Composer-generated Oracle applications targeting multiple platforms



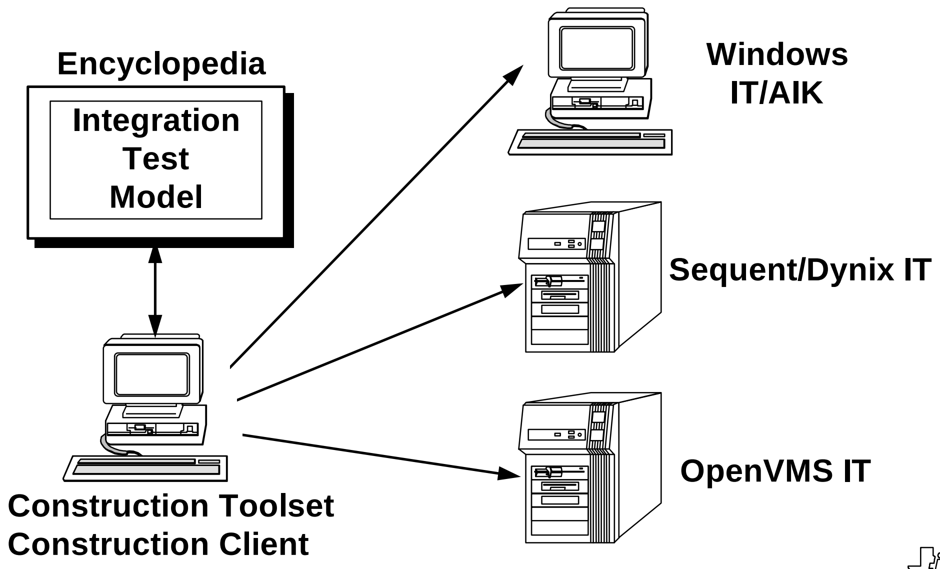
Technical Environment



Shared Database

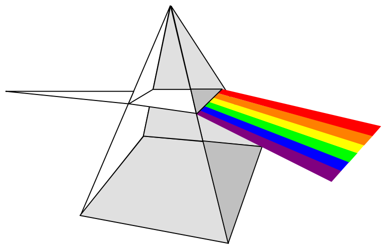


Generate & Install Code



Oracle Tuning Approach

- Installing and configuring Oracle
- ***Application design***
- ***Data access methods***
- Memory allocation tuning
- Tuning disk I/0
- CPU usage tuning
- Tuning resource contention



Installing and Configuring Oracle

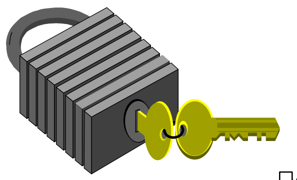
- For initial install, review Oracle default settings
- Review application parameters with system DBA
  - – Determine concurrent transactions–rollback segments

Application Design

- Data modeling guidelines
- Design considerations
- Construction options
- Assessing performance

Data Modeling Guidelines

- Modifying vs. Referencing relationships
  - Modifying – generates a *SELECT* for *UPDATE* when reading the table to ensure referential integrity on foreign key fields.
  - Referencing – generates *SELECT* without update. Referential integrity is ensured by the application. Can be used for optional relationships with low risk for updates in foreign key field.
- Minimize joins – Denormalization



Design Considerations

- Action diagrams
- Optimization of SQL statements
- Selecting the optimizer
- Technical design
- Using External Action Blocks (EABs)
- Distributed Processing vs. Remote Data Access (RDA)

Action Diagrams–GUI List Box

- READ EACH... SORTED BY
  - – May sort on table – index may be ignored unless all of the attributes in the sorted by clause are contained in a single index in the same sequence & are defined as not null in the DSD
- READ EACH...WHERE...SORTED BY
  - – Where clause forces new index path, resulting in table scan and sort
- READ EACH.....WHERE index column(s) > value
  - – Optimizer will choose index if the attribute defined in value is contained in an index in specified sequence (< or >)

Action Diagrams–Batch

- Persistent Views
  - – Used to reduce reads by maintaining currency on data
  - – Set persistent view *locked* only for update and delete
- Starve views – reference only required fields
- Reduce view matching
  - – Create group view with cardinality of one to match views between action diagrams instead of matching multiple views
- High performance view passing
- Checkpoint/restart logic to increase commit level or use EAB to issue commit

Optimization of SQL Statements

- Oracle optimizes all *SELECT, INSERT, UPDATE, and DELETE* statements
- Evaluates expressions and conditions containing constants
- Original statements may be transformed to equivalent joins
- Merges the view's query into the original statement, or the original statement into the view's query, then optimizes the result

Selecting the Optimizer

- Rule-Based – determines a retrieval path by applying a set of pre-programmed rules
- Cost-Based – uses physical table characteristics to perform the access path calculations
- Use EAB to dynamically set/reset optimizer

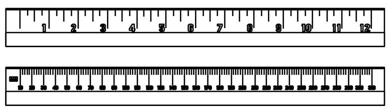
```
External _set_opt

** Insert user defined code here

EXEC SQL
SET optimizer_mode = choose
```

Rule-Based

- Fixed set of rules
- Oracle V6 – not planned for future Oracle releases
- Consistent results
- Rules can be coded in action diagrams
- Order of entity types in READ statements can be used to determine index used
  - Fewer row tables first



Cost-Based

- Three options available in cost-based:
  - *Choose* optimizer chooses optimal path
  - *First Row* optimizes for singleton select
  - *All Rows* optimizes for multiple rows returned
- Recommended by Oracle
- Varying set of rules by release
- Requires statistics

Technical Design

- Tune Oracle indexes
- Index design
- Sequential indexes
- Update generated DDL
- Define Oracle roles

Tuning Oracle Indexes

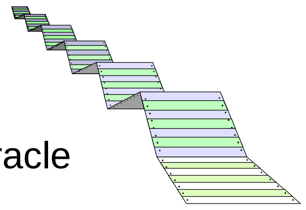
- Design optimal identifiers – short, numeric, unique
- Customize Entry Points (EPs)
  - Review entry points – add or delete
  - Reorder fields in records in data model
  - Reorder fields in entry point
- Customize generated indexes
  - Update default indexes defined for each identifier

Index Design

- User-designed
  - Create indexes to optimize access paths defined in READ statements
- Consider a composite instead of two inefficient indexes
  - Ex: Last name, first name combined instead of separate indexes on both fields
  - Oracle tries to combine indexes whenever possible during execution

Sequential Indexes

- Next sequential
  - – Code sets value in sequential order
- Oracle SEQUENCE
  - – Optimal method of using a sequential identifier
  - – Allow generation of numbers for system-assigned identifiers
  - – Sequence cache controlled in Oracle data dictionary
  - – Use EAB to retrieve value for sequence stored in database

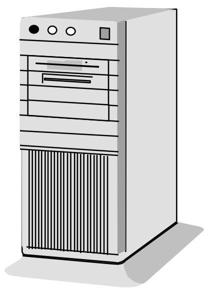


Update Generated DDL

- Split tables and indexes into separate tablespaces
- Add storage clause information based on entity type properties
  - – Minimum occurrences
  - – Maximum occurrences
  - – Average number of occurrences
  - – Expected growth rate
- Declarative data integrity constraints
- DBMS-enforced referential integrity

Defining Oracle Roles

- Use the Client/Server Encyclopedia to determine roles for Oracle application
- Query Public Interface views for each action diagram to determine entity actions

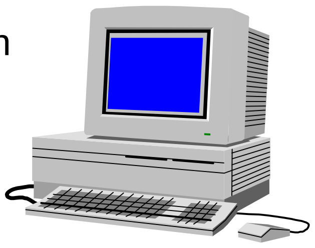


Using EABs

- Use EABs in a Composer application for:
  - Sequential key assignment
  - Set/reset optimizer goal
  - Optimize SQL – utilize HINTS
  - Array processing – read multiple rows
    - » Composer-generated Pro*C repeats single read
    - » Faster method is to read multiple rows into array

Distributed Processing vs. Remote Data Access

- Remote Data Access
  - Limited number of users
  - Processing in client application
  - Access to Oracle via SQL*Net
  - Results in heavy data traffic
  - Over network
- Distributed Processing
  - Optimal for large number of users
  - Oracle processing distributed to server application on database server



Construction Options–Windows

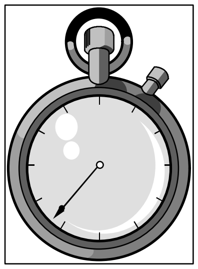
- Modify script files
  - Modify IDW*****.SCR files precompiler options
  - DOS limit on length of command line is 256 characters
  - Oracle RDA
  - Advanced Installation Kit (AIK)
- Use to control parsing of SQL statements
  - RELEASE_CURSOR=NO
  - HOLD_CURSOR=YES
  - MAXOPENCURSORS=100

Construction Options–Batch

- Modify script files
  - Customize script file – update precompiler options
  - Load into target configuration database
  - Regenerate install scripts
- Use to control parsing of SQL statements
  - RELEASE_CURSOR=NO
  - HOLD_CURSOR=YES
  - MAXOPENCURSORS=100

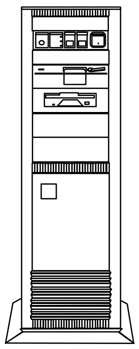
Assessing Performance

- Define performance levels
  - Application performance
  - Oracle engine performance
  - SQL*Net/communications
  - System performance
- Set goals
- “Just the facts, ma’am”



System Performance

- UNIX commands
  - – System Activity Report – sar
  - – Processor Status – ps
- Processing time – Oracle vs. application
- Contention with other users
- Assess data file disk activity



Data Access Methods

- Explain plan
- SQL*Trace
  - – Use EAB to turn trace on for Composer application
- TKPROF
  - – Review TKPROF report to look for:
    - » Joins
    - » Cursor reparsing
    - » Sorts

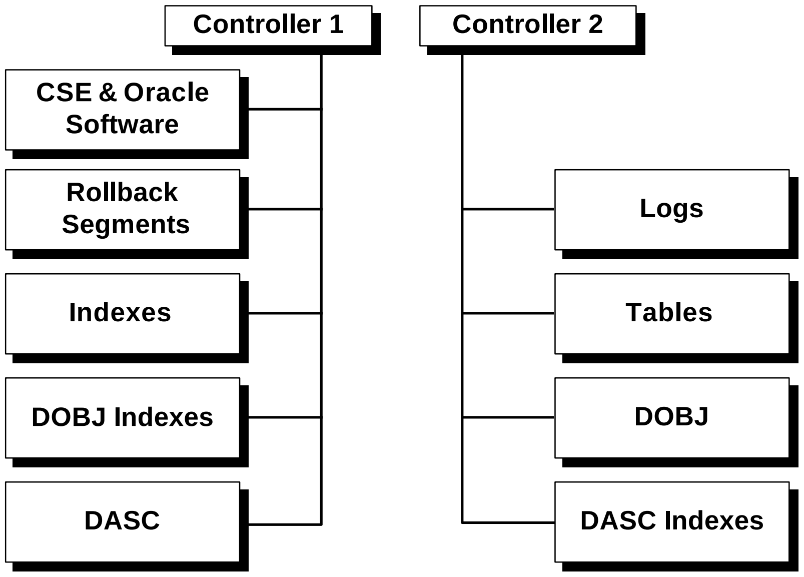
Tuning Memory Management

- Tune number of database buffers and redo buffers
- Tune data dictionary cache – Version 6
- Tune shared pool size – Version 7
- Reduce swapping and paging
  - Swapping – swap memory pages to disk when physical memory becomes constrained
  - Paging – move individual processes to disk when physical memory becomes constrained

Tuning Disk I/O

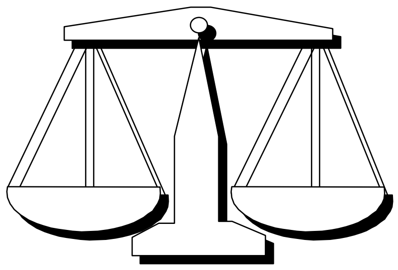
- Distribute I/O and applications across drives & controllers
- Tune number of database writers
- Check for large disk request queues
  - *sar* report details disk statistics across entire server
  - Determine location of data/indexes/redo logs
- Check for disk and tablespace fragmentation

HP/Oracle CSE Data Placement



Tuning CPU Usage

- Balance CPU loads
- Reorganize usage patterns
- Example: use batch programs to offload system-intensive Composer C/SE programs to off-hours



Tuning Resource Contention

- Determine contention bottlenecks
- Assess contention based on number of users
- Assess contention background vs. client/server applications

Summary

- Determine application performance goals
- Assess impact of tuning database on different types of applications
- Utilize optimal target platform

Optimizing Composer Multi-Platform Oracle Applications

Session 750

Rebecca Lawson
Texas Instruments