Oracle 8i: New Features for Administrators

Electronic Presentation

30071GC11
Production 1.1
May 2000
M011240

ORACLE®

Author

Jean-Francois Verrier

Technical Contributor and Reviewer

David Austin
Christian Boure
Christian Burgevin
Cyrille Drouard
Bruce Ernst
Joel Goodman
Scott Gossett
Patrick Jacob
Dominique Jeunot
Stefan Lindblad
Howard Ostrow
Richard Powell
Hans Proetzl
Rick Pulliam
Ulrike Schwinn
Mark Smith
Etienne Tiphine
Bram Van Der Vaus
Nitin Vengurlekar
Sergiusz Wolicki
Anthony Woodell

Publisher

Janine Katter

1

Introduction

Course Objectives

After completing this course, you should be able to do the following:

- Describe the new features introduced in Oracle8*i*
- Describe the features of Java in the database
- Identify features added to the optimizer in Oracle8*i*

Course Objectives

- Use summary management features
- Create and manage the different types of indexes supported by Oracle8*i*
- Describe the new features introduced in Oracle8*i* regarding partitioned tables and indexes

Course Objectives

- Describe Oracle8*i* installation and migration
- Create and manage the different types of tablespace supported by Oracle8*i*
- Monitor long-running operations
- Deploy the database resource manager

Course Objectives

- Identify new networking options offered by Net8
- Implement bounded recovery time
- Manage standby databases for automatic recovery and read-only access
- Use new archive logging options
- Describe the functionality of LogMiner

Course Objectives

- Manage corrupt block detection and repair
- Describe the virtual database
- Create and manage roles and objects to support new Oracle Advanced Queuing features
- Use SQL*Plus to manage database startup, shutdown, and related activities

Course Objectives

- Describe event triggers
- Identify Oracle8*i* national language support (NLS) features
- List new constraints capabilities

Suggested Course Schedule

Day	Lessons	Labs
1	1 - 3	3
	4	4
2	5 - 8	5,7
3	9 - 12	9,10,12
4	13-16	15

2

Java in the Database

Objectives

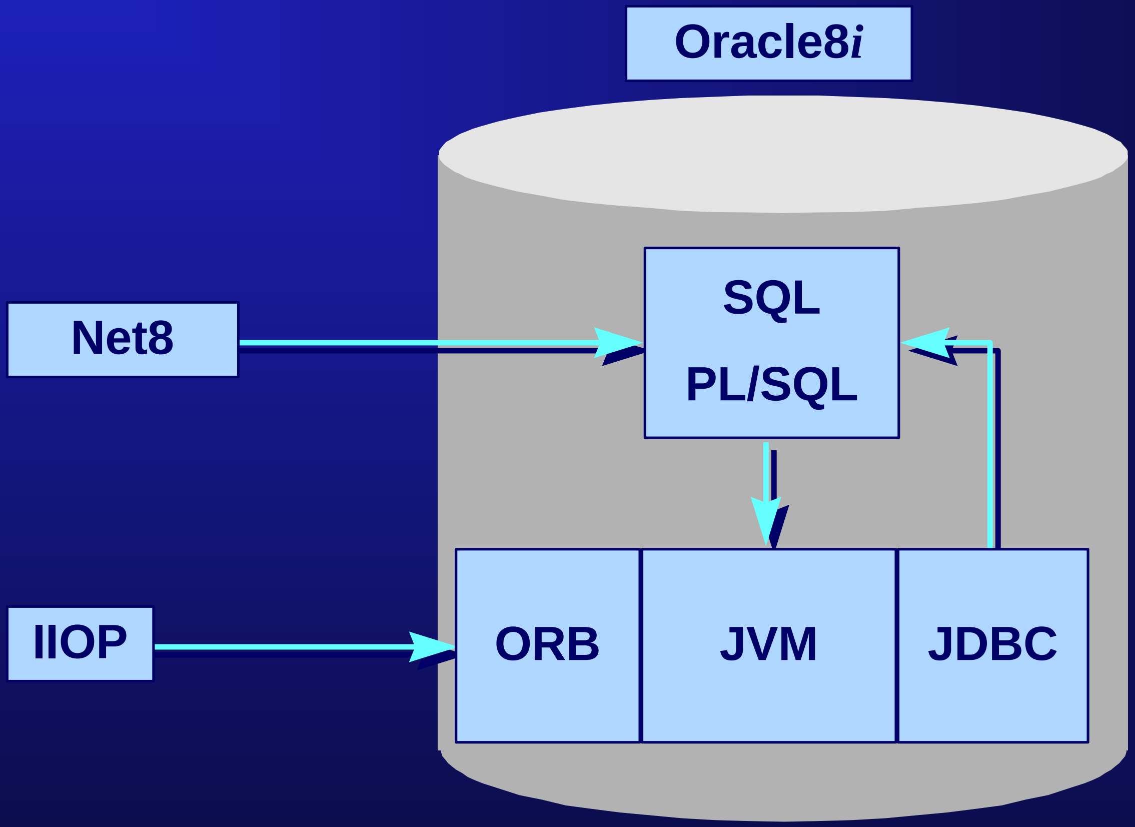
After completing this lesson, you should be able to do the following:

- Describe Oracle Java components
- Describe JServer installation
- Tune JServer
- Remove Java classes from the database

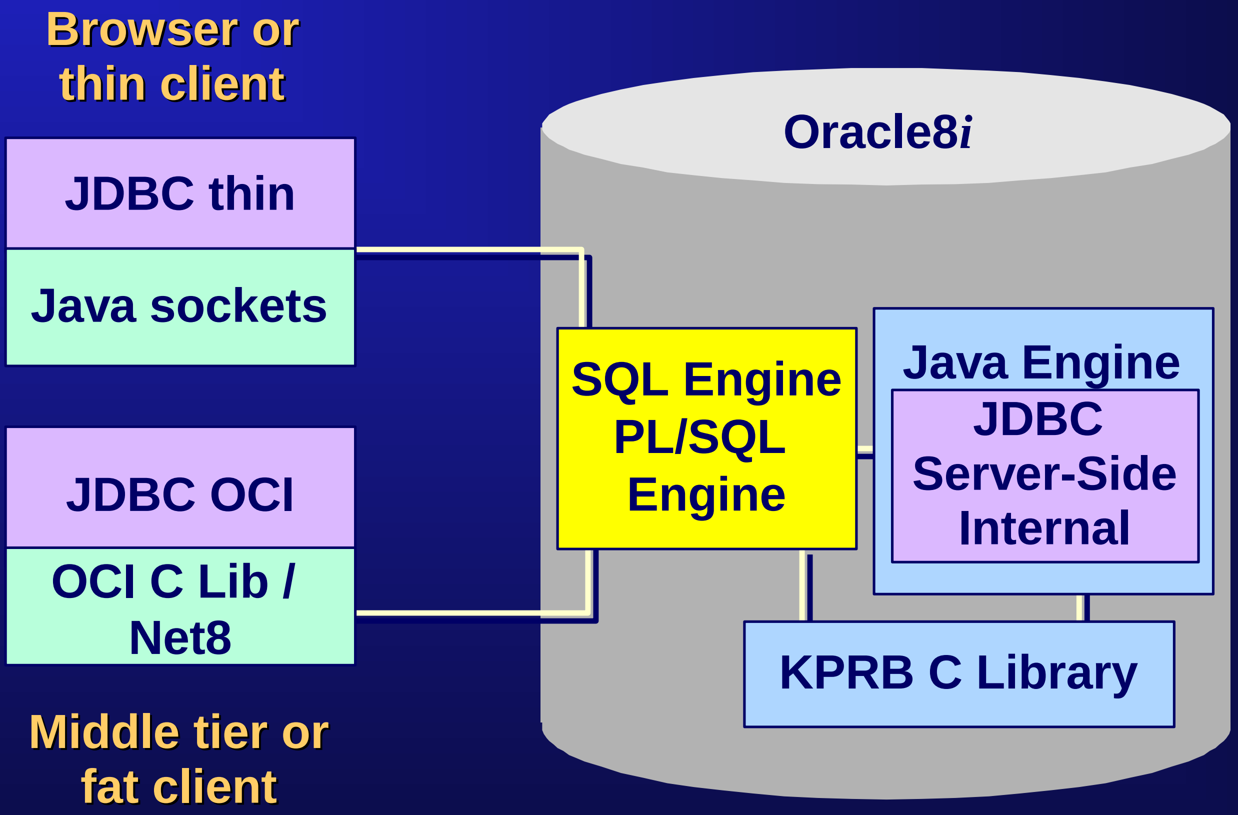
Java Overview

- Open, portable, productive language
- The language of Internet computing
- Oracle8*i* JServer
  - Enterprise class Java server
  - Java integrated with DB
  - Java Database Connectivity (JDBC) and SQLJ access the database
  - Can use Java anywhere that PL/SQL used
  - Industry-standard components
  - EJBs and CORBA built in
  - Productive programming tools

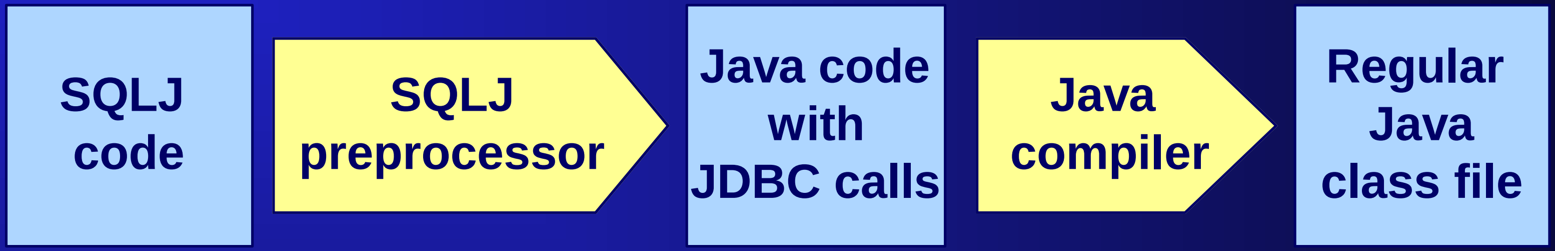
Integrating Java into Oracle8*i*



Java Database Connectivity Drivers

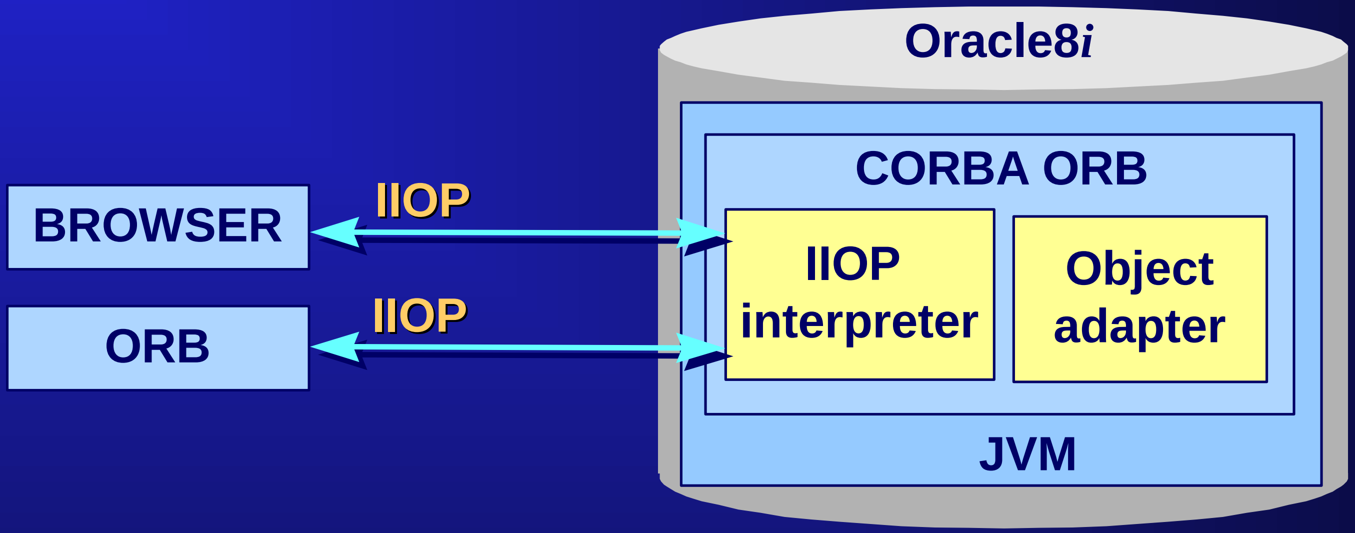


SQLJ



- Enables SQL statements embedded in Java
- Generates Java with JDBC calls
- Is significantly more compact than JDBC
- Is easier to write and maintain
- Is a de facto standard

Oracle8*i* Java Components



- Can improve developer productivity
- Includes two programming models:
  - Enterprise JavaBeans (EJBs)
  - Common Object Request Broker Architecture (CORBA)

Development Tools

- Sun Java Development Kit
  - Basis of all Java development tools
  - Included with JServer
  - Command-line-driven tools
- Oracle JDeveloper 3.0
  - GUI-driven tool
  - Server-side Java development, deployment, and debugging
  - JDBC, SQLJ, EJB, and CORBA
  - Client-side database support as well

Installation Overview

- JServer installation is part of the typical Oracle software and database installation.
- JServer can be installed in one of three ways:
  - Select a minimal or typical install
  - Select the JServer option for custom installs
  - Run the `initjvm.sql` script for manual installs
- This course looks at the manual installation of JServer.

Preparing the Instance and Database Before Installing the JVM Classes

Before executing `initjvm.sql`, you should ensure the following:

- SHARED_POOL_SIZE = 50MB
- JAVA_POOL_SIZE = 20MB
- SYSTEM tablespace can grow to 160 MB (8.1.6) or to 105 MB (8.1.5)
- Rollback segment can grow to 50 MB

Installing the JVM Classes Using the initjvm.sql Script

- Located in $ORACLE_HOME/javavm/install
- Creates database objects to support Java
- Installs initial Java classes with the CREATE OR REPLACE JAVA SYSTEM; command
- Creates database startup and shutdown triggers
- Configures JServer for CORBA and EJB
- Creates the dbms_java support package with the initdbj.sql script
- Creates various roles: JAVASYSPRIVS, JAVAUSERPRIVS, JAVADEBUGPRIVS
- Network configuration

Installation Verification

`initjvm.sql` checks for a successful install:

- Number of Java objects created is 8643 in 8.1.6 and 4023 in 8.1.5:

```
SQL> SELECT count(*)
  2    FROM dba_objects
  3    WHERE object_type LIKE 'JAVA%';
```

- Number of Java objects with an invalid status is 0:

```
SQL> SELECT count(*)
  2    FROM dba_objects
  3    WHERE object_type like ‘JAVA%’ AND
  4    status != ‘VALID’;
```

Sizing Shared Pool for Java

- `SHARED_POOL_SIZE`
  - 8 KB per loaded class
  - 50 MB for loading large JAR files
- `LARGE_POOL_SIZE`
  - Used with MTS
  - Contains part of the UGA

Sizing Java Pool Memory

- Dedicated server:
  - Java pool stores the shared part of each Java class, which uses 4 to 8 KB for each class
  - UGA in PGA contains per-session Java state
- Multithreaded server:
  - Stores the shared part of each Java class, which uses 4 to 8 KB for each class
  - Some of the per-session Java state
  - Up to 1 GB for EJB or CORBA
- Server-side compilation requires extra memory
- EJBs and CORBA require extra memory

Check Java Pool Memory Usage

- Run the Java application
- Monitor the Java memory usage

```
SQL> SELECT *
  2    FROM v$sgastat
  3    WHERE name = 'java pool';
POOL        NAME                            BYTES
----------- -------------------------- ----------
java pool   free memory                  30261248
java pool   memory in use                19742720
SQL>
```

Limiting Java Session Memory Usage

- `JAVA_SOFT_SESSIONSPACE_LIMIT`
  - 1 MB default
  - Only writes a warning message to a trace file
- `JAVA_MAX_SESSIONSPACE_SIZE`
  - 4 GB default
  - Ends the session with an out-of-memory error

Deinstallation Steps

1. Run the script that removes the JVM:
   - In the `ORACLE_HOME/rdbms/admin` directory
   - In 8.1.5, `utljavarm.sql`
   - In 8.1.6, `utljavrm.sql`
2. Change JAVA_POOL_SIZE to 1 MB in 8.1.5 or to 0 in 8.1.6
3. Reduce SHARED_POOL_SIZE by tuning the shared pool
4. Remove references to GIOP and SGIOP from:
   - `init.ora`
   - `listener.ora`

Developing Java Stored Procedures

Step 1: Write the Java stored procedure

Step 2: Load the Java program into Oracle8*i*:

- Use `CREATE JAVA DDL`
- Use the `loadjava` utility

Step 3: Publish the Java program to SQL:

- Expose top-level Java entrypoint
- Map Java arguments and datatypes to SQL
- Set up appropriate user privileges

Step 4: Call Java program from SQL or PL/SQL

Example: Publishing Java to SQL

Create the Java class:

```
CREATE OR REPLACE JAVA SOURCE NAMED "Hello" AS
  public class Hello {
    static public String Msg(String tail) {
      return "Hello " + tail;
    }
  }
```

Publish the class to SQL:

```
CREATE OR REPLACE FUNCTION hello
  ( str VARCHAR2 )
  RETURN VARCHAR2
AS
  LANGUAGE JAVA NAME
    'Hello.Msg (java.lang.String)
      return java.lang.String';
```

Summary

You should now have a basic understanding of the Jserver functionalities.

In this lesson, you should have learned how to:

- Use the `initjvm.sql` script to load the initial Java classes
- Set instance configuration parameters to:
  - Size the Java pool
  - Limit Java session memory usage
- Tune and monitor the Java pool
- Deinstall JServer

3

Optimizer and Query Improvements

Objectives

After completing this lesson, you should be able to do the following:

- Describe the features of optimizer plan stability
- Describe the contents of the `DBMS_STATS` package
- Explain sharing cursor improvements
- Explain Top-N SQL queries
- Identify new SQL keywords for computing subtotals
- Identify new sort processing options

Optimizer Plan Stability

- Allows well-tuned applications to force the use of the desired SQL access path
- Consistent execution paths maintained through certain database changes
- Implemented using a stored outline consisting of hints

Factors Not Addressed by Outlines

- Degree of parallelism
- Predicate placement
- OR expansion
- Partition access
- Recursive queries
- *_ENABLED parameters
- Controlling the execution plan for third-party applications

Creating Stored Outlines

CREATE_STORED_OUTLINES parameter:

```
ALTER SESSION
  SET CREATE_STORED_OUTLINES = train;
SELECT   co.crs_id, ...
```

CREATE OUTLINE command:

```
CREATE OR REPLACE OUTLINE co_cl_join
  FOR CATEGORY train
  ON SELECT  co.crs_id, ...
       FROM  courses    co,
             classes    cl
    WHERE   co.crs_id = cl.crs_id;
```

Using Stored Outlines

- Set the USE_STORED_OUTLINES parameter to TRUE or to a category name.

```
ALTER SESSION
  SET USE_STORED_OUTLINES = train;
SELECT   co.crs_id, ...
```

- Both CREATE_STORED_OUTLINES and USE_STORED_OUTLINES can be set at the instance or session level, but they are not init.ora parameters.
- V$SQL contains the OUTLINE_CATEGORY column.

Outlines Security

- `CREATE ANY OUTLINE`: Needed by user to create outlines
- `DROP ANY OUTLINE`: Needed by user to drop outlines
- `ALTER ANY OUTLINE`: Needed by user to modify outlines
- No special privilege required to use an outline

Maintaining Stored Outlines

- Use `OUTLN_PKG` to:
  - Drop unused outlines
  - Drop categories of outlines
  - Rename categories
- Use `ALTER OUTLINE` to:
  - Rename an outline
  - Rebuild an outline
  - Change the category of an outline
- Stored outlines in tables in `OUTLN` schema

Moving Outlines Tables Example

```
EXP OUTLN/OUTLN TABLES = ('OL$','OL$HINTS')
```

```
DROP TABLE OL$;
DROP TABLE OL$HINTS;
```

```
CREATE TABLESPACE outln_ts DATAFILE
   'tspace.dat' SIZE 2MB;
```

```
ALTER USER OUTLN DEFAULT TABLESPACE
   outln_ts;
```

```
IMP OUTLN/OUTLN TABLES = ('OL$','OL$HINTS')
```

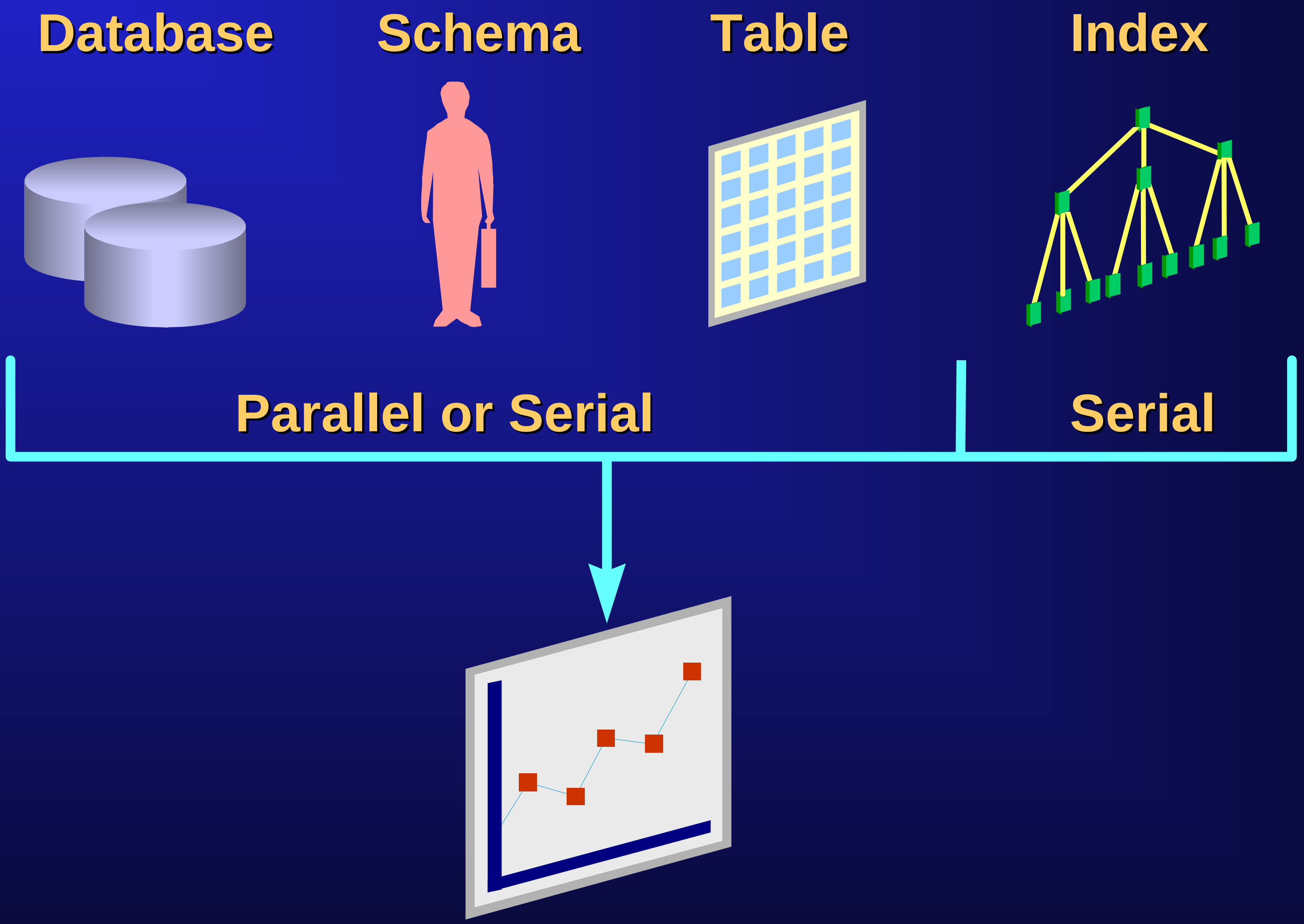
Sharing Cursors

- Text of SQL statements must be identical
- The only tolerated differences are literals, if `CURSOR_SHARING` is set to `FORCE`
- SQL statements must reference the same objects
- Bind variables in the SQL statements must match in name and data type
- The SQL statements must be optimized using the same optimization approach

Checking for Cursor Sharing

- CURSOR_SHARING is an init.ora parameter and is session- or system-modifiable. Possible values are:
  - FORCE
  - EXACT
- V$ views showing system-generated bind variables:
  - V$SQL
  - V$SQL_BIND_DATA
  - V$SQL_BIND_METADATA

DBMS_STATS Package



Database

Schema

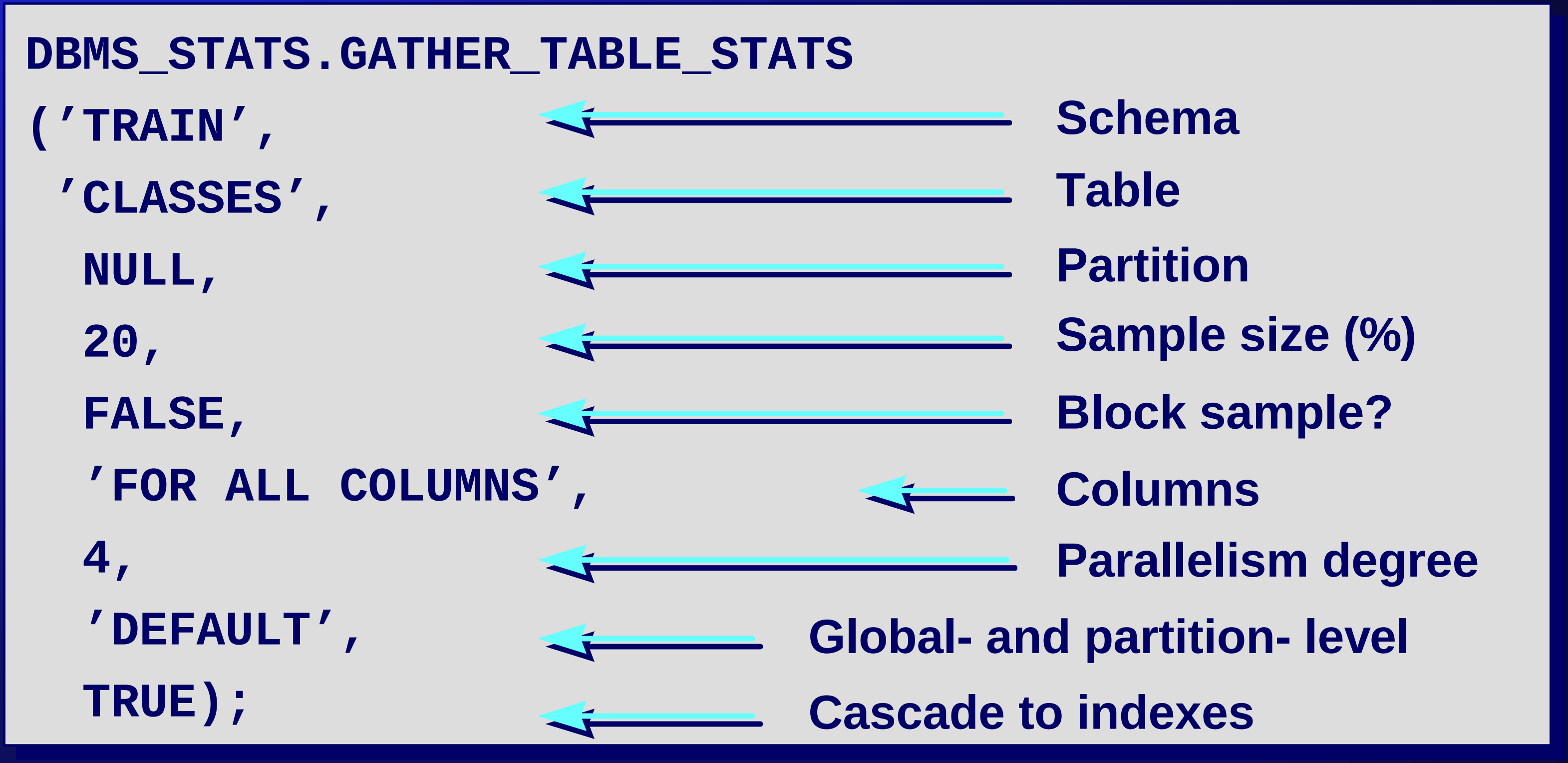
Table

Index

Parallel or Serial

Serial

DBMS_STATS: Generating Statistics



```
DBMS_STATS.GATHER_TABLE_STATS
('TRAIN',
 'CLASSES',
  NULL,
  20,
  FALSE,
  'FOR ALL COLUMNS',
  4,
  'DEFAULT',
  TRUE);
```

Schema

Table

Partition

Sample size (%)

Block sample?

Columns

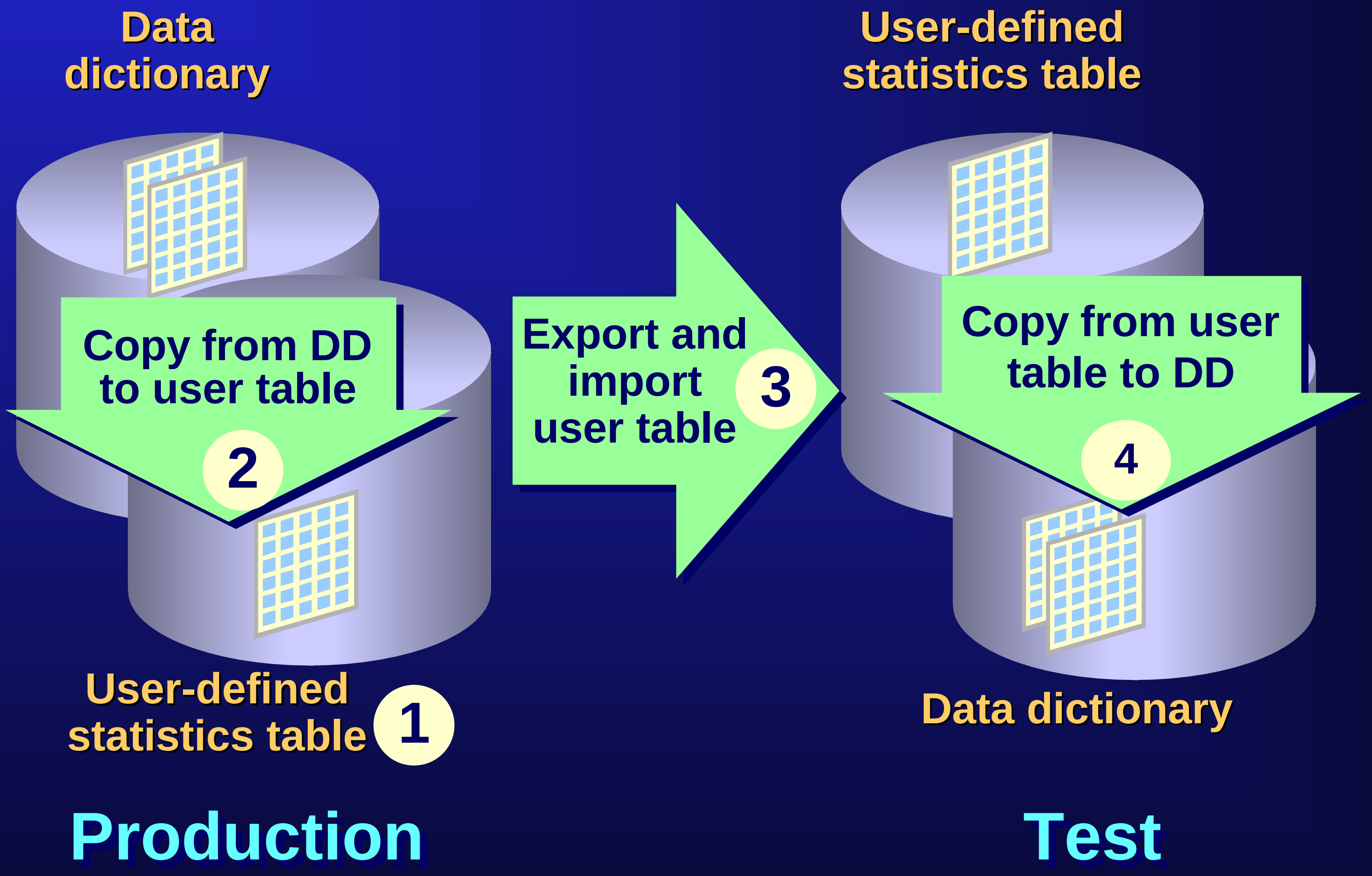
Parallelism degree

Global- and partition- level

Cascade to indexes

The syntax is not complete on this example

Copying Statistics Between Databases



Data dictionary
User-defined statistics table
Copy from DD to user table
2
Export and import user table
3
Copy from user table to DD
4
User-defined statistics table
1
Data dictionary
Production
Test

Example: Copying Statistics

```
DBMS_STATS.CREATE_STAT_TABLE (
   'TRAIN',              ←     Schema
   'STATS',              ←     Statistics table name
   'USERS' );            ←     Tablespace
```

```
DBMS_STATS.EXPORT_TABLE_STATS(
   ’TRAIN’,              ←     Schema
   ’COURSES’,            ←     Table name
   NULL,                 ←     No partitions
   ’STATS’,              ←     Statistics table name
   ’CRS 980501’,         ←     ID for statistics
   TRUE );               ←     Index statistics
```

Monitoring Tables

1

```
ALTER TABLE *tabname* MONITORING
```

2

```
DBA_TAB_MODIFICATIONS
```

3

```
DBMS_STATS.GATHER_SCHEMA_STATS(
  ownname => 'OE',
  options => ’GATHER STALE’,
  objlist => lt               );
```

Sort Performance Improvements

- More predictable sort performance
- Implicit use of direct writes
- Sort-related `init.ora` parameters obsolete:
  - `SORT_DIRECT_WRITES`
  - `SORT_WRITE_BUFFERS`
  - `SORT_WRITE_BUFFER_SIZE`
  - `SORT_READ_FAC`
- New `init.ora` parameter: `SORT_MULTIBLOCK_READ_COUNT`

Top-N SQL

- Views and in-line views now allow ordering
- Sorts only the required number of rows

```
SELECT  *
  FROM  ( SELECT  class_id,
                  crs_id,
                  start_date
            FROM  classes
            ORDER BY start_date DESC )
  WHERE ROWNUM < 10
```

ROLLUP Operation

```
SELECT   type, status, SUM(days)
  FROM   classes
  GROUP BY ROLLUP(type, status);
```

```
TYPE STAT   SUM(DAYS)
IDL  AVAI          72
...
S/EC ERRO         195
S/EC FULL           2
S/EC MOVE         312
S/EC SENT         136

S/EC             5394
                18760
```

CUBE Operation

```
SELECT      type, status, SUM(days)
  FROM      classes
  GROUP BY CUBE(type, status)
  ORDER BY type, status;
```

```
TYPE STAT  SUM(DAYS)
---- ---- ----------
IDL  AVAI         72
...
S/EC SENT        136


  S/EC              5394
       AVAI          294
       BOOK         8034
  ...
       MOVE          521
       SENT          366
                   18760
```

ROLLUP totals

CUBE totals

Summary

In this lesson, you should have learned the following:

- Stored outlines ensure that execution plans stay consistent
- The `DBMS_STATS` package can manipulate statistics to affect execution plans
- `CURSOR_SHARING` extends cursor sharing
- Top-N SQL queries allow sort in in-line views
- `ROLLUP` and `CUBE` keywords compute subtotals
- New sort processing options improve sort performance

Practice 3 Overview

- Creating statistics tables
- Exporting data dictionary statistics
- Creating categories
- Using outline categories in SQL statements
- Importing data dictionary statistics
- Droping statistics tables

4

Summary Management

Objectives

After completing this lesson, you should be able to explain how to build and manage:

- Materialized views for Oracle Summaries
- Dimensions

What Is Summary Management ?

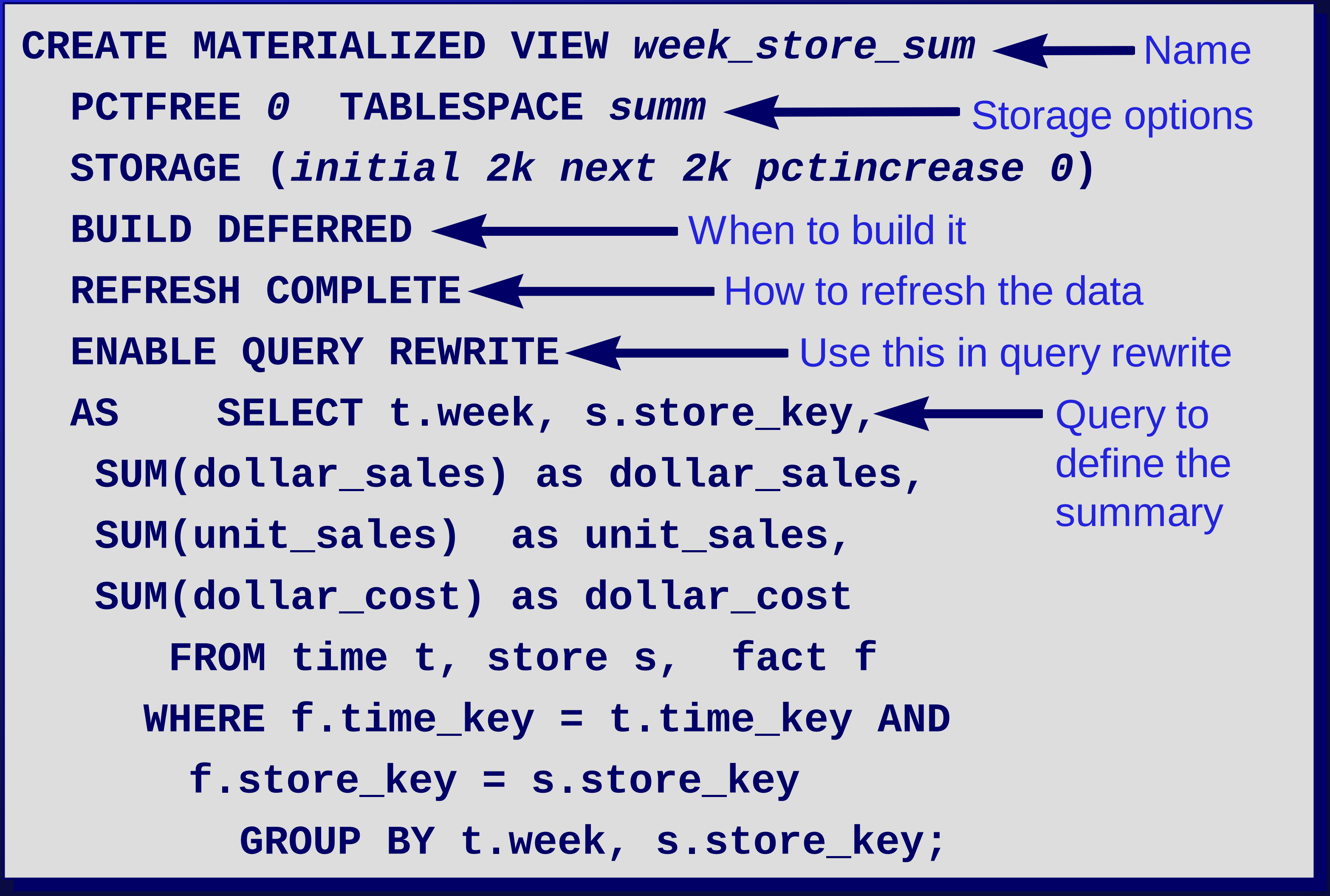
- Create predefined summary tables using materialized view in Oracle8*i*
- Application queries use summary segments for improved performance
  - No need to change application queries
    - User queries reference base tables
  - Oracle optimizer rewrites the query
    - Access materialized views
- Refresh materialized views as needed

What Is a Materialized View ?

A materialized view:

- Is an instantiation of a SQL statement
- Has its data stored in tables, offering:
  - Space management options
  - Use of indexes and partitions
- Used for:
  - Data warehouses
  - Distributed computing

Materialized View Example



```
CREATE MATERIALIZED VIEW week_store_sum
  PCTFREE 0  TABLESPACE summ
  STORAGE (initial 2k next 2k pctincrease 0)
  BUILD DEFERRED
  REFRESH COMPLETE
  ENABLE QUERY REWRITE
  AS    SELECT t.week, s.store_key,
   SUM(dollar_sales) as dollar_sales,
   SUM(unit_sales)  as unit_sales,
   SUM(dollar_cost) as dollar_cost
      FROM time t, store s,  fact f
    WHERE f.time_key = t.time_key AND
      f.store_key = s.store_key
        GROUP BY t.week, s.store_key;
```

Name

Storage options

When to build it

How to refresh the data

Use this in query rewrite

Query to define the summary

Build Methods

- `BUILD DEFERRED`: MV created but not populated
- `BUILD IMMEDIATE`: MV created and populated
- `ON PREBUILT TABLE`: MV created over existing table

Refresh Triggering Events

- `ON DEMAND`: Manual
- `ON COMMIT`: On transaction commit
- `Schedule`: At regular intervals

Materialized Views: Manual Refresh

- Refresh specific materialized views

```
DBMS_MVIEW.REFRESH(’SF_SALES’, PARALLELISM => 10);
```

- Materialized views based on one or more base tables

```
DBMS_MVIEW.REFRESH_DEPENDENT
(’SALES’);
```

- All materialized views due for refresh

```
DBMS_MVIEW.REFRESH_ALL_MVIEWS;
```

Refresh Methods

- `COMPLETE`
- `FAST` or `INCREMENTAL`
  - Using materialized view logs
  - Using direct loader log: `ALL_SUMDELTA`
- `FORCE`
- `NEVER`

General Restrictions on Fast Refresh

- No views in the `FROM` list
- No nonrepeating expressions like `SYSDATE` and `ROWNUM`
- No `RAW` or `LONG RAW` columns
- No `HAVING` or `CONNECT BY` clauses
- Only AND equijoin predicates
- No subqueries, inline views, or set functions like `UNION` or `MINUS`

Restrictions on Fast Refresh on Materialized Views with Joins Only

- No `GROUP BY` clauses or aggregates
- If outer joins, then unique constraints must exist on the join columns of the inner join table
- Rowids of all the tables must be in the `SELECT` list
- Rowids materialized view logs for all tables
- FAST refreshable after DML or direct load

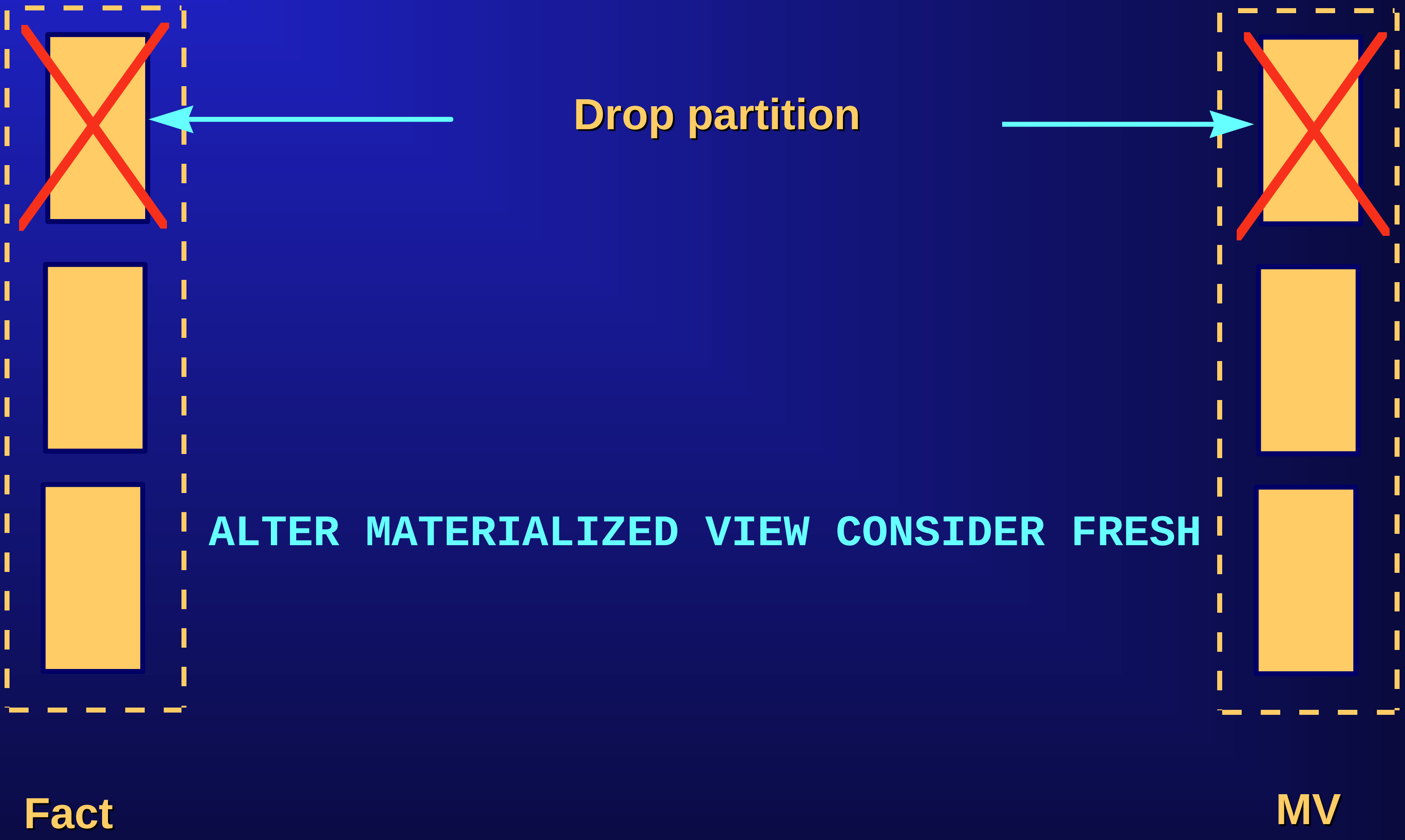
Restrictions on Fast Refresh on MVs with Single-Table Aggregates

- They can only have a single table
- `SELECT` list must contain all groupings
- They cannot have a `WHERE` clause
- `COUNT(*)` must be present
- They cannot have a `MIN` or `MAX` function
- `INCLUDING NEW VALUES` MV logs on all tables with all referenced MV columns
- If `AVG(expr)` or `SUM(expr)` is specified, then you must have `COUNT(expr)`

Restrictions on Fast Refresh on MVs with Joins and Aggregates

- The `WHERE` clause can contain inner equijoins only
- FAST refreshable after direct load only
- Can have only the `ON DEMAND` option

Considering a Materialized View as Fresh



ALTER MATERIALIZED VIEW CONSIDER FRESH

Privileges and Materialized Views

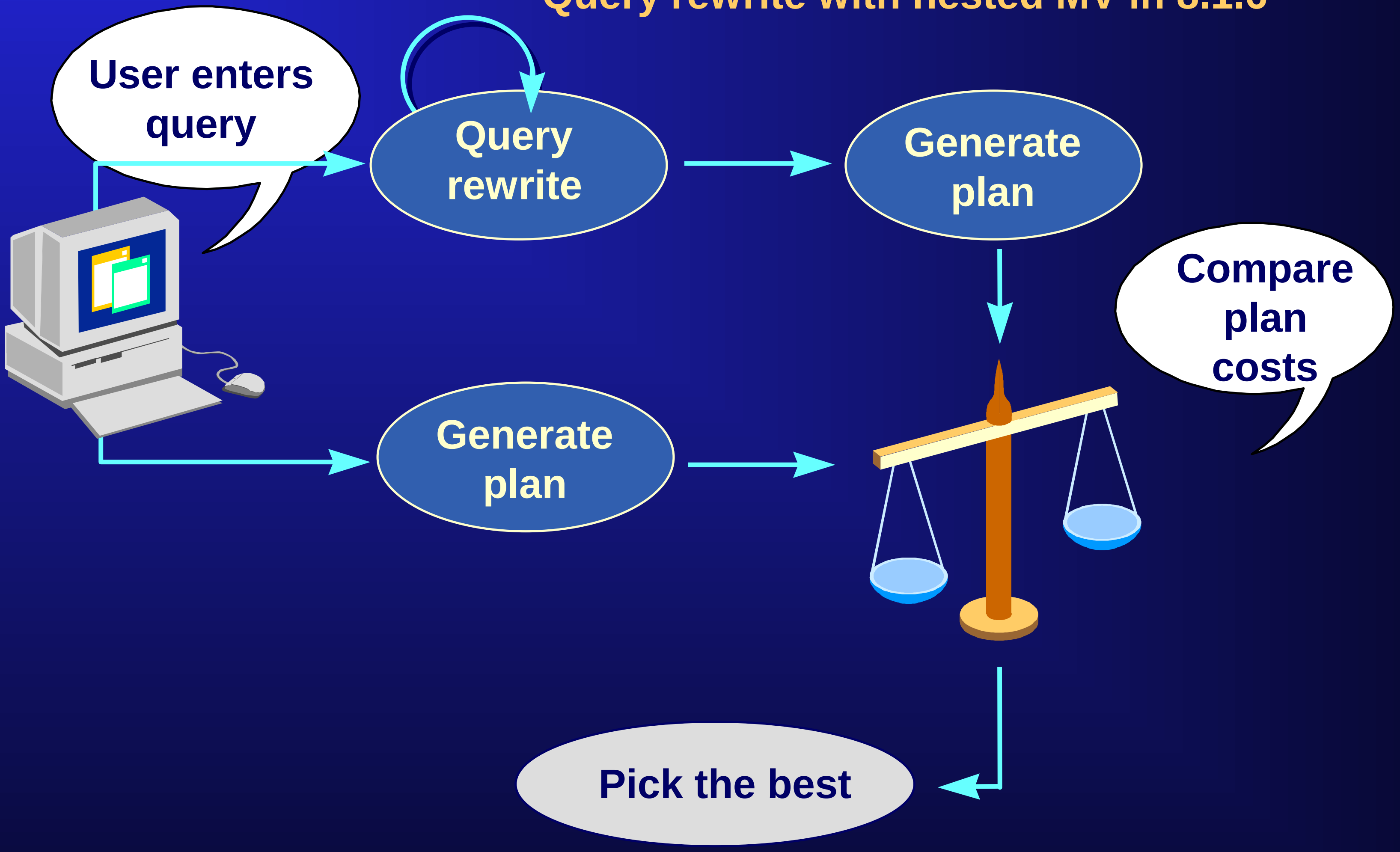
- `CREATE MATERIALIZED VIEW`
- `CREATE ANY MATERIALIZED VIEW`
- `QUERY REWRITE`
- `GLOBAL QUERY REWRITE`
- `CREATE TABLE`
- `CREATE INDEX`
- `CREATE VIEW`

Materialized Views: Data Dictionary

- ALL_REFRESH_DEPENDENCIES
- DBA_MVIEW_AGGREGATES
- DBA_MVIEW_ANALYSIS
- DBA_MVIEW_DETAIL_RELATIONS
- DBA_MVIEW_JOINS
- DBA_MVIEW_KEYS
- DBA_MVIEWS (New in 8.1.6)

Cost-Based Query Rewrite Process

Query rewrite with nested MV in 8.1.6



ORACLE®

Enabling Query Rewrite

To enable query rewrite you should:

- Flag individual MVs for query rewrite
- Set `QUERY_REWRITE_ENABLED` to `TRUE`
- Set `OPTIMIZER_MODE` to
  - `ALL_ROWS`
  - `FIRST_ROWS`
  - `CHOOSE` if statistics are generated
- Avoid the `NOREWRITE` hint in the query

When Does Oracle Rewrite a Query?

- Query rewrite must be enabled for the session
- The rewrite integrity level must allow the use of the MV
- Either all or part of the result requested by the query must be obtainable from the MV
  - View-Based MV (New in 8.1.6)
  - CUBE/ROLLUP rewrite (New in 8.1.6)
  - Complex Materialized Views
  - Date Folding (New in 8.1.6)

Did Query Rewrite Occur?

- Execute query

```
SELECT
    s.zip, p.product_type, sum(s.amount)
FROM sales s, product p
WHERE s.prod_id = p.prod_id
GROUP BY s.zip, p.prod_type;
```

- Examine execution plan

```
OPERATION              NAME
---------------------- -----------------
SELECT STATEMENT
  TABLE ACCESS FULL    SALES_SUMMARY
```

Types of Query Rewrite: Exact Match Example

```
SELECT month,
 avg(quantity) AS avgqty
FROM time t,sales s,product p
WHERE p.prod_id=s.prod_id
 and t.saledate=s.saledate
GROUP BY prod_id, month
HAVING sum(quantity) > 5000;
```

REWRITE

```
SELECT month,

sumqty/cntqty

FROM sumsales

WHERE

 sumqty>5000;
```

Application Query

```
SELECT month, prod_id, sum(quantity) AS sumqty,
count(quantity) AS cntqty
FROM time t,sales s,product p
WHERE p.prod_id=s.prod_id and t.saledate=s.saledate
GROUP BY prod_id, month;
```

SUMSALES Query Definition

Types of Query Rewrite: Aggregation to All Example

```
SELECT month,
 sum(quantity) AS sumqty
FROM time t,sales s
WHERE t.saledate=s.saledate
GROUP BY month
HAVING sum(quantity) > 5000;
```

Application Query

REWRITE

```
SELECT month,

sumqty

FROM sumsales

GROUP BY month

HAVING
sumqty>5000;
```

```
SELECT month, prod_id, sum(quantity) AS sumqty
FROM time t,sales s,product p
WHERE p.prod_id=s.prod_id and t.saledate=s.saledate
GROUP BY prod_id, month;
```

SUMSALES Query Definition

Types of Query Rewrite: Rollup Example

```
SELECT year, prodid
 sum(quantity) AS sumqty
FROM time t,sales s, product p
WHERE p.prod_id=s.prod_id
and t.saledate=s.saledate
GROUP BY prod_id, year;
```

Application Query

REWRITE

```
SELECT year,
prod_id,
sum(sumqty)
FROM sumsales s,
(SELECT DISTINCT
month, year FROM
time) v WHERE
v.month=s.month
GROUP BY
 prod_id,  year;
```

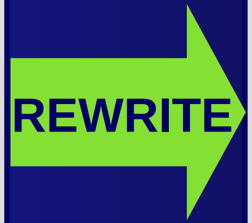
```
SELECT month, prod_id, sum(quantity) AS sumqty
FROM time t,sales s,product p
WHERE p.prod_id=s.prod_id and t.saledate=s.saledate
GROUP BY prod_id, month;
```

SUMSALES Query Definition

Types of Query Rewrite: Join Back Example

```
SELECT month,
 sum(quantity) AS sumqty
FROM time t,sales s, product p
WHERE p.prod_id=s.prod_id
and t.saledate=s.saledate
and t.year=2000
GROUP BY prod_id, month;
```

Application Query



```
SELECT month,
sumqty        FROM
sumsales s WHERE
s.month IN (
SELECT t.month
FROM time t
WHERE
t.year=2000);
```

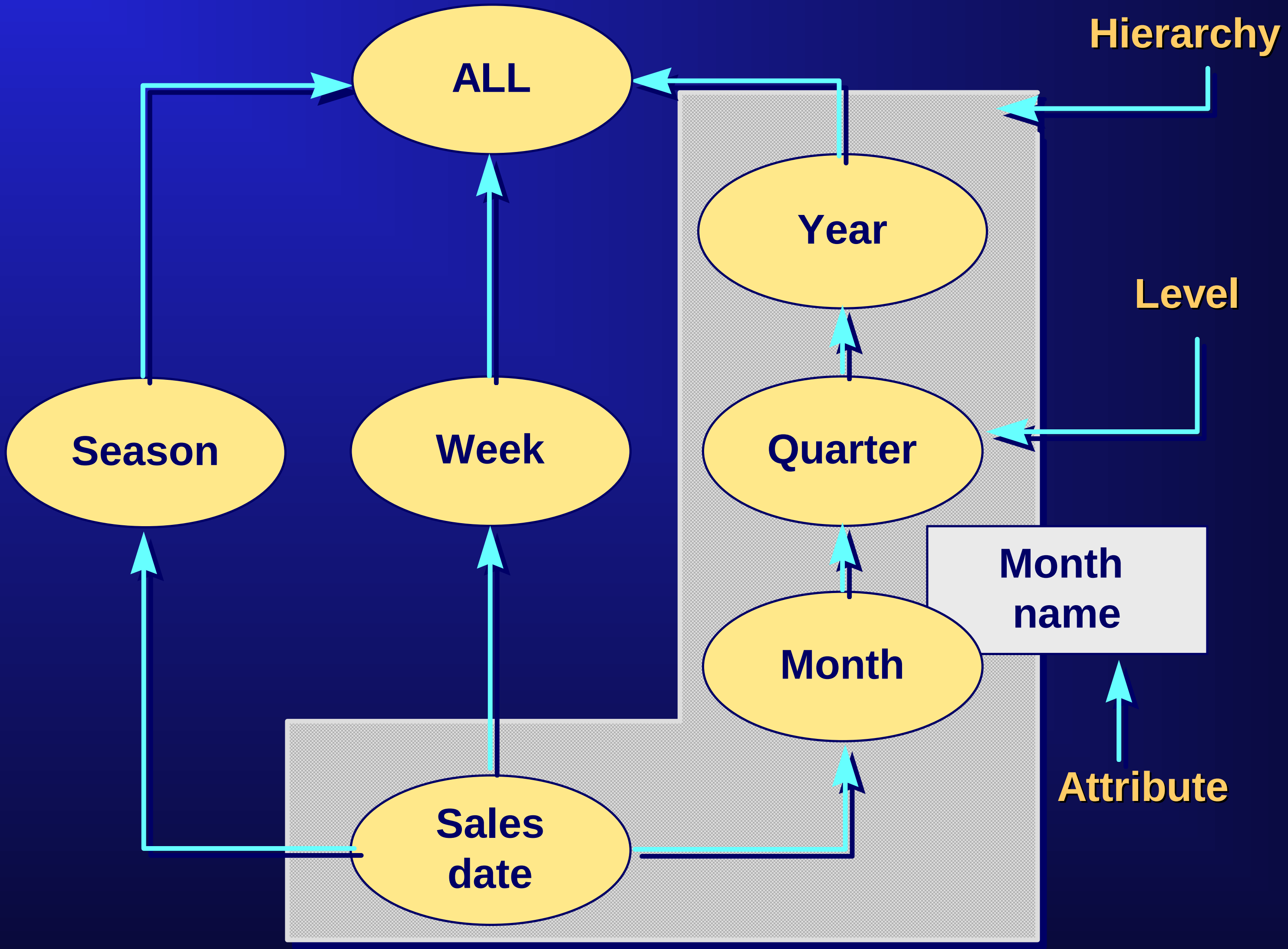
```
SELECT month, prod_id, sum(quantity) AS sumqty
FROM time t,sales s,product p
WHERE p.prod_id=s.prod_id and t.saledate=s.saledate
GROUP BY prod_id, month;
```

SUMSALES Query Definition

About Dimensions

- Dimensions are data dictionary structures that define hierarchies based on existing columns
- Dimensions are optional but highly recommended because they:
  - Enable additional query rewrites without the use of constraints
  - Help document hierarchies
  - Can be used by OLAP tools

Dimensions and Hierarchies



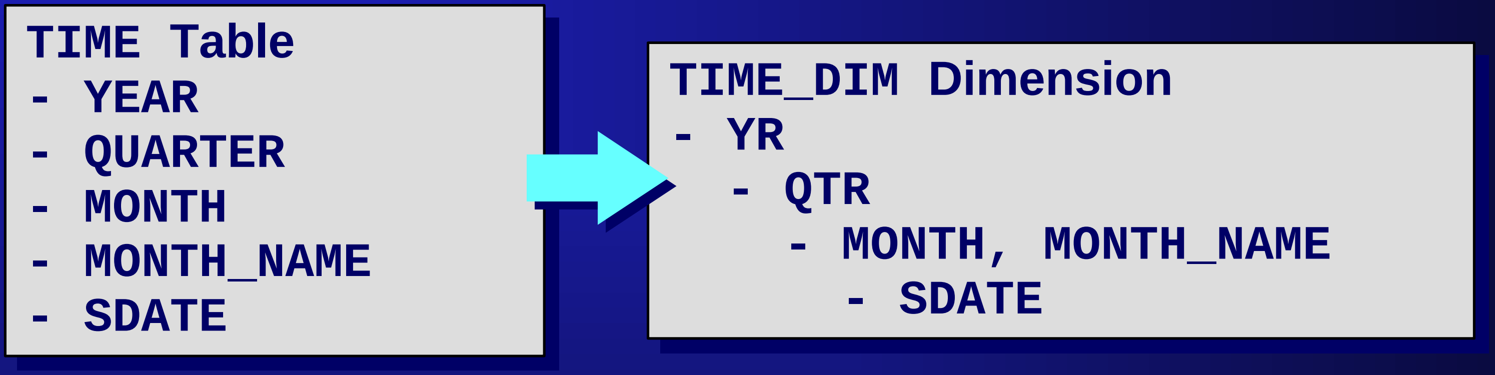
Dimension: Example

```
SELECT * FROM time;
```

```
SDATE          MONTH  MONTH_NAM  QUARTER       YEAR
---------  -----  ---------  -------  ---------
01-JAN-98      1  January          1       1998
02-JAN-98      1  January          1       1998
03-JAN-98      1  January          1       1998
04-JAN-98      1  January          1       1998
...

30-DEC-98     12  December         4       1998
31-DEC-98     12  December         4       1998
```

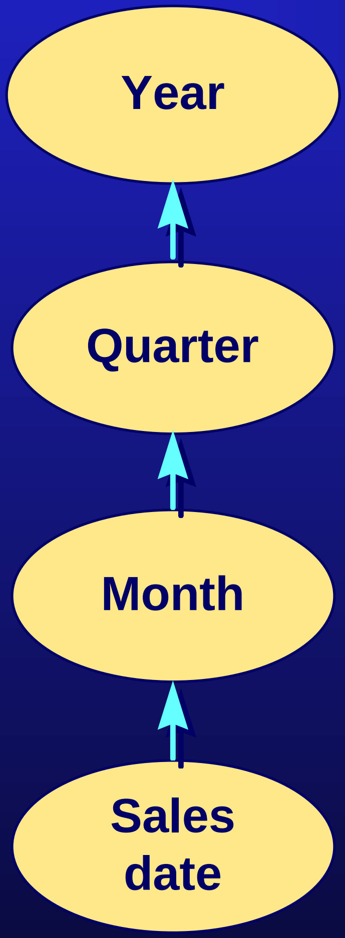
Dimension: Example



```
TIME Table
- YEAR
- QUARTER
- MONTH
- MONTH_NAME
- SDATE
```

```
TIME_DIM Dimension
- YR
   - QTR
      - MONTH, MONTH_NAME
         - SDATE
```

Defining Dimensions and Hierarchies

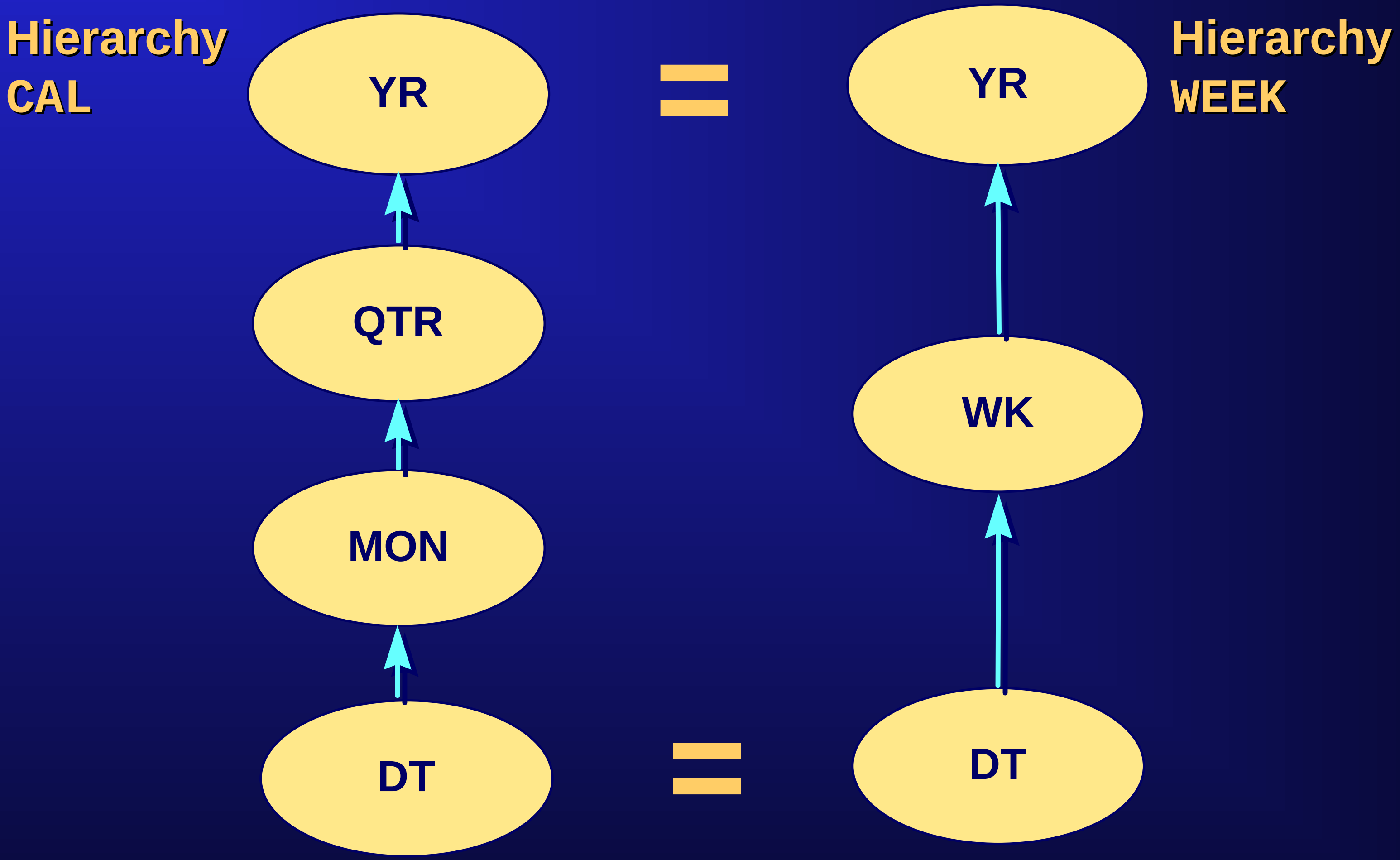


```
CREATE DIMENSION time_dim
 LEVEL sdate IS time.sdate
 LEVEL month IS time.month
 LEVEL qtr   IS time.quarter
 LEVEL yr    IS time.year
HIERARCHY calendar_rollup (
 sdate CHILD OF
 month CHILD OF
 qtr   CHILD OF  yr )
ATTRIBUTE month
  DETERMINES month_name;
```

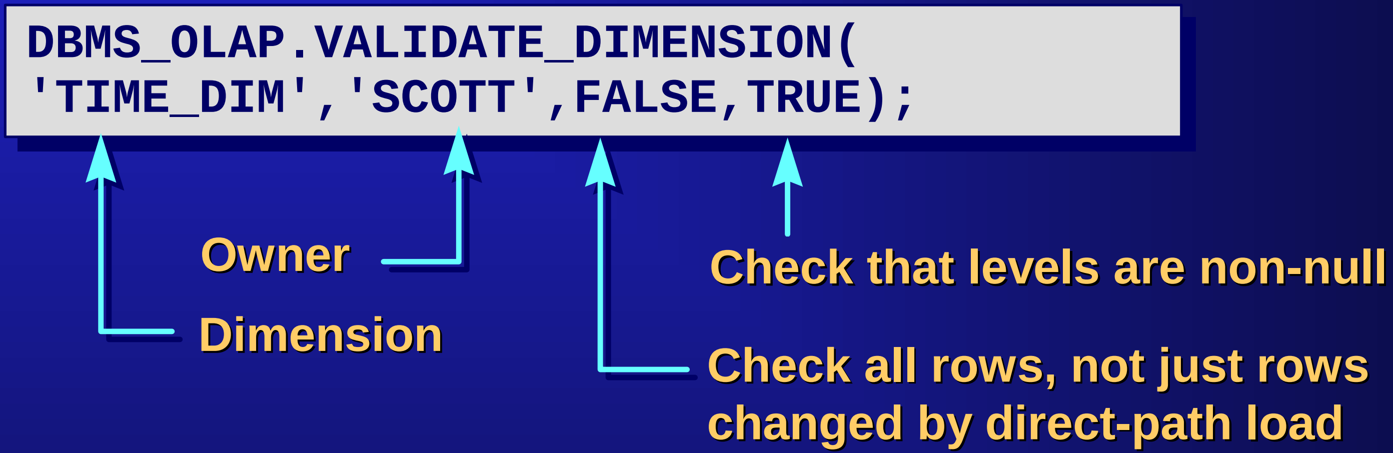
Dimensions Based on Multiple Tables

- Data dictionary objects
- Columns from one or more tables
- Keys and attributes correspond to columns
- All attributes and keys for one level must belong to one table

Dimensions with Multiple Hierarchies



Verifying Relationships in a Dimension



```
DBMS_OLAP.VALIDATE_DIMENSION(
'TIME_DIM','SCOTT',FALSE,TRUE);
```

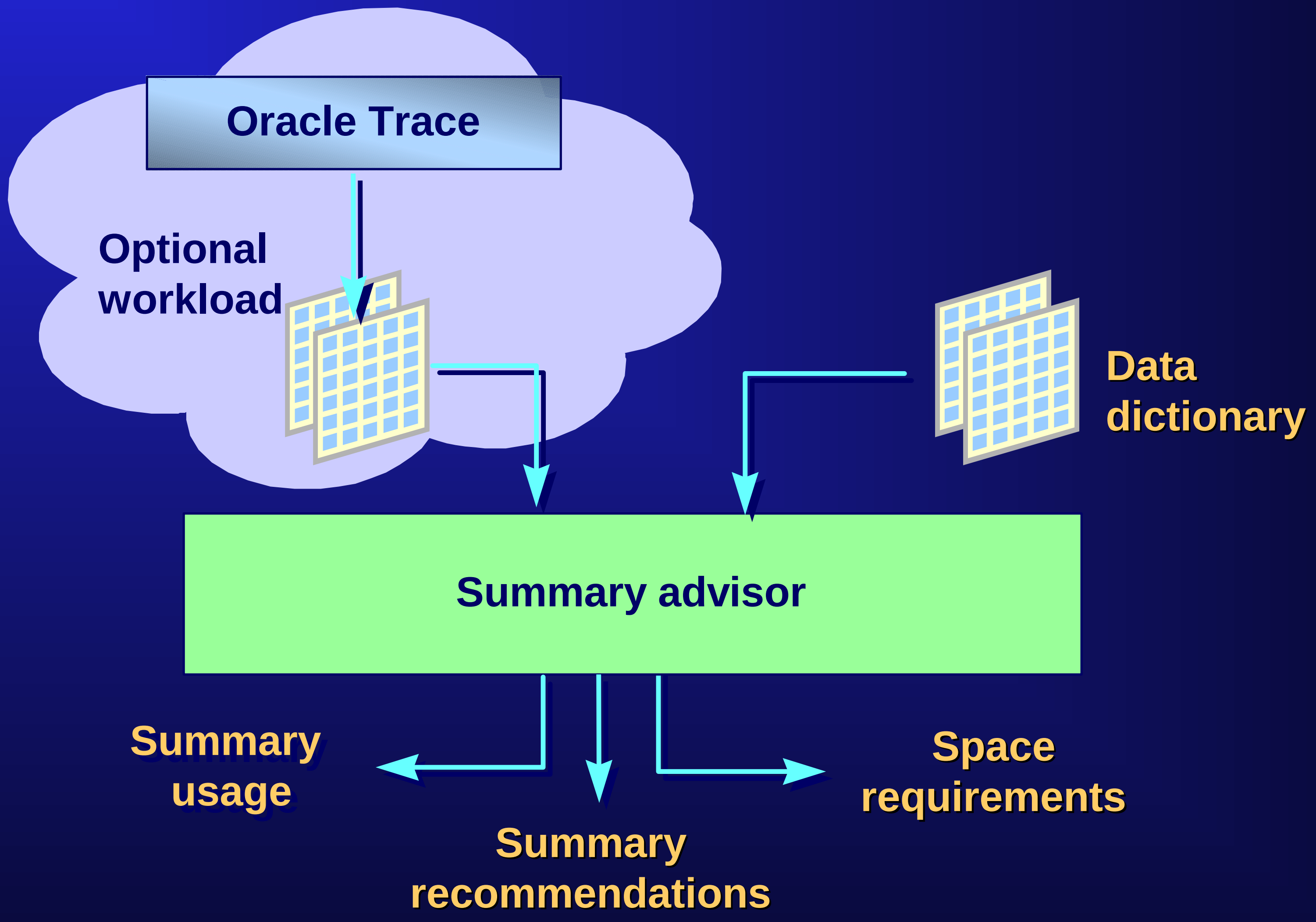
Owner
Dimension
Check that levels are non-null
Check all rows, not just rows changed by direct-path load

- Check hierarchical, attribute, and join relationships
- Updates `MVIEW$_EXCEPTIONS` with type of relationship and ROWIDs of violating rows

Dimensions: Data Dictionary

- `DBA_DIMENSIONS`
- `DBA_DIM_LEVELS`
- `DBA_DIM_LEVEL_KEY`
- `DBA_DIM_ATTRIBUTES`
- `DBA_DIM_HIERARCHIES`
- `DBA_DIM_CHILD_OF`
- `DBA_DIM_JOIN_KEY`

Summary Advisor



Oracle Trace

Optional workload

Data dictionary

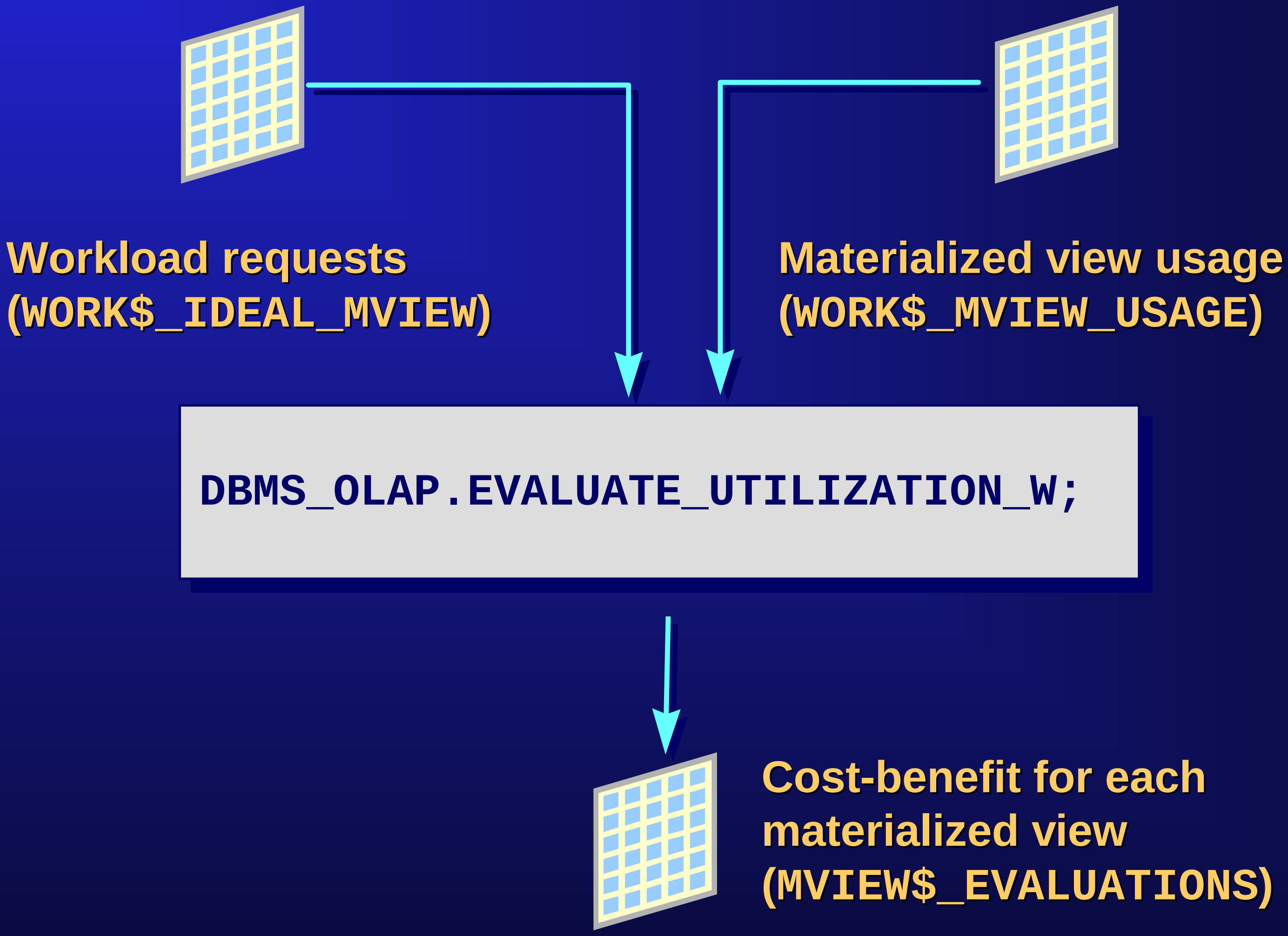
Summary advisor

Summary usage

Summary recommendations

Space requirements

Utilization of Existing Summaries



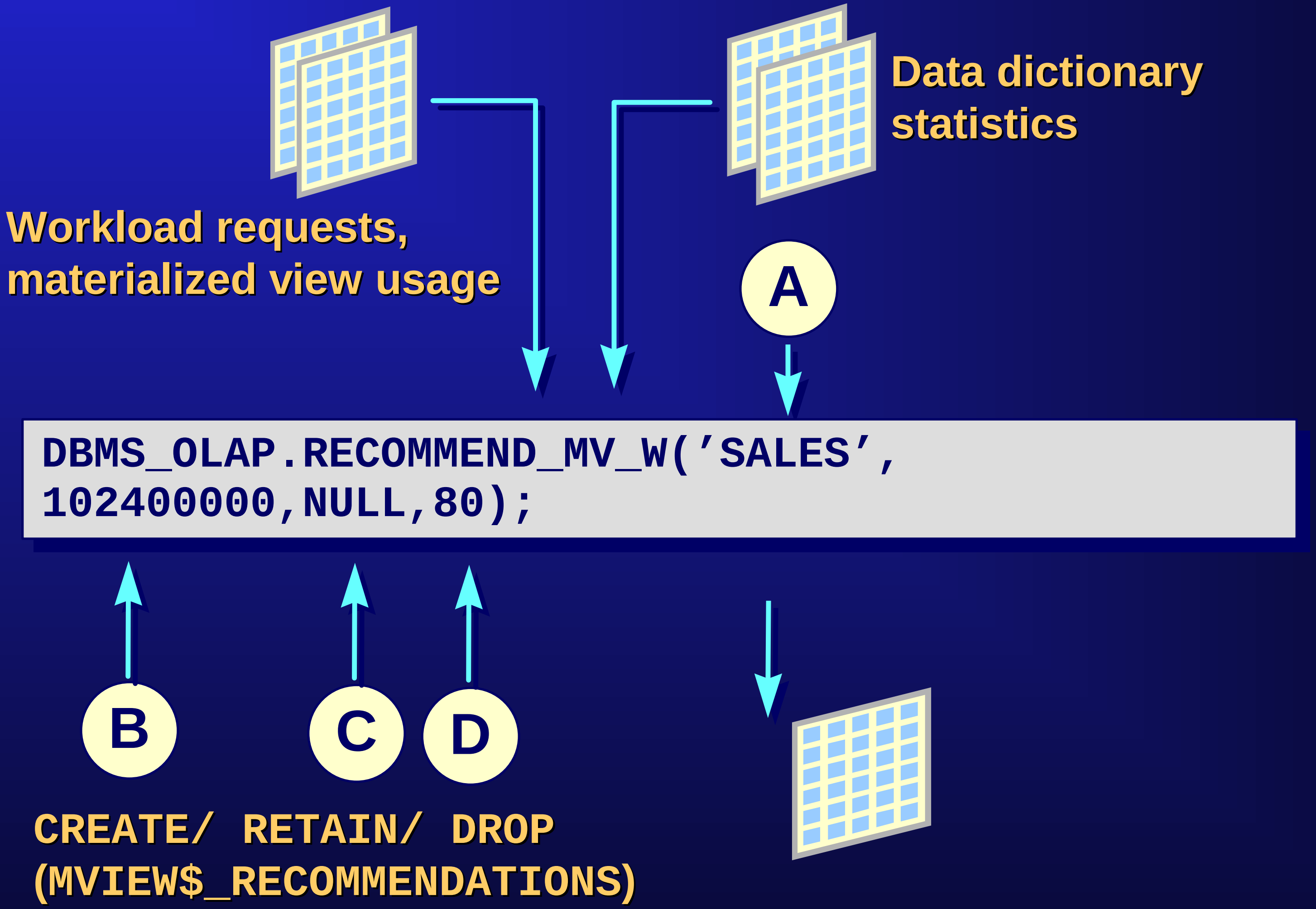
Workload requests
(WORK$_IDEAL_MVIEW)

Materialized view usage
(WORK$_MVIEW_USAGE)

```
DBMS_OLAP.EVALUATE_UTILIZATION_W;
```

Cost-benefit for each
materialized view
(MVIEW$_EVALUATIONS)

Obtaining Summary Recommendations



Data dictionary statistics

Workload requests, materialized view usage

A

```
DBMS_OLAP.RECOMMEND_MV_W('SALES',
102400000,NULL,80);
```

B

C

D

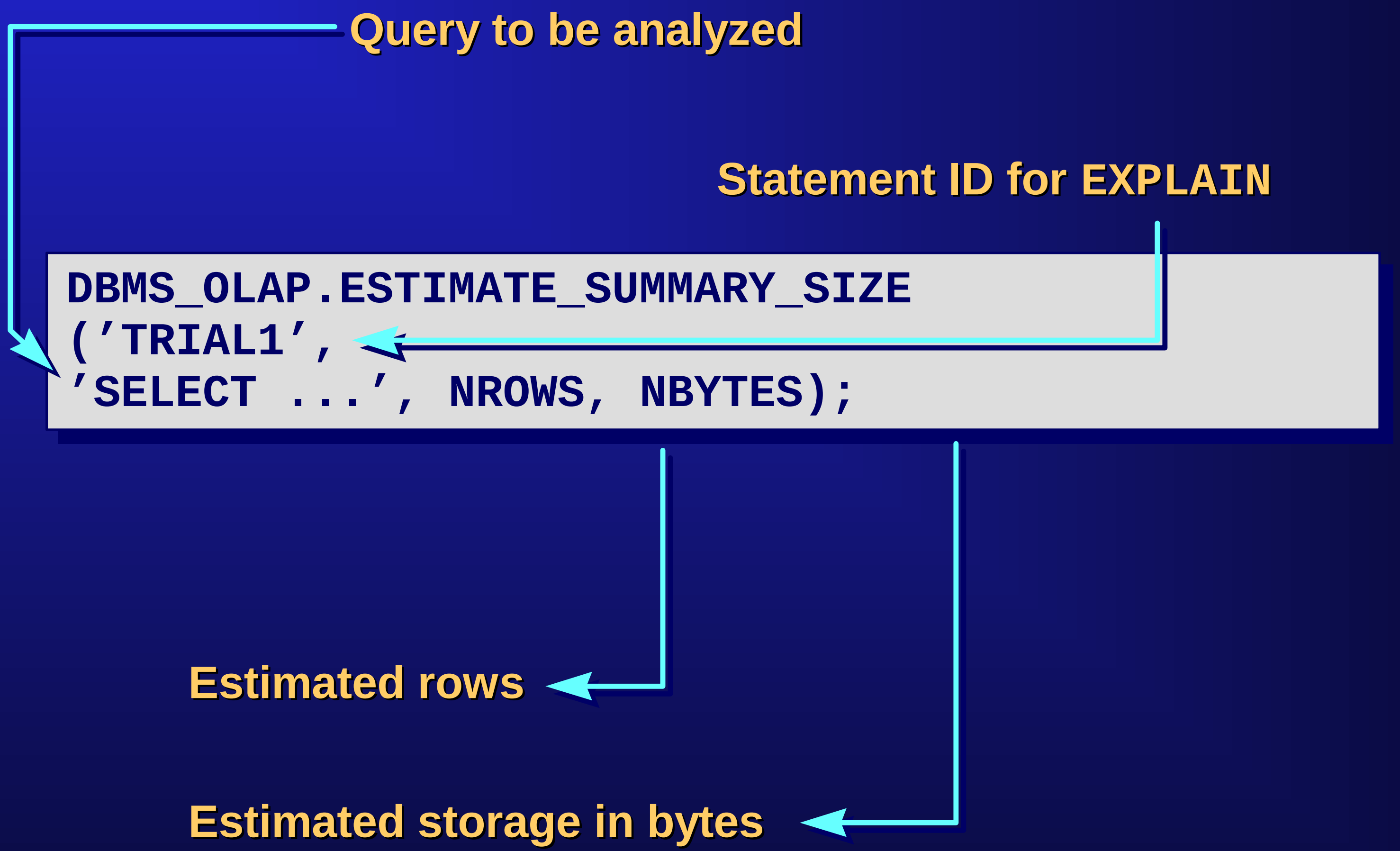
CREATE/ RETAIN/ DROP
(MVIEW$_RECOMMENDATIONS)

Viewing Recommendations

```
SELECT recommended_action, mview_name
FROM mview$_recommendations;
```

```
RECOMMENDED_ACTION MVIEW_NAME
------ ----------- ----------------
RETAIN             SALES_SUMRY
DROP               COMM_SUMRY
RETAIN             BRAND_SUMRY
...
CREATE
```

Estimating Storage Requirements



Query to be analyzed

Statement ID for EXPLAIN

```
DBMS_OLAP.ESTIMATE_SUMMARY_SIZE
('TRIAL1',
'SELECT ...', NROWS, NBYTES);
```

Estimated rows

Estimated storage in bytes

Summary

In this lesson, you should have learned the following:

- Materialized views significantly improve query execution:
  - Data is stored and manually or automatically refreshed synchronously or asynchronously
  - Summarized data is stored and used in combination with dimensions and hierarchies
  - The query is automatically rewritten by the optimizer

Summary

- Query rewriting can be enabled or disabled:
  - At the instance level
  - At the session level
  - At the statement level
  - At the object level

Summary

- A summary advisor helps to:
  - Collect summary usage statistics
  - Provide recommendations to create, retain, or drop summaries
  - Provide space estimation for possible new summaries
- Utilities such as `SQL*Loader`, `Export`, and `Import` support this new feature

Practice 4 Overview

- Creating materialized views
- Enabling query rewrites for materialized views
- Comparing resource consumption of queries against base tables and materialized views
- Investigating storage requirements for materialized views

5

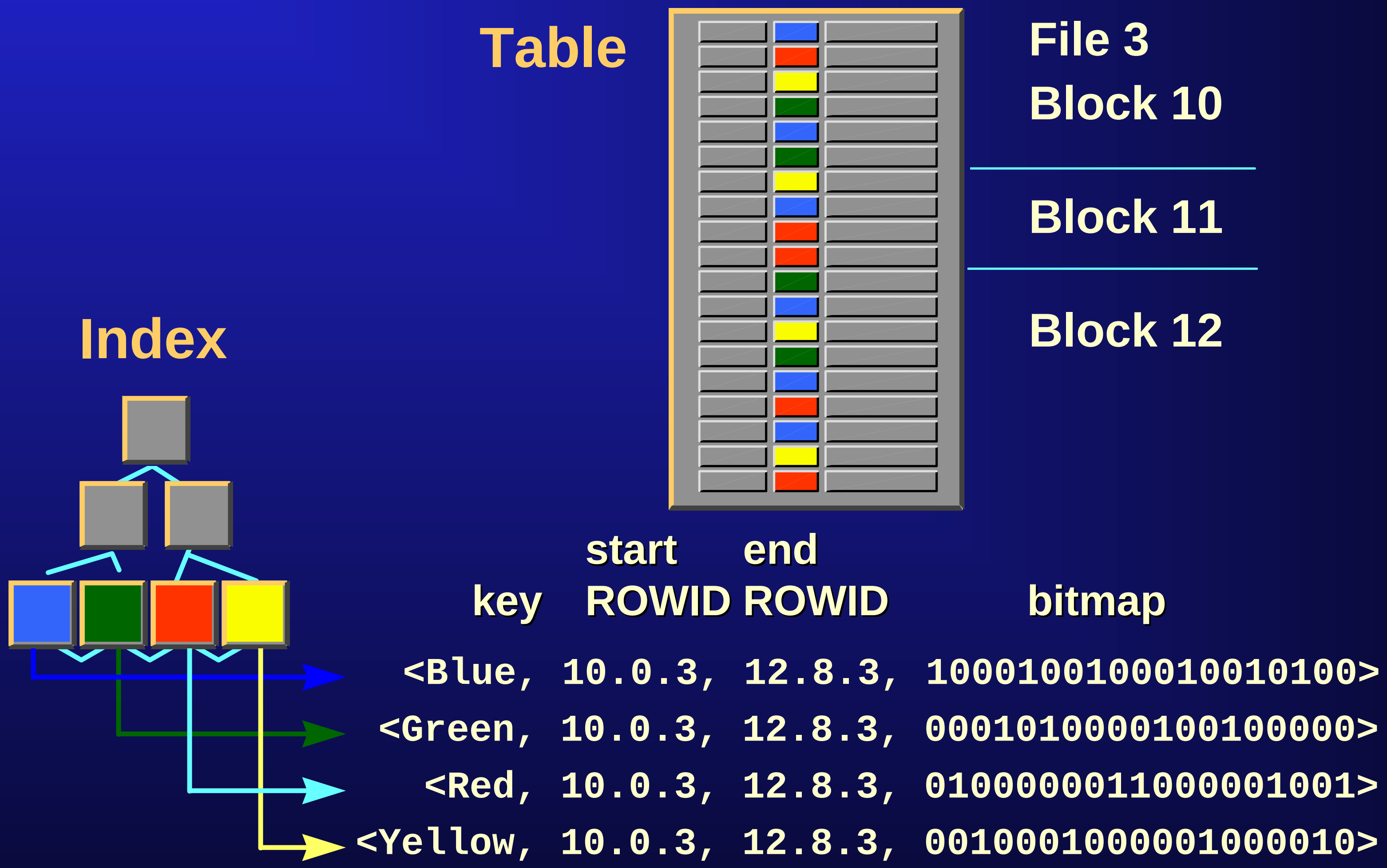
Indexes and Index-Organized Tables

Objectives

After completing this lesson, you should be able to do the following:

- Describe bitmap indexes improvements
- Describe a function-based index
- Build an index online
- Compute index statistics
- Describe an index-organized table (IOT)
- Explain logical ROWIDs
- Create multiple indexes on an IOT
- Explain how to partition an IOT

Bitmap Index



```
        start    end
key     ROWID    ROWID          bitmap
 <Blue, 10.0.3, 12.8.3, 1000100100010010100>
<Green, 10.0.3, 12.8.3, 0001010000100100000>
  <Red, 10.0.3, 12.8.3, 0100000011000001001>
<Yellow, 10.0.3, 12.8.3, 0010001000001000010>
```

Bitmap Index Improvements

- Bitmaps are compressed in bitmap indexes
- Compression/decompression algorithm has been improved:
  - DML is more feasible (smaller bitmaps)
  - Allows bitmaps of higher cardinality than in Oracle 7.3
- Bitmap indexes can be partitioned (local index only)

Function-Based Indexes (B*tree or Bitmap)

- Dramatically improves query performance

```
CREATE INDEX sales_city_margin_idx
     ON sales(city_name ASC,
           (revenue - cost) DESC));
```

- Queries using expressions can use the index

```
SELECT city_name, ordid,
       (revenue - cost) AS MARGIN     FROM sales
ORDER BY city_name ASC, margin DESC;
```

Rebuilding Indexes Online

- Rebuilding indexes and index-organized tables can be done with minimal table locking
- Helps achieve goal of 7 x 24 availability
- Consistency is maintained in the new index while DML is performed on base table
- Works for indexes on columns and the primary structure for index-organized tables

The Rebuilding Indexes Online Process

- Processing occurs in three stages:
  - Prepare
  - Build
  - Merge
- Can run in parallel; is not restartable
- This operation may take a significant amount of space; possibly more than double the space of the existing index

Rebuilding Indexes Online: Examples

```
CREATE INDEX ord_idx ON orders(ord_id) ONLINE;
```

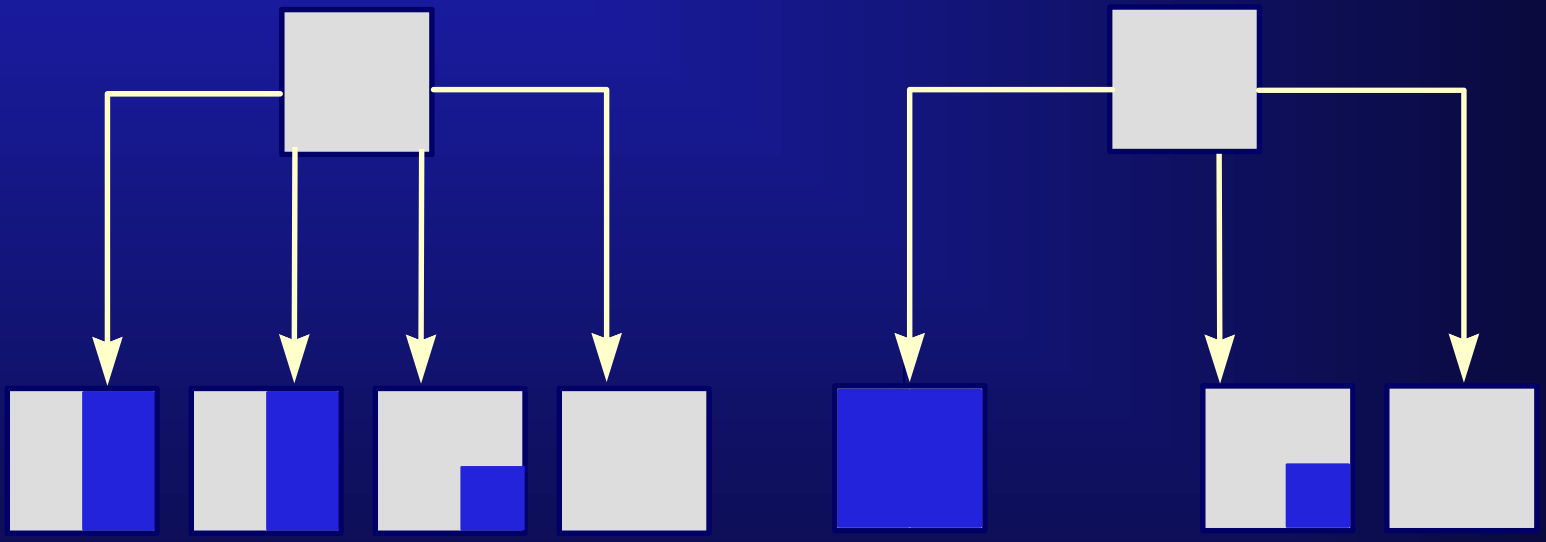
```
ALTER INDEX ord_idx REBUILD ONLINE;
```

```
ALTER TABLE iot_ord MOVE ONLINE;
```

Coalescing Free Space in Indexes

```
ALTER INDEX ord_idx COALESCE;
```

PCTFREE=50%



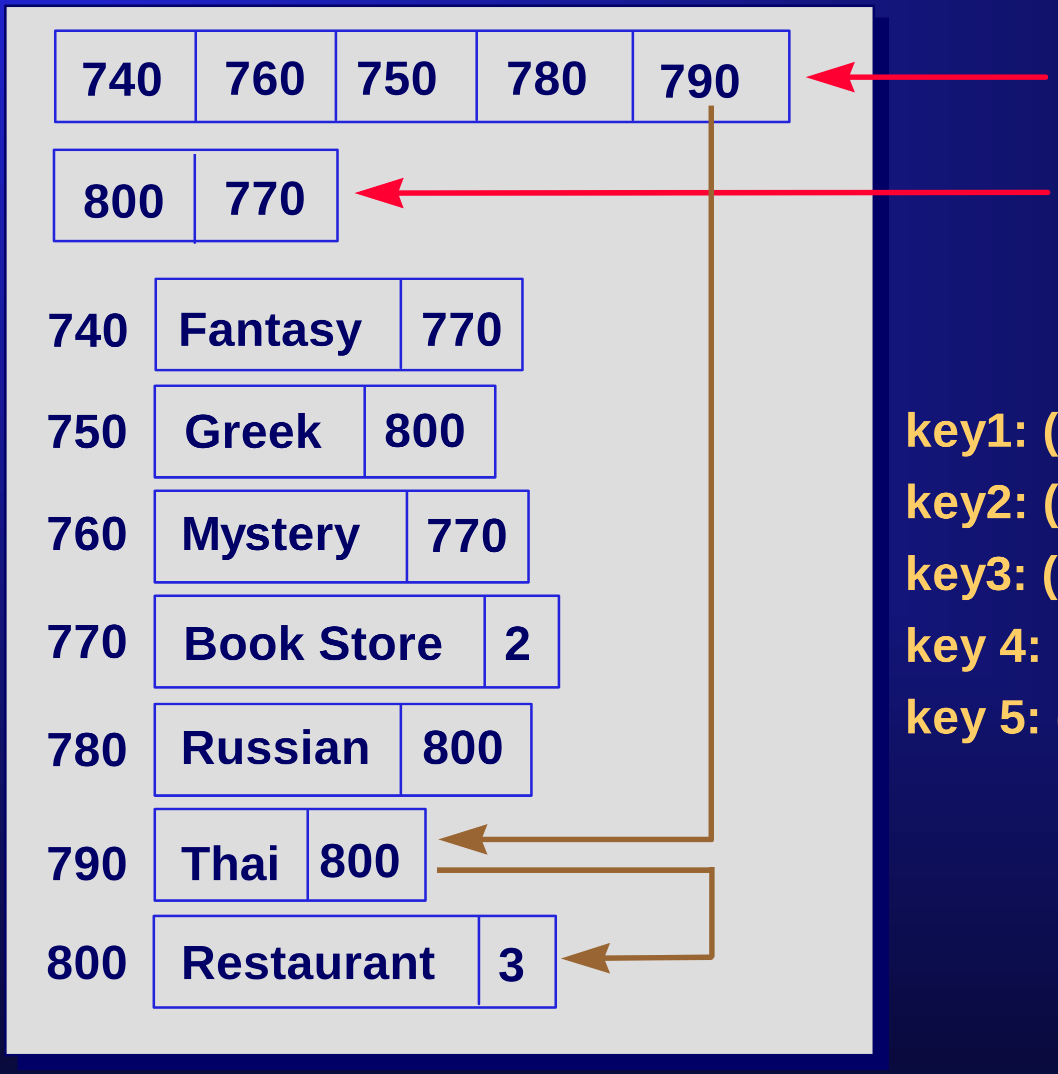
Before

After

Computing Statistics on Indexes

- The `COMPUTE STATISTICS` clause has been added to the `CREATE INDEX` and `ALTER INDEX REBUILD` commands
- Requests generation of statistics when an index is created
- If the index is composite, statistics are for the leading column only

B*Tree Key Compression



740	760	750	780	790

Suffix slot array

800	770

Prefix slot array

740	Fantasy	770
750	Greek	800
760	Mystery	770
770	Book Store	2
780	Russian	800
790	Thai	800
800	Restaurant	3

key1: (Restaurant, Thai)

key2: (Restaurant, Russian)

key3: (Book Store, Mystery)

key 4: (Restaurant, Greek)

key 5: (Book Store, Fantasy)

B*Tree Key Compression Maintenance

- **ALTER TABLE MOVE COMPRESS [*n*]**
- **ALTER INDEX REBUILD NOCOMPRESS**
- **CREATE INDEX**
- **CREATE TABLE**
- **PARTITION MAINTENANCE**
- **DBA_IND_PARTITIONS**
- **DBA_INDEXES**

Row Overflow in IOTs

- You can store nonkey columns in an overflow area
- You can specify:
  - The overflow tablespace
  - A threshold size for overflow rows
  - The column name where the row is split
- `UTLCHN1.SQL` (new in 8.1.5) or `DBMS_IOT.BUILD_CHAIN_ROWS_TABLE` to store chained rows

Example: Index-Organized Tables

```
ALTER SESSION SET NLS_DATE_FORMAT=‘DD-MON-YYYY’;
```

```
SQL> CREATE TABLE ord_over_iot
(id NUMBER, odate DATE, amount number,
notes VARCHAR2(1000),
PRIMARY KEY(id, odate))
     ORGANIZATION INDEX INCLUDING amount
     PCTTHRESHOLD 20
     OVERFLOW TABLESPACE all_overflow
PARTITION BY RANGE(odate)
(PARTITION p1 VALUES LESS THAN (‘01-FEB-1998’)
 TABLESPACE q1,
 PARTITION p2 VALUES LESS THAN (MAXVALUE)
 TABLESPACE q2);
```

Dictionary Views

```
select table_name, tablespace_name, iot_name, iot_type
from dba_tables;
TABLE_NAME          TABLESPACE_NAME    IOT_NAME     IOT_TYPE
------------------  ---------------   --------     --------------
SALES                                                IOT
SYS_IOT_OVER_2268   USER_DATA          SALES         IOT_OVERFLOW
```

```
select index_name,index_type,tablespace_name,table_name
from dba_indexes;
INDEX_NAME                     INDEX_TYPE   TABLESPACE TABLE_NAME
------------------------------ ------------ ---------- -----------
SYS_IOT_TOP_2268               IOT - TOP    INDX       SALES
```

```
select segment_name,tablespace_name,segment_type
from dba_segments;
SEGMENT_NAME         TABLESPACE          SEGMENT_TYPE
------------------   ----------          ----------------
SYS_IOT_OVER_2268    USER_DATA           TABLE
SYS_IOT_TOP_2268     INDX                INDEX
```

Dictionary Views

```
select table_name, tablespace_name, iot_name, iot_type
from dba_tables;
TABLE_NAME          TABLESPACE_NAME    IOT_NAME     IOT_TYPE
------------------  ---------------    --------     --------------
SALES                                                IOT
SYS_IOT_OVER_2268   USER_DATA          SALES         IOT_OVERFLOW
```

```
select index_name,index_type,tablespace_name,table_name
from dba_indexes;
INDEX_NAME                     INDEX_TYPE   TABLESPACE TABLE_NAME
------------------------------ ------------ ---------- ------------
SYS_IOT_TOP_2268               IOT - TOP    INDX       SALES
```

```
select segment_name,tablespace_name,segment_type
from dba_segments;
SEGMENT_NAME         TABLESPACE           SEGMENT_TYPE
------------------   ----------           ----------------
SYS_IOT_OVER_2268    USER_DATA            TABLE
SYS_IOT_TOP_2268     INDX                 INDEX
```

Restrictions on Index-Organized Tables

- Must have a primary key
- Cannot use unique constraints
- Cannot be clustered
- Cannot contain `LONG` columns
- Distribution and replication not supported

Restrictions on Index-Organized Tables

- **Cannot create an IOT of object types**
- **An IOT can contain columns of LOB and nested table types, but only if the table is not partitioned**
- **IOTs must be reorganized using the`MOVE` clause of the `ALTER  TABLE` command**
- **Use `UTLEXPT1.SQL` to create the `EXCEPTIONS` table**

Logical ROWIDs

- Provide fastest access to rows in index-organized tables
- Based on the primary key value of a row and an optional “guess”
- Accessed via the Universal ROWID (`UROWID`) datatype
- PL/SQL includes `UROWID` support
- Used to create secondary indexes on index-organized tables

Multiple Indexes on Index-Organized Tables

- An IOT can have additional indexes
- The index contains a key value and a logical ROWID
- Cannot be bitmap Indexes
- Cannot be reversed
- Cannot use the NOSORT option

Summary

In this lesson, you should have learned the following:

- Bitmap index performance has been improved
- Function-based indexes can now be created
- Indexes can be rebuilt online
- Index statistics can be gathered as the index is created
- Logical ROWIDs point to rows in IOTs
- IOTs can have secondary indexes

Practice 5 Overview

This practice covers the following topics:

- Collecting statistics while creating an index
- Creating a function-based index
- Creating an index-organized table
- Creating a secondary index on an index-organized table

6

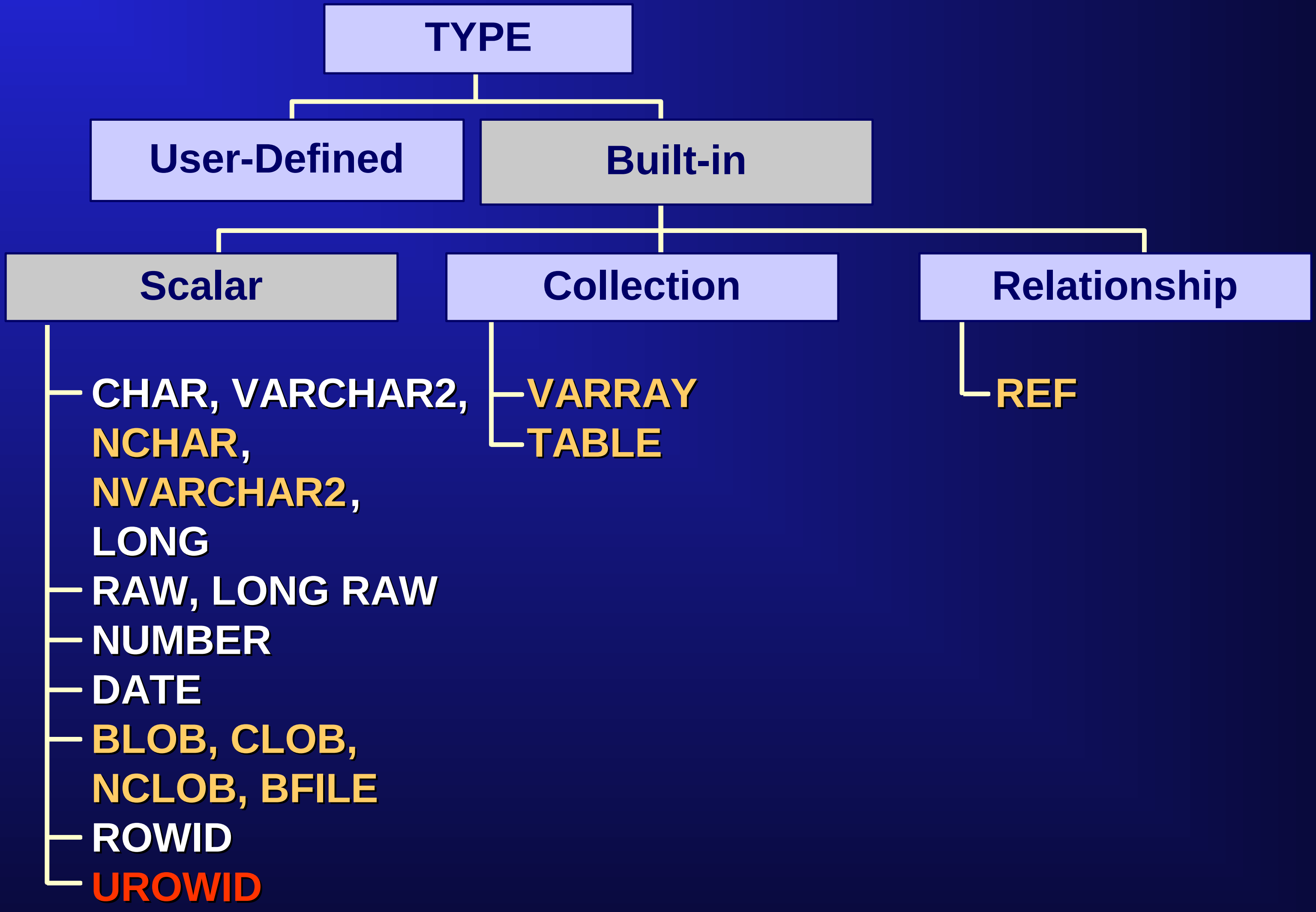
LOBs

Objectives

After completing this lesson, you should be able to:

- Define LOBs from a DBA perspective
- Understand Temporary LOBs

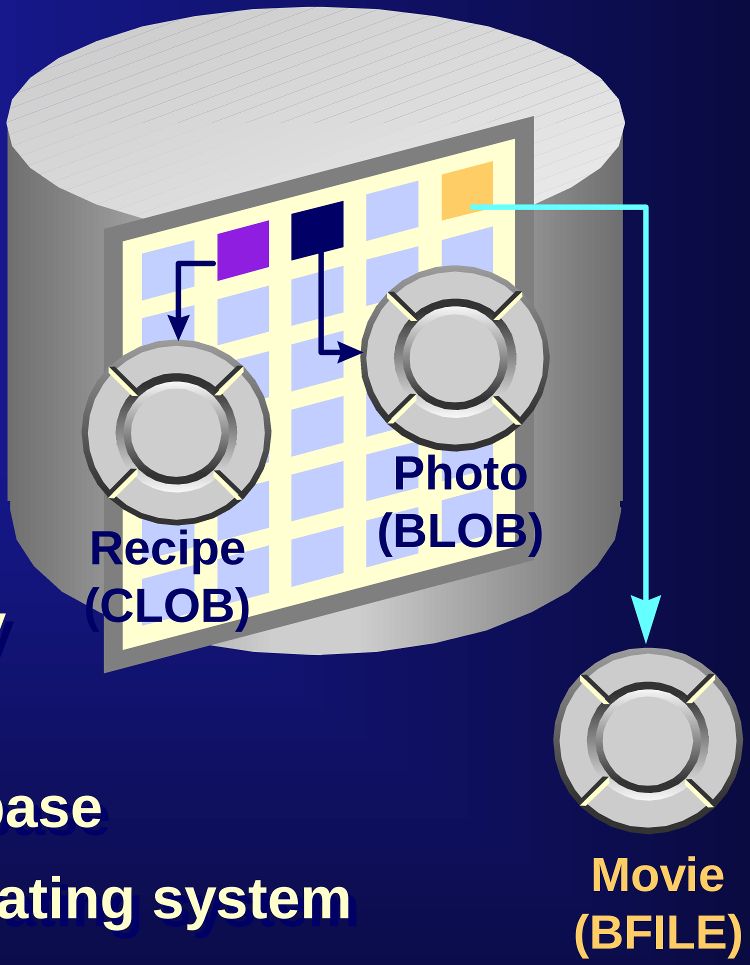
Hierarchy of Attribute Types



- TYPE
  - User-Defined
  - Built-in
    - Scalar
      - CHAR, VARCHAR2, NCHAR, NVARCHAR2, LONG
      - RAW, LONG RAW
      - NUMBER
      - DATE
      - BLOB, CLOB, NCLOB, BFILE
      - ROWID
      - UROWID
    - Collection
      - VARRAY
      - TABLE
    - Relationship
      - REF

LOB Overview

- LOB storage:
  - Unstructured data
  - Binary or character
  - Size to 4 GB
  - Special storage
  - Special concurrency
- Storage method:
  - Internal, in the database
  - External, in the operating system



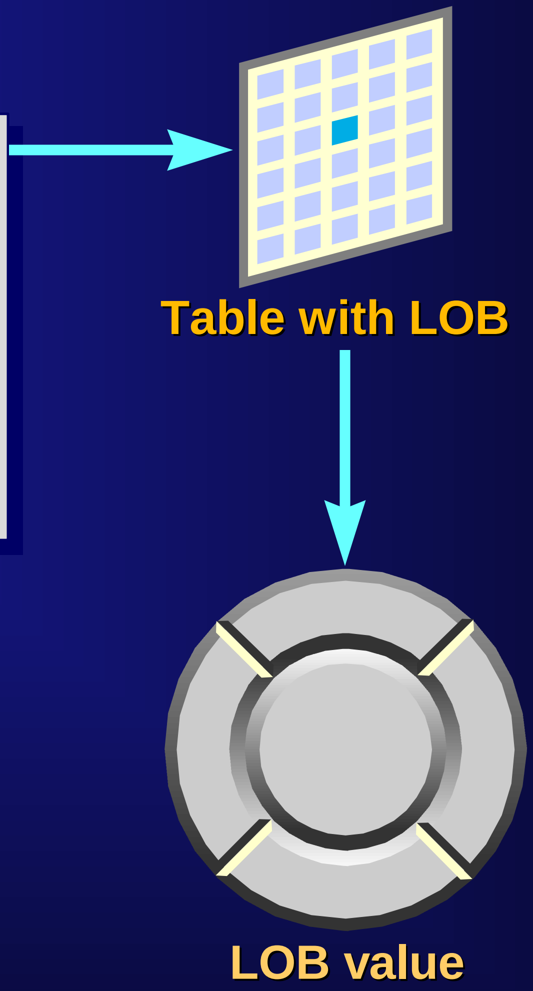
Contrasting LONG and LOB Data Types

LONG, LONG RAW	LOB
Single column per table	Multiple columns per table
Up to 2 gigabytes	Up to 4 gigabytes
SELECT returns data	SELECT INTO returns locator
Data always stored in-line	Data stored in-line or out-of-line
Cannot be an object attribute	Can be an object attribute
Cannot be partitioned	Can be partitioned
Cannot be used in IOTs	Can be used in IOTs
No replication	Can be replicated
Sequential access to chunks	Random access to chunks
Limited PL/SQL support	Extensive PL/SQL support

Characteristics of a LOB

Program with LOB locator

```
DECLARE
  jobloc BLOB;
BEGIN
  SELECT description INTO jobloc
   FROM job_table
   WHERE title='Analyst';
END;
```



Two distinct parts of a LOB:

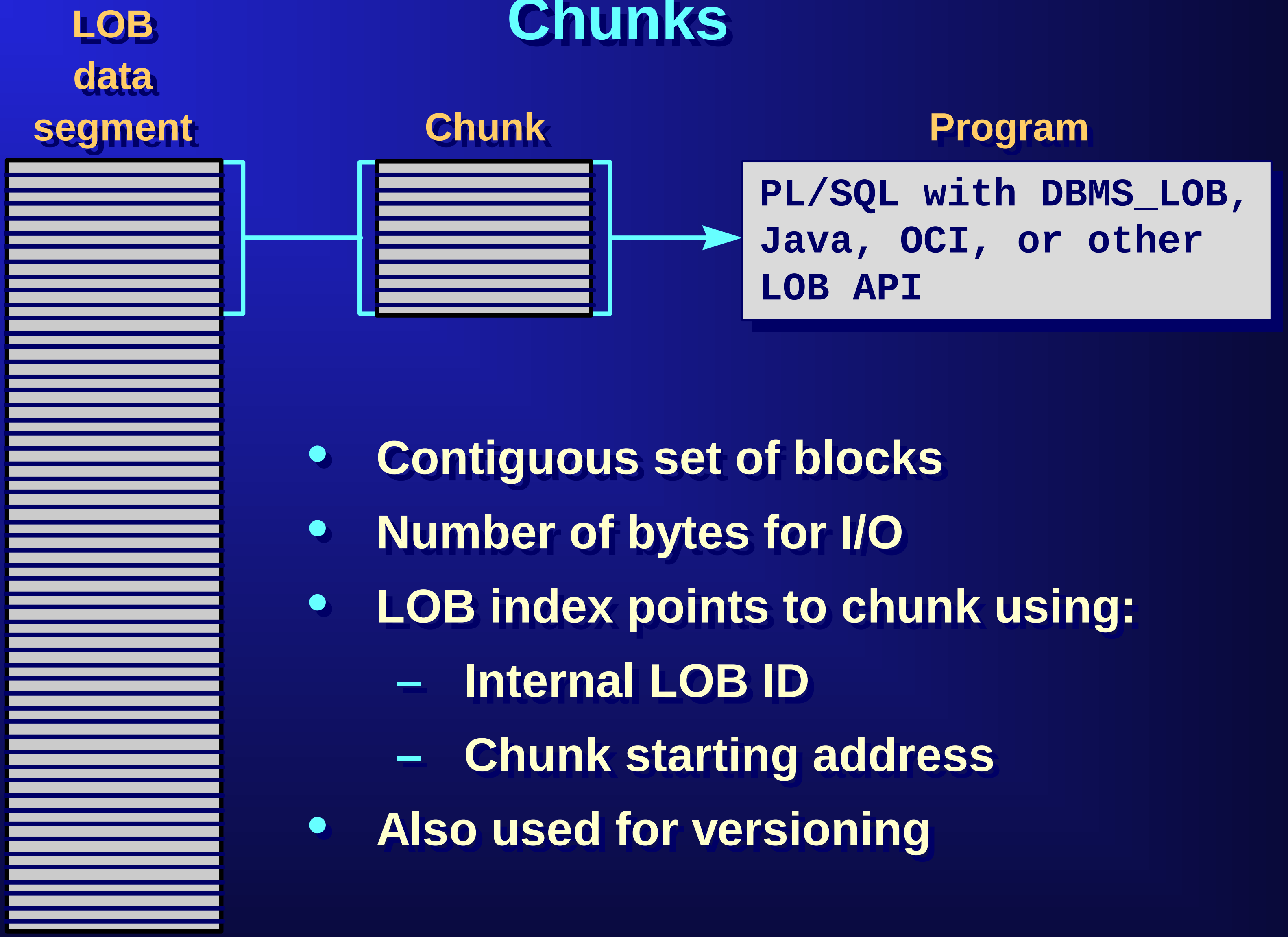
- Locator is a pointer to the LOB
- Value is the actual data

Internal LOB Storage

- LOB value can be stored:
- In-line
  - Stored with the other row data
  - Only if 4000 bytes or less
- Out-of-line
  - Stored in a separate segment
  - LOB index segment accesses LOB values
  - LOB data segment stores LOB values

Internal LOBs use copy semantics

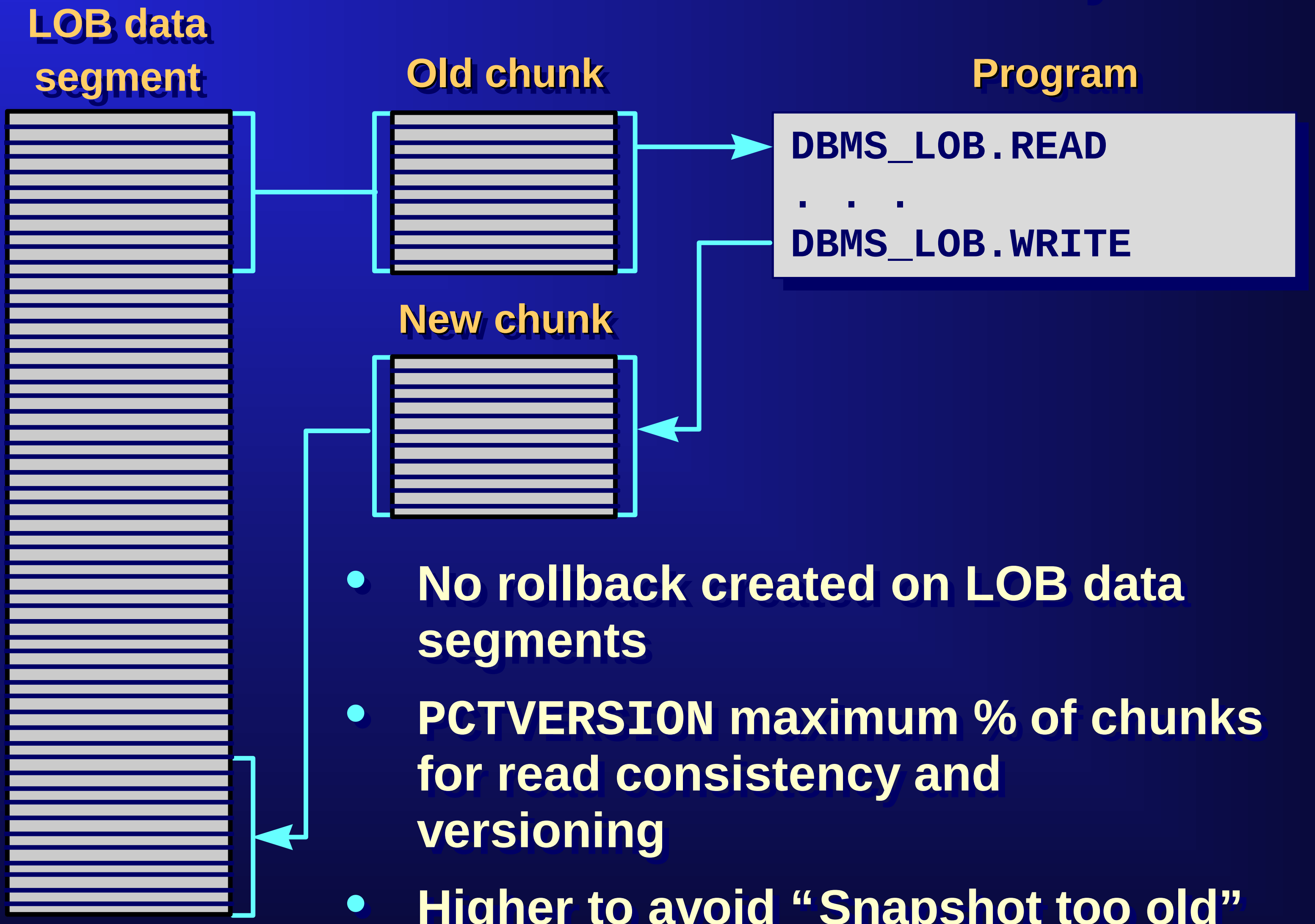
Chunks



PL/SQL with DBMS_LOB, Java, OCI, or other LOB API

- Contiguous set of blocks
- Number of bytes for I/O
- LOB index points to chunk using:
  - Internal LOB ID
  - Chunk starting address
- Also used for versioning

Internal LOB Read Consistency



```
DBMS_LOB.READ
. . .
DBMS_LOB.WRITE
```

- No rollback created on LOB data segments
- PCTVERSION maximum % of chunks for read consistency and versioning
- Higher to avoid “Snapshot too old”

Guidelines for Internal LOBs

- `CHUNK`: Use a multiple of chunk size when:
  - Setting `INITIAL` and `NEXT` extent sizes
  - Manipulating LOBs
- Use the same chunk size for database I/O and network traffic
- The chunk size can be queried using an OCI or `DBMS_LOB` function

Guidelines for Internal LOBs

- `PCTVERSION`: Set depending on LOB access: Zero if read-only
- Low if:
  - Updates and reads not concurrent
  - Written once and then read-only
- High if:
  - Large queries
  - Heavy write and read activity

Guidelines for Internal LOBs

- CACHE , NOCACHE , and CACHE READS options
  - Use CACHE if reads and updates are frequent
  - Use NOCACHE if never modified
    - Avoids DB buffer cache
    - Out-of-line LOB redo in chunk sizes
  - Use CACHE READ if modified occasionally
- LOGGING / NOLOGGING: Use NOLOGGING when recovery is not required

Guidelines for Internal LOBs

- `DISABLE STORAGE IN ROW` improves performance when other columns are frequently read without the LOB value
- `ENABLE STORAGE IN ROW`:
  - Moved out of line when size > 4000 bytes
  - Improves performance when small LOBs are frequently read with rows
  - Hurts performance of full table scans that do not access LOB value

Temporary LOBs

- Nonpersistent, internal LOBs
- Transient workspace for LOB manipulation
- No rollback or redo logging generated
- Manipulated using DBMS_LOB or other APIs
- Exist for the session, transaction, or call
- Monitored via the data dictionary
- To create a temporary LOB:

```
DBMS_LOB.CREATETEMPORARY(v_lob_loc, true,
   DBMS_LOB.TRANSACTION);
```

LOB Data Dictionary Views

- **DBA_LOBS**
- **DBA_SEGMENTS**
- **DBA_INDEXES**
- **V$TEMPORARY_LOBS**
- **V$SORT_USAGE**

LOB APIs

- SQL DML manipulates the entire LOB value
- For more extensive manipulation, use `DBMS_LOB` or other API
- `DBMS_LOB` package:
  - Created via `dbmslob.sql` and `prvtlob.plb`
  - Two types of procedures:
    - Mutators modify LOBs
    - Observers read LOBs and return properties
- Similar capabilities in precompilers, OCI, OO4O, and Java

Summary

In this lesson, you should have learned the following:

- With Oracle8*i*, users can define LOB data.
- Temporary LOBs can speed up LOBs manipulation.

Practice 6 Overview

This practice covers the following topics:

- Creating tables containing LOBs
- Inserting into tables containing LOBs
- Reading LOBs by using DBMS_LOB
- Querying data dictionary LOBs information
- Creating temporary LOBs
- Querying data dictionary Temporary LOBs information

7

Partitioning Improvements

Objectives

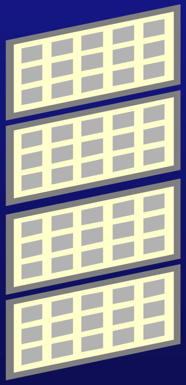
After completing this lesson, you should be able to do the following:

- Revise the general partitioning concepts
- Implement range, hash, and composite partitioning
- Explain `ENABLE/DISABLE ROW MOVEMENT`
- Explain the new partition pruning capabilities
- Describe partition-wise join
- Review partition maintenance operations
- Explain automatic parallel execution

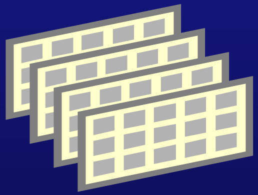
Partitioning Methods

Three partitioning methods are available:

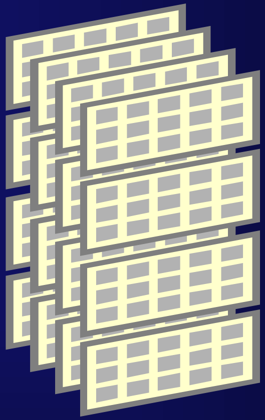
- Range
- Hash
- Composite



Range partitioning



Hash partitioning



Composite partitioning

Range Partitioning Example

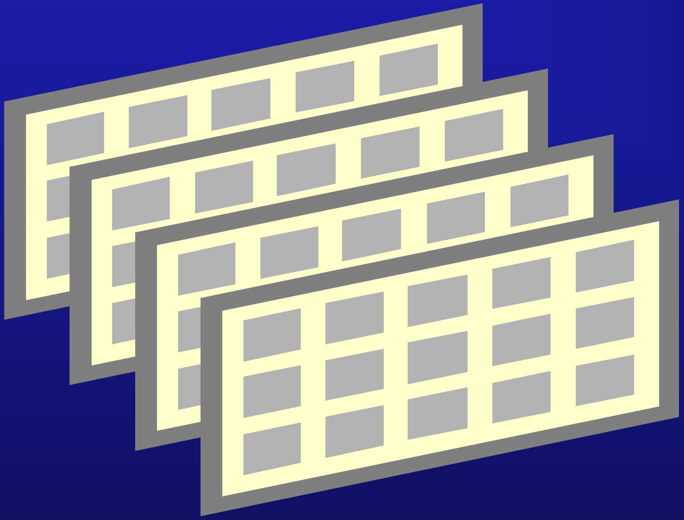
```
CREATE TABLE sales
  (acct_no      NUMBER(5),
   person       VARCHAR2(30),
   sales_amount NUMBER(8),
   week_no      NUMBER(2))     1
   PARTITION BY RANGE (week_no)    2                    3
    (PARTITION P1 VALUES LESS THAN    (4)  TABLESPACE data0,
     PARTITION P2 VALUES LESS THAN    (8)  TABLESPACE data1,
     . . .…
    PARTITION P13 VALUES LESS THAN    (53)TABLESPACE data12
);
```

1 The partition key is week_no.

2 VALUES LESS THAN must be specified as a literal.

3 Physical attributes can be set per partition.

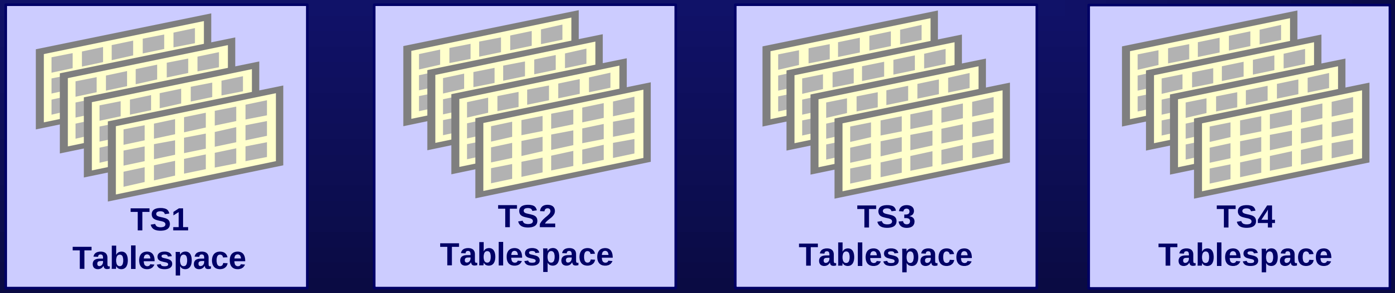
Hash Partitioning Overview



- Easy to implement
- Enables better performance for PDML and partition-wise join
- Inserts rows into partitions automatically based on hash of partition key
- Supports «hash» local indexes
- Does not support «hash» global indexes

Hash Partitioning Example 1

```
CREATE TABLE product
(id     NUMBER(5),
 name   VARCHAR2(30),
 amount NUMBER(5))
     STORAGE (INITIAL 10M)
     PARTITION BY HASH(id) PARTITIONS 16
     STORE IN (ts1,ts2,ts3,ts4);
```



TS1 Tablespace

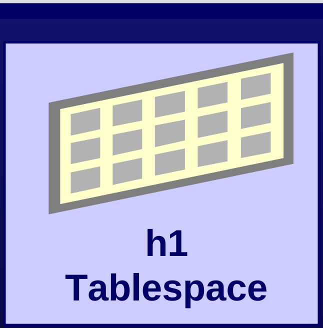
TS2 Tablespace

TS3 Tablespace

TS4 Tablespace

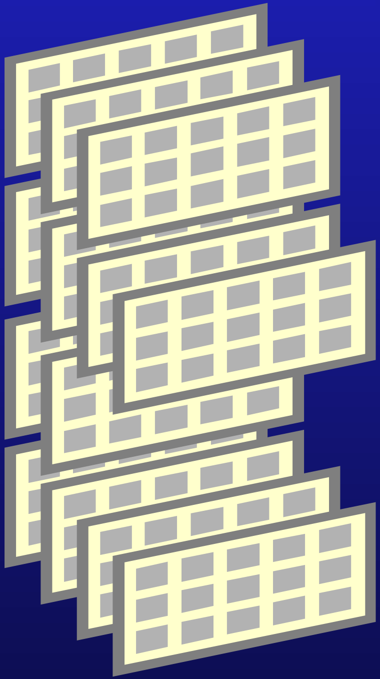
Hash Partitioning Example 2

```
CREATE TABLE product
(id     NUMBER(5),
 name   VARCHAR2(30),
 amount NUMBER(5))
     STORAGE (INITIAL 10M)
     PARTITION BY HASH(id)
      (PARTITION p1 TABLESPACE h1,
       PARTITION p2 TABLESPACE h2);
```



h2
Tablespace

Composite Partitioned Table: Overview



- Ideal for both historical data and data placement
- Provides high availability and manageability, like range partitioning
- Improves performance for parallel DML and supports partition-wise joins
- Allows more granular partition elimination
- Supports composite local indexes
- Does not support composite global indexes

Composite Partitioning Example 1

```
ALTER SESSION SET NLS_DATE_FORMAT='DD-MON-YYYY';
```

```
CREATE TABLE orders(
            ordid                  NUMBER,
            orderdate              DATE,
            productid   NUMBER,
            quantity               NUMBER)
      PARTITION BY RANGE(orderdate)
SUBPARTITION BY HASH(productid) SUBPARTITIONS 8
STORE IN (ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8)
(PARTITION q1 VALUES LESS THAN('01-APR-1998'),
 PARTITION q2 VALUES LESS THAN('01-JUL-1998'),
 PARTITION q3 VALUES LESS THAN('01-OCT-1998'),
 PARTITION q4 VALUES LESS THAN(MAXVALUE));
```

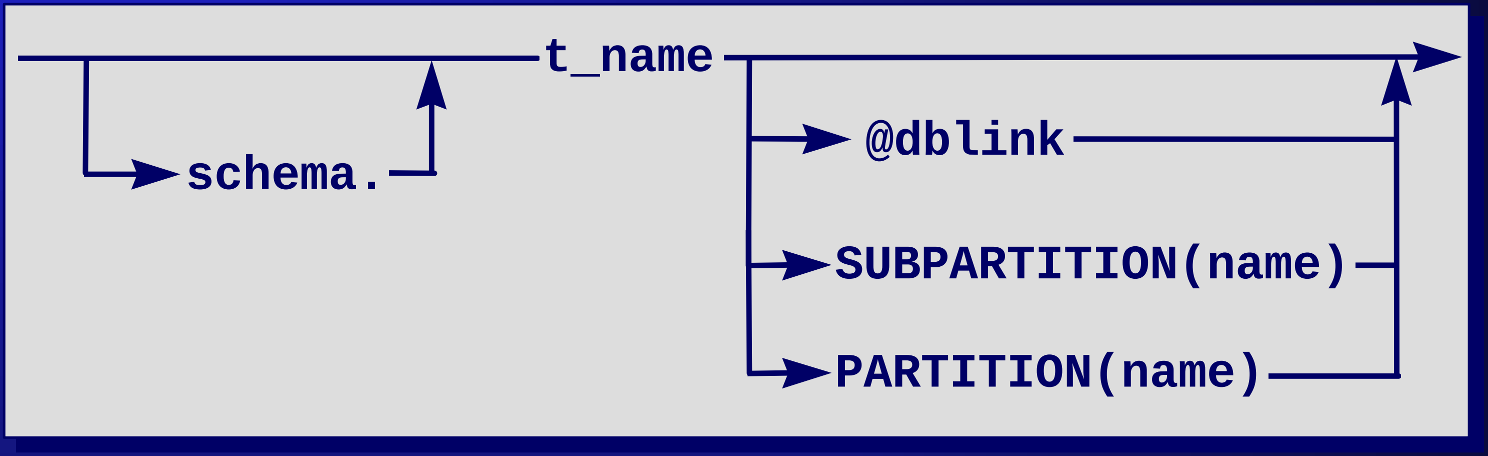
Composite Partitioning Example 2

Table-level and partition-level default attributes:

```
CREATE TABLE orders (
 ordid NUMBER,
 orderdate DATE,
 prodid NUMBER)
PARTITION BY RANGE (orderdate)
   SUBPARTITION BY HASH(prodid)SUBPARTITIONS 8
   STORE in (ts1,ts2,ts3,ts4)
 (PARTITION p1 VALUES LESS THAN (‘01-APR-1998’) PCTFREE 40,
  PARTITION p2 VALUES LESS THAN (‘01-JUL-1998’)
    STORE IN (ts5,ts6,ts7,ts8),
  PARTITION p3 VALUES LESS THAN (‘01-OCT-1998’)
    (SUBPARTITION S1, SUBPARTITION S2),
  PARTITION p4 VALUES LESS THAN (MAXVALUE) SUBPARTITIONS 6);
```

ORACLE®

Partition and Subpartition Extended Table Names



```
SELECT * FROM ORDERS PARTITION(Q1) ;
```

- Names for subpartitions and partitions must be unique within the table or index
- Extended table names can be used in all DML statements.

Updatable Partition Keys

- A partitioned table can be created or altered to allow row movement between partitions:

```
ALTER TABLE sales ENABLE ROW MOVEMENT;
```

- This clause can only be applied to partitioned tables.
- Disabled is the default behavior.

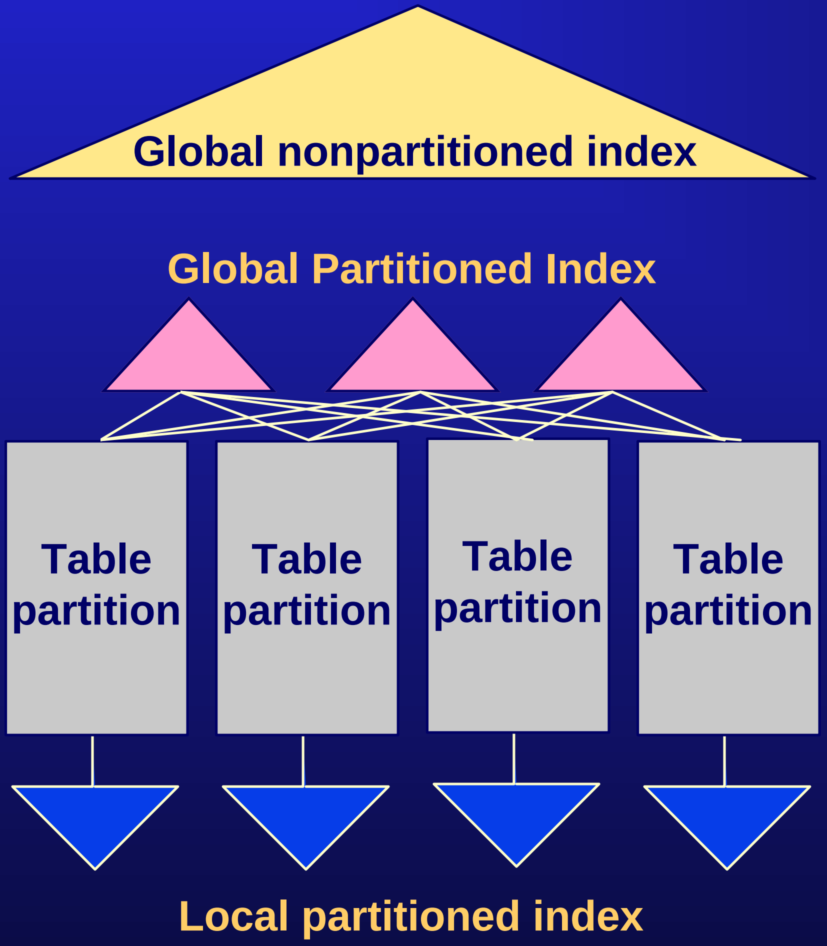
Equipartitioning

S R	Hash(C1,n3)	Range(C2,n4)	Composite Range(C2,n4)/ Hash(C1,n3)
Hash(C1,n1)	n1=n3	🟦	n1=n3
Range(C2,n2)	🟦	n2=n4	n2=n4

**Composite
Range(C2,n2)/
Hash(C1,n1)**
	n1=n3	n2=n4	n2=n4
			or
			n1=n3

`Partitioning_Method(Partition_key,Number_of_fragments)`

Partitioned Indexes for Scalable Access



- Indexes can be partitioned, delivering improvements in:
  - Manageability
  - Availability
  - Performance and scalability
- Enables parallel index scans
- Choices in index configuration:
  - Local prefixed index
  - Local nonprefixed index
  - Global prefixed index
  - Global nonpartitioned index
- Flexibility to suit a variety of access patterns, index sizes

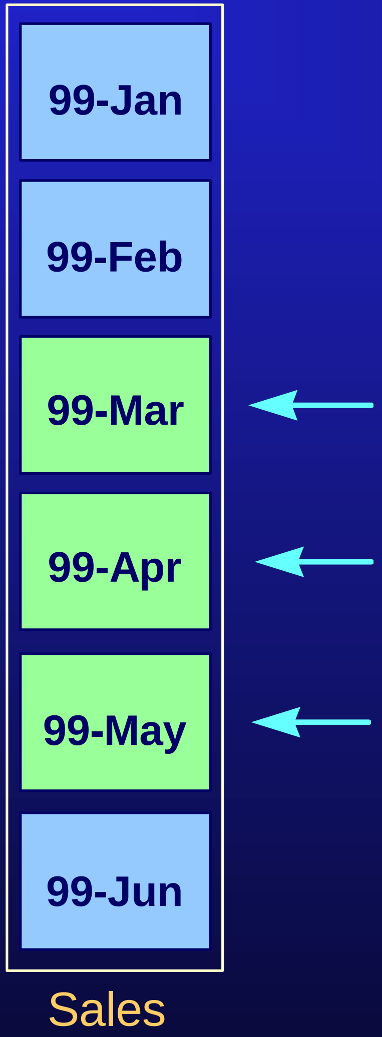
Composite Partitioned Indexes

- Composite partitioned indexes are always local and stored to the table subpartition by default.

```
CREATE INDEX order_ind
ON orders(orderdate, productid) LOCAL;
```

- If required, tablespaces can be specified at either the index or index subpartition levels.
- Range partitioned global indexes on composite partitioned tables are supported.

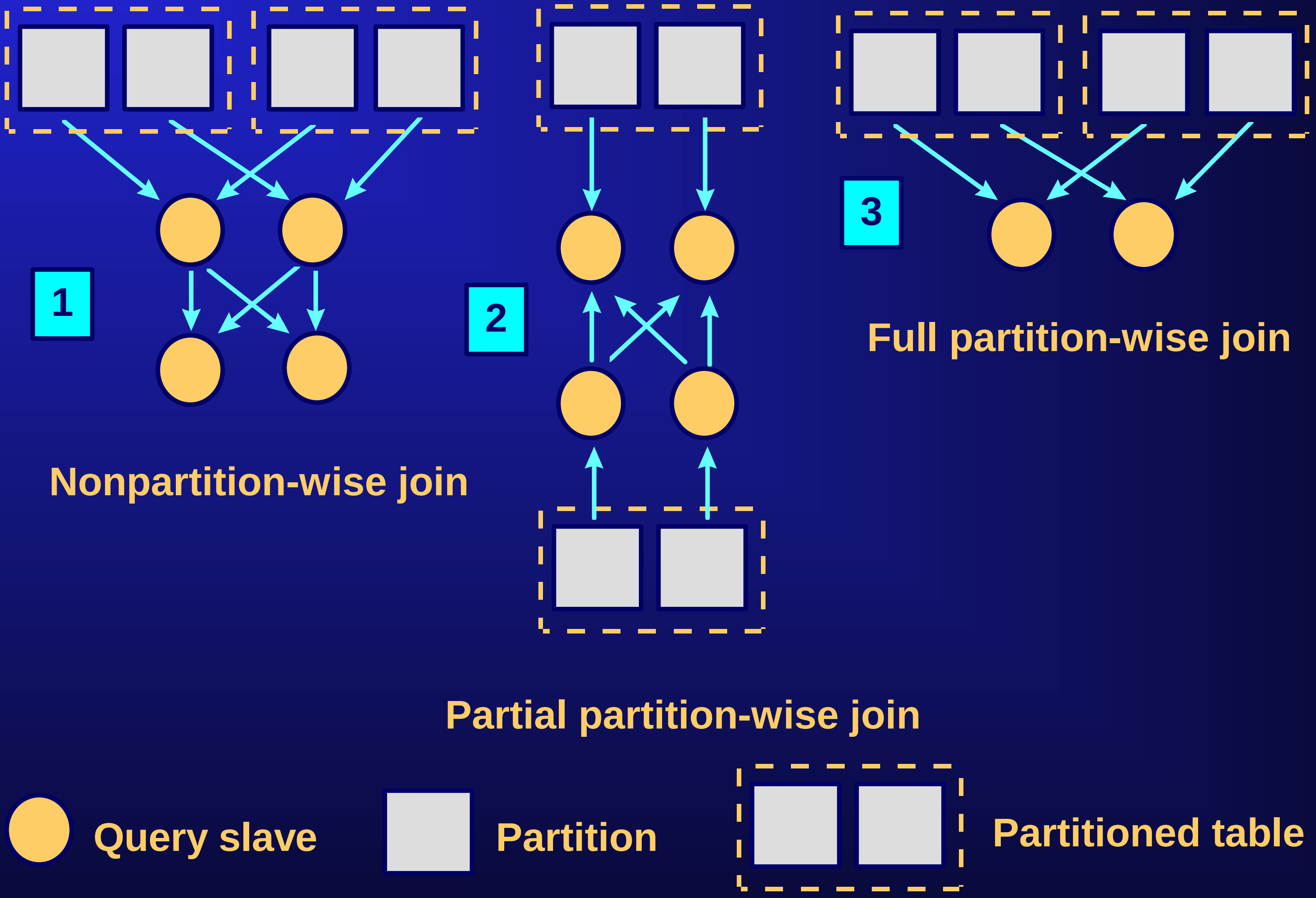
Partition Pruning



Partition pruning: only the relevant partitions are accessed

```
SQL> SELECT SUM(sales_amount)
      2      FROM sales
  3  WHERE sales_date BETWEEN
  4  TO_DATE(‘01-MAR-1999’,
  5          ‘DD-MON-YYYY’) AND
  6  TO_DATE(‘31-MAY-1999’,
  7          ‘DD-MON-YYYY’);
```

Partition-Wise Join



1

Nonpartition-wise join

2

Partial partition-wise join

3

Full partition-wise join

Query slave

Partition

Partitioned table

Statistics Collection for Partitioned Objects

- You can gather object-, partition- or subpartition-level statistics.
- There are `GLOBAL` or `NON-GLOBAL` statistics.
- `DBMS_STATS` gather global statistics at any level for tables only.
- `ANALYZE` cannot gather global statistics.
- The `ANALYZE` statement merges statistics from one level to obtain statistics at a higher level.
- It is not possible to gather global histograms.
- it is not possible to gather global statistics for indexes.

DBMS_STATS and ANALYZE Examples

```
CALL DBMS_STATS.GATHER_TABLE_STATS(
           ownname => ‘o816’, tabname => ‘sales’,
           partname => ‘feb97’, granularity => ‘partition’);
```

```
CALL DBMS_STATS.GATHER_INDEX_STATS(
           ownname => ‘o816’, indname => ‘isales’,
           partname => ‘s1’);
```

```
ANALYZE TABLE sales PARTITION (feb97) COMPUTE STATISTICS;
```

```
ANALYZE INDEX isales SUBPARTITION (s1) COMPUTE STATISTICS;
```

```
ANALYZE TABLE sales PARTITION (feb97)
               VALIDATE STRUCTURE INTO INVALID_ROWS;
```

Data Dictionary Views

- DBA_PART_TABLES
- DBA_TAB_PARTITIONS
- DBA_PART_KEY_COLUMNS
- DBA_TABLES
- DBA_OBJECTS
- DBA_IND_PARTITIONS
- DBA_PART_INDEXES
- DBA_PART_HISTOGRAMS
- DBA _TAB_SUBPARTITIONS
- DBA _SUBPART_KEY_COLUMNS
- DBA_PART_COL_STATISTICS
- DBA_SUBPART_COL_STATISTICS
- DBA_SUBPART_HISTOGRAMS
- DBA_IND_SUBPARTITIONS
- DBA_SEGMENTS

- SYS_Pn: Partition name
- SYS_SUBPn: Subpartition name

Maintenance Operations for Table Partitions

Operation	Range	Hash*	Composite*
Adding partitions	ADD PARTITION		•ADD PARTITION* •MODIFY PARTITION ADD SUBPARTITION*
Coalescing partitions		COALESCE PARTITION*	MODIFY PARTITION COALESCE SUBPARTITION*
Dropping partitions	DROP PARTITION		DROP PARTITION
Exchanging partitions	EXCHANGE PARTITION		•EXCHANGE PARTITION** •EXCHANGE SUBPARTITION*
Merging partitions*	MERGE PARTITIONS*		MERGE PARTITIONS*
Modifying default attributes of partitions	MODIFY DEFAULT ATTRIBUTES		•MODIFY DEFAULT ATTRIBUTES •MODIFY DEFAULT ATTRIBUTES FOR PARTITION*
Modifying real attributes of partitions	MODIFY PARTITION		•MODIFY PARTITION •MODIFY SUBPARTITION*
Moving partitions	MOVE PARTITION		MOVE SUBPARTITION*
Renaming partitions	RENAME PARTITION		•RENAME PARTITION •RENAME SUBPARTITION*
Splitting partitions	SPLIT PARTITION		SPLIT PARTITION
Truncating partitions	TRUNCATE PARTITION		•TRUNCATE PARTITION •TRUNCATE SUBPARTITION*

Maintenance Operations for Index Partitions

Operation	Type	Range	Hash*	Composite*
Dropping partitions	Global	DROP PARTITION		
	Local	n/a	n/a	n/a
Modifying default attributes of partitions	Global	MODIFY DEFAULT ATTRIBUTES		
	Local	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	•MODIFY DEFAULT ATTRIBUTES •MODIFY DEFAULT ATTRIBUTES FOR PARTITION*
Modifying real attributes of partitions	Global	MODIFY PARTITION		
	Local	MODIFY PARTITION	MODIFY PARTITION	•MODIFY PARTITION •MODIFY SUBPARTITION*
Rebuilding partitions	Global	REBUILD PARTITION		
	Local	REBUILD PARTITION	REBUILD PARTITION	REBUILD SUBPARTITION*
Renaming partitions	Global	REBUILD PARTITION		
	Local	REBUILD PARTITION	RENAME PARTITION	RENAME SUBPARTITION*
Splitting partitions	Global	SPLIT PARTITION		
	Local	n/a	n/a	n/a

Data Manipulation Language (DML) Partition and Subpartition Locks

Additional levels of locking hierarchy:

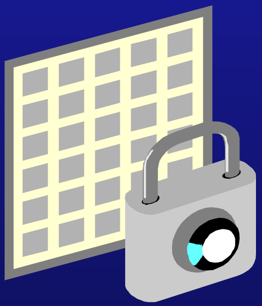
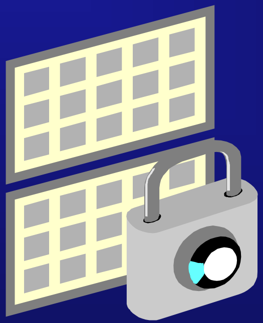
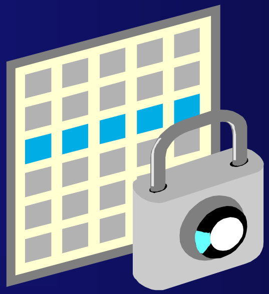


Table locks



Fragment locks



Row locks

SQL*Loader and Partitioned Objects

- You can load a partitioned table through the conventional path.
- You can sequentially load a partitioned table through the direct path.
- You can parallel load a single table partition through the direct path.

Automated Parallel Query Tuning

- Instance level:
  - `PARALLEL_AUTOMATIC_TUNING=TRUE` sets intelligent defaults for all parallel query initialization parameters
  - Reduces tuning complexity in most cases
  - Hand tuning by sophisticated high-end users is still possible
- Table/index level:

Automatic allocation of parallel query slaves across instances

Controlling Parallel Query Execution

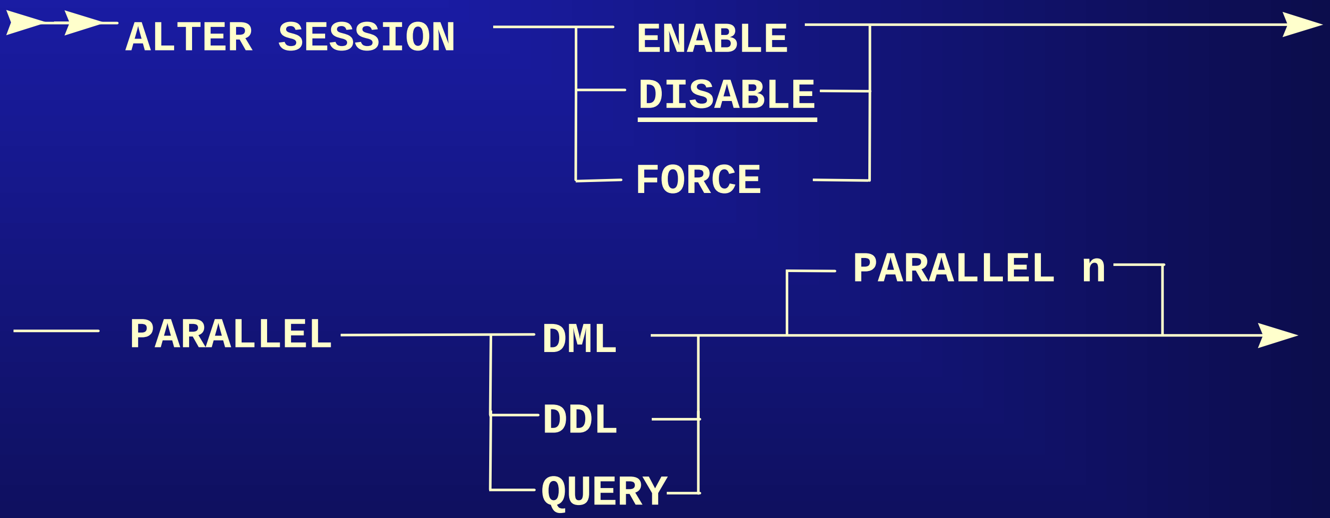
- Allocation of parallel query slaves can be controlled with:
  - New initialization parameters:

    `PARALLEL_ADAPTIVE_MULTI_USER`

    `PARALLEL_THREADS_PER_CPU`
  - Database resource manager
- Parallel query uses large pool for messages; increase large pool and decrease shared pool as needed

Enabling Parallel DML/DDL/QUERY

- The ALTER SESSION statement enables parallel mode



ALTER SESSION

ENABLE

DISABLE

FORCE

PARALLEL

DML

DDL

QUERY

PARALLEL n

- QUERY only starting with 8.1.6

Dynamic Performance Views

- V$PX_PROCESS
- V$PX_PROCESS_SYSSTAT
- V$PX_SESSION
- V$PX_SESSTAT

Summary

In this lesson, you should have learned the following:

- Large tables and indexes can be partitioned into smaller, more manageable pieces.
- Partitioning can improve performance.
- Partitioning can provide higher data availability.
- Maintenance operations can be applied to a smaller subset of data.
- Automatic parallel execution eases tuning

Practice 7 Overview

This practice covers the following topics:

- Creating partitioned tables :
  - Range
  - Hash
  - Composite
- Enabling row movement
- Creating local and global indexes
- Querying data dictionary partitions information
- Manipulating partitions
- Pruning partitions
- Enabling PDML (Optional)

8

Oracle Universal Installer: Migration and Upgrade

Objectives

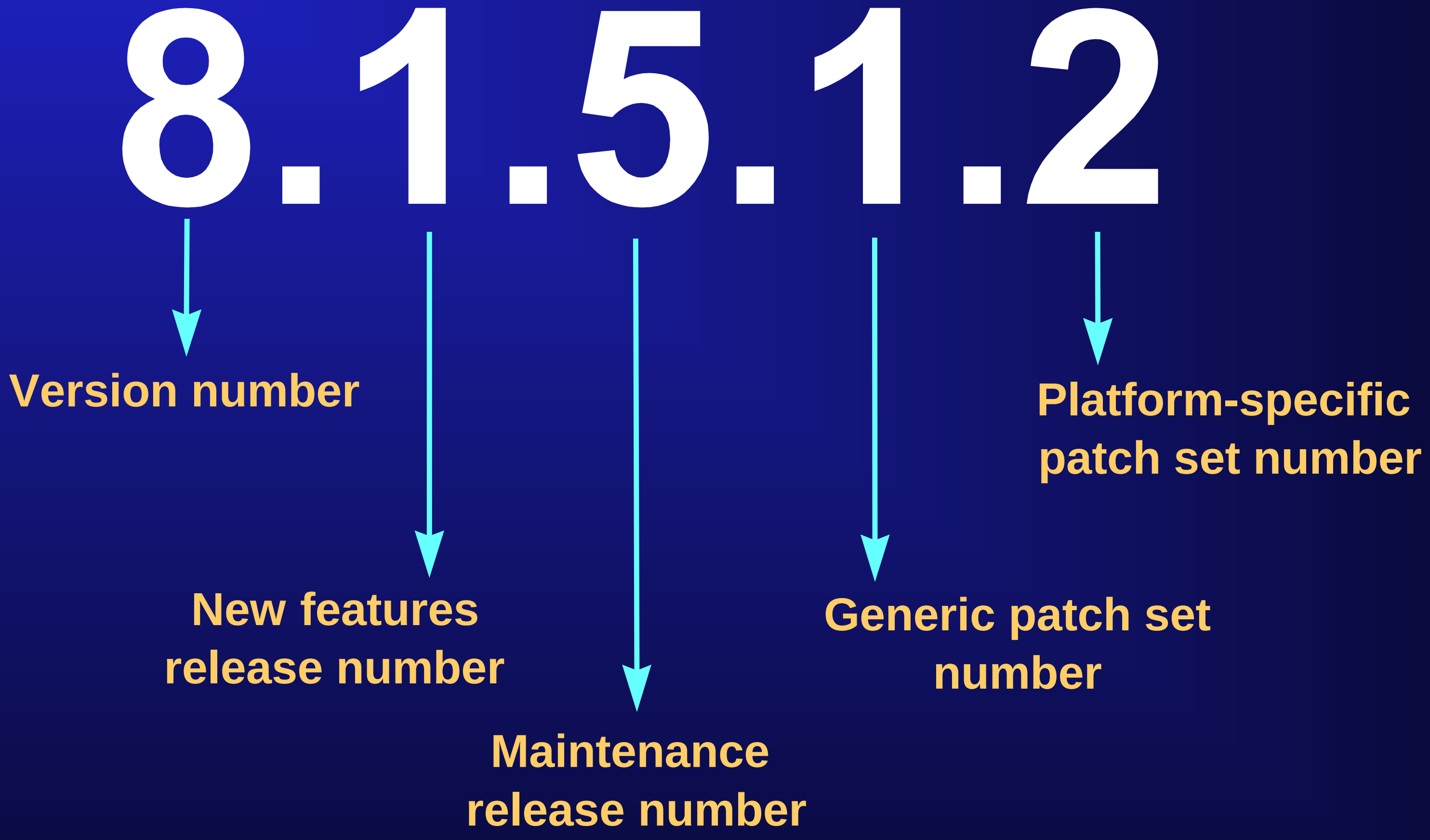
After completing this lesson, you should be able to do the following:

- List the features of the Oracle Universal Installer
- Learn how to migrate an Oracle7 database to Oracle8*i*
- Learn how to upgrade an Oracle8 database to Oracle8*i*

Oracle Universal Installer

- Java-based
- Inventory directories
- Multiple Oracle Homes support
- NLS support
- Configuration tools integration
- Web installs
- Logging
- Silent installations
- Optimal Flexible Architecture (OFA)

Migration or Upgrade?



8.1.5.1.2

Version number

New features release number

Maintenance release number

Generic patch set number

Platform-specific patch set number

Migration Methods

- Migration utility
- Oracle Data Migration Assistant
- Export/import
- Copying data

Common Migration Problems

- Running out of space
- Quiesce and disable symmetric replication
- No recovery needed
- No Save Undo
- No outstanding distributed transactions
- No role or user named MIGRATE or OUTLN
- Prepare the SYSTEM rollback segment
- PCTINCREASE around 50% in SYSTEM
- Ensure that SYS and SYSTEM have the SYSTEM tablespace as default and temporary
- AUD$ is inside the SYSTEM tablespase

Common Migration Problems

- `init.ora` parameters
- Remember control files location, database character set, and SYSDBA and SYSOPER users
- Making backups
- Duration of migration
- Avoiding ROWIDs problem
- Compatibility
- Invalid objects and lost statistics
- Read-only tablespace issues
- Preventing large restores
- Testing

Migration Steps

- Install Oracle8*i* software in a `NEW` home.
- Run `migprep` from Oracle8*i* to transfer migration files in the Oracle7 home (only on UNIX).
- Shut down the Oracle7 database cleanly and make a complete backup.
- Run migration from the Oracle8*i* home.
- Copy `convSID.dbf` (generated file) into Oracle8*i* home.
- Delete control files.
- Start up `nomount` and covert database.
- Open resetlogs the database.
- Execute `U0703040.SQL`.

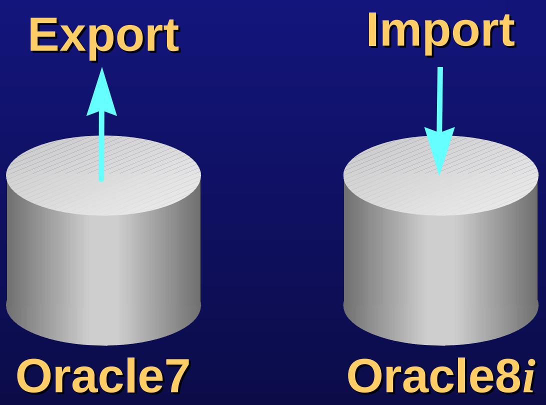
Migration Parameters

Identify the parameters you need for the migration utility:

- `PFILE`
- `CHECK_ONLY`
- `DBNAME`
- `NO_SPACE_CHECK`
- `NEW_DBNAME`
- `MULTIPLIER`
- `SPOOL`

Export/Import

1. Export from the source database.
2. Install Oracle8 software.
3. Prepare the Oracle8*i* database.
4. Import into the target database.



Upgrading from 8.0 to 8.1

Old Release	Upgrade Path
8.0.4S 8.0.1 8.0.2	Use the following steps: 1. Upgrade to release 8.0.5 2. Upgrade to 8.1
8.0.3, 8.0.4, 8.0.5, 8.0.6 8.1.5	Upgrade to 8.1.6 or 8.1.5 directly
8.1.3 8.1.4	1. Upgrade to 8.1.5 2. Upgrade to 8.1.6
8.1.1 8.1.2	No upgrade path

Direct Manual Upgrades

- Ensure that there is no `OUTLN` user or role.
- Ensure there is enough space in the `SYSTEM` tablespace.
- Ensure there are enough rollback segments.
- Make a backup.
- Install 8*i* software in a new home.
- Move and modify your `init.ora` file.
- Set your environment to identify the Oracle8*i* database.
- Start up your Oracle8*i* instance.
- Run the appropriate upgrade script.
- Upgrade specific components.

Data Migration Assistant

- Seamless migration from the selected versions of Oracle 7.1, 7.2, 7.3, and 8.0 to Oracle8*i*, version 8.1
- Restrictions:
  - Parallel server migrations
  - No raw devices

Summary

In this lesson, you should have learned the following:

- The Universal Installer presents a standard interface across all platforms.
- The Database Configuration Assistant provides:
  - Custom or automatic installation based on hardware discovery
  - Pretuned starter databases and sample schemas
  - Scripted, silent mode installs

Summary

- You can use one of three methods to migrate based on:
  - Resources available
  - User needs
- Creating a migration plan is essential.
- The Data Migration Assistant can help with migrations and with upgrades from Oracle8 to Oracle8*i*.

9

Tablespace Management

Objectives

After completing this lesson, you should be able to do the following:

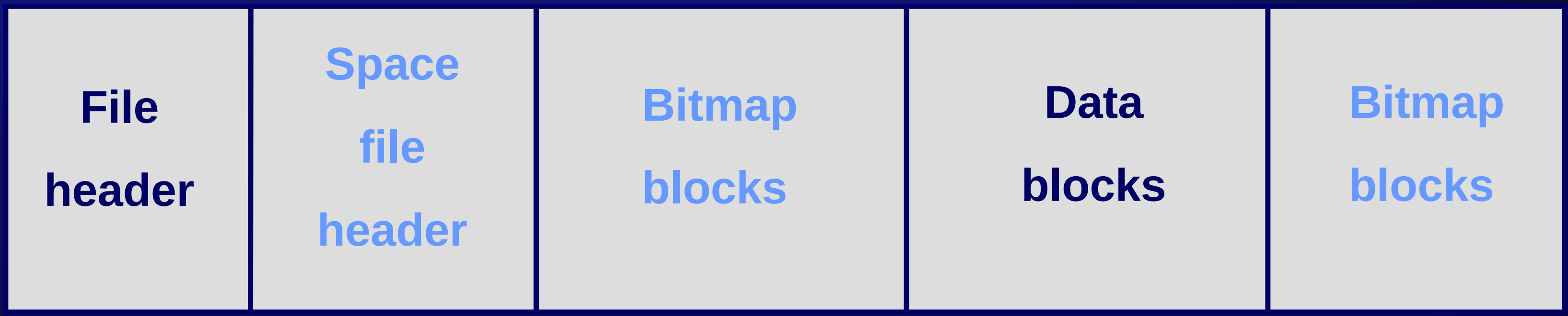
- Manage locally managed tablespaces
- Manage transportable tablespaces
- Use read-only tablespace enhancements

Locally Managed Tablespaces Overview

- Better space management
  - Uniform extent sizes
  - Reduced data dictionary access
- Reduced fragmentation
- Better management of temporary space
- More reliability

Features of Locally Managed Tablespaces

- Bitmaps are stored within files.
- Locally and dictionary-managed tablespaces can coexist.
- The `SYSTEM` tablespace cannot be locally managed.



File header

Space file header

Bitmap blocks

Data blocks

Bitmap blocks

Permanent Locally Managed Tablespaces

Create a permanent locally managed tablespace:

```
CREATE TABLESPACE user_data_1
...
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

```
CREATE TABLESPACE user_data_1
...
EXTENT MANAGEMENT LOCAL UNIFORM [SIZE 2M];
```

Migration of Locally Managed Tablespaces

```
CALL DBMS_SPACE_ADMIN.
TABLESPACE_MIGRATE_TO_LOCAL(name,[unit,rfno])
```

```
CALL DBMS_SPACE_ADMIN.
TABLESPACE_MIGRATE_FROM_LOCAL(name)
```

Temporary Locally Managed Tablespaces

Create a temporary, locally managed tablespace:

```
CREATE TEMPORARY TABLESPACE user_temp_1
TEMPFILE ‘......’
EXTENT MANAGEMENT LOCAL
[UNIFORM [SIZE 10M]];
```

Maintaining Temporary Locally Managed Tablespaces

```
ALTER TABLESPACE lmtemp ADD TEMPFILE
'/u02/lmtemp02.dbf' SIZE 2M;
```

```
ALTER DATABASE TEMPFILE
'/u02/lmtemp02.dbf' OFFLINE|ONLINE;
```

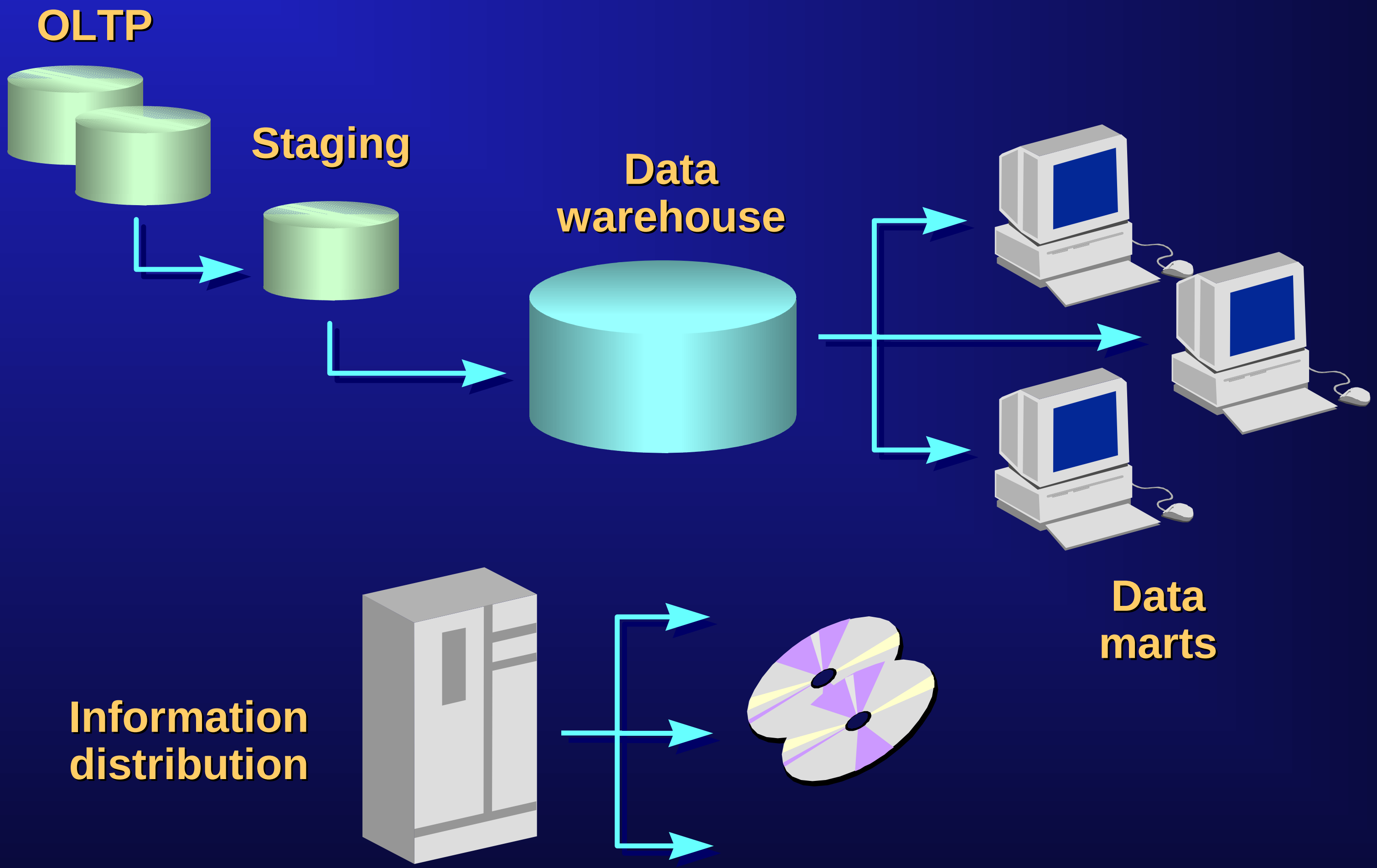
```
ALTER DATABASE TEMPFILE
'/u02/lmtemp02.dbf' RESIZE 4M;
```

```
ALTER DATABASE TEMPFILE
'/u02/oracle/data/lmtemp02.dbf' DROP;
```

Locally Managed Tablespaces Views

- `DBA_FREE_SPACE, DBA_EXTENTS`
- `DBA_TABLESPACES`
- `V$TEMPFILE, V$DATAFILE`
- `DBA_TEMP_FILES, DBA_DATA_FILES`
- `V$TEMP_EXTENT_MAP`
- `V$TEMP_EXTENT_POOL`
- `V$TEMP_SPACE_HEADER`

Data Transportation: Transportable Tablespaces



Copying Tablespaces

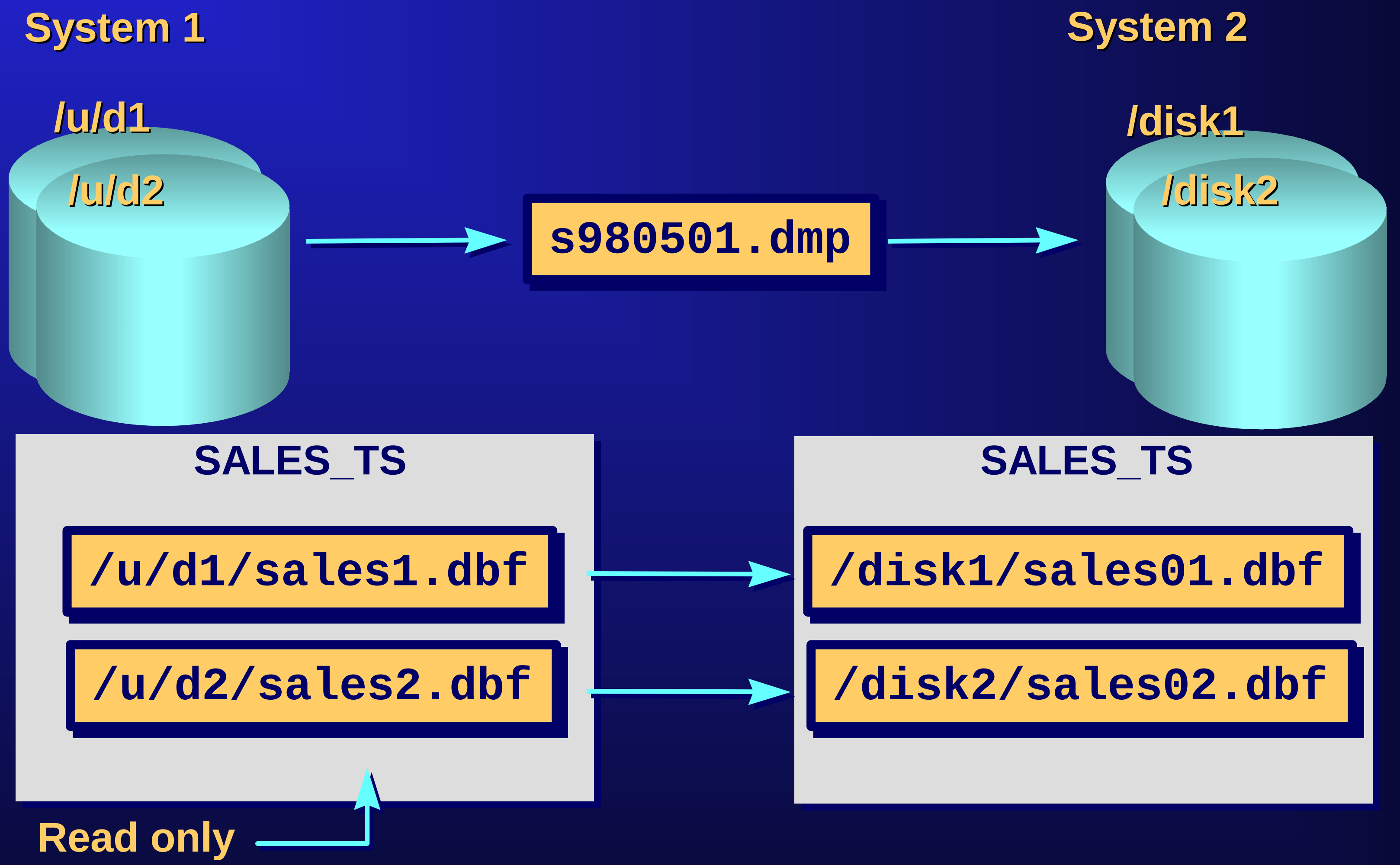
1. Make tablespace read-only.
2. Export metadata from source.
3. Copy data files to target system.
4. Transfer export file.
5. Import metadata into target.
6. Make tablespace in target read-write, if necessary.

Exporting and Importing Metadata

```
exp ‘/ AS SYSDBA’
FILE=s980501.dmp
TRANSPORT_TABLESPACE=y
TABLESPACES=sales_ts
TRIGGERS=N CONSTRAINTS=N
```

```
imp ‘/ AS SYSDBA’
FILE=s980501.dmp
TRANSPORT_TABLESPACE=y
DATAFILES=(/disk1/sales01.dbf,
               /disk2/sales02.dbf)
```

Transporting a Tablespace



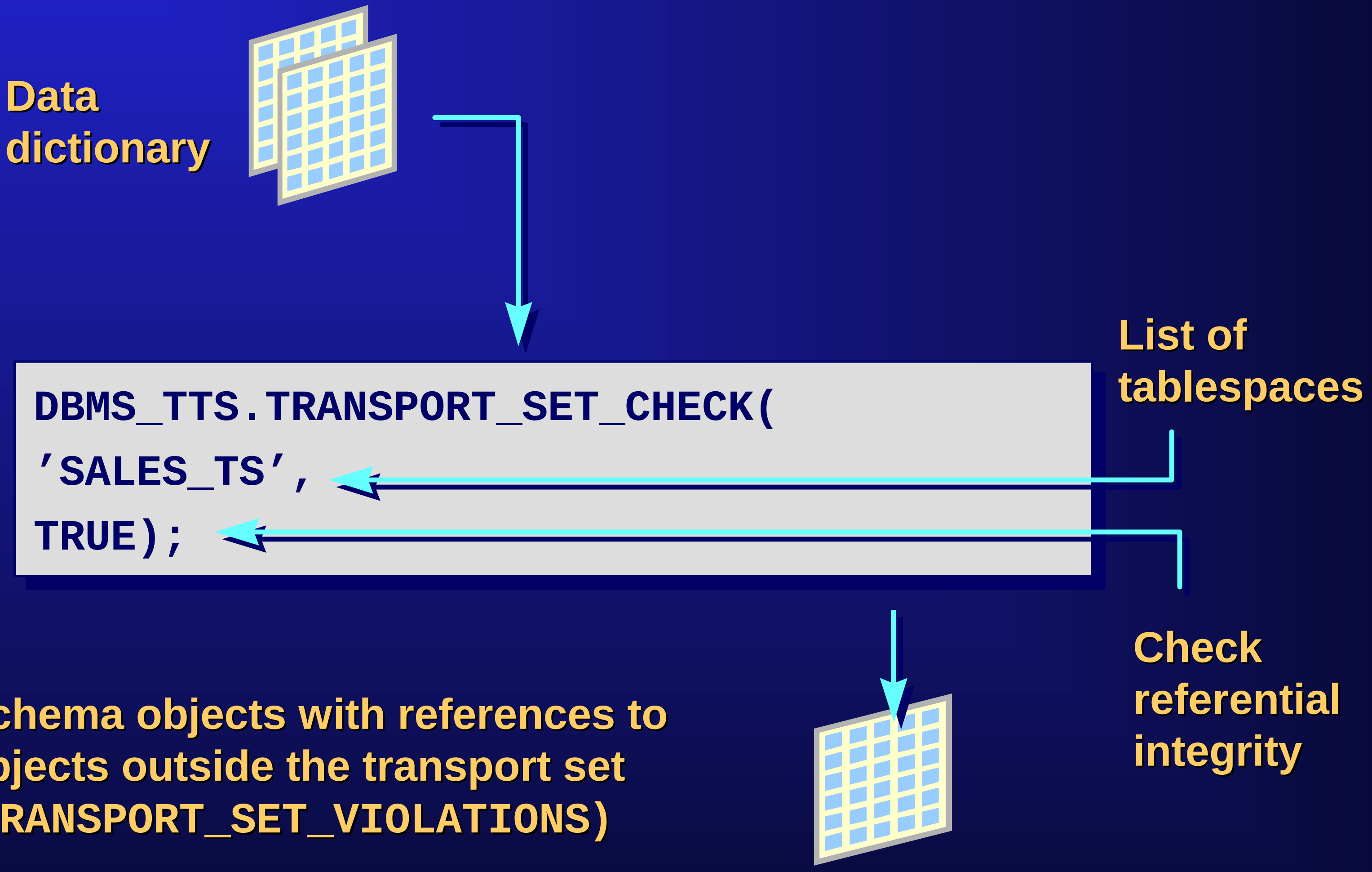
Uses of Transportable Tablespaces

- Moves entire tablespaces data that do not contain SYS objects
- Supports media recovery
- Source and target databases must:
  - Be on the same operating system
  - Run Oracle8*i*, release 8.1, or above
  - Have the same block size
  - Use the same character sets
- Looks at the `PLUGGABLE_SET_CHECK` view

Transportable Tablespaces and Schema Objects

- Tablespaces transported in one run must be self-contained.
- The following objects cannot be transported:
  - Snapshot and replication
  - Function-based indexes
  - Scoped REFs
  - Domain indexes
  - 8.0-compatible AQ with multiple recipients

Checking Transport Set



Data dictionary

```
DBMS_TTS.TRANSPORT_SET_CHECK(
'SALES_TS',
TRUE);
```

List of tablespaces

Check referential integrity

Schema objects with references to objects outside the transport set (**TRANSPORT_SET_VIOLATIONS**)

Read-Only Tablespaces

- Allow no DML against stored segments
- Very useful for transportable tablespaces
- The command occurs in two phases:
  - Prepare files for read-only
  - Wait until current transactions on the database complete

Summary

In this lesson, you should have learned the following:

- Locally managed tablespaces provide:
  - Better space management
  - Higher availability
- Transportable tablespaces provide:
  - Easy exchange of data between data marts and data warehouses
  - Archiving of OLTP data to storage or data warehouse
  - Publication of data with a database
- Read-only tablespaces can be readily prepared for transport.

Practice 9 Overview

This practice covers the following topics:

- Building a locally-managed tablespace
- Examining space allocation with uniform extent sizes
- Plugging the same tablespace in multiple databases at the same time

10

Database Resource Manager

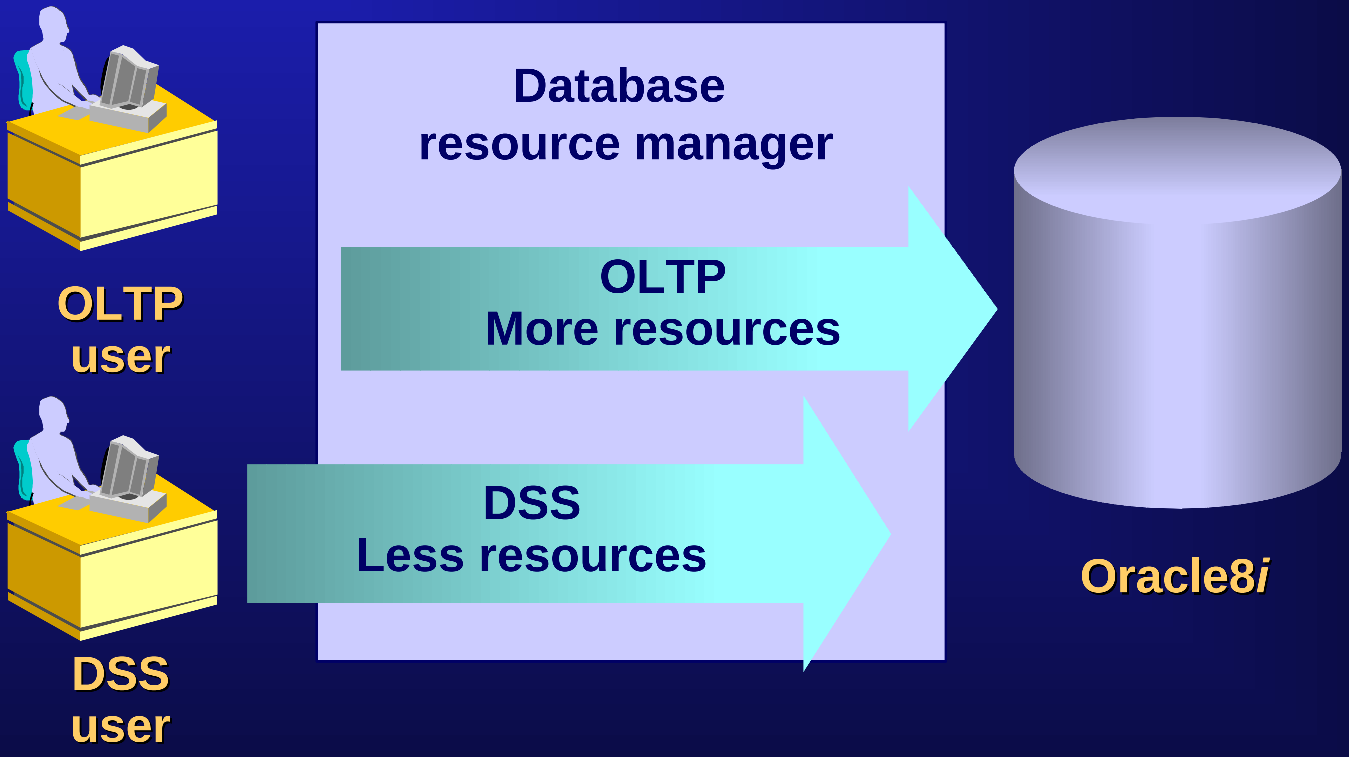
Objectives

After completing this lesson, you should be able to do the following:

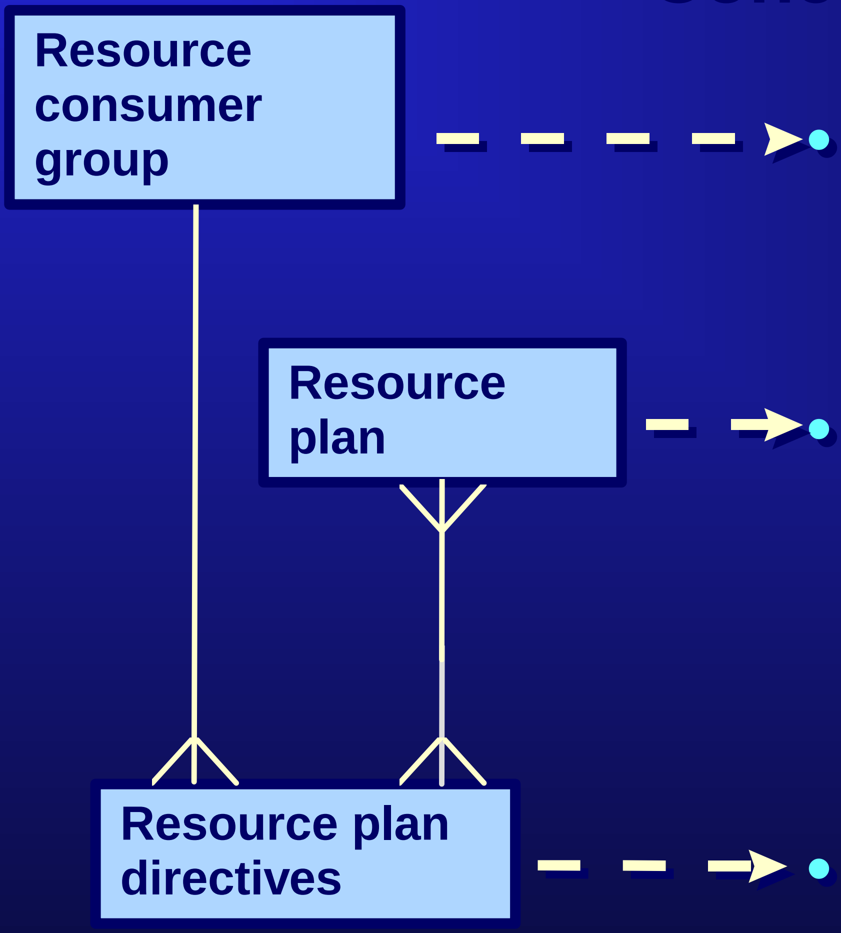
- List the features of the database resource manager
- Limit the use of resources using the database resource manager

Overview

- Manage mixed workload
- Control system performance



Database Resource Management Concepts



Resource consumer group

Resource plan

Resource plan directives

- User groups with similar resource needs
- Allocates CPU and PQ servers (one active plan)
- Assigns groups and resources to plans

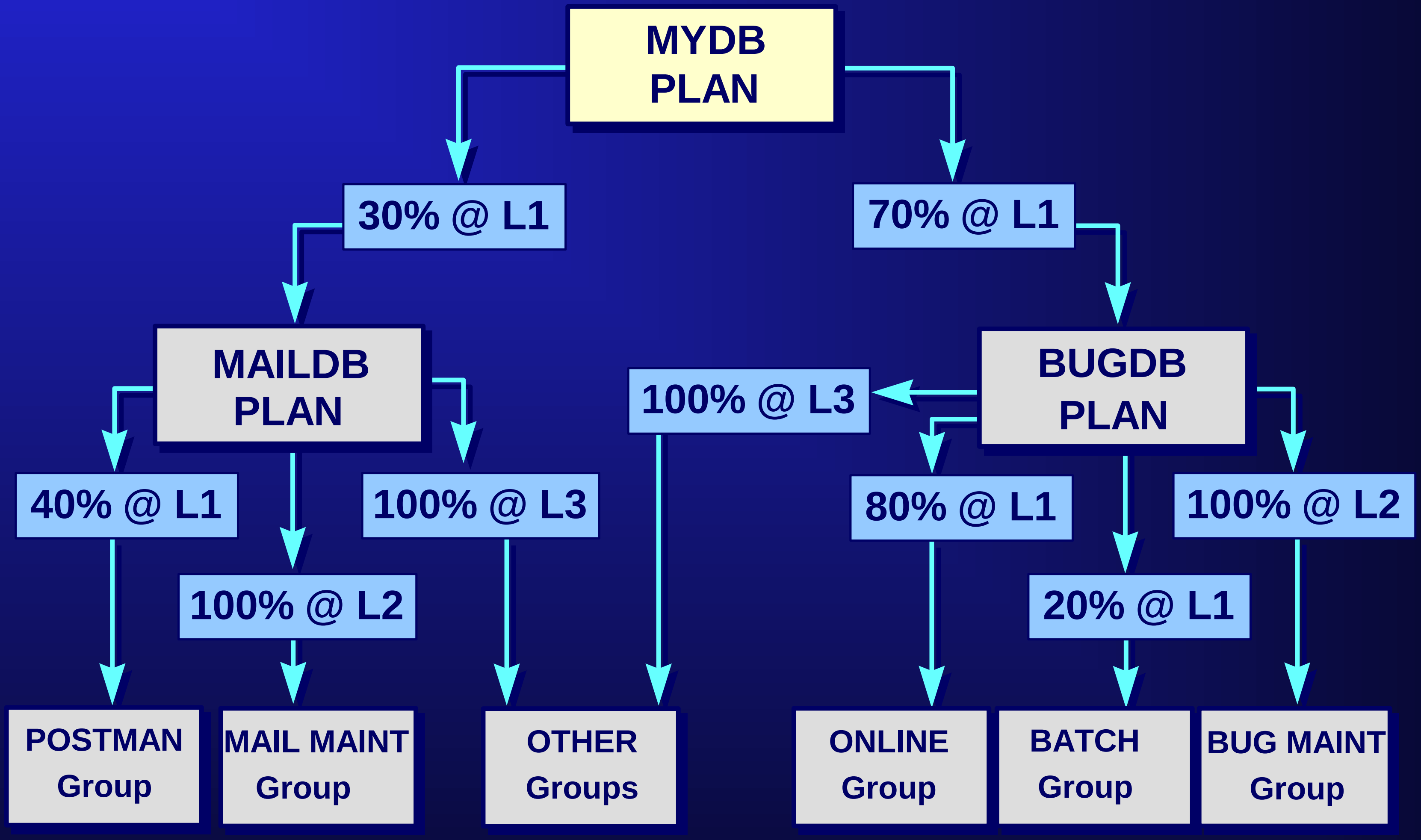
Resource Allocation Methods

Method	Resource Allocation	Recipient
Round-robin	CPU to sessions	Groups
Emphasis	CPU to groups	Plans
Absolute	Parallel degree	Plans

The Original Plan: SYSTEM_PLAN

Resource Consumer Group	Allocation Methods			
	P1CPU	**P2**CPU	**P3**CPU	**P1** //
SYS_GROUP	100%	0%	0%	0
OTHER_GROUPS	0%	100%	0%	0
LOW_GROUP	0%	0%	100%	0

Using Subplans



MYDB PLAN

30% @ L1

70% @ L1

MAILDB PLAN

BUGDB PLAN

40% @ L1

100% @ L2

100% @ L3

100% @ L3

80% @ L1

20% @ L1

100% @ L2

POSTMAN Group

MAIL MAINT Group

OTHER Groups

ONLINE Group

BATCH Group

BUG MAINT Group

Administering the Database Resource Manager

1. Assign the resource manager system privileges to the administrator.
2. Create resource objects with the package `DBMS_RESOURCE_MANAGER`:
   - Resource consumer groups
   - Resource plans
   - Resource plan directives
3. Assign users to groups with the package `DBMS_RESOURCE_MANAGER_PRIVS`.
4. Specify the plan to be used by the instance.

Assigning the Resource Manager Privilege

1. **Assign the resource manager system privileges to the administrator.**

```
DBMS_RESOURCE_MANAGER_PRIVS.
  GRANT_SYSTEM_PRIVILEGE (
    grantee_name => 'SCOTT',
    privilege_name
      => 'ADMINISTER_RESOURCE_MANAGER',
    admin_option => FALSE  );
```

Creating Database Resource Manager Objects

2. Create resource objects with the package **DBMS_RESOURCE_MANAGER.**

   a. Create a pending area.

```
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```

   b. Create resource consumer groups.

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
  consumer_group => 'OLTP',
  comment =>           'Online users' );
```

Creating Database Resource Manager Objects

c. Create resource plans.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (
    plan =>    'NIGHT',
    comment => 'DSS/Batch priority, ...' );
```

d. Create resource plan directives.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    plan =>                     'NIGHT',
    group_or_subplan =>         'SYS_GROUP',
    comment =>                  '...',
    cpu_p1 =>                      100,
    parallel_degree_limit_p1 => 20);
```

Creating Database Resource Manager Objects

e. Validate the pending area.

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
```

f. Commit the pending area.

```
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

Assigning Users to Consumer Groups

3. Assign users to groups.

```
DBMS_RESOURCE_MANAGER_PRIVS.
  GRANT_SWITCH_CONSUMER_GROUP (
    grantee_name =>     'MOIRA',
    consumer_group =>   'OLTP',
    grant_option =>     FALSE );
```

Set the initial consumer group for users.

```
DBMS_RESOURCE_MANAGER.
  SET_INITIAL_CONSUMER_GROUP (
    user =>             'MOIRA',
    consumer_group =>   'OLTP' );
```

Setting the Resource Plan for an Instance

4. Specify the plan to be used by the instance.
   - Specify the RESOURCE_MANAGER_PLAN initialization parameter.

```
RESOURCE_MANAGER_PLAN=day
```

   - Change the resource plan without shutting down and restarting the instance.

```
ALTER SYSTEM
  SET RESOURCE_MANAGER_PLAN=night;
```

Changing a Consumer Group Within a Session

The user or the application can switch the current consumer group.

```
DBMS_SESSION.
  SWITCH_CURRENT_CONSUMER_GROUP (
    new_consumer_group => 'DSS',
    old_consumer_group => v_old_group,
    initial_group_on_error => FALSE );
```

Changing Consumer Groups for Sessions

- Can be set by DBA for a session

```
DBMS_RESOURCE_MANAGER.
  SWITCH_CONSUMER_GROUP_FOR_SESS (
    session_id => 7,
    session_serial => 13,
    consumer_group => 'OLTP');
```

- Can be set by DBA for all sessions for a user

```
DBMS_RESOURCE_MANAGER.
  SWITCH_CONSUMER_GROUP_FOR_USER (
    user => 'MOIRA',
    consumer_group => 'OLTP');
```

Database Resource Manager Information

- DBA_RSRC_PLANS: Plans and status
- DBA_RSRC_PLAN_DIRECTIVES: Plan directives
- DBA_RSRC_CONSUMER_GROUPS: Consumer groups
- DBA_RSRC_CONSUMER_GROUP_PRIVS: Users and roles
- DBA_USERS Column : INITIAL_RSRC_CONSUMER_GROUP
- DBA_RSRC_MANAGER_SYSTEM_PRIVS: Users and roles

Current Database Resource Manager Settings

- **V$SESSION: Contains the RESOURCE_CONSUMER_GROUP column that shows the current group for a session**
- **V$RSRC_PLAN: A view that show the active resource plan**
- **V$RSRC_CONSUMER_GROUP: A view that contains statistics for all active groups**

Summary

In this lesson, you should have learned how to control the use of CPUs and the degree of parallelism by using the database resource manager.

11

Manageability Enhancements

Objectives

After completing this lesson, you should be able to do the following:

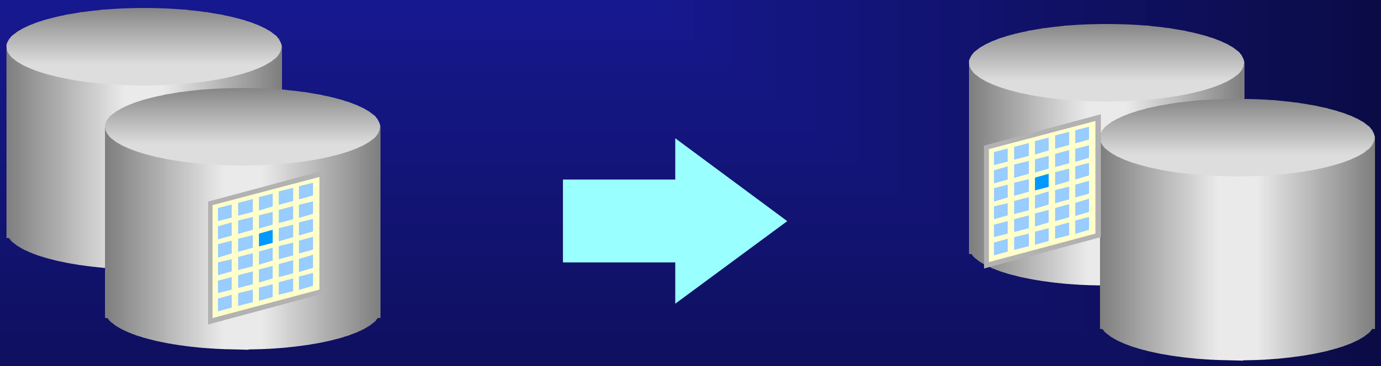
- Identify database limits
- Relocate and reorganize tables
- Remove unused columns from a table
- Define temporary tables
- Identify SQL*Loader enhancements
- Monitor long-running operations
- Change database character sets
- Define new constraints features
- Define new Export/Import features

Database Limits

Columns per table	1,000
Columns per index	32/30
CHAR, NCHAR	2,000 bytes
VARCHAR2, NVARCHAR2	4,000 bytes
CLOB, BLOB, BFILE	4 GB
Data files per database	65,533
Data files per tablespace	1,022
Tablespaces per database	65,533

Relocating and Reorganizing a Table

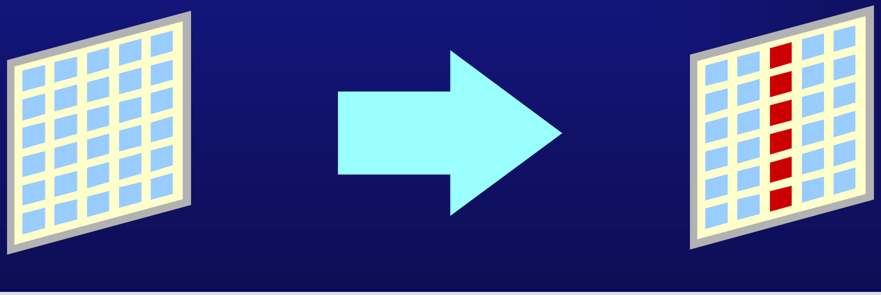
- Builds new segments and drops old segments
- Retains index definition, constraints, and grants on the table



```
ALTER TABLE orders
     MOVE [ONLINE] TABLESPACE data1
```

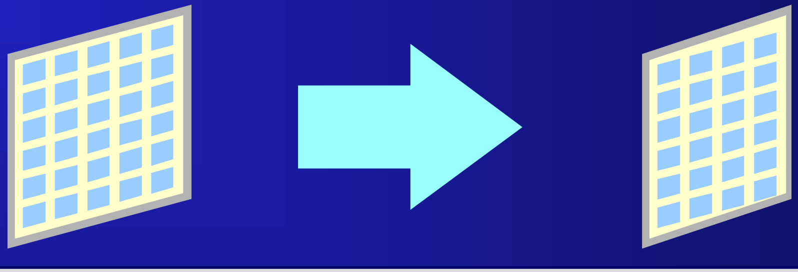
Marking a Column as Unused

- Marks column as unused
- Completes the task quickly
- Does not release space
- Column not visible to users
- Is not reversible

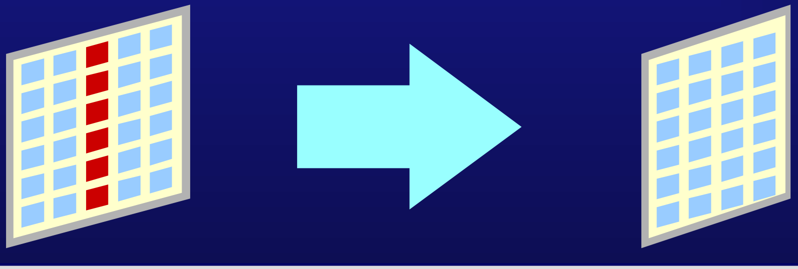


```
ALTER TABLE orders
     SET UNUSED COLUMN comments;
```

Removing a Column from a Table



```
ALTER TABLE orders
     DROP COLUMN order_date
                    CASCADE CONSTRAINTS;
```



```
ALTER TABLE orders
     DROP UNUSED COLUMNS;
```

Removing a Column from a Table: Minimizing Rollback Usage

- Define number of rows for saving changes:

```
ALTER TABLE orders
     DROP COLUMN order_date CHECKPOINT 1024;
```

Marks table INVALID until operation completes

- Resume interrupted operation using:

```
ALTER TABLE orders
     DROP COLUMNS CONTINUE;
```

Temporary Tables

- Retain data only for the duration of a transaction or session.
- Definitions persist in the data dictionary.
- Data is only visible to the session.
- Use sort space to store data.
- Allocate extents in user’s temporary tablespace, if needed.
- DMLs do not generate redo.
- DMLs generate rollback.

Creating and Monitoring Temporary Tables

```
CREATE GLOBAL TEMPORARY TABLE emp_temp
(eno NUMBER,ename VARCHAR2(20),sal NUMBER)
ON COMMIT DELETE ROWS;
```

```
SELECT table_name, temporary, duration
 FROM dba_tables
 WHERE table_name='EMP_TEMP';

TABLE_NAME                T DURATION
------------------------- - ---------------
EMP_TEMP                  Y SYS$TRANSACTION
```

SQL*Loader Enhancements

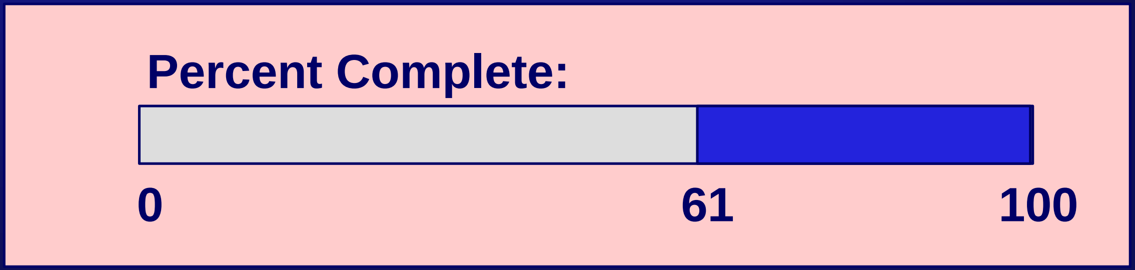
- String delimited fields
- New variable-length field types that use length-value pairs
- Use of compound predicates in NULLIF or DEFAULTIF clauses
- Increased record size limits
- Can load arbitrarily complex object-relational data
- OCI Direct Path API provided

SQL*Loader Example

```
LOAD DATA
INFILE ‘company_data.dat’
RECSEPARATOR ‘;’
FIELDS TERMINATED BY ‘::’
INTO TABLE DEPT (
 compname VARCHAR,
 loc VARCHAR NULLIF (compname=BLANKS AND loc=BLANKS),
 slogan VARCHARC(3),
 dept_mgr COLUMN OBJECT (name CHAR, empid INTEGER),
 resume CHAR DEFAULTIF resume=BLANKS,
 extfname1 FILLER CHAR(40),
 desc LOBFILE(extfname1) TERMINATED BY EOF,
 extfname2 FILLER CHAR(30),
 pict BFILE(scott_dir, extfname2)
)
```

Monitoring Long-Running Operations: Overview

- Many long-running database operations populate the V$SESSION_LONGOPS view.
- Applications can also populate this view by using a new PL/SQL interface:



Monitoring Long-Running Operations

Obtain progress statistics

```
SELECT sid, serial#, opname,
 TO_CHAR(start_time,‘HH24:MI:SS’)  "START",
 (sofar/totalwork)*100 "PERCENT_COMPLETE"
FROM v$session_longops;
```

```
SID  SERIAL#   OPNAME        START       PERCENT_
10       235   Sort Output   13:03:05    35.98098
10       235   Table Scan    13:02:56    90.56454
10       235   SQL Execution 13:02:56    60.25644
```

Changing Database Character Sets After Creation

Example of changing the database character set from US7ASCII to WE8ISO8859P1:

```
SQL> SHUTDOWN IMMEDIATE; -- or NORMAL
<do a full backup>
SQL> STARTUP MOUNT;
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
SQL> ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
SQL> ALTER DATABASE OPEN;
SQL> ALTER DATABASE CHARACTER SET WE8ISO8859P1;
SQL> SHUTDOWN IMMEDIATE; -- or NORMAL
SQL> STARTUP;
```

Deferred Constraint Checking

- Can a constraint can be deferred?
  - NOT DEFERRABLE (default)
  - DEFERRABLE
- The default behavior of the constraint:
  - INITIALLY IMMEDIATE (default)
  - INITIALLY DEFERRED
- Use SET CONSTRAINTS or ALTER SESSION SET CONSTRAINTS to change the behavior

Validating Constraints

- ENABLE/DISABLE affects future changes:
  - ENABLE to check future changes
  - DISABLE to stop checking future changes
- VALIDATE/NOVALIDATE affects current table data:
  - VALIDATE to check the current table data
  - NOVALIDATE to avoid checking the current table data

New Constraint Functionality: RELY Flag

- Lets DBA indicate validity of data without enabling or validating a constraint
- Used by query rewrite

```
ALTER TABLE state
  ADD PRIMARY KEY(state_code) DISABLE NOVALIDATE;

ALTER TABLE state MODIFY PRIMARY KEY RELY;
```

Unique or Primary Key Constraint Using a Nonunique Index

- **Enforce a unique or primary key (PK) constraint with a nonunique index.**
- **The nonunique index must have the unique or primary key columns as the prefixed columns.**
- **The columns in the index and constraint do not have to be in the same order.**

```
CREATE TABLE acct
  ( acct_no    NUMBER(10),
    customer_id   NUMBER(10),
    acct_comment  VARCHAR2(200),
     CONSTRAINT pk_cid_aid
     PRIMARY KEY(customer_id, acct_no) DISABLE);
```

```
CREATE INDEX I_ANO_CNO_ACOMM
  ON acct(acct_no, customer_id, acct_comment);
```

Data Dictionary Changes

DBA_CONSTRAINTS new columns:

- DEFERRABLE indicates whether the constraint can be deferred.
- DEFERRED indicates whether the constraint is currently deferred.
- VALIDATED indicates whether the table data is validated.
- RELY contains the setting of the RELY flag.

Export

```
Export: Release 8.1.6.0.0 - Production on Wed Oct 6 15:23:43 1999
(c) Copyright 1999 Oracle Corporation.  All rights reserved.
...
Keyword  Description (Default)        Keyword      Description (Default)
--------------------------------------------------------------------------
USERID      username/password            FULL         export entire file (N)
BUFFER      size of data buffer          OWNER        list of owner usernames
FILE        output files (EXPDAT.DMP)    TABLES       list of table names
COMPRESS    import into one extent (Y)   RECORDLENGTH length of IO record
GRANTS      export grants (Y)            INCTYPE      incremental export type
INDEXES     export indexes (Y)           RECORD       track incr. export (Y)
ROWS        export data rows (Y)         PARFILE      parameter filename
CONSTRAINTS export constraints (Y)       CONSISTENT   cross-table consistency
LOG         log file of screen output    STATISTICS   analyze objects (ESTIMATE)
DIRECT      direct path (N)              TRIGGERS     export triggers (Y)
FEEDBACK    display progress every x rows (0)
FILESIZE    maximum size of each dump file
QUERY       select clause used to export a subset of a table
VOLSIZE     number of bytes to write to each tape volume
The following keywords only apply to transportable tablespaces
TRANSPORT_TABLESPACE export transportable tablespace metadata (N)
TABLESPACES list of tablespaces to transport
```

Import

```
Import: Release 8.1.6.0.0 - Production on Wed Oct 6 15:26:12 1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.

Keyword Description (Default) Keyword Description (Default)
--------------------------------------------------------------------------
USERID username/password                FULL import entire file (N)
BUFFER size of data buffer              FROMUSER list of owner usernames
FILE input files (EXPDAT.DMP)           TOUSER list of usernames
SHOW just list file contents (N)        TABLES list of table names
IGNORE ignore create errors (N)         RECORDLENGTH length of IO record
GRANTS import grants (Y)                INCTYPE incremental import type
INDEXES import indexes (Y)              COMMIT commit array insert (N)
ROWS import data rows (Y)               PARFILE parameter filename
LOG log file of screen output           CONSTRAINTS import constraints (Y)
DESTROY overwrite tablespace data file (N)
INDEXFILE write table/index info to specified file
SKIP_UNUSABLE_INDEXES skip maintenance of unusable indexes (N)
ANALYZE execute ANALYZE statements in dump file (Y)
FEEDBACK display progress every x rows(0)
TOID_NOVALIDATE skip validation of specified type ids
FILESIZE maximum size of each dump file
RECALCULATE_STATISTICS recalculate statistics (N)
VOLSIZE number of bytes in file on each volume of a file on tape
TRANSPORT_TABLESPACE transportable tablespace metadata (N)
TABLESPACES tablespaces to be transported into database
DATAFILES datafiles to be transported into database
TTS_OWNERS users that own data in the transportable tablespace set
```

Summary

In this lesson, you should have learned how to:

- Identify new database limits
- Relocate and reorganize tables
- Drop columns
- Define temporary tables
- Identify the new SQL*Loader options
- Monitor long-running operations
- Change database character sets
- Identify new constraints features
- Identify new Export/Import features

Practice 11 Overview

This practice covers the following topics:

- Moving tables to different tablespaces
- Unusing columns
- Droping columns
- Creating temporary tables
- Deferring constraints checking
- Disabling constraints

12

Availability and Recoverability Enhancements

Objectives

After completing this lesson, you should be able to do the following:

- Learn RMAN new features
- Implement duplex and multiple archive logs
- Set up a standby database in sustained recovery mode
- Start up a database for read operations
- Suspend database I/Os
- Describe the functionality of LogMiner
- Implement fast-start fault recovery
- Manage corrupt block detection and repair
- Describe the new possibility of dynamically change the number of free lists

RMAN New Features

Version 8.1.6:

- The `AUTOLOCATE` option of the set command
- The `CONFIGURE COMPATIBLE` command
- The `ALTER DATABASE OPEN` reset logs
- The `CHANGE DELETE`, `DELETE EXPIRED`, and `BACKUP DELETE INPUT` commands can now remove catalog records rather than update them to status DELETED. This behavior depend on the compatibility of the recovery catalog.

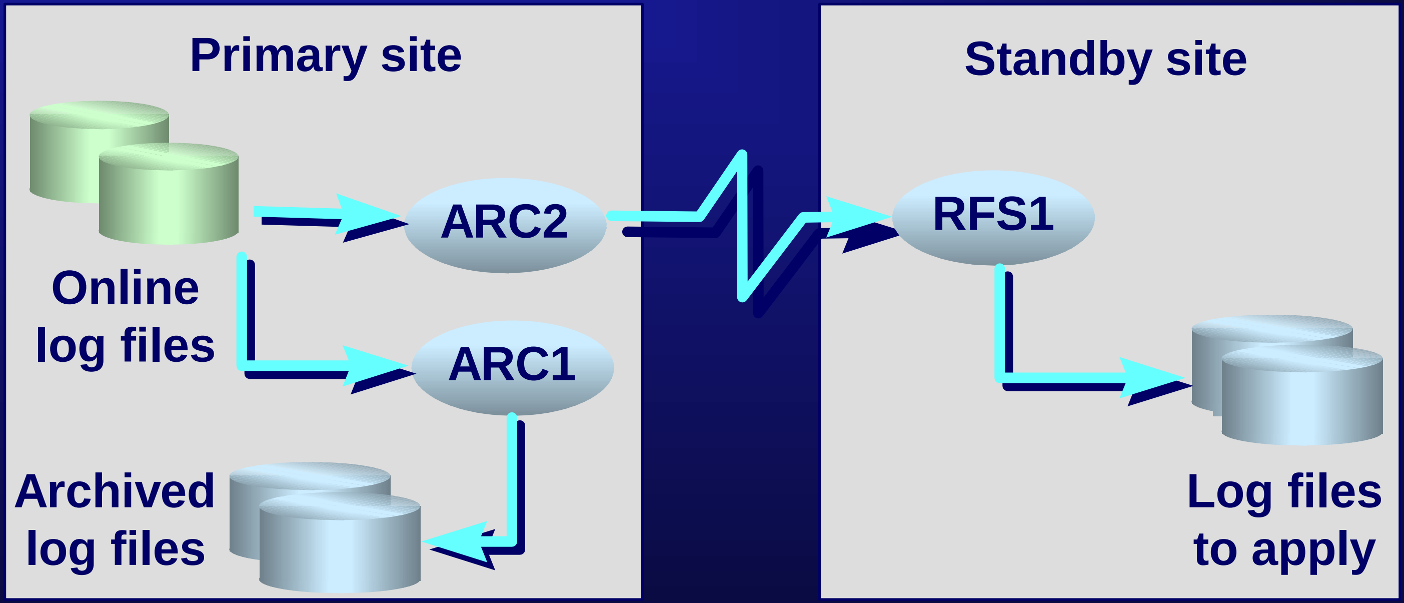
RMAN New Features

Version 8.1.5:

- Media Management API, version 2.0
- The cross-check commands
- CREATE, DROP, UPGRADE CATALOG commands
- STARTUP, SHUTDOWN, MOUNT, OPEN commands
- Duplicate databases
- Node affinity recognition by RMAN in Oracle Parallel Server
- Duplex backup sets
- You can perform TSPITR without a RC
- New views: V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO
- DBMS_RCVCAT.UNREGISTERDATABASE (8.0)

Archiver Enhancements

- Multiple archive destinations: Local disk or remote database
- Multiple archive processes



Specifying Multiple Archive Locations

- Specify up to five archival destinations by using `LOG_ARCHIVE_DEST_`*`n`*:

  Either local disk or remote database

```
log_archive_dest_1 = "LOCATION=/archive1"
log_archive_dest_2 = "SERVICE=standby_db1"
```

- Use `LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST`:

```
log_archive_dest = /archive1/arch
log_archive_duplex_dest = /archive2/arch
```

Multiple Archive Options

- Set archive location as `MANDATORY` or `OPTIONAL`.
- Define time before retry in case of failures.

```
log_archive_dest_1="LOCATION=/archive/
                        MANDATORY REOPEN"
log_archive_dest_2="SERVICE=standby_db1
                        MANDATORY REOPEN=600"
log_archive_dest_3="LOCATION=/archive2/
                        OPTIONAL"
```

Specifying Minimum Number of Local Destinations

- **LOG_ARCHIVE_MIN_SUCCEED_ DEST** parameter:

```
log_archive_min_succeed_dest = 2
```

- An online redo log group can be reused only if:
  - Archiving has been done to all mandatory locations
  - The number of local locations archived is greater than or equal to **LOG_ARCHIVE_MIN_SUCCEED_ DEST**

Controlling Archiving to a Destination

- An archival destination can be disabled by a new (dynamic) initialization parameter: **LOG_ARCHIVE_DEST_STATE _*n***

```
log_archive_dest_state_2 = DEFER
log_archive_dest_state_3 = DEFER
```

- Archiving to a destination can be enabled again:

```
log_archive_dest_state_2 = ENABLE
```

```
ALTER SYSTEM SET log_archive_dest_state_3 =
ENABLE
```

Multiple Archive Log Processes

- **Support multiple archive locations**
- **Increase archiving throughput**
- **Reduce the need to perform manual archives**
- **Controlled by new dynamic parameter:**

  `LOG_ARCHIVE_MAX_PROCESSES`
- **Started if `LOG_ARCHIVE_START=TRUE` and automatic archiving is enabled**
- **Default value is 1**

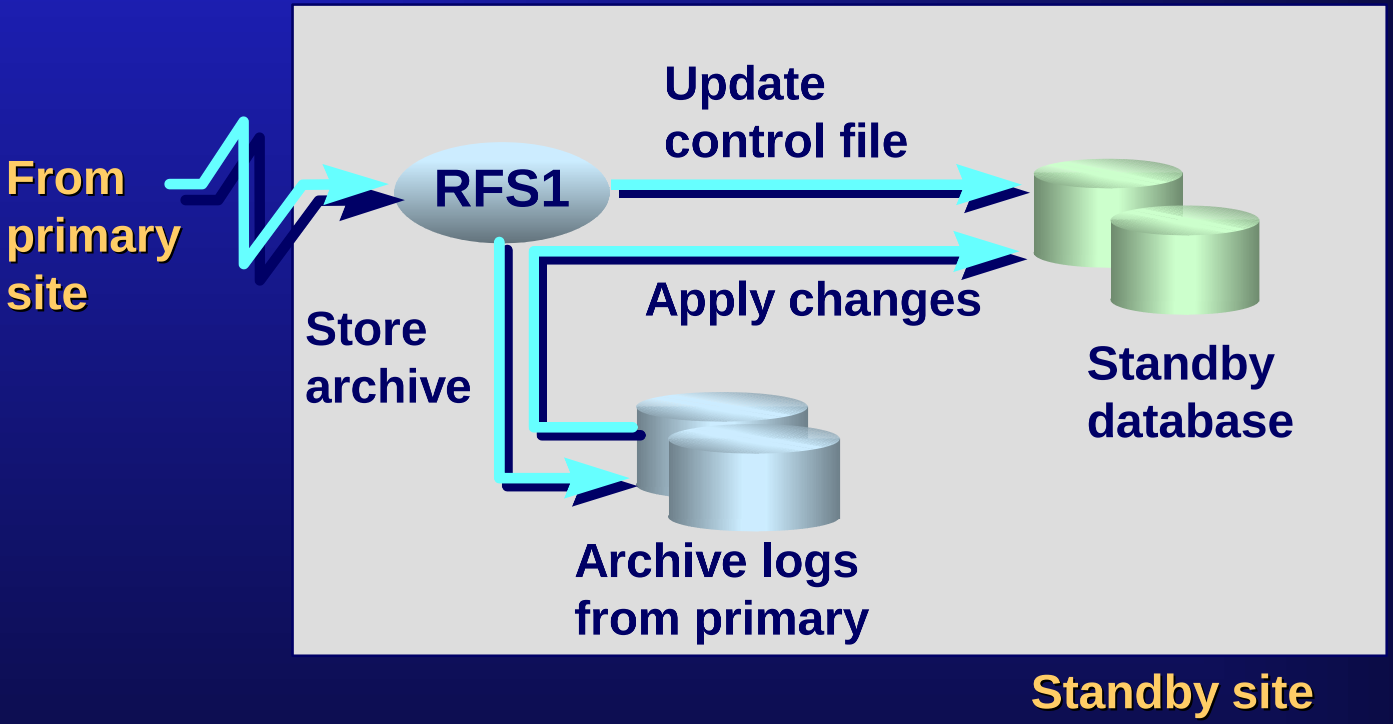
Controlling Archive Trace Information

- LOG_ARCHIVE_TRACE parameter
- Controls trace output for all archive processes and destinations
- Trace files stored in the USER_DUMP_DEST and BACKGROUND_DUMP_DEST directories
- Is system modifiable (ALTER SYSTEM)

Obtaining Archiving Information

- V$ARCHIVE_DEST
  - BINDING: Optional or mandatory
  - STATUS: Valid, inactive, deferred, error, disabled, or bad parameter
  - TARGET: Primary or standby
  - FAIL_SEQUENCE: Log if error occurred
  - ERROR: Error text
- V$ARCHIVE_PROCESSES
  - STATUS: Stopped, scheduled, starting, active, stopping, or terminated
  - STATE: Busy or idle
  - LOG_SEQUENCE: Current log archived

Standby Database Enhancements



Enabling Sustained Recovery at Standby Site

- Specify new initialization parameter STANDBY_ARCHIVE_DEST to store archives from primary site

```
standby_archive_dest=/primary_archive/
```

- Enable sustained recovery

```
ALTER DATABASE RECOVER
     MANAGED STANDBY DATABASE;
```

- Set a time-out, if necessary, to stop sustained recovery

Opening a Database for Read Operations

- Any database can be opened as a read-only database
- Standby databases can be used for reports or DSS operations
- A read-only database can be used to:
  - Execute queries
  - Make data files off-line or online

Disk sorts are only possible using locally managed tablespaces.

Using a Standby Database for Reads

1. Cancel sustained recovery.
   - Logs received are stored, but not applied.

```
ALTER DATABASE RECOVER
MANAGED STANDBY DATABASE CANCEL [IMMEDIATE];
```

2. Open database in read-only mode.

```
ALTER DATABASE OPEN READ ONLY;
```

3. Restart sustained recovery, when needed, after ensuring that there are no active sessions.

Suspending and Resuming Databases

- In order to backup, don't forget:

```
ALTER TABLESPACE ts_name BEGIN BACKUP;
```

- Suspend I/O activity:

```
ALTER SYSTEM SUSPEND;
```

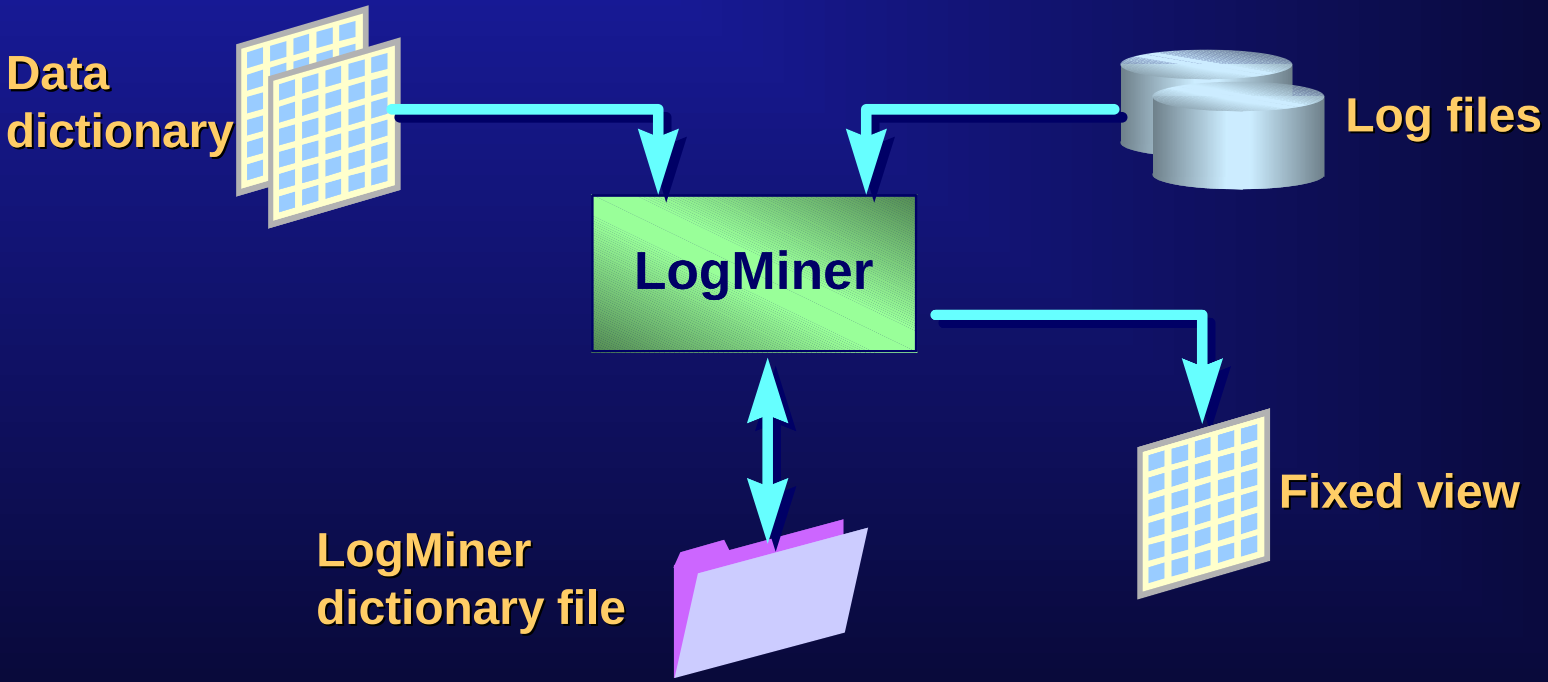
```
SELECT database_status FROM v$instance;
```

- Users can still access cached data.
- Any new I/O is blocked.
- Resume database I/Os:

```
ALTER SYSTEM RESUME;
```

Analyzing Redo Log Files

- Track changes to database
- Undo changes to the database
- Perform tuning and capacity planning



Build LogMiner Dictionary File

- Source can be an Oracle8 database:
  - Same platform as the analyzing instance
  - Same character set
- File is used to resolve object names:
  - Perform this step when new objects are added.
- Directory must be specified using UTL_FILE_DIR:

```
dbms_logmnr_d.build(
'orc1dict.ora','/oracle/database');
```

Specify Log Files to Analyze

- Online or archived files may be specified.
- Database does not need to be mounted.

1. Create a new list and specify first file:

```
dbms_logmnr.add_logfile('log1orc1.ora',
                        dbms_logmnr.NEW);
```

2. Specify additional files to be analyzed:

```
dbms_logmnr.add_logfile(
            'log2orc1.ora', dbms_logmnr.ADDFILE);
```

Analyze Specified Files

- Initiate log analysis:
  - Extracts details of transactions that occurred between the times specified
  - Can also use System Change Number ranges

```
dbms_logmnr.start_logmnr(dictfilename=>'orc1dict.ora',
starttime=>to_date('01/01/98:08AM','DD/MM/YY:HHAM'),
endtime=>to_date('03/01/1998:09AM','DD/MM/YYYY:HHAM'));
```

- Release resources:

```
dbms_logmnr.end_logmnr;
```

Viewing Log Information

V$LOGMNR_CONTENTS.SQL_REDO column contains statements reflecting changes:

```
SELECT timestamp, username, sql_redo
 FROM v$logmnr_contents
 WHERE seg_name = ‘EMP’;

TIMESTAMP USER SQL_REDO
--------  ----- ----------------------------
01-JAN-98  SCOTT  delete from EMP where rowid =
01-JAN-98  SCOTT  insert into EMP(...) ...
02-JAN-98  SCOTT  update EMP set SAL = … where …
```

Perform Logical Recovery

SQL_UNDO column shows statements that can be used to reverse the changes:

```
SELECT sql_redo, sql_undo
 FROM v$logmnr_contents
 WHERE seg_name = ‘EMP’;

SQL REDO                    SQL UNDO
---------------------      -----------------------
delete from EMP ...        insert into EMP(...) ...
insert into EMP ...        delete from EMP ...
update EMP set SAL = ...   update EMP set SAL = ...
```

Obtaining Information About Logs Analyzed

- `V$LOGMNR_DICTIONARY`
  - `TIMESTAMP`
  - `FILENAME`
- `V$LOGMNR_LOGS`
  - `LOG_ID, FILENAME`
  - `LOW_SCN, HIGH_SCN`
  - `LOW_TIME, HIGH_TIME`
- `V$LOGMNR_PARAMETERS`
- Arguments supplied during analysis

Initial Release of LogMiner Features and Limitations

- Records visible only to the analyzing session
- One row per redo record
- DML on scalar data and transaction control statements are supported
- DDL statements shown as DML on data dictionary
- SQL on chained or migrated data rows is not reconstructed
- Hex values for segment names shown if:
  - Object definition not in dictionary file
  - Changes are to clustered tables

Why Checkpointing?

- Ensure all dirty buffers are written to disk:

  Because of the LRU list, it could be possible that the oldest dirty buffer is never written because oldest does not mean least recently used.

- Faster Instance or Crash Recovery:

  The Oracle server rolls forward only after the last checkpoint.

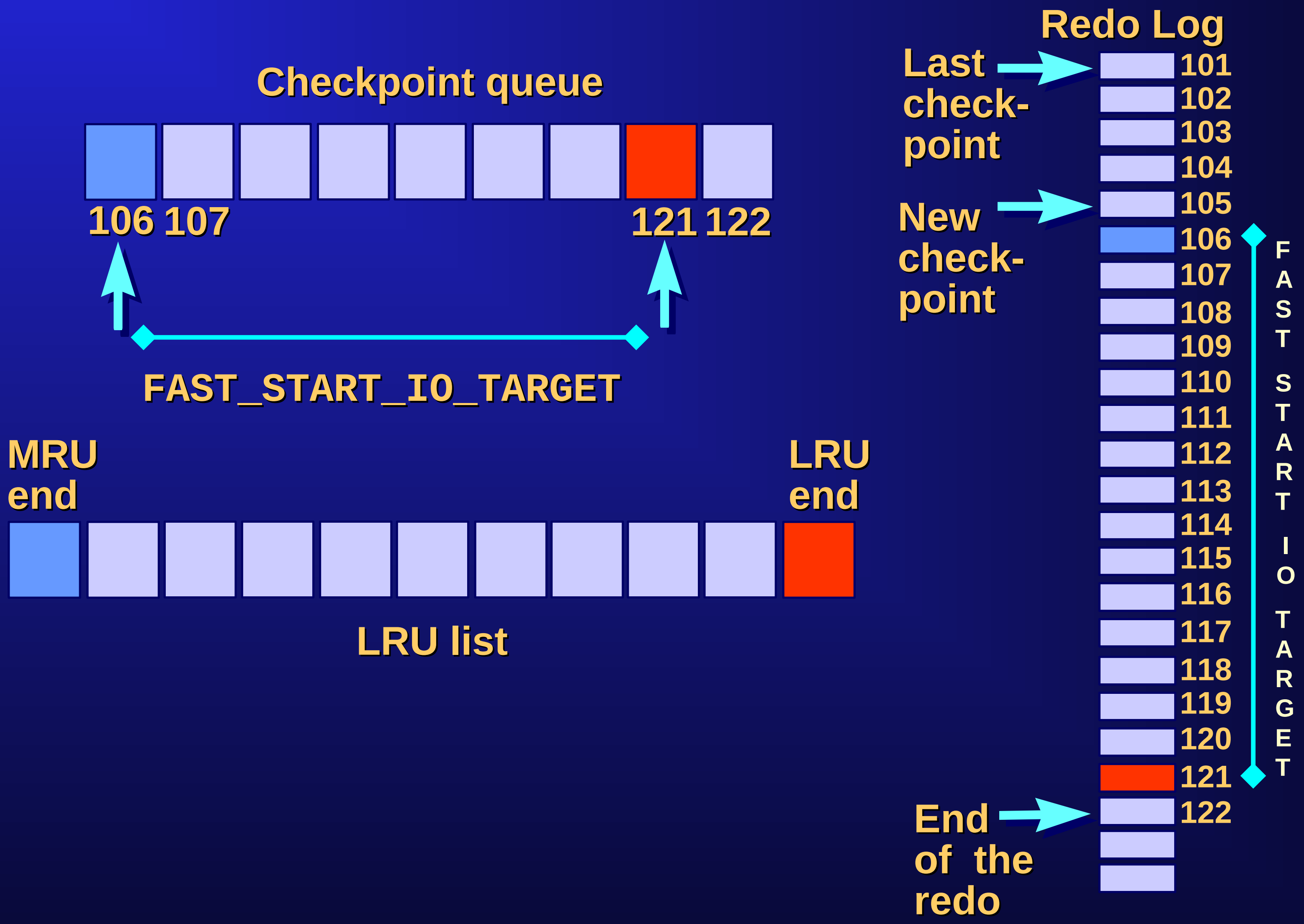
Fast-Start Checkpointing

New dynamic parameter to limit data file I/O during recovery:

```
fast_start_io_target = 1000
```

- Useful in establishing service level agreements with users
- Used in conjunction with other parameters to determine target for checkpointing

Use of FAST_START_IO_TARGET



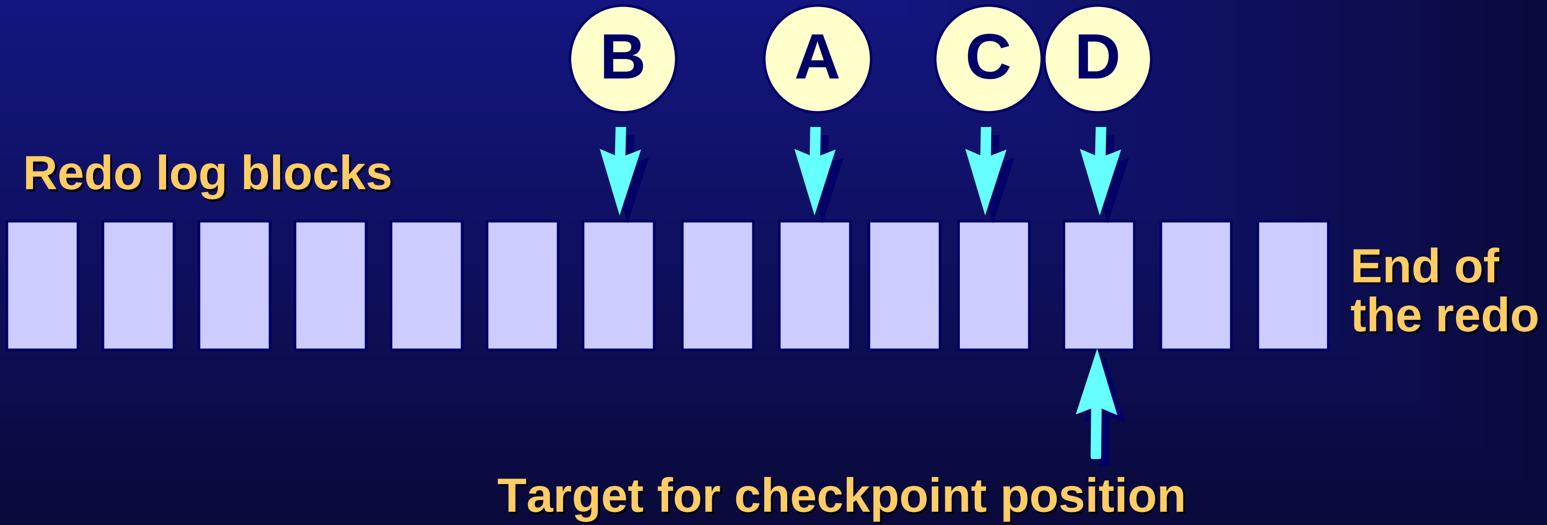
Other Factors Affecting Checkpointing

A **Target based on FAST_START_IO_TARGET**

B **90% of size of smallest redo log**

C **End of the log LOG_CHECKPOINT_TIMEOUT seconds ago**

D **LOG_CHECKPOINT_INTERVAL blocks from the end**



ORACLE®

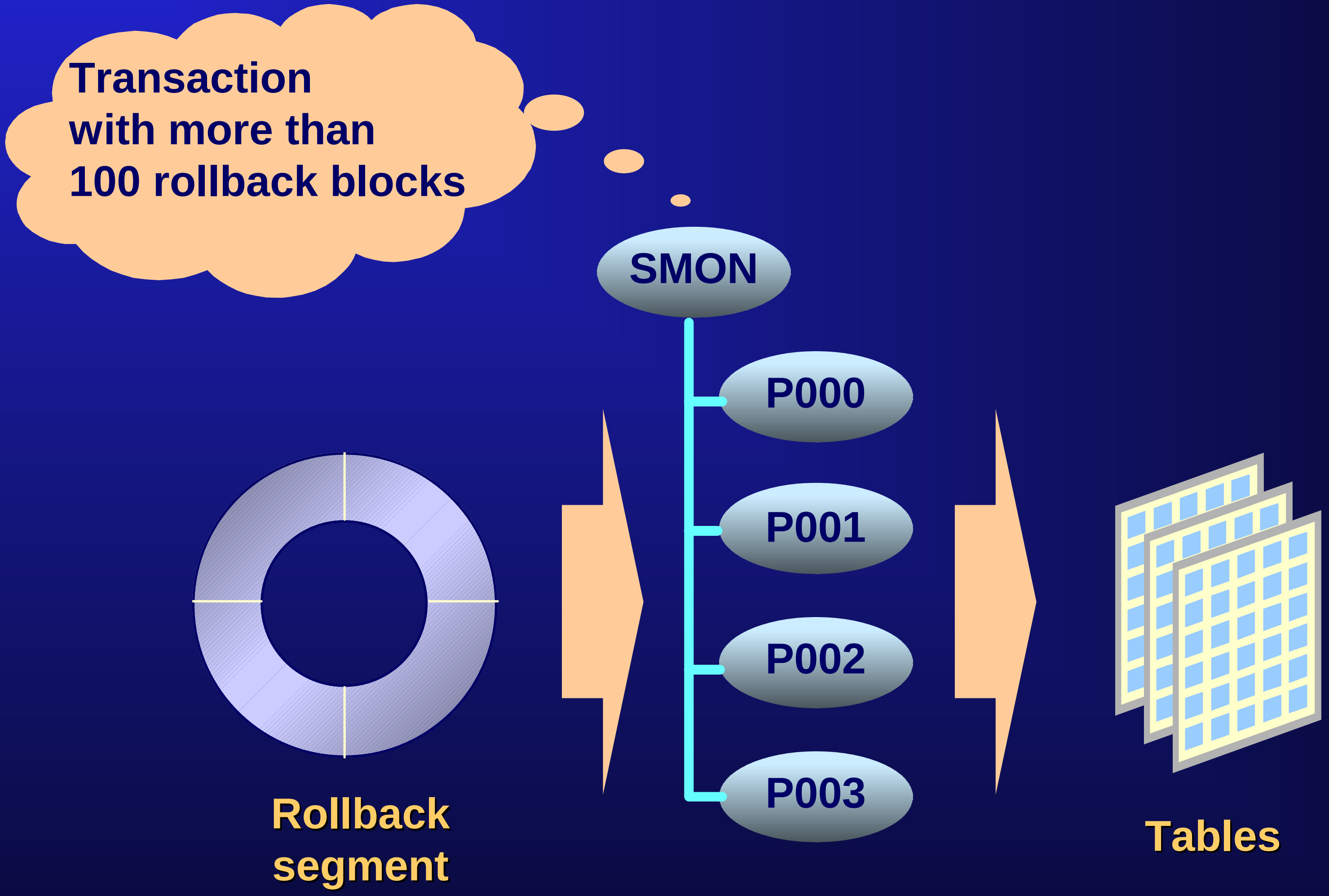
Factors Influencing Time to Recover

- Fast-start recovery time is at best an estimate.
- Recovery may take longer because:
  - Checkpoint target changed only at specific time intervals
  - Additional recovery activities such as reading logs not accounted for
  - Recovery time may be faster if parallel recovery is used

Monitoring Bounded Recovery Time

- Define FAST_START_IO_TARGET based on:
  - Service level required
  - AVGIOTIM column in V$FILESTAT
- Check impact of parameters from V$INSTANCE_RECOVERY:
  - RECOVERY_ESTIMATED_IOS
  - TARGET_REDO_BLKS
  - LOG_FILE_SIZE_REDO_BLKS
  - LOG_CHKPT_TIMEOUT_REDO_BLKS
  - LOG_CHKPT_INTERVAL_REDO_BLKS
  - FAST_START_IO_TARGET_REDO_BLKS

Fast-Start Parallel Rollback



Controlling Fast-Start Parallel Rollback

Define dynamic parameter

`FAST_START_PARALLEL_ROLLBACK:`

Value	Maximum Parallel Recovery Servers
FALSE	None
LOW	# slaves no more than 2 * CPU_COUNT
HIGH	# slaves no more than 4 * CPU_COUNT

Monitoring Parallel Rollback

- V$FAST_START_SERVERS
  - STATE: recovering or idle
  - PID, UNDOBLKSDONE
- V$FAST_START _TRANSACTIONS
  - USN, SLT, SEQ: Transaction ID
  - UNDOBLKSDONE
  - UNDOBLKSTOTAL
  - CPUTIME: Time in seconds

Fast-Start On-Demand Rollback

Server process encountering data to be rolled back performs the following:

- Roll back the block containing the required row
- Hand off further recovery, which may be in parallel, to SMON

Improved response

Detecting Block Corruption

- New dynamic (session, system deferred) initialization parameter:

```
db_block_checking = true
```

  - Performs logical block check on data and index blocks when they are changed
  - Dumps trace information
  - Raises ORA-1578 when trying to re-read an already found corrupted block
  - On corruption detection, data is lost
  - Check data off-line before using it
- In 8.1.6, this is always enabled on the SYSTEM tablespace.

Detecting Block Corruption Using DBMS_REPAIR

- Connect as user SYS.
- Create repair table:

```
dbms_repair.admin_tables('REPAIR_TABLE',
 DBMS_REPAIR.REPAIR_TABLE,
 DBMS_REPAIR.CREATE_ACTION,'temp_data');
```

- Check object for corruption:

```
dbms_repair.check_object('ORATRAIN',
'LOCATIONS',corrupt_count=>:cc);
```

  – Populates repair table
  – Can check table, partition or index

Making Objects Usable

- Mark blocks as corrupt:

```
dbms_repair.fix_corrupt_blocks('ORATRAIN',
 'LOCATIONS',fix_count=>:fc);
```

  – Uses repair table to identify blocks
  – Records time of fix in repair table

- Enable skipping of corrupt blocks:

```
dbms_repair.skip_corrupt_blocks(
 'ORATRAIN', 'LOCATIONS');
```

- Recover available data using table scan

Implications of Skipping Blocks

- All rows in blocks marked as corrupt are inaccessible
- Indexes may point to blocks marked corrupt
- Referential integrity constraints may be violated
- Disable and reenable constraints identify violations
- If the head of a freelist is corrupt, the freelist is initialized:
  - Use `REBUILD_FREELISTS`

Checking Index Entries Pointing to Rows in Corrupt Data Blocks

- Create table to hold results:

```
dbms_repair.admin_tables('ORPHAN_TAB1',
 DBMS_REPAIR.ORPHAN_TABLE,
 DBMS_REPAIR.CREATE_ACTION,'temp_data');
```

- Check index:

```
dbms_repair.dump_orphan_keys('ORATRAIN',
'LOC_PK', orphan_table_name=>'ORPHAN_TAB1',
 key_count=>:kc);
```

- Populates orphan table

Limitations of DBMS_REPAIR

- Tables with LOBs, nested tables, and VARRAYs can be analyzed, but out-of-line columns are ignored.
- Index-organized tables and LOB indexes cannot be analyzed.
- DUMP_ORPHAN_KEYS does not operate on bitmap and function-based indexes.
- DUMP_ORPHAN_KEYS only processes keys up to 3,950 bytes long.
- Clusters are supported in the SKIP_CORRUPT_BLOCKS and REBUILD_FREELISTS procedures only.

Adding Free Lists

- Dynamically add or reduce process free lists:

```
ALTER TABLE emp STORAGE (FREELISTS 5);
```

- Requires an exclusive lock on the segment
- Can be done on any segment where the FREELISTS option was possible before

Summary

In this lesson, you should have learned how to:

- Explain RMAN new features
- Specify multiple archive log locations
- Automate standby database recovery
- Start up a read-only database
- Suspend database I/Os
- Analyze log files using LogMiner
- Implement fast-start fault recovery
- Detect corrupt blocks and recover them
- Change the segment’s free lists number

Practice 12 Overview

This practice covers the following topics:

- Creating an RMAN catalog and using it for some basic commands
- Setting up multiple archive log locations
- Running queries in a read-only database
- Creating temporary locally-managed tablespaces
- Building LogMiner dictionary file
- Analyzing redo log information with LogMiner

13

Features of Net8

Objectives

After completing this lesson, you should be able to do the following:

- Describe the new service naming scheme
- Explain automatic registration
- Describe load balancing
- Configure the network for JServer
- Explain MTS new features introduced in 8.1.6

Automatic Listeners Registrations

- When started, PMON and Dispatchers register themselves to the listener (no `listener.ora`)
- Registration information:
  - Instance name
  - Service names
  - Instance and dispatchers load
- Registration can be done for one listener or multiple listeners (local or remote) by using:
  - `LOCAL_LISTENER` or `LISTENER`
  - `INSTANCE_NAME`
  - `SERVICE_NAMES`

Service Names

- Before Oracle8*i*, SID was specified in connect descriptors.
- With Oracle8*i*:
  - `SERVICE_NAME` and `INSTANCE_NAME` should be used in connect descriptors.
  - Database service names:
    - `SERVICE_NAME` and `INSTANCE_NAME` can be used in `init.ora` files.
  - Dispatcher service names:
    - `SERVICE` can be used in the `MTS_DISPATCHERS` parameter.
    - `MTS_SERVICE` is obsolete.

Load Balancing Overview

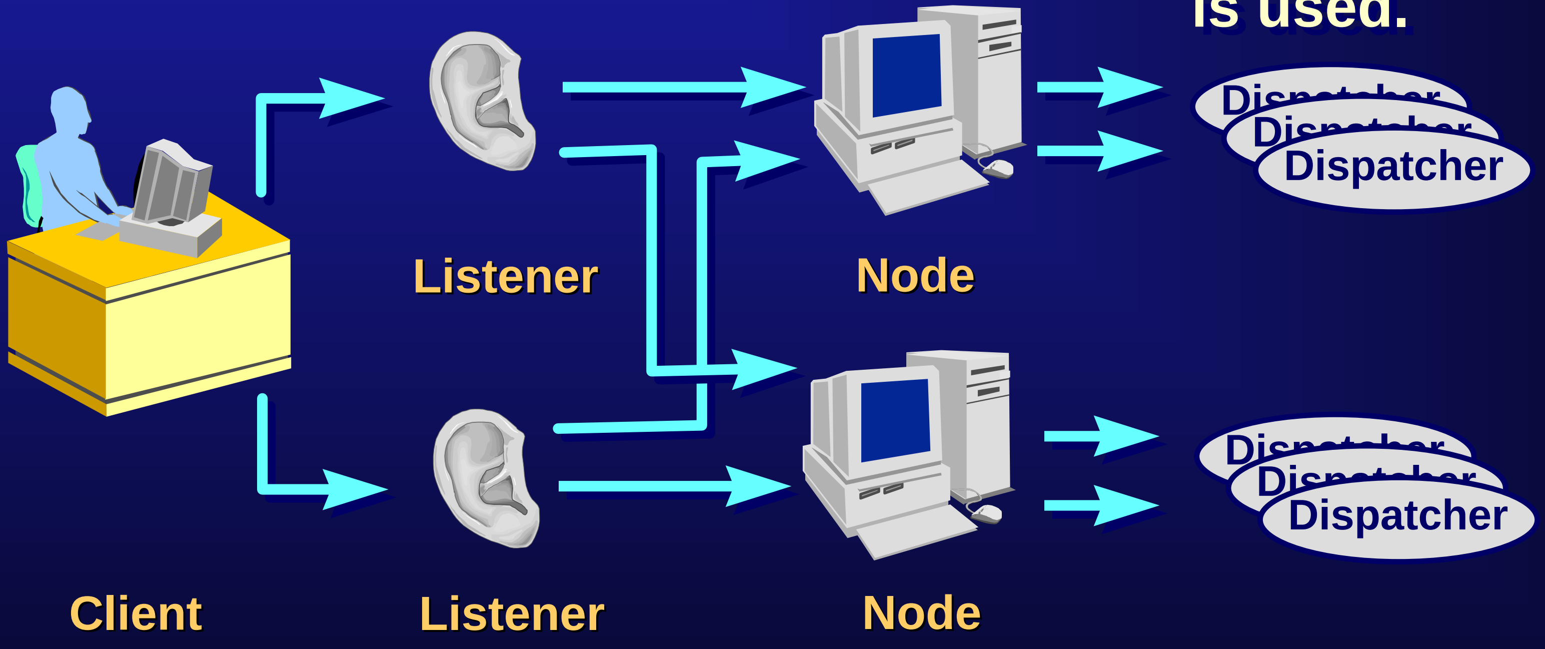
- Client Load Balancing for multiple listeners distributes the load over multiple listeners.
- Connect-Time Failover for Multiple Listeners failover connection requests in case listeners are down.
- Connection load balancing balances:
  - The number of active connections among various nodes
  - The number of active connections among various instances
  - The number of active connections among various dispatchers for the same service

Connection Load Balancing

1. Client randomly chooses from available listeners.

2. Node with least CPU usage is identified.

3. Dispatcher with least number of connections is used.

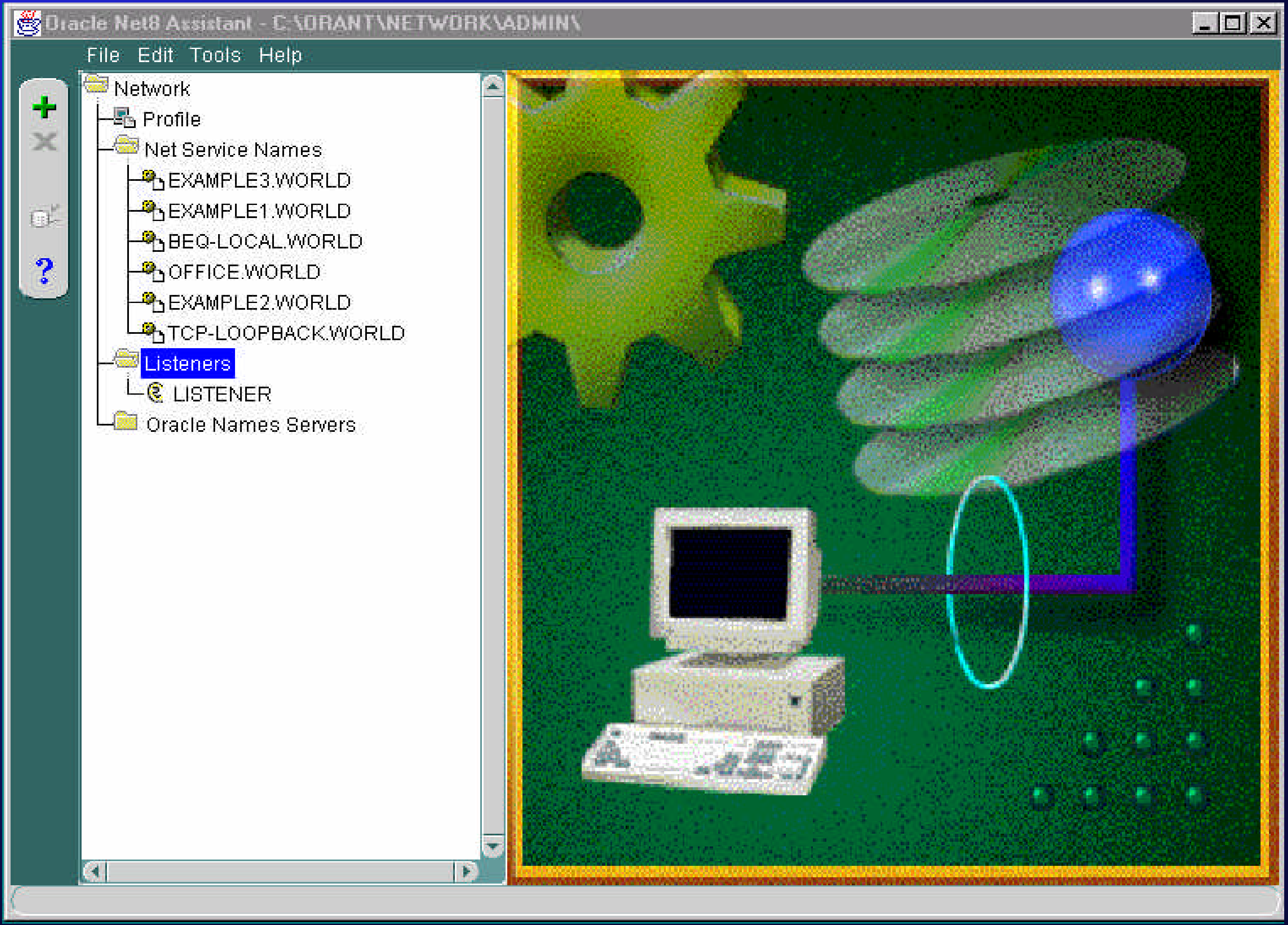


Client Load Balancing and Failover: Example

Enabling load balancing and failover in the `TNSNAMES.ORA` file:

```
TST8 = (DESCRIPTION=
                    (FAILOVER=on)
                    (LOAD_BALANCE=on)
                    (ADDRESS =  (PROTOCOL=tcp)
                                (HOST=host1)
                                (PORT=1521))
                    (ADDRESS =  (PROTOCOL=tcp)
                                (HOST=host2)
                                (PORT=1521))
         (CONNECT_DATA=(SERVICE_NAME=sales))
        )
```

Net8 Assistant



- Network
  - Profile
  - Net Service Names
    - EXAMPLE3.WORLD
    - EXAMPLE1.WORLD
    - BEQ-LOCAL.WORLD
    - OFFICE.WORLD
    - EXAMPLE2.WORLD
    - TCP-LOOPBACK.WORLD
  - Listeners
    - LISTENER
  - Oracle Names Servers

Configuring the Network for JServer

- Network protocols include the following:
  - Net8 for Java stored procedures
  - IIOP for CORBA and EJBs
- IIOP requires multithreaded server.
- These components may require configuration:
  - Dispatchers in `init.ora`
  - Shared servers in `init.ora`
  - Listener in `listener.ora`
- Configuration is best done with the Net8 Assistant.

Configuring the Network for Stored Procedures

- **listener.ora**

```
(ADDRESS = (PROTOCOL = TCP)(HOST = ed-bssun8)
  (PORT = 1521))
(PROTOCOL_STACK = (PRESENTATION = TTC)
  (SESSION = NS)))
```

- **init.ora**

```
MTS_DISPATCHERS = "(PROTOCOL = TCP)
                   (DISPATCHERS = 2)
                   (PRESENTATION=TTC)"
MTS_SERVERS = 5
MTS_MAX_SERVERS = 20
```

Configuring the Network for IIOP

- `listener.ora`

```
(ADDRESS = (PROTOCOL = TCP)(HOST = ed-bssun8)
           (PORT = 2481))
(PROTOCOL_STACK =
(PRESENTATION = GIOP)
(SESSION = RAW))
```

- `init.ora`

```
MTS_DISPATCHERS=  "(PROTOCOL=TCP)(DISPATCHERS=2)
(PRESENTATION=oracle.aurora.server.SGiopServer)"
MTS_SERVERS = 5
MTS_MAX_SERVERS = 20
```

MTS enhancement in 8.1.6

- MTS_CIRCUITS: Controls the maximum number of virtual circuits
- MTS_SESSIONS: Controls the maximum number of MTS sessions
- ALTER SYSTEM SHUTDOWN [IMMEDIATE]: Enables specific dispatcher shutdown
- INDEX attribute to the ALTER SYSTEM SET MTS_DISPATCHER command: Designates an entry in the initialization file

Summary

In this lesson, you should have learned the following:

- You can use service naming for load balancing and automatic registration.
- Automatic registration simplifies network configuration.
- MTS must be configured to use the IIOP protocol.

14

SQL*Plus, PL/SQL, and National Language Support Enhancements

Objectives

After completing this lesson, you should be able to do the following:

- Use SQL*Plus for database management
- Describe the use of PL/SQL for:
  - Event triggers
  - Autonomous transactions
  - Native Dynamic SQL
- Describe other PL/SQL enhancements
- Describe the National Language Support enhancements

Using SQL*Plus for Database Administration

- Server Manager functionality is moved into SQL*Plus.
- Migrate Server Manager scripts to SQL*Plus.
- Server Manager is no longer supported.
- Because `CONNECT INTERNAL` will no longer be supported after Oracle 8.1, you should start using `CONNECT / AS SYSDBA` instead

New Event Triggers

New triggering events:

- STARTUP
- SHUTDOWN
- SERVERERROR
- LOGON
- LOGOFF
- CREATE
- ALTER
- DROP
- ANALYZE*
- AUDIT*, NOAUDIT*
- COMMENT*
- GRANT*, REVOKE*
- RENAME*
- TRUNCATE*
- DDL*

New trigger levels:

- Database
- Schema

```
CREATE TRIGGER db_pin AFTER
STARTUP ON DATABASE
BEGIN
sys.dbms_shared_pool.keep('SYS.
STANDARD');  -- Add any others
END;
```

Note: * is for 8.1.6 only

Autonomous Transactions

```
PROCEDURE atm_trans
...
  log_card_usage (cardnum, loc);
  INSERT INTO txn VALUES (9001,1000,...);
END;
```

```
PROCEDURE log_card_usage
  (   p_cardno     IN    NUMBER,
      p_loc        IN    NUMBER )
IS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO usage VALUES (p_cardno, p_loc);
  COMMIT;
END;
```

Other PL/SQL Enhancements

- Support for very large packages
- PL/SQL bulk binds
- Dynamic SQL in PL/SQL
- Parameter passing by reference
- PL/SQL package for REF-based operations
- Monitoring and analysis of program execution

National Language Support: New National Database Character Set in 8.0

A national database character set clause at database creation time has been added:

```
CREATE DATABASE U16
LOGFILE
  GROUP 1 ('/DISK3/log1a.rdo',’/DISK4/log1b.rdo’)SIZE 1 M,
  GROUP 2 ('/DISK3/log2a.rdo',’/DISK4/log2b.rdo’)SIZE 1 M
DATAFILE
  '/DISK1/system01.dbf' size 50M autoextend on
  CHARACTER SET US7ASCII
  NATIONAL CHARACTER SET JA16SJISFIXED;
```

National Language Support: Euro Currency Symbol Support

- Dual currency support (for euro):

```
ALTER SESSION SET NLS_DUAL_CURRENCY=‘EUR’;
```

```
 SELECT TO_CHAR(123,’U999’) FROM dual;
 TO_CHAR(123,'U
 --------------
        EUR123
```

- All participating member states have territory files updated to accommodate NLS_DUAL_CURRENCY parameter.

National Language Support: New Local Data

- Expanded Asian character set support: MS Windows Code Pages 932, 949, 936, 950 (Japanese, Korean, and simplified and traditional Chinese)
- Programming interfaces (OCI) provide cartridge developers and application developers access to international information and services.
- Fixed-width Unicode (UCS2) character support is provided in the following client interfaces: OCI, Pro*C/C++, and ODBC.
- Expanded NLS data has been included. New territories have been added.

National Language Support: Linguistic Index Support

- High-performance with local sorting:

```
CREATE INDEX nls_ename ON
     emp (NLSSORT(ename, ‘NLS_SORT = German’));
```

- NLS_COMP parameter for linguistic comparisons

Summary

In this lesson, you should have learned the following:

- SQL*Plus replaces Server Manager.
- PL/SQL event triggers include DB events, user events, and DDL events.
- PL/SQL supports Native Dynamic SQL.
- It is possible to have `AUTONOMOUS PL/SQL` blocks.
- National language support enhancements include:
  - Support for the euro currency symbol
  - Expanded territory support
  - Linguistic indexing

15

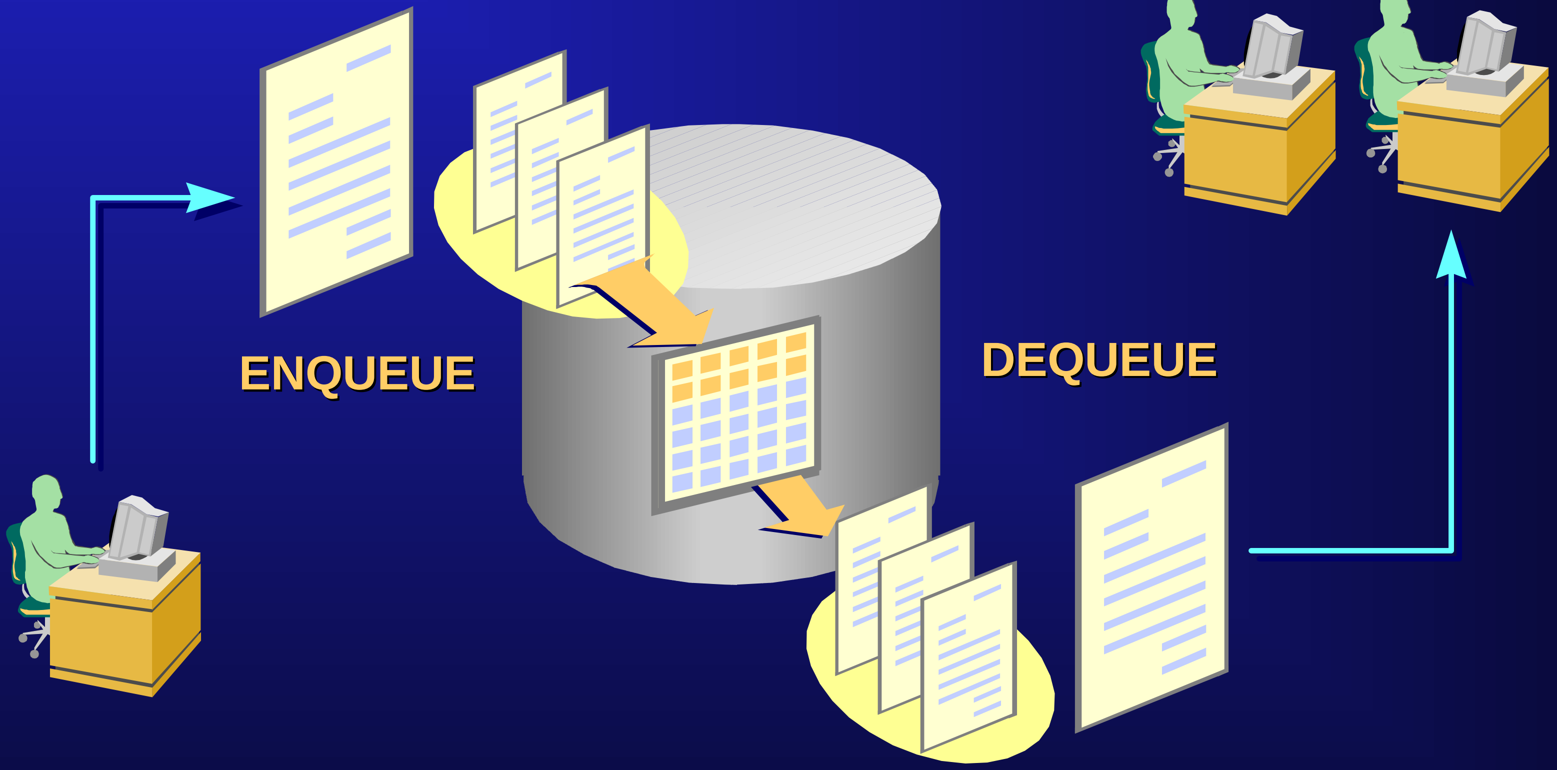
Advanced Queuing

Objectives

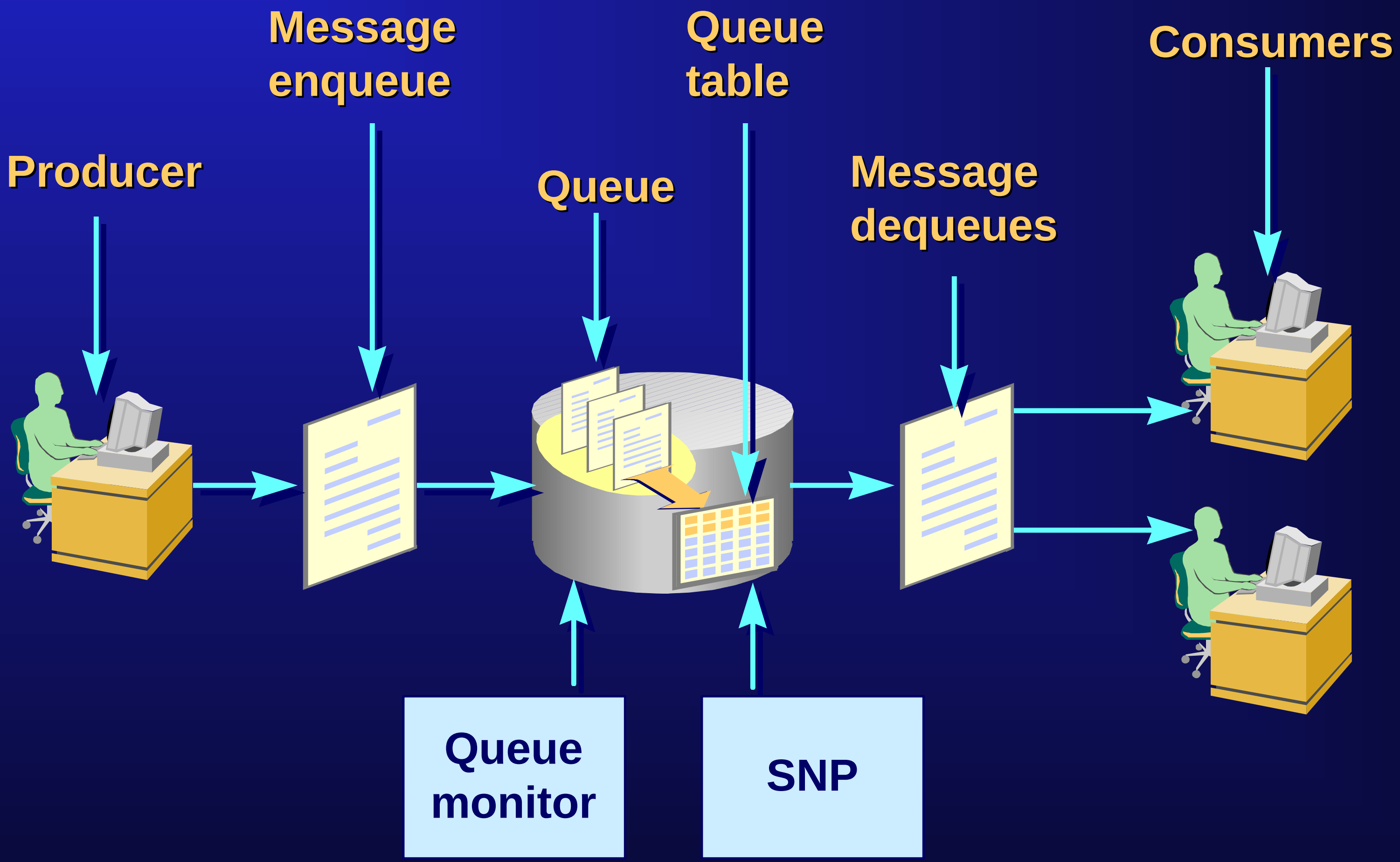
After completing this lesson, you should be able to do the following:

- List the features used to manage queue tables and queues
- Start a Queue Monitor process to support message expiration, retry, and delay
- Start an SNP process to propagate messages
- Create roles and objects to support Advanced Queuing (AQ)

Overview



Queuing Components



ORACLE®

Features of Advanced Queuing

Flexibility to meet the application requirements:

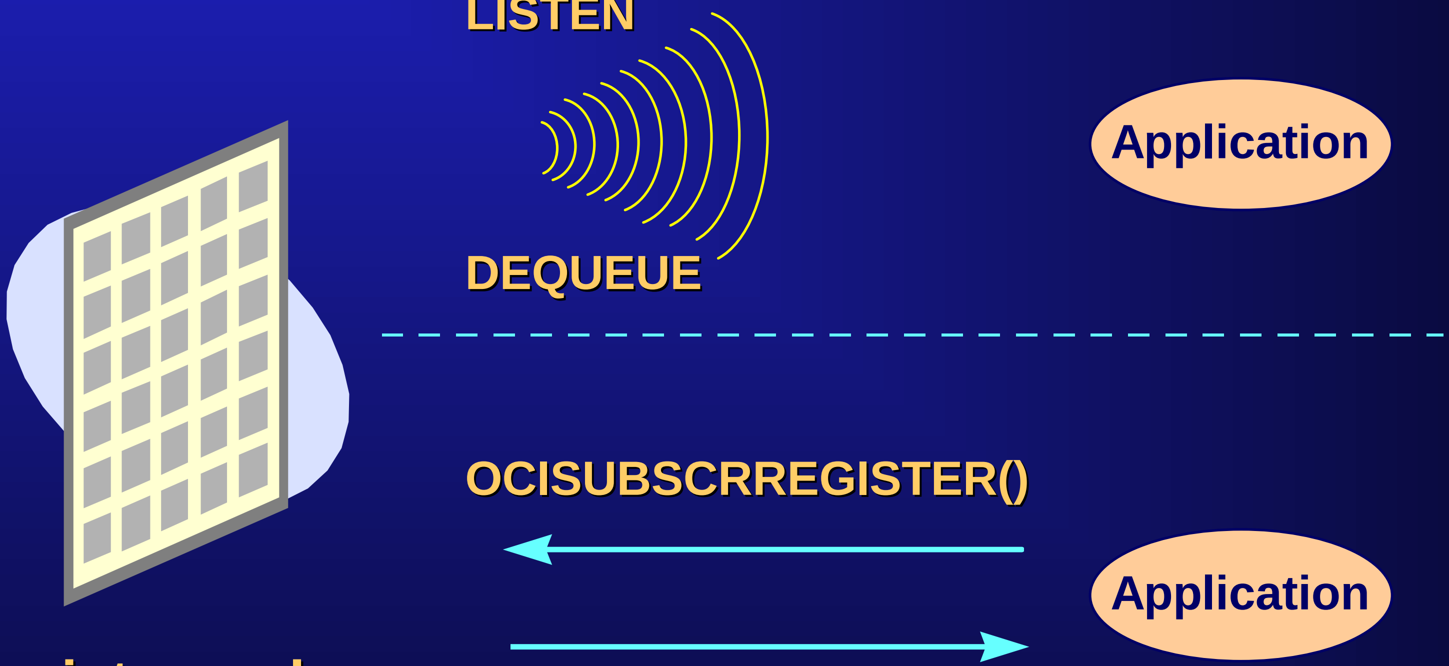
- Programmatic interfaces for AQ:
  - PL/SQL using `DBMS_AQADM` and `DBMS_AQ`
  - C++ using OCI
  - C or C++ using the Pro*C/C++ precompiler
  - Visual Basic using Oracle Objects for OLE
  - Other languages using Oracle Objects for OLE
  - Java Native API or JMS API
- Structured payload using object types
- Message priority and ordering

Features of Advanced Queuing

Flexibility to meet the application requirements:

- Time delay or expiration
- Independent queue transactions
- Dequeue with `browse`, `lock`, `remove`, or `remove_nodata`
- Message grouping by process or transaction
- Multiple recipients
- SQL access to payload and properties
- Exception queues for error-handling
- Persistent or non-persistent queues

Features of Advanced Queuing



Listen

LISTEN

DEQUEUE

Application

OCISUBSCRREGISTER()

Application

Register and callback

OCINOTIFICATIONCALLBACK()

OCIAQDeq()

Features of Advanced Queuing

- Queue security at the object and system level
- AQ statistics:
  - Current state of the queuing system
  - History of each message
  - Propagation
- Interfaces to Oracle utilities:
  - Import/export done at queue table granularity
  - Oracle Enterprise Manager GUI
- Support for Oracle Parallel Server environments

AQ Implementation Tasks

- DBA configures the instance.
- DBA creates the AQ Administrator.
- AQ Administrator uses `DBMS_AQADM` to:
  - Create queuing objects
  - Grant privileges to AQ developers
- AQ developers use `DBMS_AQ` to:
  - Enqueue messages
  - Dequeue messages

DBA Configures the Instance

Timer processes:

- Required for messages that require a delay, expiration, or timed retention
- Example of `init.ora` parameter:

```
AQ_TM_PROCESSES = 1
```

Job queue processes:

- Required to propagate queues
- Example of `init.ora` parameter:

```
JOB_QUEUE_PROCESSES = 2
```

DBA Creates the AQ Administrator

Create user AQ as the AQ Administrator:

```
CREATE USER aq IDENTIFIED BY aq;
GRANT AQ_ADMINISTRATOR_ROLE TO aq;
GRANT CONNECT, RESOURCE TO aq;
[EXECUTE DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (
privilege   => 'MANAGE_ANY',
grantee  => 'AQ',
admin_option   => FALSE );]
```

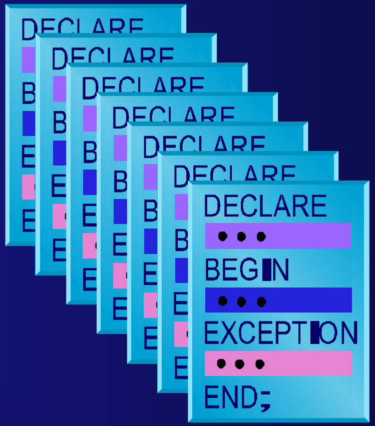
Security in Oracle 8.0 and 8*i*

- In Oracle8*i:*
  - Compatibility queues: 8.0 and/or 8.1
  - `AQ_USER_ROLE` for user’s schema 8.1 compatible queues only and any 8.0 compatibility queues
  - Execute on `DBMS_AQADM` to maintain user’s AQ objects only
  - Execute on `DBMS_AQ` for user’s schema queues only
  - New system privileges
  - New queues privileges
  - `GRANT_TYPE_ACCESS` is made obsolete
  - To migrate or downgrade compatibility, use `DBMS_AQADM.MIGRATE`

DBMS_AQADM Package

Categories of DBMS_AQADM procedures:

- Security
- Queue table maintenance
- Queue maintenance
- Subscriber maintenance
- Propagation maintenance



DBMS_AQ Package

ENQUEUE adds a message to a queue:

```
dbms_aq.enqueue (
  queue_name               IN   VARCHAR2,
  enqueue_options          IN   enqueue_option_t,
  message_properties       IN   message_properties_t,
  payload                  IN   <object_type|RAW>,
  msgid                    OUT  RAW )
```

DEQUEUE retrieves a message from a queue:

```
dbms_aq.dequeue (
  queue_name               IN  VARCHAR2,
  dequeue_options          IN  dequeue_options_t,
  message_properties       OUT message_properties_t,
  payload                  OUT <object_type_name>,
  msgid                    OUT RAW)
```

LISTEN monitors multiple queues for a message.

Data Dictionary Views

- **DBA_QUEUE_TABLES describes the names and types of all queue tables created in the database.**
- **DBA_QUEUES contains operational characteristics for every queue in a database.**
- **DBA_QUEUE_SCHEDULES describes the current schedules for propagating messages.**
- **QUEUE_PRIVILEGES describes queues for which the user is the grantor, grantee, or owner; or access to the queue is granted to an enabled role or PUBLIC.**
- **V$AQ describes statistics for the queues in the database.**

Other Queue Objects

CREATE_QUEUE_TABLE creates:

- aq$<queue_table_name> is a read-only view.
- aq$_<queue_table_name>_e is the default exception queue.
- aq$_<queue_table_name>_t is an index for queue monitor operations.
- aq$_<queue_table_name>_i is an index or an IOT for dequeues on multiple consumer queues.
- aq$<queue_table_name>_s contains subscribers.
- aq$<queue_table_name>_r contains rules.
- aq$<<queue_table_name>_h contains history.

Export/Import

- Export is done at the queue table granularity
- Export in `FULL` or `USER` mode takes care of queues.
- When exporting in `TABLE` mode, the user must export manually all important tables (see previous slide).
- Ignore obsolete `ROWID` warnings for queues.
- Incremental exports are not supported.
- Try to avoid the `IGNORE=Y` parameter during import.

Summary

In this lesson, you should have learned the following:

- Oracle Advanced Queuing offers the ability to defer execution of work.
- DBA configures the database for queues by:
  - Setting `AQ_TM_PROCESSES` or `JOB_QUEUE_PROCESSES`
  - Creating the AQ Administrator
- The AQ Administrator uses package `DBMS_AQADM`.
- The AQ developer uses package `DBMS_AQ`.

Practice 15 Overview

This practice covers the following topics:

- Adding Time Manager processes
- Adding Job Queue processes
- Creating AQ administrator users
- Creating queues and queue tables
- Viewing queues data dictionary information
- Enqueing and dequeing messages

16

Database Security

Objectives

After completing this lesson, you should be able to do the following:

- Explain data encryption in tables
- Explain Unique Schemas and Schemas-Independent Users
- Describe N-Tier authentication
- Describe invoker's rights security management
- Implement application context areas
- Implement fine-grained access control

Data Encryption

- Encryption is easy, but key management is a killer!
- DBMS_OBFUSCATION_TOOLKIT package
  - DESEncrypt(*input*,*key*,*encrypted_data*)
  - DESDecrypt(*input*,*key*,*decrypted_data*)
- Created by:
  - dbmsobtk.sql (package specification)
  - prvtobtk.plb (package body)

Unique Schemas

- No direct connections through them
- Contain only objects and rights

```
CONNECT OE/OE;
```

```
ALTER SESSION SET CURRENT_SCHEMA=APP;
```

```
SELECT schemaname FROM V$SESSION
WHERE username=‘OE’;
```

```
SELECT SYS_CONTEXT('USERENV','CURRENT_SCHEMA’)
FROM DUAL;
```

Shared Schema

- Enables Enterprise Users to share a single schema
- Avoids the one-to-one mapping between Enterprise and Global Users
- Only available over SSL authentication of Enterprise Users

```
CREATE USER application_user IDENTIFIED
GLOBALLY AS '';
```

N-tier Authentication/Authorization

- Limit the application user power
- Keep the client user identity

```
ALTER USER <client> GRANT CONNECT
THROUGH <application> WITH ROLE <role>;
```

```
ALTER USER <client> REVOKE CONNECT
THROUGH <application> WITH ROLE <role>;
```

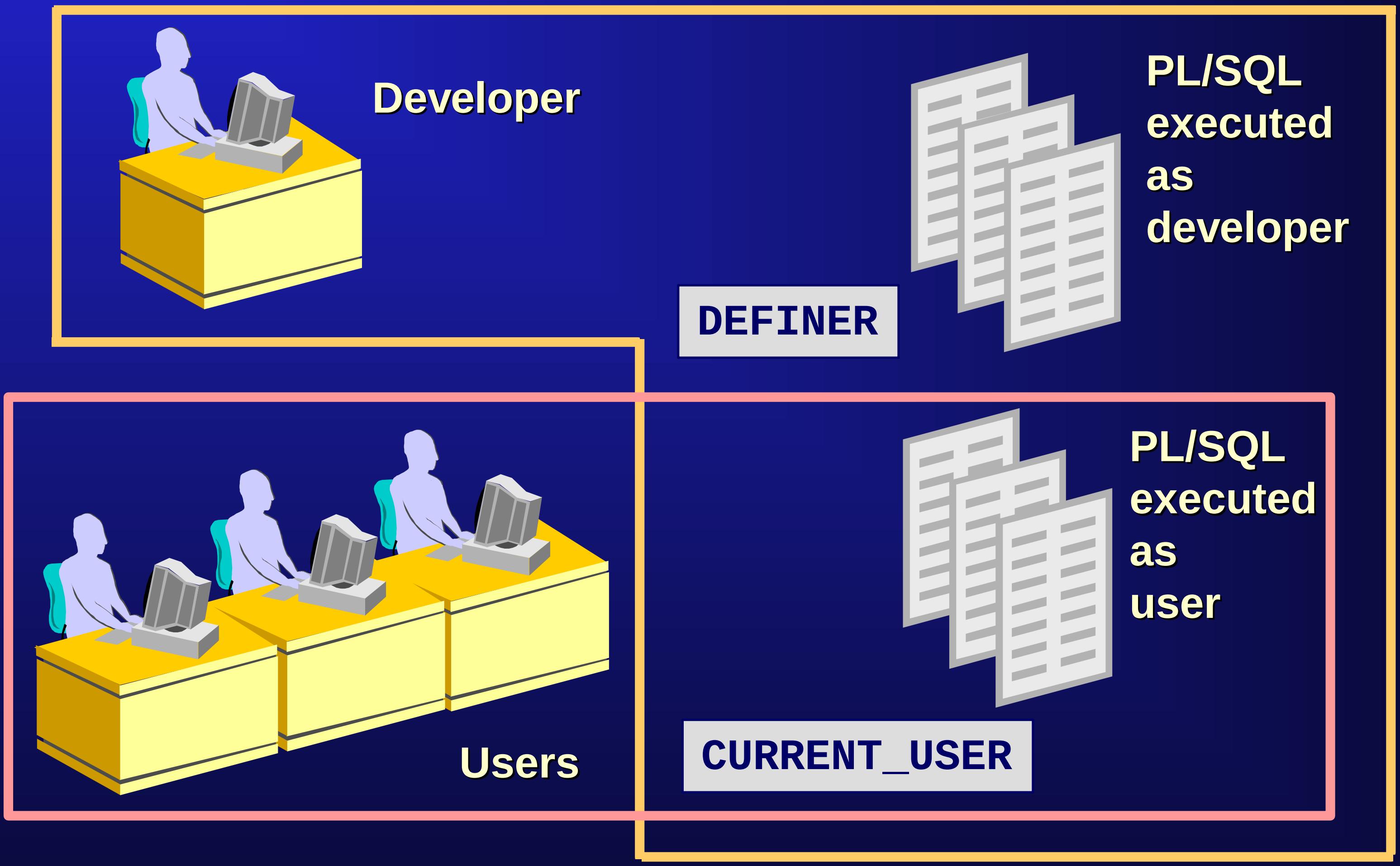
```
AUDIT <operation> BY <application> ON
BEHALF OF <client>;
```

Enterprise User Management: Overview

Useful for large user communities accessing numerous databases and applications

- Enables single sign-on over Secure Socket Layer (SSL)
- Oracle Wallet Manager for n-tier authentication
- Integrate standard Lightweight Directory Access Protocol (LDAP) version 3 with Oracle8 release 8.1
- Support for RADIUS authentication
- Native authentication on Windows NT

Invoker’s Rights



Developer

PL/SQL
executed
as
developer

DEFINER

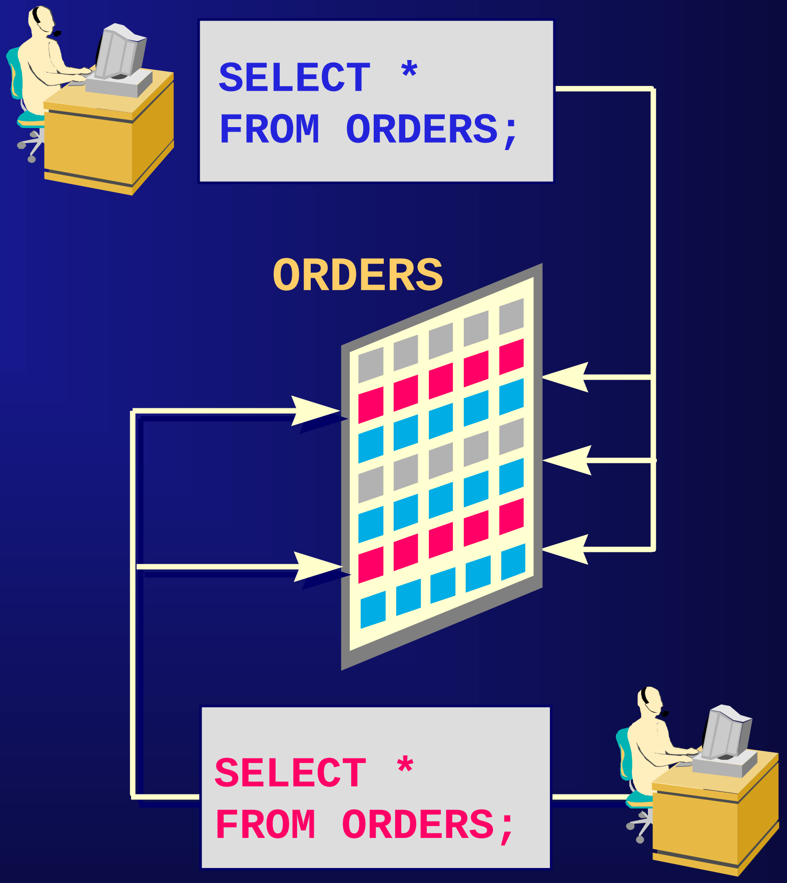
Users

PL/SQL
executed
as
user

CURRENT_USER

Fine-Grained Access Control Overview

- Associate security policies (implemented by functions) with tables or views
- Server automatically enforces security policies (no matter how data is accessed)
- Application context (optional) enables flexible access control definition



```
SELECT *
FROM ORDERS;
```

```
SELECT *
FROM ORDERS;
```

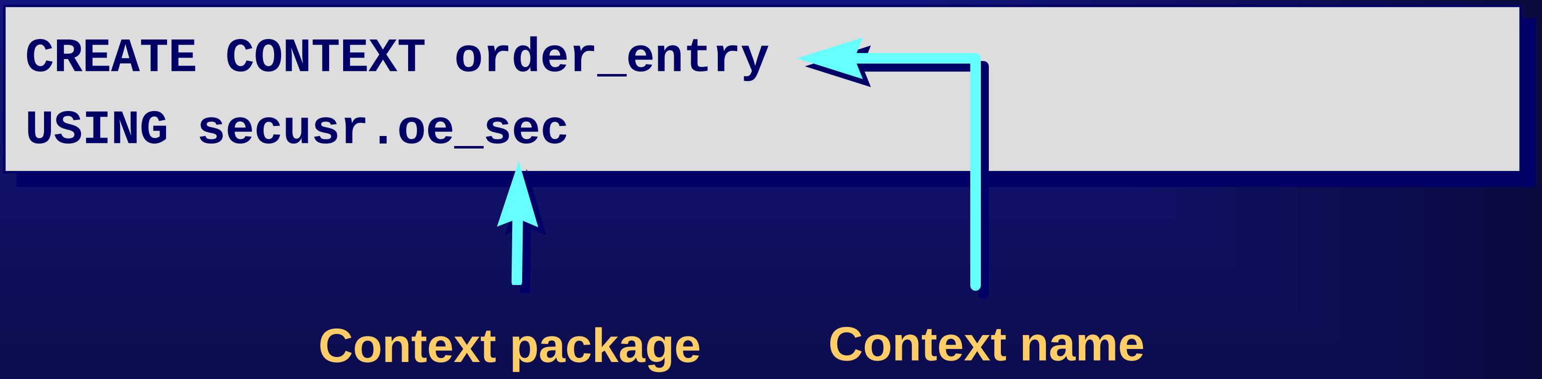
Application Context Features Overview

- The USERENV built-in context:
  - SYS_CONTEXT('userenv', '<*attribute*>')
- Application-specific contexts:
  - Each application can have its own context, with its own attributes.
  - SYS_CONTEXT('<*app_ctx*>', '<*attribute*>')
- Reparse any cursor using context information to see context modifications:
  - Context attributes are static.

Building Application Context

Create a context:

- Each context has unique name within the database.
- Database binds context name to context package that implements it.



```
CREATE CONTEXT order_entry
USING secusr.oe_sec
```

Context package

Context name

ORACLE®

Building Application Context

```
PROCEDURE set_cust_num IS custnum number
BEGIN
  SELECT cust_no INTO custnum
  FROM customers where username =
  SYS_CONTEXT('USERENV','session_user');

  DBMS_SESSION.SET_CONTEXT
   ('order_entry','cust_num', custnum);
END;
```

Create a Logon Trigger

```
GRANT EXECUTE ON secusr.oe_sec TO
Public;
```

```
CREATE OR REPLACE TRIGGER secusr.set_ctx
AFTER LOGON ON DATABASE
BEGIN
  secusr.oe_sec.Set_Cust_Num;
END;
/
```

Implementing Fine-Grained Access Control

```
CREATE PACKAGE BODY oe_security IS
  FUNCTION custnumsec
    RETURN VARCHAR2 IS
  BEGIN
    IF SYS_CONTEXT('order_entry', 'rôle')=
     'customer'
    THEN return 'cust_no = SYS_CONTEXT(
             ''order_entry'',''cust_num'')';
    ELSIF SYS_CONTEXT('order_entry', 'role') ='clerk'
    THEN
      return 'sales_region = SYS_CONTEXT(
               ''order_entry'',''region'')';
    ELSE return '';
    END IF;
  END;
END;
```

```
GRANT EXECUTE ON secusr.oe_security TO Public;
```

Implementing Fine-Grained Access Control

Associate policy package with tables and views by using PL/SQL package (DBMS_RLS):

- ADD_POLICY
- DROP_POLICY
- ENABLE_POLICY
- REFRESH_POLICY

```
DBMS_RLS.ADD_POLICY
('apps','orders','order_policy',
'secusr','oe_security.custnum_sec','select')
```

Using Fine-Grained Access Control

Direct or indirect access to tables with attached policy automatically invokes the policy:

- Data server calls package policy
- Package policy returns predicate (which uses application context)
- Data server rewrites query using predicate

Summary

In this lesson, you should have learned how to do the following:

- Implement fine-grained access control for application-specific access
- Select invoker's rights to control schema access from stored PL/SQL
- Manage Enterprise Users and Global Users using the new functionalities introduced in 8.1.6
- Encrypt and decrypt data in tables