
ORACLE®

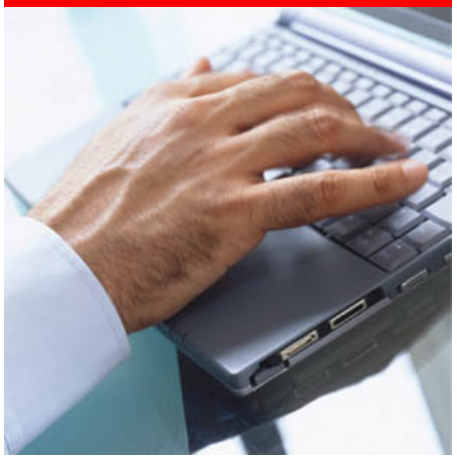
Your. Open. World.

Long Transactions with Oracle Database 11g Workspace Manager - S298742

Bill Beauregard
Senior Principal Product Manager
Server Technologies, Oracle USA

ORACLE

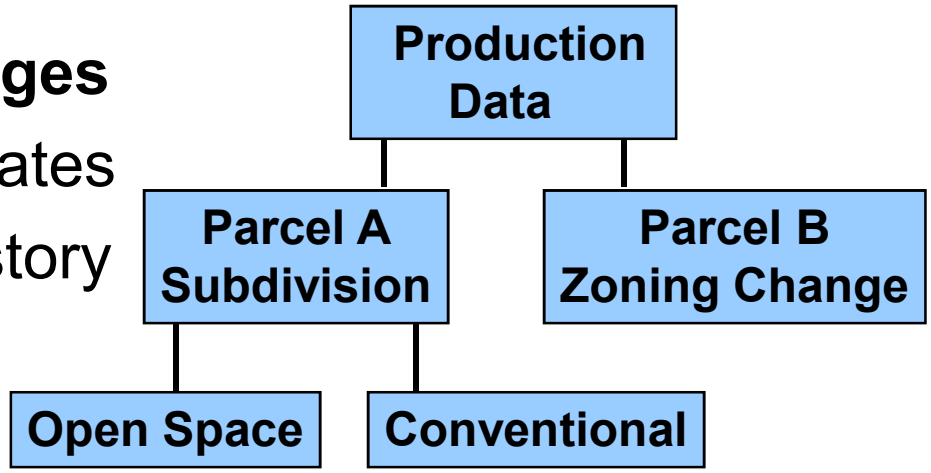
Agenda



- Workspace Manager overview
- Major features
- Use cases
- Best Practices
- Enhancements by release

Managing Long Duration Change to Data

- **Isolating a group of changes**
  - Data in multiple valid states
  - Current – Planned - History
  - Publish all or nothing



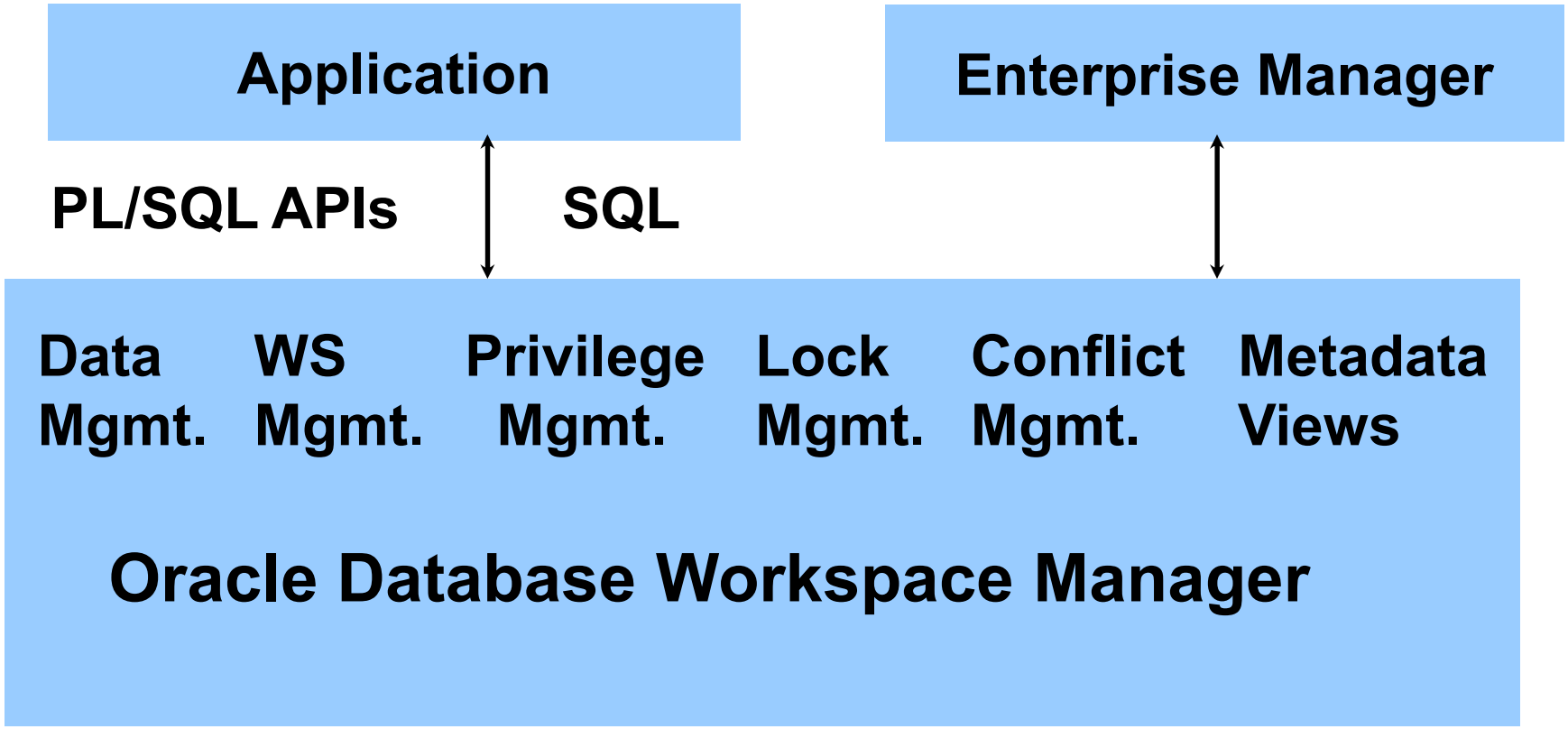
- **Creating multiple scenarios**
  - "What if" analysis
  - Multiple application testers using the same data set
- **Retaining a history of changes indefinitely**
  - Go back to any point in time in past years
  - See a transactionally consistent view of the database

Oracle Workspace Manager

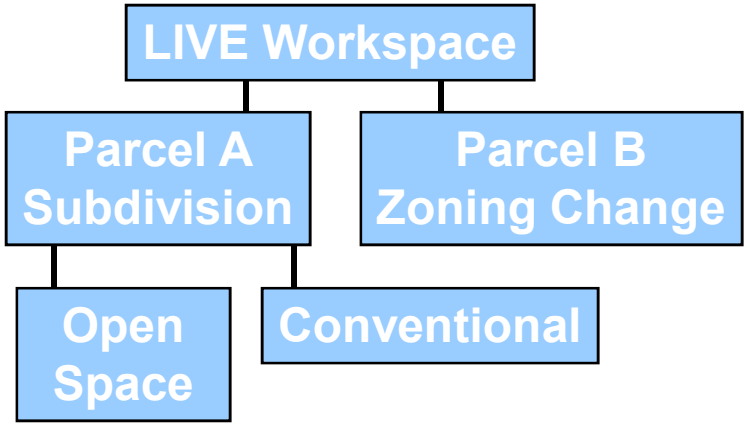
- A feature of Oracle Database (PL/SQL package)
- Manages long transactions for isolation, “what if” scenarios and history
- Saves time, money and resources
- Unit of versioning is a table
- Creates virtual workspaces & savepoints
- Incorporated in 13 GIS vendors’ solutions

http://search.oracle.com “Workspace Manager”

Workspace Manager Architecture



Workspace Manager Features



- Workspace hierarchies
- No changes to application SQL
- Optimistic and pessimistic locking
- Persistent workspace locks
- Continually Refreshed workspaces
- Multi-Parent Workspaces
- Differencing and Conflict detection/resolution
- Partial and Full Merge/Refresh of workspace/table
- Garbage collection operations to optimize version storage

Workspace Manager Operations

- **Table**: EnableVersioning, DisableVersioning, DDL, merge, refresh, rollback, topologies
- **Workspace**: create, goto, refresh, merge, rollback, remove, multiparent, compress, alter, events
- **Savepoints**: create, alter, goto, rollback
- **History**: goto date
- **Valid Time**: valid from & till, set session valid time
- **Privileges**: access, create, delete, rollback, merge
- **Access Modes**: read, 1 writer, WS operations, none
- **Locks**: exclusive and shared
- **Differences**: compares savepoints and workspaces
- **Detect / Resolve Conflicts**: choose version to merge
- **Bulk data movement**: SQL*Loader, Import/Export, Replication

Customer Examples

Cheaper

- 49 databases consolidated - City of Edmonton
- 50% reduction in hardware requirements - HMO
- 4x work w/ ½ resources - City of Virginia Beach

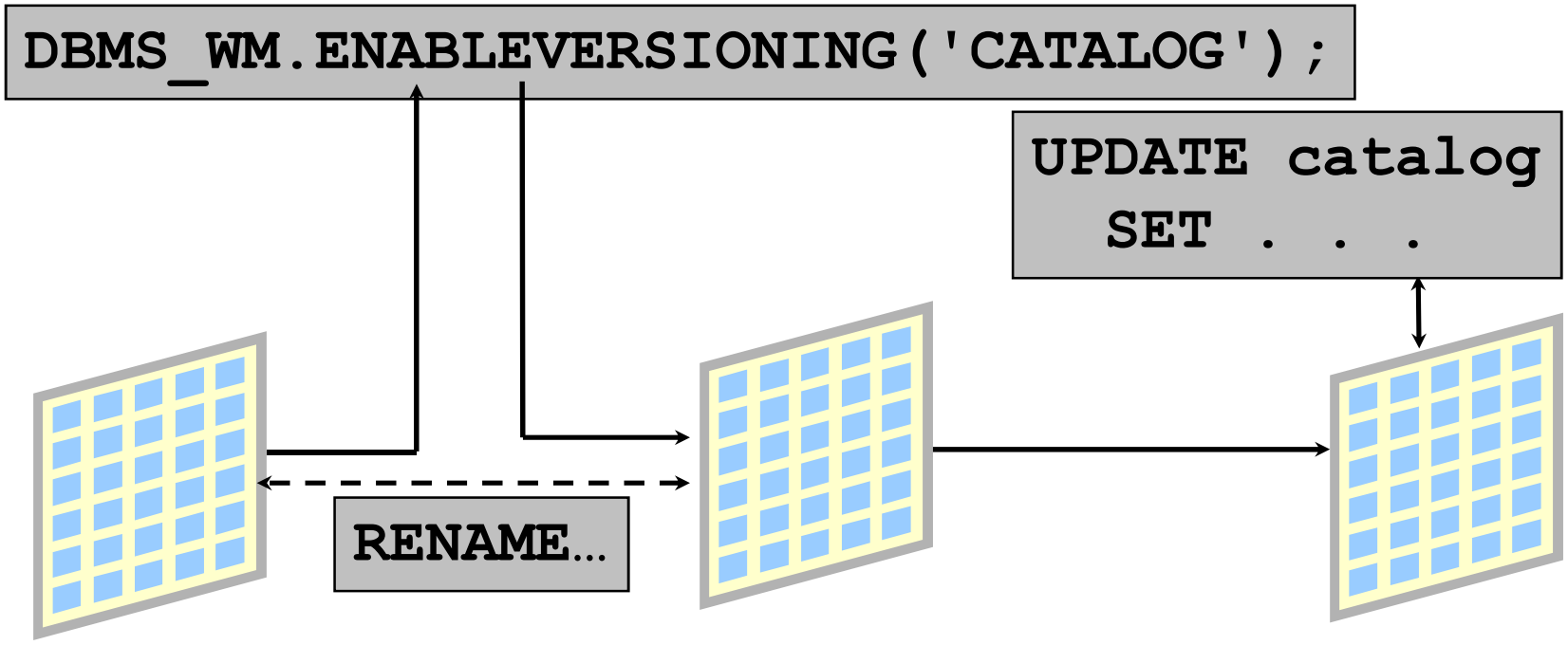
Faster

- 10x efficiency increase - City of Virginia Beach

Better

- Plan scenarios and isolate of groups of changes to data about the grid - Powel Utility customers
- Disconnected field editing - Bentley Utility customers
- Keep historical data - German Lignite Mining Co.

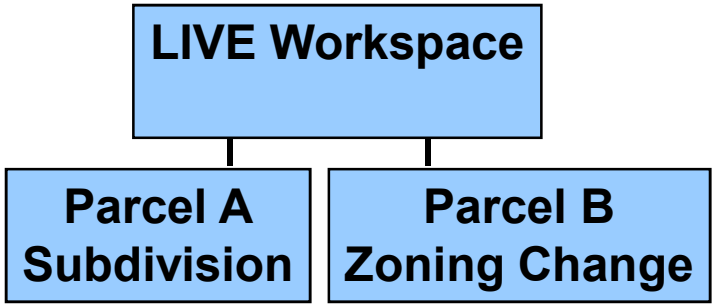
Version-Enabling a Table



- All row versions stored in the _LT table
- All access to table is through Workspace Manager

What is a Workspace?

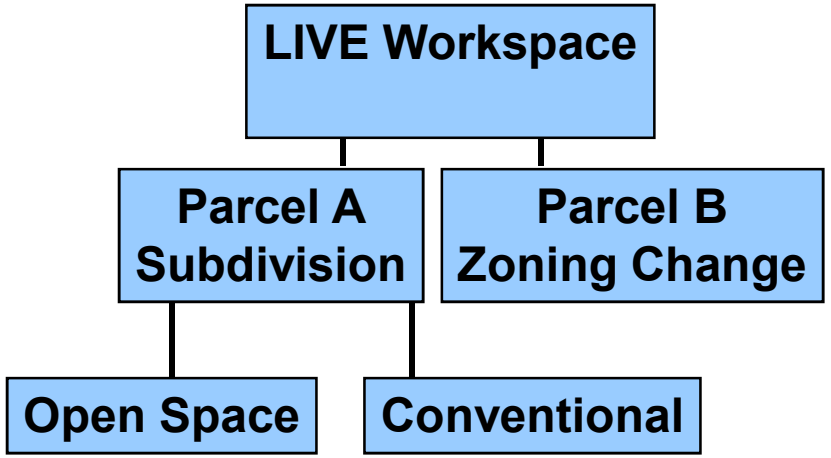
- A WS is a logical view (not physical storage) that can be shared by users
- A WS isolates one or more changes (row versions) from one or more version-enabled tables



- Workspace hierarchies can be of any depth and width
- The LIVE WS is automatically created at installation time
- By default, a workspace is created as a child of LIVE
- Changes made in a workspace by conventional short txn.

Workspace Facts

- GotoWorkspace procedure sets session context to a WS
- The current version is the one that is queried and changed



- Changes are only accessible in the child workspace until merged with the parent workspace
- Changes in the parent can be refreshed to the child
- A workspace is a transactionally consistent view of data.
- No changes to application SQL are required
- Database optimizer hints are supported
- Workspace privileges, access modes & locks provided

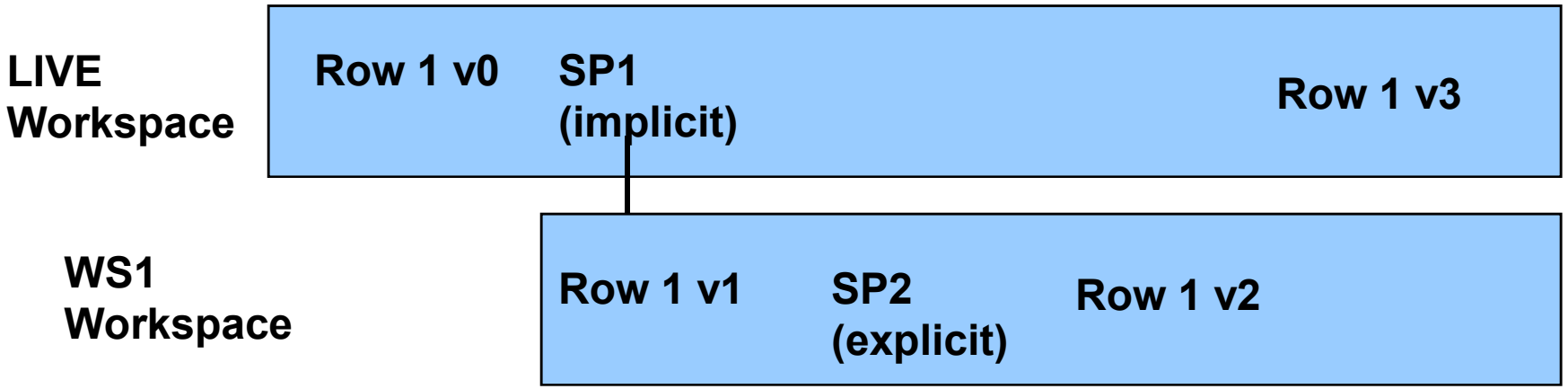
What is a Savepoint?

- A savepoint causes a new version to be created
- A savepoint groups changes within a workspace
- Two types of savepoints: Explicit and Implicit
  - An Explicit savepoint is created and named by a user
  - An Implicit savepoint is created automatically in the parent WS when a child WS is created
- A savepoint is usually created to mark a unit of work
- Subsequent changes are made to the current version until another savepoint is created
- Changes made in a WS can be rolled back to a savepoint
- GotoSavepoint procedure provided

When is a New Row Version Created?

PARCELS table:

- **Row 1 (V0) –present when (or inserted after) PARCELS table is versioned**
- **Row 1 (V1) – 1st update in WS1 (since implicit SP1 was created)**
- **Row 1 (V2) –1st update in WS1 after explicit SP2 is created**
- **Row 1 (V3) –1st update in LIVE after WS1 / SP1 created (it creates a conflict)**



Workspace Manager Integration

- Supports Oracle Spatial
- Supports all datatypes (including nested tables)
- DDL operations on version-enabled tables
- Constraints (Referential Integrity, Unique, Check)
- Triggers
- Import / Export (database and workspace)
- SQL*Loader bulk loading
- Replication
- VPDs & Label Security
- Materialized Views (full refresh)
- Partitioning – Add, Merge, Split
- Manage via Enterprise Manager & metadata views

DDL Operations on Versioned Tables

- Support details in Developer's Guide, Section 1.8
- BeginDDL procedure
  creates an empty temporary table <*table-name*>_LTS
- CommitDDL or RollbackDDL procedure
- Support for table, column, index, trigger, referential & unique integrity constraint, and privilege-related DDL
- Index support: normal, bitmap, function-based normal, function-based bitmap, and domain

Constraints and Triggers

- **Referential Integrity Support**
  - If parent table is version-enabled, child table must be also
  - If child table is version-enabled, parent table need not be
  - Multilevel referential integrity constraints are permitted on version-enabled tables
- Tables with unique constraints can be versioned
  - UNIQUE constraint on a single column or multiple columns
  - Unique index on a single column or multiple columns
  - Functional unique index on the table
- SET NULL constraints not supported.
  - Converted to the RESTRICT option

DataPump Import/Export Considerations

- Full database import and export supported
- Workspace-level import and export
  - Call the Export procedure for each versioned table
  - Import the staging table
  - call the Import procedure for each versioned table
  - Specify workspace where the data resided on the source database and the workspace into which the data should be stored.

Bulk Loading with SQL*Loader

- Direct-path and conventional-path bulk supported
- Load into latest version of any workspace or into the root version (version 0) of LIVE

1. GetBulkLoadVersion fetches reserved version
2. BeginBulkLoading prepares table for bulk loading
3. Run SQL*Loader – change control file to control file, to specify the <table_name>_LT name version number fetched in step 1.
4. CommitBulkLoading or the RollbackBulkLoading

Multi-Master Replication

- Replicate all workspace-related:
  - entities - workspaces and savepoints
  - operations – e.g., CreateWorkspace and MergeWorkspace
  - DML and DDL operations on version-enabled tables.
- Only the master definition site can perform workspace, DML & DDL operations on versioned tables.
- Read operations on versioned tables allowed on all sites in the replication environment.

Advanced Security Support

- Virtual Private Database
  - Row-level security policies not enforced during workspace operations
  - Row-level security policies must be defined on a versioned table and _LOCK, _CONF, _DIFF, and _HIST tables.
- Label Security
- apply_table_policy, remove_table_policy, enable_table_policy, and disable_table_policy

Materialized Views Support

- Materialized view can be created a on a versioned table
- Complete refresh method (REFRESH COMPLETE) must be used
- Cannot version-enable a materialized view or the base table of a materialized view.

Partitioning Support

- Add, merge, and split table partitions in a version-enabled table
- Use AlterVersionedTable procedure, alter_option parameter

Workspace Metadata Views

Read-only views monitor all aspects of the Workspace Manager Environment:

- Version-enabled tables:
  - Conflicts, Differences, Locks, History & Multiworkspace
- Workspaces
- Savepoints
- Users
- Privileges
- Locks
- Conflicts

Code Sample

```
--Version enable the PERSONNEL table with history and
 timestamp all changes
  DBMS_WM.EnableVersioning('PERSONNEL',
   Hist=>'VIEW_WO_OVERWRITE');
-- Create a workspace called PERSONNEL_UPDATES
  dbms_wm.createWorkspace('PERSONNEL_UPDATES'
   );
-- Go to workspace PERSONNEL_UPDATES and update
  dbms_wm.gotoWorkspace('PERSONNEL_UPDATES');
   update PERSONNEL....
```

Code Sample (Continued)

```
-- Create a savepoint called POTENTIAL_CHANGES in the
   PERSONNEL_UPDATES workspace & make more changes
   dbms_wm.CreateSavepoint('PERSONNEL_UPDATES',
     'POTENTIAL_CHANGES');
     update PERSONNEL....
-- Undo the last set of changes
   dbms_wm.RollbackToSP('PERSONNEL_UPDATES','POTENTIAL
     _CHANGES');
```

Code Sample (Continued)

```
-- Merge changes into LIVE (production) Workspace and
  remove the workspace PERSONNEL_UPDATES
dbms_wm.gotoWorkspace('LIVE');

dbms_wm.MergeWorkspace('PERSONNEL_UPDATES',
  remove_workspace => true);

-- Disable versioning on the PERSONNEL table
   dbms_wm.DisableVersioning('PERSONNEL');
```

Workspace Manager Locks

- Pessimistic mode eliminates row conflicts between a parent and child workspace by locking changed rows
- Workspace Manager locks are used in conjunction with conventional Oracle Database short transactions locks
- Locking can be set for a workspace, session or row.
- This example sets shared workspace locking on workspace `WS1` and allows a session to override the locking mode

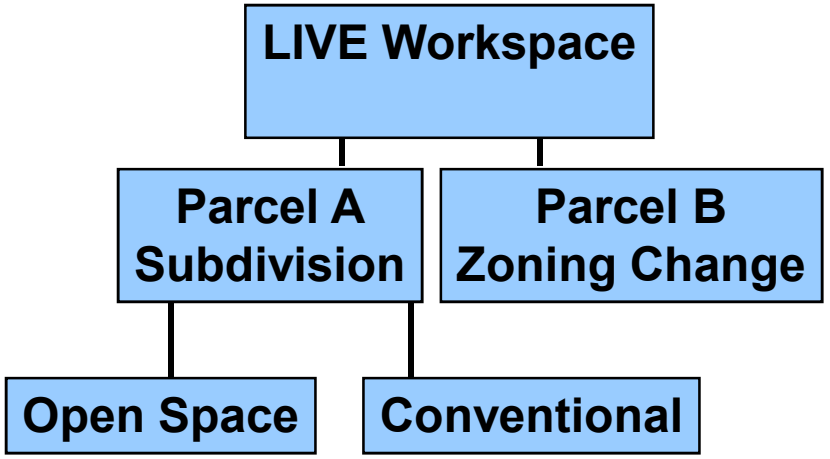
```
DBMS_WM.SETWORKSPACELOCKMODEON
   ('WS1', 'S', TRUE);
```

Workspace Lock Modes

- Shared (S): Any user in the workspace can modify a row changed in the workspace
- Exclusive (E): Only the user that set the lock can modify the row in the workspace
- Some exclusive locks prevent access but not conflicts
  - Version Exclusive (VE): Only the user that set the lock can modify the row in any workspace
  - Workspace Exclusive (WE): Only the user that set the lock can modify the row in the workspace in which the lock was set. Other users can modify the row in other workspaces

Resolving Workspace Conflicts

- Optimistic mode increases concurrency by allowing conflicts
- A conflict is created between a child and parent if a row is modified in both after the common base version
- Unit of conflict detection is a row
  - If different columns are updated in parent and child workspaces a conflict still exists (false conflict)



- Conflicts must be resolved before merge or refresh
- Conflicts automatically detected when a Merge or Refresh is attempted or when `SetConflictWorkspace` is called
- Workflow to resolve conflicts:, `BeginResolve`, `ResolveConflicts`, `CommitResolve`, `RollbackResolve`

ResolveConflicts cont'd

- A save point is created in the child workspace
- `ResolveConflicts Keep` parameter
  - PARENT: The row from the parent is copied into the child workspace
  - CHILD: No row is copied into the parent until the workspace or table is merged
  - BASE: The base row is copied to the child workspace. No copy is made in the parent workspace until a merge operation.
- After resolution, both rows need to be modified in order for a conflict to reoccur.
  - Except when parent row is modified after resolution in favor of child

History Options

- Timestamps row changes with transaction time
- A table can be version enabled with one of the following history options:
  - *VIEW_WO_OVERWRITE* ⇒ each update is preserved
  - *VIEW_W_OVERWRITE* ⇒ subsequent updates in the same version overwrite previous updates
- *_HIST view* shows all versions in the _LT table
- *dbms_wm.GotoDate* to view data at a given instant
- Rollback, Compress and Remove operations delete the history of changes made in the workspace
- Compress Workspace[Tree] can optionally preserve history (in this case it only deletes savepoints)

Valid Time (Effective Dating)

- Uniquely specifies a valid date and time range for a row version.
- Sets the valid time (VT) range for your session before a query or DML
- Insert, update or delete a row in one of two ways:
  - Default to the session VT range
  - Specify a VT range (past, present and/or future)
- Use Valid Time comparison operators to refine queries within the valid time of the session context.
- Specify the Valid Time option for a table when or after a table is version-enabled

Sequenced & Nonsequenced Updates

Sequenced: Used to update user data in a row

- If a VT range is not specified in an Update or Delete the session VT range is used to split the original row. For example:
  - VT range of the original row = T1 - T10
  - VT range of the session = T4 – T6
  - Update: The original row is split into three rows with the following VT ranges:

    T1 - T4, T4 - T6, T6 – T10
  - Delete: The original row is split into two rows with the following VT ranges: T1 - T4, T6 – T10 (If the session VT is not set the entire row is deleted)

Nonsequenced: used to update valid time range in a row

- If a VT range is specified in an update:
  - No additional row is created
  - Workspace Manager will return an error if the VT range overlaps that of an existing row

Valid Time Mode and Filter

Mode: Use to correct errors in the data

- OFF: Disables sequenced & nonsequenced operations
  - Use to correct errors in a row w/o affecting the row's VT range
- ON: Enables sequenced & nonsequenced operations
  - Use to update or delete a row in a specific VT range

Filter: Use to return a single row for a session VT range

- OFF: Can return multiple rows
  - Session VT range = T1 – T10
  - Two versions of a row: T1 – T5, T5 – T10 (query returns both)
- ON: Specifies a point in time within the session VT range
  - A query only returns the version valid for that point in time

Valid Time Operators

Use Valid Time operators as query filters

Relationship operators

- WM_OVERLAPS – do two periods overlap
- WM_CONTAINS - does one period contain another
- WM_MEETS - is the end of one period the start of another
- WM_EQUALS - are the start / end of two periods the same
- WM_LESSTHAN - is the end of one period earlier than the start of another
- WM_GREATERTHAN - is the start of one period later than end of another

Set Operators

- WM_INTERSECTION - returns the time range common to two periods.
- WM_LDIFF - returns the difference between the two periods that is earlier in time
- WM_RDIFF returns the difference between the two periods that is later in time

Versioning Oracle Spatial topologies

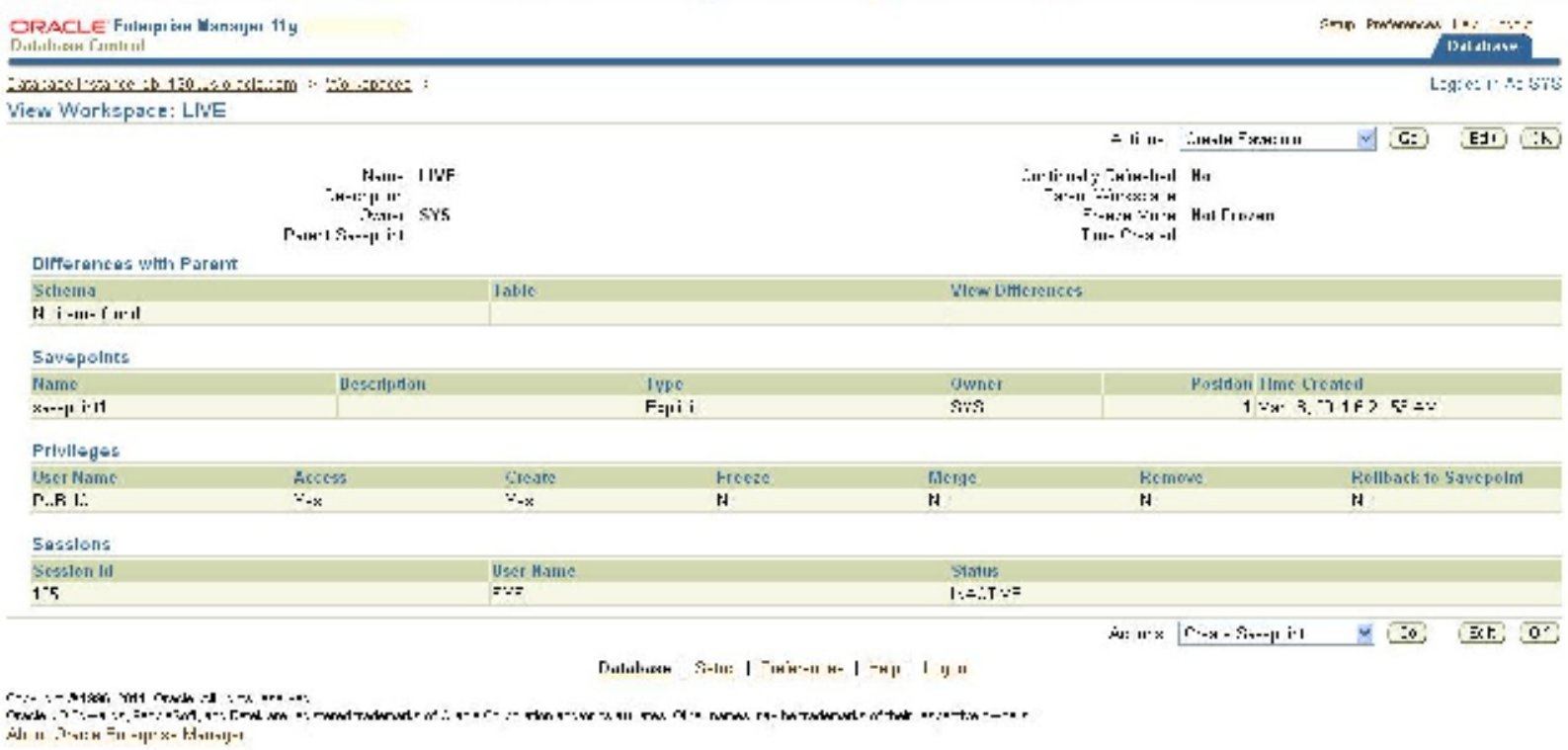
- Oracle Spatial topologies can be versioned by specifying the the topology name as the `table_name` in the `EnableVersioning` procedure

```
EXECUTE DBMS_WM.EnableVersioning(table_name
=> 'xyz_topo', isTopology => TRUE);
```

- `ADD_TOPO_GEOMETRY_LAYER` adds a topology geometry layer in a version-enabled topology
- The following example adds a topology geometry layer to the `CITY_DATA` topology. The topology geometry layer consists of polygon geometries in the `FEATURE` column of the `LAND_PARCELS` table.

```
EXECUTE
DBMS_WM.ADD_TOPO_GEOMETRY_LAYER('CITY_DATA',
'LAND_PARCELS', 'FEATURE','POLYGON');
```

Database Control for Workspace Mgr.



Workspace Manager Case Studies

Intergraph GeoMedia Transaction Mgr.

- Provides Long Transaction support for GeoMedia GIS
- Workspaces to isolate groups of changes
- Locking can be optimistic or just-in-time pessimistic
- History management
- Valid Time (effective dating) used by cities and DOT

Two Case Studies

- City of Edmonton
- City of Virginia Beach

Case Study: City of Edmonton

Spatial Land Inventory Management System provides a single mgt. environment for city's land based assets

- Application platform:
  - Oracle Workspace Manager and Oracle Locator
  - Intergraph GeoMedia Pro, GeoMedia Transaction Manager
- Data feeds:
  - Land registry and surveys
  - Utilities and phone co.
  - Tax assessments
  - Dept. of Public Works

SLIM Data

- Legal Survey parcels
- Assessment parcels
- Title parcels
- Civic holdings
- Parkland Assets
- Zoning and Land Use
- Underground utilities
- Street Lights and Trolley
- Addresses
- Single Line Street Network
- Sidewalk structure/condition
- Road structure/condition
- Buildings, entryways
- Demographic data

Administrative areas such as:

- Community leagues
- Neighbourhoods
- Wards
- Voting subdivisions
- Business Revitalization Zones
- Residential parking program
- Neighbourhood structure plans
- Area structure plans
- Inspection areas
- Traffic districts / zones
- Major commercial corridors

THE CITY OF Edmonton

City of Edmonton (continued)

- Users:
  - 1000's of end users - city officials, departments, mortgage lenders, citizens
  - 150 professionals - Engineers, planners, cartographers
  - 50 data entry personnel
- Client access:
  - Internet, mobile and thick client tools
- Database: 30gb and growing

Number of Current / Historical Records

Land Parcel related tables

- Title-Assessment-Civic Properties: 197,297/ 891,274
- Title related information: 928,182 / 1,251,509
- Legal Descriptions (lot-block-plan): 817,027 / 1,692,009

Address related tables

- Addresses: 395,243 / 1,175,994
- Buildings-floors-entryways-suites: 908,012 / 1,066,799

Assessment: 182,943 / 205,311

Assets

- Street Lights: 49,460 Poles, 89,641 Luminaires, 46,948 HW
- Future data: More Parkland Assets, Bus Stops, Scanned Roadways As-Built images, Traffic Signals, Street Markings, Parking Meters and more

THE CITY OF Edmonton

ORACLE

City of Edmonton's Problem

- Data duplication was common
- Data formats were inconsistent
- Data quality was inconsistent
- Data currency was often a problem
- Some required data did not exist
- Historical data was limited

Requirements

- Single, centralized data store
- Store data in three states:
  - Proposed
  - Current
  - Historical
- Maintain audit trail for data maintainers
- Maintain historical and proposed states for business users

Workspace Manager in Production

- Data maintainers
  - Create workspaces to isolate changes
  - Merge workspaces when changes are completed and approved
- 112 version enabled tables
- Referential constraints and triggers used heavily
- Average 75 workspaces in use
- Average rows merged at a time
  - Registries data load - 13 tables - 8800 rows
  - Addressing - 5 tables - 80 rows
  - Parcel Maintainers - 2 tables - 140 rows

Results

- Integrated, centralized, high quality data
  - Replaced 49 disparate land apps., 166 databases
  - Single point of update and management
  - Citywide sharing of consistent data with controlled access
- Concurrency and historical perspective
  - Concurrency: end users access current data while data entry and updates are isolated in workspaces
  - History: all changes retained,“goto date” capability

City of Virginia Beach Using GeoMedia

Data Profile

- 450,000 Population
- 150,000 Parcels
- 23,000 Condos
- 30,000 Apartment Units
- 4 Million Elements
- 7200 GIS Files
- 12,000 DGN files
- 2200 Tax Maps
- 19,879 Street Segments
- Staff of 14

Project Requirements

- Integrate parcel data & workflow for Assessment & Mapping
- Minimize duplication
- Create more efficient processes
- Improve performance & output
- Centralize situs address mgt.
- Add spatial controls, validation, analysis and spatial attribution
- More advanced data modeling
- Role-based, concurrent access

City of Virginia Beach Project Results

The Intergraph GeoMedia Solution

- In production over year, City staff are very satisfied
- The city has an accurate picture of ground conditions
- Accurate joining of Assessment and Mapping Data
- 10x efficiency improvement
- 4x more work in the same time w/ half the resources

Oracle Spatial & Workspace Manager provided:

- Time savings w/ increased data accuracy, security and synchronization
- Single point of control over spatial data editing, attribution and validation
- Advanced spatial relationships and analysis
- Long transactions w/ isolation, effective dating & history

Powel ASA

Business critical decision support systems for electricity generators, traders and utilities

- Market leader in the Nordic region
- Over 1,000 public and private customers in Europe and U.S
- NetBas is a GIS-based grid information system used for planning, design, operation and maintenance of utility assets, and engineering analysis
- Workspace Manager enables scenario planning and isolation of groups of changes to data about the grid

Powel ASA - A Major Power Partner

- Spatial data specifying geographical position of overhead lines, transformers, etc. and attributes
- Numbers of users: Up to 800 concurrent users
- Database size: ~ 30 GB.
- 14000 workspaces, max. 2-3 levels deep, usually 1
- 430 versioned tables, Largest table: 10 mill. rows in _LT table, 6300 distinct versions
- Uses Optimistic locking and continually refreshed workspaces with conflict resolution via application GUI

HMO – Operational Data Store

A Health Maintenance Operator (HMO) built an ODS to:

- Support key operational business processes
- Aggregate transaction processing data from multiple legacy applications
- Provide subject-oriented, integrated, near realtime, detailed data for a number of financial applications and reports

Requirements

- Continuous access and updates to current data
- Historical snapshots: weekly, monthly
  - 500GB Oracle Database
  - Hardware has 1TB storage limit
- Load 60 MB (120,000 transactions) per hour
- No changes to application SQL or queries

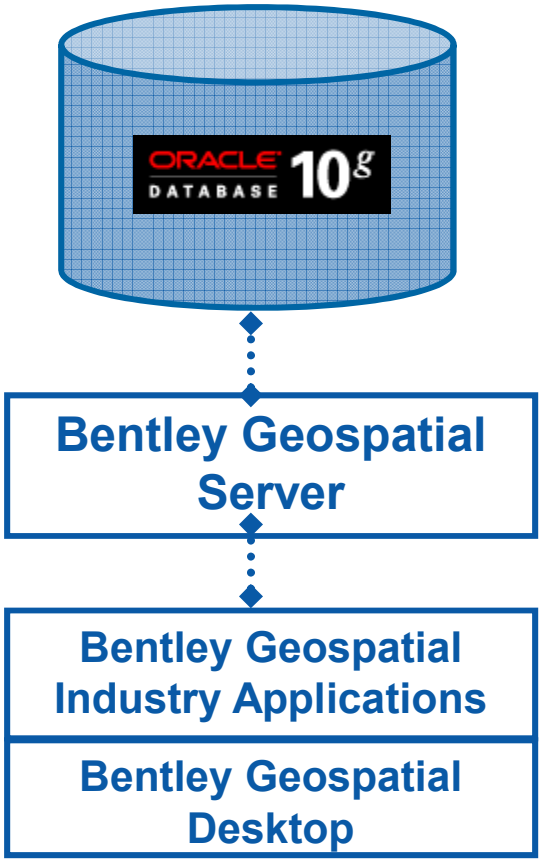
Two Alternatives for Snapshots

	Workspace Manager	Multiple Instances
Hardware and software	Same HW & SW	More HW & SW
Data Access	Continuous	None during snapshot refresh
Operational decision making	Current data	Stale data
Data Mgt.	No Change	Multiple copies
Add another snapshot	Easy	Hard

Solution – Workspace Manager

- Continuously update and access data in LIVE
- Workspaces provide historical views:
  - Weekly = end of the previous week
  - Monthly = end of the previous month
- Workspace Refresh - updates the workspace with the latest data
- Workspace Compress - removes old versions from LIVE

Bentley Geospatial Server



- Live & Disconnected viewing / editing
- Oracle Workspace Manager
  - Pessimistic/ optimistic long txn. locking
  - Multi-user editing w/ Conflict resolution
  - History & Valid Time
- Geometry and topology support
- Coordinate systems
- Text, feature-linked text
- Coded domains
- Loading tools (DGN2SDO)
- Fully customizable – web services

Bentley Customers Examples

Using Oracle Spatial, Workspace Manager

- Large Utility Company, Torino, Italy
  - Work order management for electric, district heating, gas and water networks
  - Disconnected spatial editing w/ pessimistic locking

- Central German Lignite Mining Company (MIBRAG)
  - Mine data mgt: surveying, updating, mapping, reporting
  - Management of historical data w/ Valid Time

In Closing

- Workspace Manager provides long transactions for:
  - Long duration projects
  - What-if scenarios
  - History
- It is tightly integrated with
  - Oracle Database
  - Oracle Spatial
- It is a defacto standard for long transactions with the most comprehensive set of features and database integration

For More Information

search.oracle.com

Workspace Manager 

OR

http://www.oracle.com/technology/products/database/workspace_manager/


Q&A
QUESTIONS
ANSWERS

Addendum

Workspace Manager Best Practices

Guidelines for Version-Enabled Tables

- Version-enabled table must have a primary key
- A table can be version-enabled by the table owner or by a user with `WM_ADMIN_ROLE`
- Tables owned by `SYS` cannot be version-enabled
- Referential integrity constraints are supported on version-enabled tables
- Triggers are supported on version-enabled tables with some restrictions
- The history option allows the end-user to track all changes made to a version-enabled table

Compress Workspaces for Performance

- The number of rows, and versions and historical copies per row in the table has an affect on performance
- Compressing a workspace removes intermediate versions and history that were created in it
- Compress when the explicit savepoints and versions in the affected workspaces are no longer needed
- Benefits:
  - Savepoint names can be reused
  - Performance is improved
  - Less storage is used

The example below compresses LIVE and any child WS

```
DBMS_WM.CompressWorkspaceTree('LIVE');
```

Refresh vs Merge Workspace

- All things being equal `RefreshWorkspace` is faster than `MergeWorkspace`
  - `RefreshWorkspace` simply updates the child workspace metadata in most cases so a user in the child WS can see the current version in the parent WS

    parentVersion(child WS) = currentVersion(parent WS)

  - `MergeWorkspce` copies changes made in the child WS into the current version of parent WS
- An application can take advantage of this by:
  - Isolating changes in the parent WS
  - Using the child WS for production data
  - Refreshing the child WS

Naming Length Restrictions

- Table name: 25 characters
- Column name: 28 characters
- Index name: 30 characters (26 for *beginDDL*)
- Trigger name: 27 characters
- Constraint name: 30 characters (26 for *beginDDL*)

Note: Index and Constraint names can be modified with the *alterVersionedTable* API without using *commitDDL*

DDL on a Versioned Table

- Skeleton table <*table name*>*_LTS* is created by *beginDDL*
- Skeleton table contains only the user-defined columns (no metadata columns)
- User defined triggers, columns & RICs on the skeleton table have the same names as the corresponding ones on the versioned table
- Other objects are created with _LTS extension
- *commitDDL* compares the structure of the skeleton table with the structure of the versioned table, performs the required changes on the versioned table & deletes the skeleton

all_wm_vt_errors

- *EnableVersioning, DisableVersioning & CommitDDL* store the current PL/SQL statement being executed persistently
- When an error occurs, the *all_wm_vt_errors view* can be queried to find the failed statement

all_wm_vt_errors cont'd

- To view a list of all statements that will be executed for EnableVersioning / DisableVersioning / CommitDDL:

```
SELECT sql_str
FROM TABLE (SELECT undo_code
            FROM wmsys.wm$versioned_tables
            WHERE owner = '<owner>' AND
            table_name = '<table_name>' )
WHERE  index_type < 1000
ORDER BY  index_type asc, index_field asc;
```

Ignore_last_error

- Can be used for *disableVersioning* and *commitDDL*
- If the *all_wm_vt_errors* view contains no rows for a particular table, then ignore_last_error will not help in solving the issue
- Some operations can always be skipped. For example, the dropping of a view during disableversioning.
- Some operations should not be skipped. For example, if the compilation of the instead of triggers is failing during *commitDDL*.

Workspace Metadata Views

Read-only views to monitor all aspects of the Workspace Manager Environment:

- Version-enabled tables:
  - Conflicts, Differences, Locks, History & Multiworkspace
- Workspaces
- Savepoints
- Users
- Privileges
- Locks
- Conflicts

<table_name_XXX> Views

- **_CONF :** (_BPKC, _PKC) : Shows the conflicts between a workspace and its immediate parent
- **_DIFF :** (_PKD, _PKDB, _PKDC) : Shows the differences between 2 arbitrary workspaces/savepoints.
- **_HIST :** Displays a history of changes for the version enabled table
- **_BASE :** Same as the top view with some extra metadata columns included (version, nextver, etc.) created for performance reasons

<table_name_XXX> Views cont'd

- **_LOCK :** Shows all rows for which a version lock is being held. This view is workspace dependent.
- **_MW :** Can be used to view changes in multiple workspaces at once. The workspaces are selected with the setMultiWorkspaces procedure.
- **_CONS :** Internal view for detecting unique constraint violations from inside the instead of triggers and during workspace operations like merge, refresh, etc.