GLOC
Great Lakes Oracle Conference

APEX and SmartDB Step-by-Step

MAY 16 & 17, 2018

CLEVELAND PUBLIC AUDITORIUM, CLEVELAND, OHIO

WWW.NEOOUG.ORG/GLOC

Anton Nielsen

- Vice President – Insum
- anielsen@insum.ca

- Formerly
  - President, C2 Consulting
  - Technical Directory, Oracle
  - Consultant, Coopers and Lybrand (now PWC)
  - Captain/Scientist, US Air Force

Agenda

- What is the Thick Data Paradigm?
- Goals Related to APEX in this Paradigm
- A Recipe for Implementing the Thick Data Paradigm with APEX
- Drawbacks
- Discussion

Overview

- PL/SQL
  - Bryn Llewellyn discussed merits of PL/SQL and the “thick database” approach
    - Later rebranded as SmartDB
  - Toon Koppelaars set up focus on performance gains

- Application to APEX
  - My focus is applying this concept to Oracle APEX
  - Consider the apex.oracle.com blueprints

Great Lakes Oracle Conference

Goals

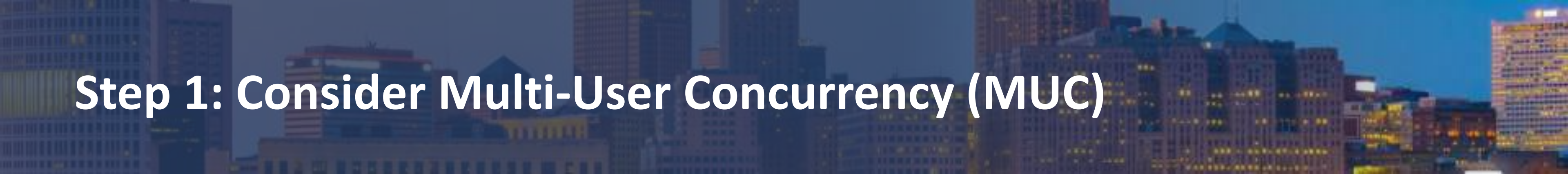
- Correctness (and Security)
- Performance
- Ease of Development
- Ease of Maintenance
- Instantiating the Thick Database Paradigm to Ensure these Goals are Met

Recipe - Ingredients

- Thought and Planning
- At Least Two Schemas
- One Workspace (with its own schema)
- Tables
- Simple Views or Editioning Views
- Logic in PL/SQL packages
- Views with Instead Of Triggers*
- Patience

*We will have a lot of discussion on this.

GLOC
Great Lakes Oracle Conference



Step 1: Consider Multi-User Concurrency (MUC)

- Optimistic vs Pessimistic Locking
- Will front end developers be responsible for managing (MUC)?
- How will you ensure correctness?

- For our example we will use an Object Version Number (OVN) column

Step 2: Store the Data

- Create a schema to hold data: GLOC_DATA
- Create tables, indexes, sequences, triggers*
- Name tables TABLE_NAME_RT
  - DEPARTMENT_RT
  - EMPLOYEES_RT
- Front End developers will never interact with the _RT tables
- Back End developers will never interact with the _RT tables
- This schema is the domain of a good DBA

*If you like triggers. These may belong on the next slide.

GLOC
Great Lakes Oracle Conference

Step 3: Consider Edition Based Redefinition (EBR)

- If using EBR, for each table create an editioning view.
- If not using EBR, for each table create a standard view.

- For my example I will create standard views.

- Discussion — Should EBR be in the same schema as the data?

Step 4: Create Schema for Logic

- Create a schema for database (pl/sql) developers: GLOC_LOGIC
- Grant select on VIEWS (EVs) with grant option* to GLOC_LOGIC
- Grant insert, update, delete on VIEWS (EVs) to GLOC_LOGIC
- Grant select on sequences GLOC_LOGIC

- Create private synonyms for DATA objects (optional)

- *The grant option will be used to allow front end developers access to data
- Note: Take care to use CREATE OR REPLACE on views going forward. Do NOT drop and recreate as synonyms become invalid.

Step 5: Create Logic

- Discussion: Table APIs
- Beyond TAPIs: Consider “Give everyone in IT a 5% raise”
- Consider triggers vs API logic (see insert code)

```
  l_id  number := p_id;
  begin
      if l_id is null then
        l_id := GLOC_seq.nextval();
      end if;
```

- Logic should be created in the LOGIC schema

Great Lakes Oracle Conference

Schema Design (so far)

LOGIC
APIs

DATA

Views (CRUD) or EVs
Data Storage Objects (Tables, Indexes)

- Grants to Views/EVs
- Select grants with grant option
- Grants to Sequences

Step 6: Create Workspace (Finally!)

- Create APEX Workspace in another schema: GLOC
- As GLOC_LOGIC
  - grant execute on GLOC_tapi to GLOC;
  - Demonstrate: grant select on employee view;
- Create synonyms as GLOC (optional)
  - synonyms for GLOC_logic APIs;
  - synonyms for GLOC_logic views
- Note: GLOC could be completely unaware of GLOC_DATA, but the APEX builder will show the views as owned by GLOC_DATA

A Word About Security

- Security should be implemented at multiple levels
  - Multiple Schemas
  - Logic defined in packages in the database
  - Access restricted to packages & Views
- SQL Injection, Post Data Tampering, etc.
  - You can not rely on browsers and front end developers to secure your data

GLOC
Great Lakes Oracle Conference

Schema Design

APEX Workspace

Packages Related to UI Only

- CRUD Grants to Select Views*
- Grants to APIs
- *We will replace this with a different approach in a subsequent step (don’t really do this!)

Logic

APIs

- Grants to Views/EVs
- Select grants with grant option
- Grants to Sequences

DATA

Views (CRUD) or EVs

Data Storage Objects (Tables, Indexes)

Step 7: Make the Workspace Work

- At this point, the workspace is functional, but lacks features
- What works
  - Reports
  - Forms Based on APIs owned by GLOC_LOGIC
  - Interactive Grids with Select Only
- What doesn't work
  - Interactive Grids with Insert, Update or Delete
  - Anything that references a ROWID
  - Forms based upon a table or view (automated DML operations)

GLOC
Great Lakes Oracle Conference

Step 7: Make the Workspace Work (ctd)

- As the LOGIC user, Create Views with instead of triggers

```
create view employee_crud as select * from employee
/


create or replace trigger employee_crud_bi_intrg
instead of insert on employee_crud
begin
  GLOC_tapi.ins_employee( p_id  => null
                        , p_emp_name => :NEW.emp_name
                        , p_department_id => :NEW.department_id
                        , p_salary =>   :NEW.salary);

end;
```

GLOC
Great Lakes Oracle Conference

Step 7: Make the Workspace Work (ctd)

- Unfortunately! In order to allow the GLOC user to insert, update and delete into the new CRUD views, with “instead of” triggers, the GLOC_LOGIC user must have grant option.
  - Also, “instead of” triggers disappear when view is recreated
- GLOC_DATA must grant with grant option.
- GLOC_LOGIC grants select, insert, update, delete on _CRUD to GLOC

Schema Design (revised)

APEX Workspace Packages Related to UI Only	• Grants to Views with Instead of Triggers • Grants to APIs
Logic APIs	• Grants to Views/EVs • Select grants with grant option • Grants to Sequences
DATA Views (CRUD) or EVs Data Storage Objects (Tables, Indexes)	

GLOC
Great Lakes Oracle Conference

Recap

- If you are satisfied with APEX features that don’t require Automated DML or updatable Interactive Grids, you can stop at “select”
- If you want additional APEX automation, create views with Instead Of triggers that exercise the logic in your APIs

Unfortunately…

- Some things still don't work
  - Add row in Interactive Grids requires PK to be set in the IG
  - Returning Value in Automated DML
  - ROWIDs
- You can't use the "returning" clause with a view with an Instead Of trigger
- You can't add a ROWID column, named ROWID, to a view

Discussion

- What is the value in separating schemas?
  - Data
  - Editioning Views
  - Logic
  - Presentation / Workspace
- Do views with Instead Of triggers encourage bad practices?
- When are the trade-offs just not worth it?
- When would you create tables just for the presentation layer?
  - Do these require similar levels of “correctness”?