

# **Data Auditing**

# **Concepts and Implementation**

**David Mann**

**Biogen Idec**



# Topics

---

- ✖ Why track changes to your data?
- ✖ Evolution Overview
- ✖ Design Considerations
- ✖ Implementation Options
- ✖ Examples



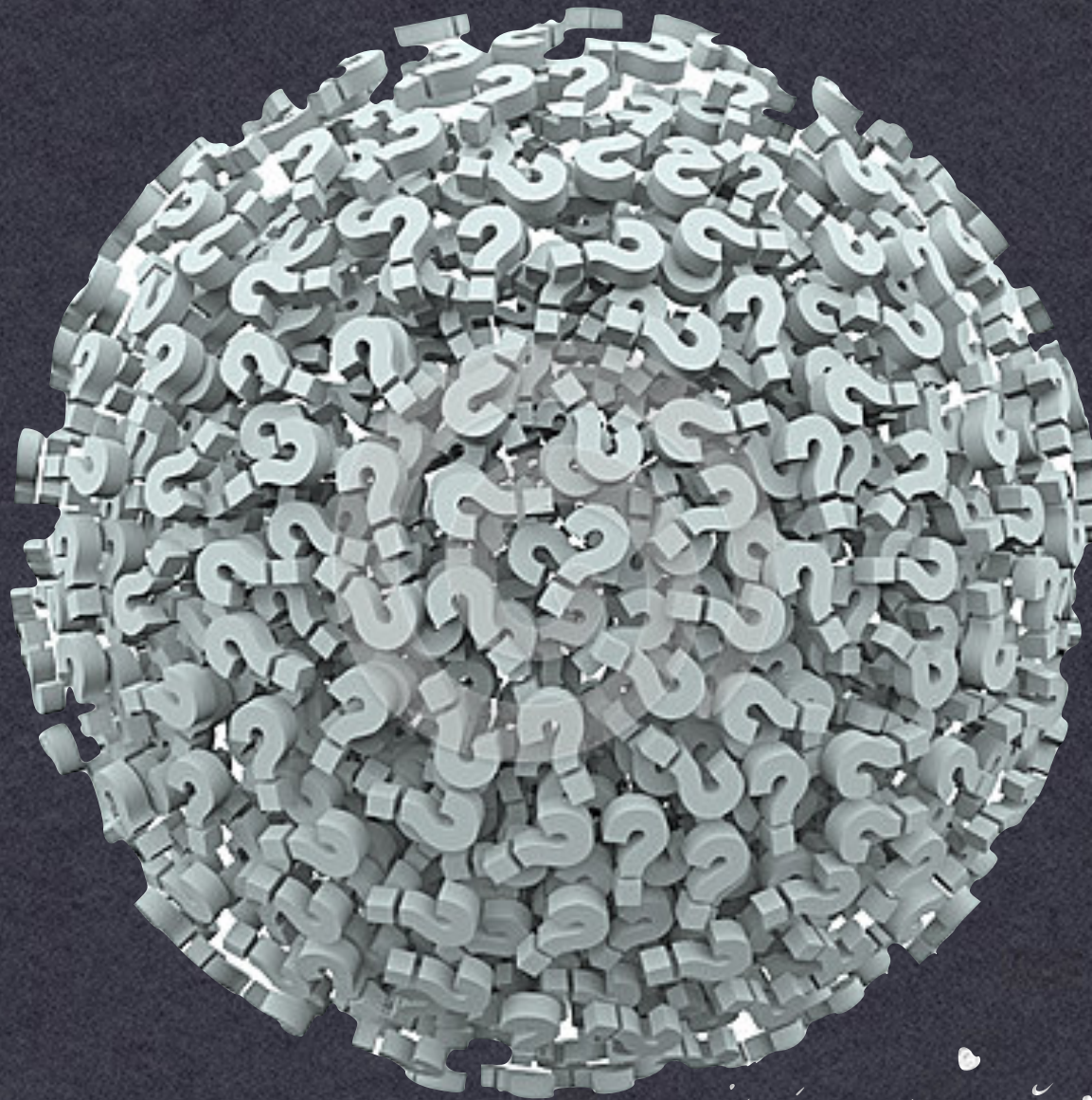
# Bio

- ✦ Lead Oracle DBA at Biogen Idec
- ✦ Developer and DBA Background
- ✦ Special interests - graphing data and developing administration tools
- ✦ I'm a data hoarder

The Biogen Idec logo is located in the bottom right corner. It consists of the words "biogen ideo" in a blue, sans-serif font, with a registered trademark symbol (®) to the right. The text is enclosed in a white rectangular box with a thin black border. The box has a small notch on its top-left corner and is intersected by a thin black crosshair.

**biogen ideo**®





**WHY TRACK DATA CHANGES?**



# Types of Oracle Auditing

- ✦ Security/Administrative Events
  - ✦ Alert/Audit Logs
  - ✦ SYSDBA activities
- ✦ Data Definition Language (DDL)
  - ✦ Structure Changes
- ✦ Data Manipulation Language (DML)
  - ✦ SELECT
  - ✦ Results of INSERTs, UPDATEs DELETEs





# We Will Focus On

---

- ✖ DML updates to data
  - ✖ Who | What | When
  - ✖ Ability to retrieve a previous state
  - ✖ Ability to list versions between states
  - ✖ Bonus: Other information modified at same time



# Why Track Data Changes?

- ✦ Requirement for Problem Domain
  - ✦ Financial
  - ✦ Medical Patient History
- ✦ Operational Support
  - ✦ Investigating, monitoring, and recording changes
- ✦ Internal Auditing Requirements
- ✦ External Auditing Requirements
  - ✦ Regulatory – SOX, HIPAA, etc.
- ✦ Accountability / deterrent
- ✦ Because customers expect it

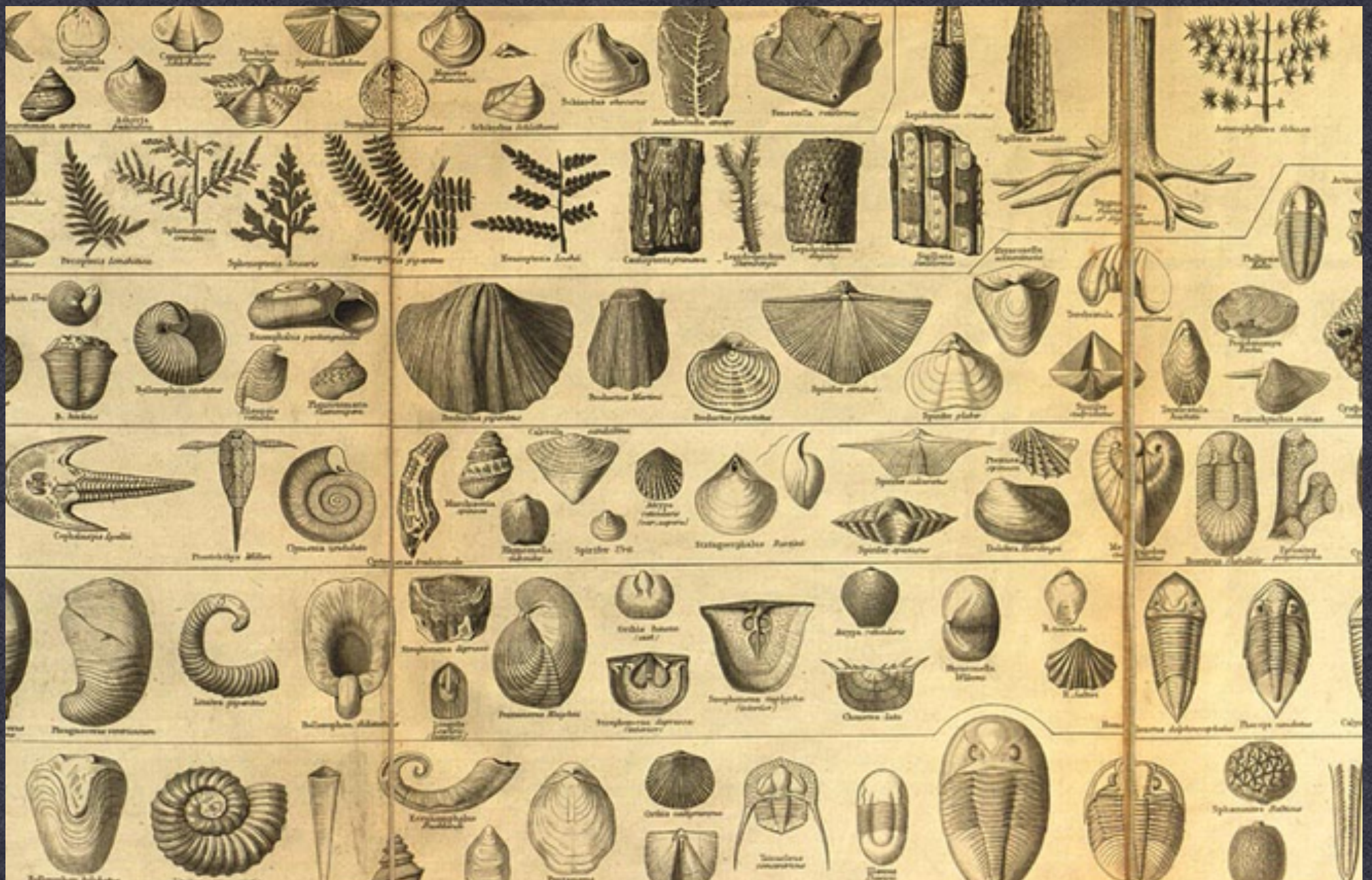


# Use Cases

---

- ✖ We found a data error on a report. We need to see the previous values for a few of the records to confirm they are sane.
- ✖ We are not sure if the user was following update procedures. Can we inspect a user's update history?
- ✖ Are there any updates to the EMP table outside of business hours? What were the updates and who did them?
- ✖ I need to inspect 1..n previous versions of a record because.... just because!





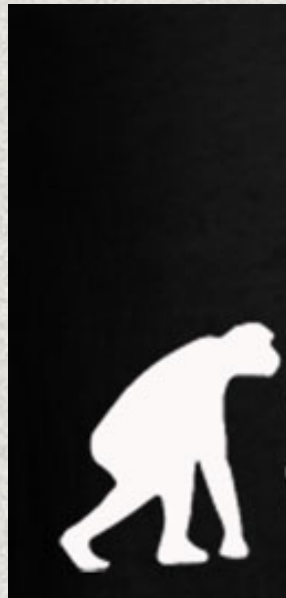
# EVOLUTION OVERVIEW

STAGES OF AUDITING GRIEF



# Evolution of Auditing - 1/4

- ✖ No Context
  - ✖ Typical “Rev 1” deliverable or trivial example schema
  - ✖ Can only hold current values
  - ✖ Physical deletions





# Evolution of Auditing - 2/4

- ✖ Limited Context
  - ✖ A novel solution “Invented Here”
  - ✖ Created User / Last Modified User
  - ✖ Created Date / Last Modified Date
  - ✖ May support logical deletions





# Evolution of Auditing - 3/4

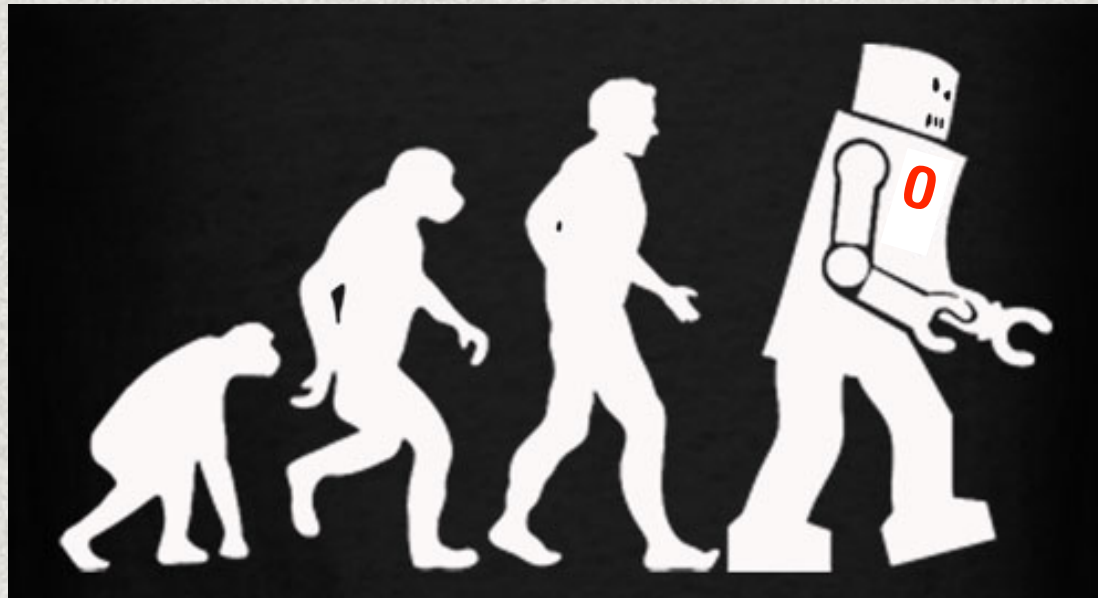
- ✦ Full Context via Informed Design
  - ✦ All changes to the data are recorded - Trans & User
  - ✦ Inputs/Outputs of systems defined
  - ✦ Flirting with Valid Time
  - ✦ Still a lot of manual work involved during development





# Evolution of Auditing - 4/4

- ✦ Full Context via DBMS Tools
  - ✦ All changes to the data are recorded
  - ✦ Transaction Time and Valid Time used where appropriate
  - ✦ Leveraging tools/automation/DBMS features to make things easier on developers and end users
  - ✦ Bonus: Changes can be tied to transactions



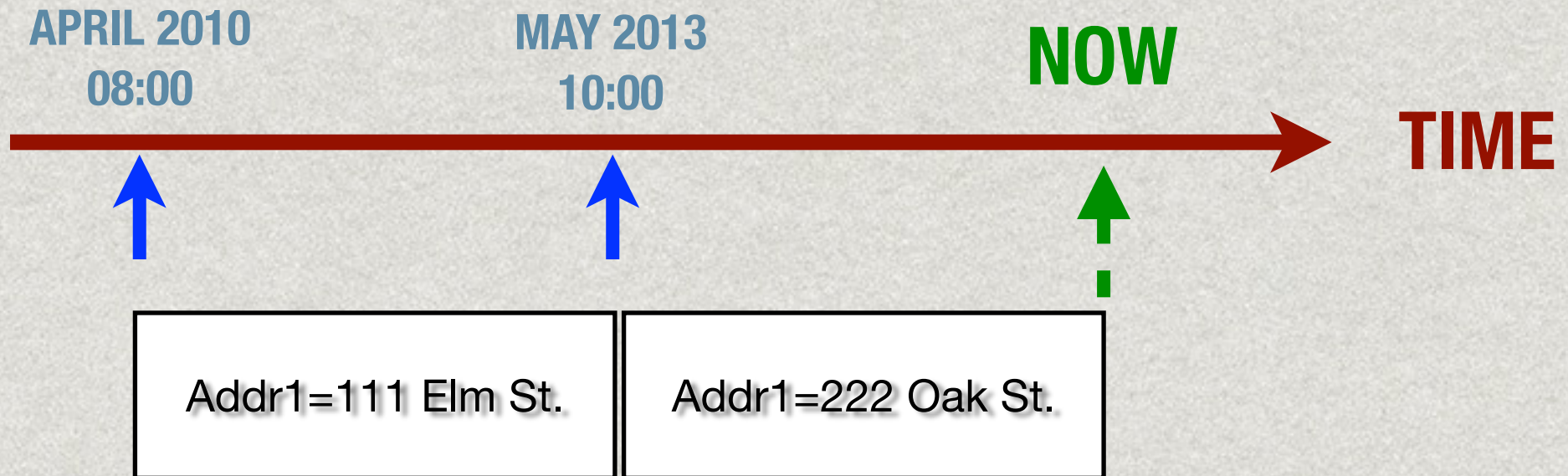




## DESIGN CONSIDERATIONS



# The Past and Present





# What should we record?

---

- ✦ Transaction Time

- ✦ System Time when a fact is stored in the database

- ✦ Identity of the user changing the data

- ✦ New Value

- ✦ Old Value?

- ✦ Old value accessible via previous auditing activity



# Where do we capture changes?

---

- ✦ Application / Business Logic Layer
- ✦ Connection level
- ✦ Database - Internal
  - ✦ Trigger
  - ✦ Oracle Features - Flashback, CDC
- ✦ Database - External
  - ✦ Redo based - Golden Gate, LogMiner



# Storage of Audit Trail Data

---

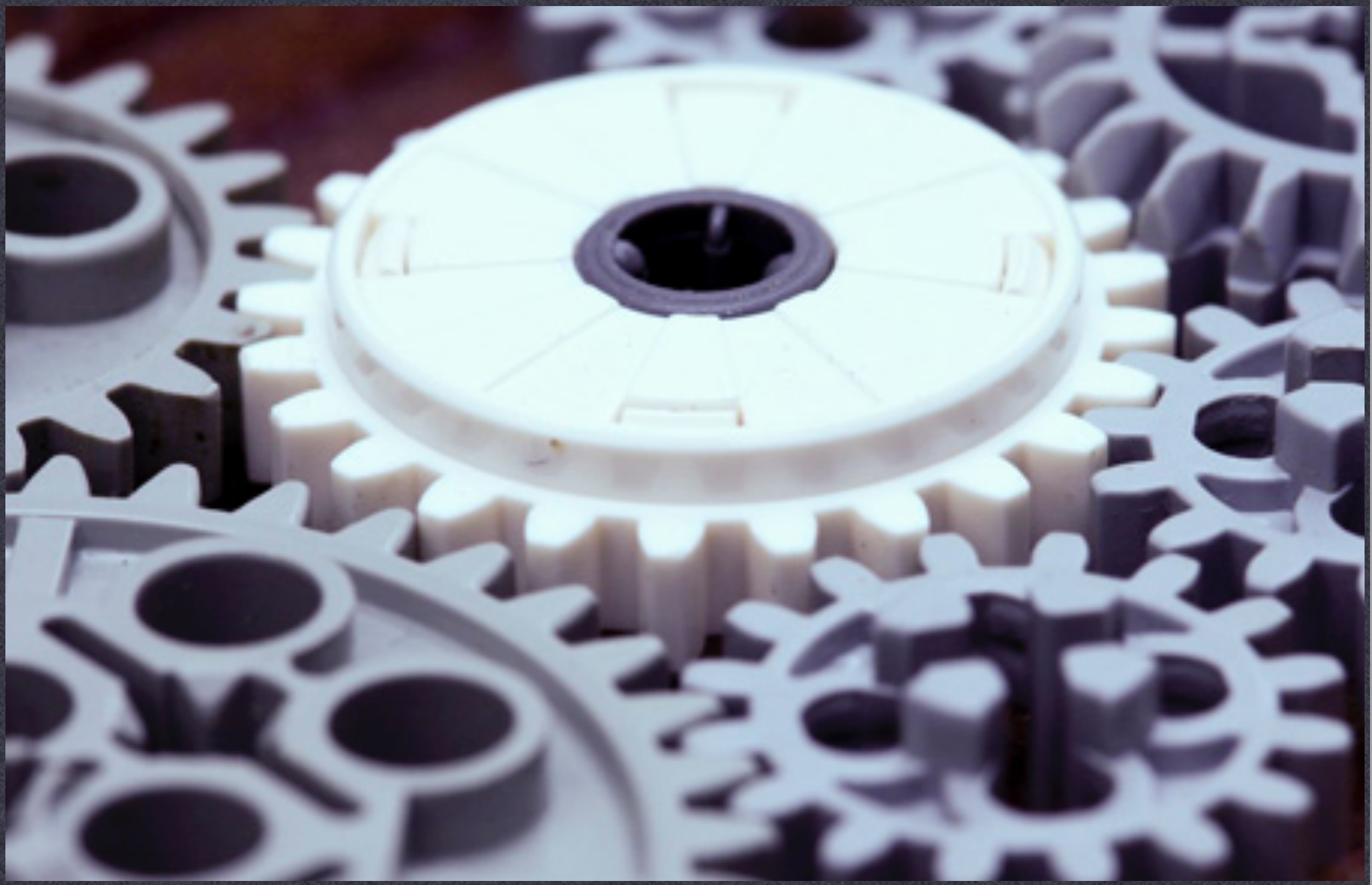
- ✦ Operating system files
- ✦ System table - AUD\$, FGA\_LOG\$
- ✦ User defined table
- ✦ System defined table
- ✦ Online Local / Online Near / Offline Remote storage



# Security of Auditing History

- ✖ Who made the change?
  - ✖ Identify via username if available
  - ✖ May be a layer of abstraction away if connection pooling
  - ✖ Proxy users, osuser, context variable
- ✖ Who should have access to changes?
  - ✖ Are audit tables protected from tampering?
  - ✖ Does your design support separation of duties?





# IMPLEMENTATION OPTIONS

FROM DATA MODELING TO OPTIONAL FEATURES



# Implementation Options

---

- ❖ Application code / business logic
- ❖ Roll your own - data modeling + triggers \*
- ❖ Roll your own - CDC-ish Offload
- ❖ Oracle Golden Gate \*
- ❖ Oracle Flashback Features \*
- ❖ Other Auditing Tools



# Application Code

---

- ✦ Generate change detail close to your business logic
- ✦ Pros:
  - ✦ Portable between database systems
  - ✦ Flexible
- ✦ Cons:
  - ✦ On the hook for all aspects of capture and management



# Data Modeling + Triggers

- ✖ Decide on Stream or Image storage of data changes
- ✖ Add columns to an 'audit' table
  - ✖ Operation - Insert / Update / Delete
  - ✖ User
  - ✖ DateTime
- ✖ Three after row triggers to record DML on each record



# Building your Auditing

---

- ✖ Issues
- ✖ Common Solutions
- ✖ Prior Art
- ✖ Example Trigger
- ✖ Pros and Cons



# Data Auditing Storage

- ✦ Stream Of Individual Changes



- ✦ Copy of Record At Each State



# Physical Storage of Changes

- ✖ Stream - if too many columns changed then might as well have just copied the record
- ✖ Point of storage inefficiency
  - ✖ Example 1: 1 record, 1 column changed
    - ✖ Stream : Create 1 record in audit table
    - ✖ Copy : Make 1 copy of record
  - ✖ Example 1: 1 record, n columns changed
    - ✖ Stream : Create n records in audit table
    - ✖ Copy : Make 1 copy of record



# All-In-One Storage

- ✦ Valid Time fields added to record, save new versions in same table
- ✦ Pros:
  - ✦ Simple concept
- ✦ Cons:
  - ✦ Locating current record logic is prohibitive. Can be simplified with views but still resource intensive.
  - ✦ May be hard to leverage built in RI because of “dupe” records



# Clone Audit Table Storage

- ✖ Add CHANGED\_DATE, CHANGED\_USER, [TRANSACTION\_ID]
- ✖ Pros:
  - ✖ Easy to recreate row-by-row what happened
  - ✖ Actions can be linked to a 'transaction' type entity.
  - ✖ RI can be used as normal on OLTP entities
- ✖ Cons:
  - ✖ Must go to separate table for audit info
  - ✖ Generation and Performance of triggers can be an issue
  - ✖ New record produced for each change
  - ✖ Logic required to show which fields updated from version to version



# Change Stream - Input

- ✦ Collect individual changes as they go against tables
- ✦ Typically recorded:
  - ✦ Table Name
  - ✦ Column Name
  - ✦ How did it change? Insert / Update / Delete Action
  - ✦ Who Changed
  - ✦ When was it changed?
  - ✦ What changed
  - ✦ Bonus: Transaction ID



# Change Stream - Insert

```
INSERT INTO EMP (EMPNO, ENAME, DEPTNO)
VALUES (100, 'John Smith', 20);
COMMIT;
```

EMP		
EMPNO	ENAME	DEPTNO
100	John Smith	20

EMP_AUDIT							
TABLE	COLUMN	PK	DATE	USER	ACT	OLD VALUE	NEW VALUE
EMP	ENAME	100	11/5/2013 5:00pm	DMANN	I	NULL	John Smith
EMP	DEPTNO	100	11/5/2013 5:00pm	DMANN	I	NULL	20



# Change Stream - Update

```
UPDATE EMP SET DEPTNO=50 WHERE EMPNO=100;  
COMMIT;
```

EMP		
EMPNO	ENAME	DEPTNO
100	John Smith	50

EMP_AUDIT							
TABLE	COLUMN	PK	DATE	USER	ACT	OLD VALUE	NEW VALUE
EMP	DEPTNO	100	11/5/2013 5:05pm	DMANN	U	20	50



# Change Stream - Delete

```
DELETE FROM EMP WHERE EMPNO=100;  
COMMIT;
```

EMP		
EMPNO	ENAME	DEPTNO

EMP_AUDIT							
TABLE	COLUMN	PK	DATE	USER	AC T	OLD VALUE	NEW VALUE
EMP	DEPTNO	100	11/5/2013 5:10pm	DMANN	D	NULL	NULL



# Change Stream

- ✖ Pros

- ✖ Easy to follow individual changes to a record

- ✖ Cons

- ✖ Inefficient access to previous states

- ✖ Complex to recreate a record at a specific point in time

- ✖ Tricky

- ✖ May be more efficient on disk space for simple/infrequent changes but can balloon for high rates of change



# Copy of Records

---

- ✖ Collect an image of the state of the record
- ✖ Pros
  - ✖ Simple mechanism to understand and implement
  - ✖ Very clear what the state of a record is at a point in time
- ✖ Cons
  - ✖ If only 1 column changes a lot of space can be wasted



# Image - Insert

```
INSERT INTO EMP (EMPNO, ENAME, DEPTNO)  
VALUES (100, 'John Smith', 20);  
COMMIT;
```

EMP		
EMPNO	ENAME	DEPTNO
100	John Smith	20

EMP_AUDIT					
EMPNO	ENAME	DEPT NO	TRANS_DATE	USER	DELETE_FLAG
100	John Smith	20	4/24/14 11:00	DMANN	N



# Image - Update

```
UPDATE EMP SET DEPTNO=50 WHERE EMPNO=100;  
COMMIT;
```

EMP		
EMPNO	ENAME	DEPTNO
100	John Smith	50

EMP_AUDIT					
EMPNO	ENAME	DEPTNO	TRANS_DATE	USER	DELETE_FLAG
100	John Smith	20	4/24/14 11:00	DMANN	N
100	John Smith	50	4/24/14 11:05	DMANN	N



# Image - Delete

```
DELETE FROM EMP WHERE EMPNO=100;  
COMMIT;
```

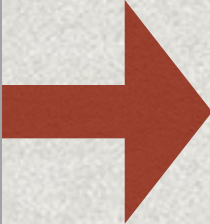
EMP		
EMPNO	ENAME	DEPTNO
100	John Smith	50

EMP_AUDIT					
EMPNO	ENAME	DEPT	TRANS_DATE	USER	DELETE_FLAG
100	John Smith	20	4/24/14 11:00	DMANN	N
100	John Smith	50	4/24/14 11:05	DMANN	N
100	John Smith	50	4/24/14 11:10	DMANN	Y



# Prior Art

- ❖ You are not the first to encounter this problem
- ❖ Data Warehouse Approach
  - ❖ “Slowly changing dimensions” - Kimball Group



SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change to attribute value	Facts associated with attribute's original value
Type 1	Overwrite attribute value	Facts associated with attribute's current value
Type 2	Add new dimension row for profile with new attribute value	Facts associated with attribute value in effect when fact occurred
Type 3	Add new column to preserve attribute's current and prior values	Facts associated with both current and prior attribute alternative values
Type 4	Add mini-dimension table containing rapidly changing attributes	Facts associated with rapidly changing attributes in effect when fact occurred
Type 5	Add type 4 mini-dimension, along with overwritten type 1 mini-dimension key in base dimension	Facts associated with rapidly changing attributes in effect when fact occurred, plus current rapidly changing attribute values
Type 6	Add type 1 overwritten attributes to type 2 dimension row, and overwrite all prior dimension rows	Facts associated with attribute value in effect when fact occurred, plus current values
Type 7	Add type 2 dimension row with new attribute value, plus view limited to current rows and/or attribute values	Facts associated with attribute value in effect when fact occurred, plus current values



# Example Trigger

```
CREATE TRIGGER emp_audit
AFTER INSERT OR DELETE OR UPDATE ON emp
for each row
begin
    insert into emp_audit (
        empno,      ename,
        job,        mgr,
        hiredate,   sal,
        comm,       deptno,
        last_modified_user, last_modified_date
    ) VALUES (
        :new.empno,    :new.ename,
        :new.job,      :new.mgr,
        :new.hiredate, :new.sal,
        :new.comm,     :new.deptno
        user,          sysdate
    );
end;
```



# Modeling/Trigger - Pros and Cons

## ✖ Pros

- ✖ “Free”
- ✖ Prior art - Lots of blog and StackOverflow posts
- ✖ Portable - to a degree

## ✖ Cons

- ✖ Recreating the wheel / “Not invented here” / Tech Debt
- ✖ Performance / Concurrency
- ✖ Cost - design/implementation/maintenance
- ✖ Security - “Chain of Custody”
- ✖ Implementation gotchas - before vs after triggers



# Data Modeling Demo

---

- ✦ Create Scott Objects
- ✦ Create \_AUDIT tables
- ✦ Add tracking fields to \_AUDIT tables
- ✦ Create triggers
- ✦ Exercise triggers
- ✦ View recorded data



# CDC-ish Offload

- ✖ Effects of DML changes recorded to a table
- ✖ Pros:
  - ✖ Logical and physical separation
  - ✖ Low impact to source DB
- ✖ Cons:
  - ✖ CDC is old tech not being actively improved
  - ✖ Separate hop to get to auditing history tables
  - ✖ Management of schema changes to base tables can be difficult/error prone



# Offload via GoldenGate

- ✦ GoldenGate observes changes to a table via Redo
  - ✦ Transports changes to 1 or more destination locations
  - ✦ Configurable to always insert instead of update
- ✦ Pros:
  - ✦ Low impact to source Database
    - ✦ No schema changes required
    - ✦ Good for auditing legacy or off-the-shelf
    - ✦ Handles high volumes with low resource usage
  - ✦ Logical and physical separation
  - ✦ Can store in local or remote tables
  - ✦ Managing source schema structure changes easier with new DDL support
- ✦ Cons:
  - ✦ Cost
  - ✦ New tool in your shop – Training, Complexity, Cooperation
  - ✦ Separate hop to get to auditing history tables



# Golden Gate Parameters

- ✦ INSERTALLRECORDS parameter
- ✦ Make sure to ADD TRANDATA or you may only see sparse population of your target records - only fields that changed
- ✦ Leverage variables to extract transaction time, userid, etc from source on target side
  - ✦ GGHEADER("USERID")
  - ✦ GGHEADER("TIMESTAMP")
  - ✦ GGHEADER("OPTYPE")
- ✦ Some other transaction information available



# Oracle Flashback

- ✦ Collection of features to capture, inspect and roll back database state changes
- ✦ Flashback Database, Flashback Table, Flashback Query, Flashback Transaction
- ✦ Flashback Data Archive
  - ✦ Managed and guaranteed retention of information
  - ✦ Oracle managed transaction time auditing for you
  - ✦ View previous state of data without destructively recovering to a point in time
  - ✦ Access to auditing features geared towards end users, not just DBAs



# Flashback Pros:

- ✦ Simplicity – Easy to get up and running
- ✦ Managed by Oracle. No triggers to debug, schema objects to deal with. Security a +
- ✦ Configurable retention times
- ✦ Flashback Version Query – point in time with “as of” syntax
- ✦ Flashback Transaction Query – list previous versions of records
- ✦ Good for adding auditing to a mature system or off-the-shelf system
- ✦ Licensing relaxed in 11.2.0.4+ versions



# Flashback Cons:

- ✖ Operations management learning curve
  - ✖ Risk of detaching history from table.
  - ✖ Handling export/import of audit data
  - ✖ Can we get data back into Flashback if it gets detached? `DBMS_FLASHBACK_ARCHIVE`
- ✖ How does it play with other features?
- ✖ Partition/subpartition operations on internal history tables not allowed
- ✖ Table DDL may be restricted in earlier versions
- ✖ Vendor Dependency



# Preparing for Flashback

---

- ✦ Create storage area for Archive Areas
- ✦ Determine retention periods
- ✦ Assign tables to Archive Areas
- ✦ Query clause or DBMS\_FLASHBACK.



# Flashback Query

- ✦ Allows viewing of data in a point in time
  - ✦ Available with default Flashback setup - to a point
  - ✦ Guaranteed with Flashback Data Archive
- ✦ Two options:
  - ✦ Construct query with “AS OF” clause
  - ✦ Set time with DBMS\_FLASHBACK.ENABLE\_AT\_TIME and query as normal

```
SELECT *  
  FROM my_flashback_table  
AS OF TIMESTAMP <timestamp>;
```

```
EXEC DBMS_FLASHBACK.ENABLE_AT_TIME(SYSTIMESTAMP - 1);  
SELECT *  
FROM my_flashback_table;
```



# Flashback Transaction

- ✦ Provides detailed metadata about a transaction
- ✦ Shows transactions that took place during a specific time period
- ✦ Use with Flashback Versions for full detail of a transaction's activity

```
SELECT xid,  
       operation,  
       undo_sql,  
FROM flashback_transaction_query  
WHERE table_owner = USER  
       AND table_name = 'MYTABLE'  
ORDER BY start_timestamp;
```



# Flashback Version Query

- ✦ Provides versions of a row between two points in time
- ✦ Pseudocolumns available to provide metadata
- ✦ VERSIONS BETWEEN clause

```
SELECT versions_startscn, versions_starttime,  
       versions_endscn, versions_endtime,  
       versions_xid, versions_operation,  
       mytable.*  
FROM mytable  
   VERSIONS BETWEEN TIMESTAMP  
        TO_TIMESTAMP('2013-03-29 08:00:00',  
                      'YYYY-MM-DD HH24:MI:SS')  
      AND TO_TIMESTAMP('2013-03-29 09:00:00',  
                      'YYYY-MM-DD HH24:MI:SS')  
WHERE id = 1;
```



# Flashback Archive Demo

---

- ✦ Define a Flashback Data Archive
- ✦ Assign tables to it
- ✦ Perform DML on tables
- ✦ Inspect the changes



# Other Auditing Tools

---

- ✦ Basic Statement Auditing
- ✦ Fine Grained Statement Auditing
- ✦ Logminer
- ✦ Misc 3rd Party Tools
- ✦ 12 Information Lifecycle Management



# Basic Auditing

- ✦ AUDIT keyword

```
AUDIT INSERT, UPDATE, DELETE ON mytable BY ACCESS;  
AUDIT ALL;
```

- ✦ Pros

- ✦ A simple, gentle introduction to auditing

- ✦ Cons

- ✦ For our purposes, it just record events and statements, not atomic detail of changes
  - ✦ Very broad brush, can be very noisy



# Fine Grained Auditing

---

## ✖ Pros

- ✖ Enhancement to basic auditing
- ✖ Provides for reducing noise in the audit log

## ✖ Cons

- ✖ Not built to record atomic data changes. Will just let you know if policy criteria has been met by a SQL statement
  - ✖ User ran a SELECT to access sensitive data
- ✖ Micro management of policies



# Logminer

---

- ✦ Pros:

- ✦ Low performance overhead
- ✦ Around since 8i
- ✦ Good for detailed inspection of activity

- ✦ Cons:

- ✦ Target user = DBA
- ✦ Outputs SQL Statements



# Other 3rd Party Tools

- ✦ “Tee” Tools

- ✦ Piggybacked JDBC or ODBC driver
- ✦ SQL statements siphoned off to other system for analysis
- ✦ No DML recording as activity is in database, not at “Tee” point

- ✦ Other approaches

- ✦ McAfee Database Activity Monitoring
- ✦ Sensitive SQL statements cancelled
- ✦ No DML recording



# 12c Info Lifecycle Mgmt

- ✖ Temporal Validity
  - ✖ Adds columns for effective/discontinued date tracking
  - ✖ Automatic predicates for valid time query

```
alter table mytable add period for track_time;
```

```
update mytable set track_time_start=sysdate-10 ;
```

```
update mytable set track_time_end=sysdate-5;
```

```
EXECUTE DBMS_FLASHBACK_ARCHIVE.enable_at_valid_time(SYSDATE-15);
```

```
EXECUTE DBMS_FLASHBACK_ARCHIVE.enable_at_valid_time(SYSDATE-8);
```

```
EXECUTE DBMS_FLASHBACK_ARCHIVE.enable_at_valid_time(SYSDATE);
```



# 12c Info Lifecycle Mgmt

- ✘ In-database archiving
  - ✘ Manage visibility of active/inactive records in a table
  - ✘ “Active Flag” filtering

```
alter table mytable row archival;
```

```
SELECT COUNT(*) FROM MYTABLE;
```

```
update sales
```

```
  set ORA_ARCHIVE_STATE=DBMS_ILM.ARCHIVESTATENAME(1)
```

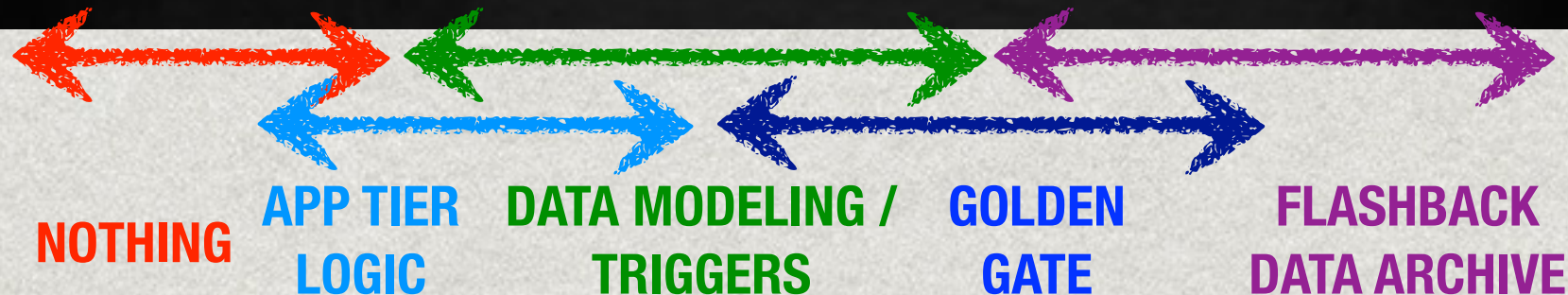
```
  where time_id < '01-JAN-2013';
```

```
alter session set row archival visibility=ALL;
```

```
alter session set row archival visibility = active;
```



# Where do the solutions fit?





# Where to next?

---

- ✦ Design for customer needs
  - ✦ Make sure desired outputs are covered by inputs
- ✦ Get familiar with implementation options
  - ✦ Lots of prior art with Data Modeling
  - ✦ Useful even if leveraging other products
- ✦ Test your approach
  - ✦ Unit test
- ✦ You've got history, exploit it with Data Warehousing
  - ✦ Kent Graziano



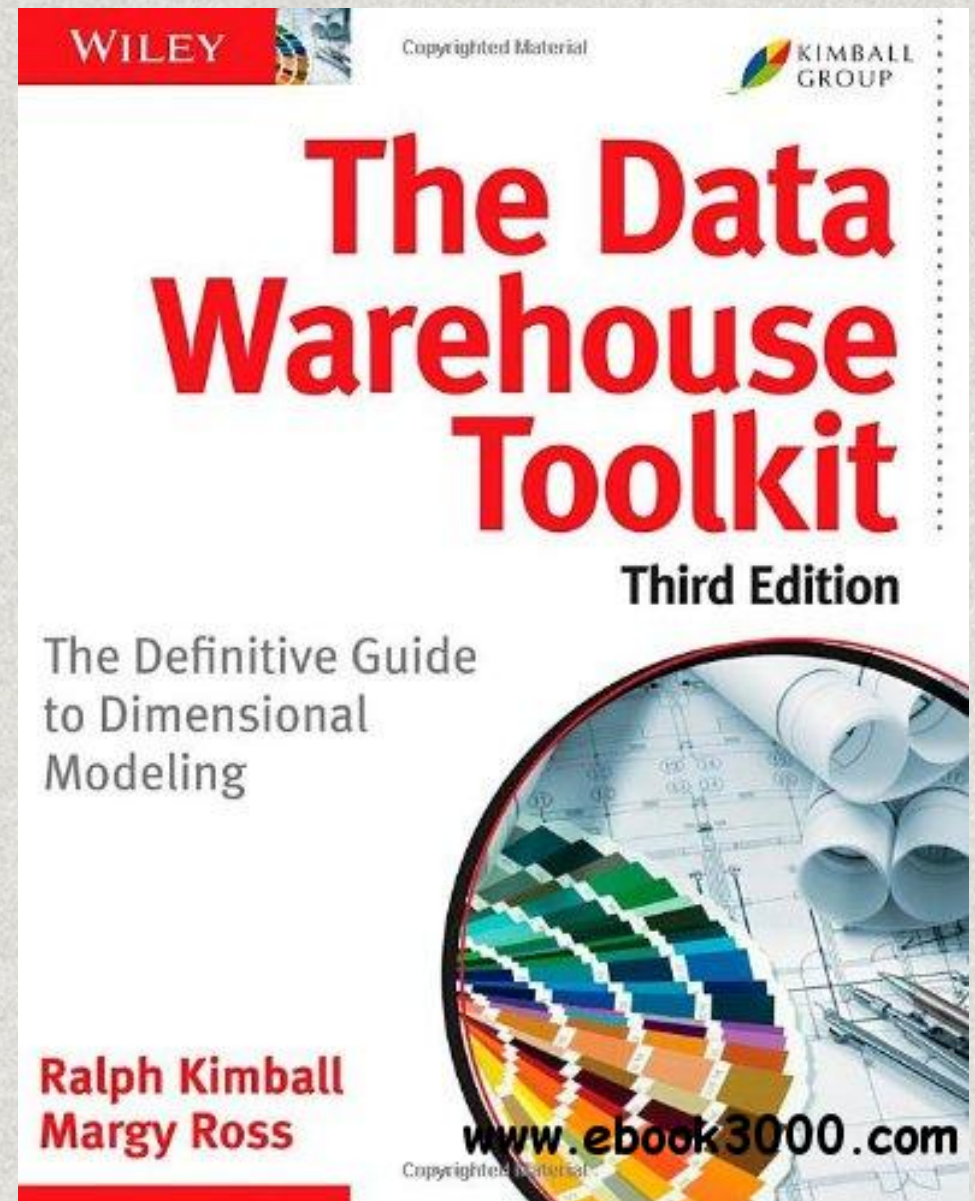
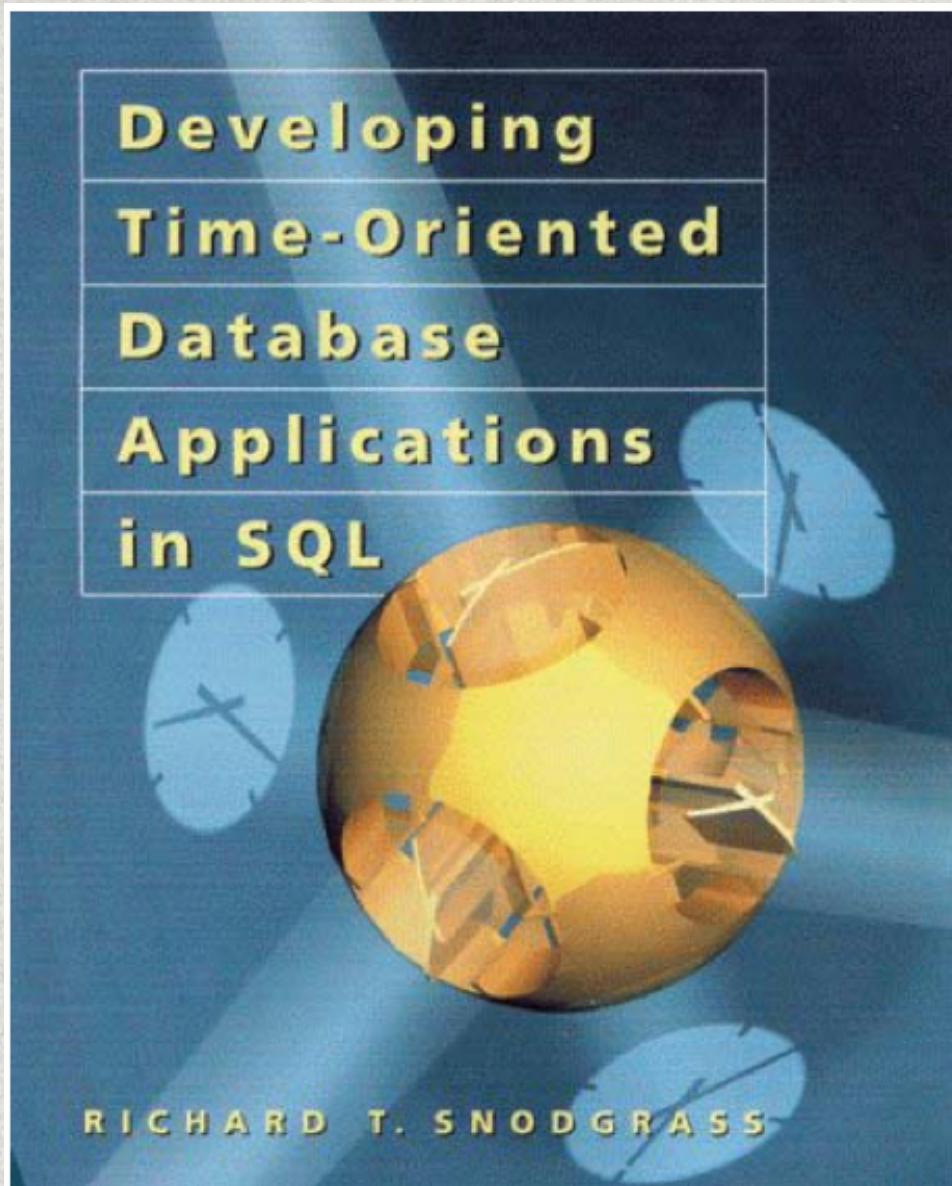
# Where to next?

---

- ✦ We have mostly explored the Transaction Time auditing
  - ✦ Explore Valid Time
    - ✦ When is the start/end date your data is considered valid?
    - ✦ This is the User's Problem Domain (ie Insurance Policy Coverage Period)
  - ✦ 12c Temporal Validity Filter
    - ✦ Flashback to version of data you want and then Filter for records valid at that point in time



# Resources





# QUESTIONS

**Slides, code, links :**

**<http://ba6.us>**

**[Email: david@ba6.us](mailto:david@ba6.us)**

**Twitter: @ba6dotus**





# References

- ❖ <http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Spradlin.pdf>
- ❖ <http://www.kimballgroup.com/1999/08/03/when-a-slowly-changing-dimension-speeds-up/>
- ❖ <http://www.rittmanmead.com/2005/10/temporal-databases/>
- ❖ Auditing in 10g  
<http://www.oracle-base.com/articles/10g/auditing-10gr2.php>
- ❖ Independent evaluation of Total Recall by Trivadis  
[http://www.trivadis.com/uploads/tx\\_cabagdownloadarea/Flashback-Data-Archives-Rechecked-v1.4\\_final.pdf](http://www.trivadis.com/uploads/tx_cabagdownloadarea/Flashback-Data-Archives-Rechecked-v1.4_final.pdf)
- ❖ <http://stackoverflow.com/questions/12321200/database-row-snapshots-revisions>
- ❖ [http://nyoug.org/Presentations/2010/March/Johal\\_Flashback.pdf](http://nyoug.org/Presentations/2010/March/Johal_Flashback.pdf)
- ❖ <http://portrix-systems.de/blog/brost/the-new-improved-and-free-flashback-data-archives-in-12c/>
- ❖ <http://infotechsworld.wordpress.com/2011/10/19/transaction-management-with-logminer-and-flashback-data-archive/>
- ❖ <http://www.oracle-developer.net/display.php?id=320>
- ❖ Temporal Databases - RT Snodgrass - <https://cs.arizona.edu/~rts/pubs/LNCS639.pdf>
- ❖ <http://www.oracle.com/technetwork/database/storage/total-recall-whitepaper-171749.pdf>