Using a Data Warehouse to Audit a Transactional System

Mike Glasser

Office of Institutional Research
University of Maryland – Baltimore County
Agenda

- Introduction
- Why Audit
- How to Audit
  - The Setup
  - The Audit
- Audit Results
- Questions
Located in suburban Baltimore County, between Baltimore, MD and Washington, DC

One of the three public research campuses in the University of Maryland System

- 10K undergraduate and 2K graduate students
- 2,200 Employees; 715 full-time faculty
- 8 Pan-Am chess championships
UMBC Data Warehouse

• Custom HR and Student tables for IR
  – 5 years
  – Oracle
  – Relational tables

• Purchased iStrategy DW for Campus
  – Legacy student > 1 year
  – PS Student Admin live for 1 month
  – SQL Server 2005
  – 15 fact tables
  – 100 custom reporting tables for IR
  – 75 GB in size
  – 3 IR staff + 1.5 IT staff
What is an “Audit”? 

Need to identify and reconcile the data that was changed yesterday
Why Audit in the DW?

• Audit HR data entry
  – Piecing together 4 pages every day

• Learn about changes to codes

• Why in the Data Warehouse?
  – Take pressure off OLTP
  – No custom tables
  – No changes to transactional tables
  – No need to turn on audit
Auditing at UMBC

- 10 tables
  - PS_Job, PS_Citizenship, Tax_Data
  - PS_Acad_Plan_Tbl (majors)
  - Translation table
  - Error Messages

- Average 825,000 records audited per night
- Average 1,000+ changes per night
- Average about 45 seconds for complete audit
- Max < 4 minutes with 825,000 changes
How to Audit

- Make a copy of “yesterday”
- Refresh the table with “today”
- Compare “yesterday” with “today”
- Report the differences
How We Audit

• Nightly_YDAY_Tables
  – Copy Source tables to _YDAY
• Refresh Source tables
• Nightly_Audit
  – Load_Data_Changes for each table
    • Data_Changes_Staging
    • Parse updates by field
  – Email_Audit_Summary
Setup

- Create YDAY table
- Create “Current” view (optional)
- Index YDAY table
- Add to Nightly YDAY procedure
- Add to Nightly Audit procedure
YDAY table

• Make a copy of “yesterday”

• Setup is one time copy of the table to be audited
  – Tablename with suffix _YDAY
  – Index on primary key

• YDAY table populated every night
  – Copy of yesterday’s source table
  – Before nightly refresh of source tables
  – Truncated to preserve indexes
Setup

- Create YDAY table
- Create “Current” view (optional)
- Index YDAY table
- Add to Nightly YDAY procedure
- Add to Nightly Audit procedure
Special Case

Effective Dating

- Transaction tables keep a history
- Current data is based on effective date
- New effective dated record inserted when something is updated
- Want to treat new record as UPDATE
“Current” View

- Create view (virtual table) of rows currently in effect
- Populate YDAY table from view
- Primary key does not include Effdt
- Compare YDAY to view of rows currently in effect
- Skip Effdt column during comparison
Effective Dating

<table>
<thead>
<tr>
<th>Emplid</th>
<th>Effdt</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2/3/2008</td>
<td>Mathematics</td>
</tr>
</tbody>
</table>

**Yesterday**

<table>
<thead>
<tr>
<th>Emplid</th>
<th>Effdt</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2/3/2008</td>
<td>Mathematics</td>
</tr>
<tr>
<td>1000</td>
<td>4/1/2009</td>
<td>English</td>
</tr>
</tbody>
</table>

**Today**
### Effective Dating

#### Yesterday

<table>
<thead>
<tr>
<th>Emplid (PK)</th>
<th>Effdt</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2/3/2008</td>
<td>Mathematics</td>
</tr>
</tbody>
</table>

#### Today View

<table>
<thead>
<tr>
<th>Emplid (PK)</th>
<th>Effdt</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>4/1/2009</td>
<td>English</td>
</tr>
</tbody>
</table>

Emplid 1000 updated major from Math to English

The changed EFFDT is ignored during comparison
Setup

- Create YDAY table
- Create “Current” view (optional)
- Index YDAY table
  - Create primary key
- Add to Nightly YDAY procedure
- Add to Nightly Audit procedure
Nightly YDAY Procedure

BEGIN TRANSACTION

TRUNCATE TABLE Stage.PS_Citizenship_YDAY

INSERT INTO Stage.PS_Citizenship_YDAY
SELECT * FROM Source.PS_Citizenship

COMMIT TRANSACTION
SELECT @table = 'PS_CITIZENSHIP'

EXECUTE Admin.Load_Data_Changes

    @p_base_table = 'Stage.PS_Citizenship_YDAY',
    @p_comp_table = 'Source.PS_Citizenship',
    @p_details = 'YES',
    @p_changes_tablename = @table
Setup

- Created YDAY table
- Created “Current” view (maybe)
- Created Primary Key
- Added to Nightly YDAY procedure
- Added to Nightly Audit procedure

- PS_Citizenship table is now going to be audited every night
How do we know what changed?

SELECT @table = 'PS_CITIZENSHIP'

EXECUTE Admin.Load_Data_Changes

@p_base_table = 'Stage.PS_Citizenship_YDAY',
@p_comp_table = 'Source.PS_Citizenship',
@p_details = 'YES',
@p_changes_tablename = @table
Delete, Add, Update?

Compare Primary Keys

Yesterday

Deletes

Updates

Today

Adds
Compare Yesterday to Today

- Read database metadata to determine primary key
- Read metadata to build dynamic SQL
- Dynamic SQL is built for ADDs and DELETEs
- Dynamic SQL is built for UPDATEs
- The dynamic SQL inserts records into staging table
- Final step parses staging table
Sample Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key1</td>
<td>Varchar(4)</td>
</tr>
<tr>
<td>Key2</td>
<td>Integer</td>
</tr>
<tr>
<td>Field1</td>
<td>Varchar(10)</td>
</tr>
<tr>
<td>Field2</td>
<td>Datetime</td>
</tr>
</tbody>
</table>
# Sample Data

## Yesterday

<table>
<thead>
<tr>
<th>Key1</th>
<th>Key2</th>
<th>Field1</th>
<th>Field2</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMBC</td>
<td>1</td>
<td>Update Me</td>
<td>7/8/2009 12:00 AM</td>
</tr>
<tr>
<td>UMBC</td>
<td>2</td>
<td>Delete Me</td>
<td>6/7/2009 12:00 AM</td>
</tr>
</tbody>
</table>

## Today

<table>
<thead>
<tr>
<th>Key1</th>
<th>Key2</th>
<th>Field1</th>
<th>Field2</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMBC</td>
<td>1</td>
<td>Updated</td>
<td>5/1/2009 12:00 AM</td>
</tr>
<tr>
<td>UMBC</td>
<td>3</td>
<td>Add Me</td>
<td>3/4/2009 12:00 AM</td>
</tr>
</tbody>
</table>
## Staging Audit Data

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Action</th>
<th>Key Fields</th>
<th>Key Values</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Delete</td>
<td>Key1~Key2</td>
<td>UMBC~2</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Add</td>
<td>Key1~Key2</td>
<td>UMBC~3</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Update</td>
<td>Key1~Key2</td>
<td>UMBC~1</td>
<td>~ Field1…</td>
</tr>
</tbody>
</table>

~~Field1=[Update Me^^Updated]  
~Field2=[Jul 8 2009 12:00AM^^May 1 2009 12:00AM]~

| ~ | field delimiter  
| ^^ | old/new value delimiter |
Audit SQL for Add

```
INSERT INTO Stage.DATA_CHANGES
    (Tablename, Action, Key_Fields, Key_Values, Changes)
SELECT 'Sample' tname, 'ADD' action, 'Key1~Key2' kfields, convert(varchar,A.Key1) + '~' + convert(varchar,A.Key2) kvalues, null
FROM mglasser.Sample A
LEFT OUTER JOIN mglasser.Sample_YDAY B
    ON A.Key1 = B.Key1 and A.Key2 = B.Key2
WHERE B.Key1 is null
```
Create Dynamic SQL for Add

SET @SQL =

'INSERT INTO Stage.DATA_CHANGES
    (Tablename,Action,Key_Fields,Key_Values,Changes) SELECT ''
    + @p_changes_tablename + '' tname, ''ADD'' action, ''
    + @Key_Fields + '' kfields, ' + @Key_Values + ' kvalues, null ' +
    + @Outer_Join + ' ON ' + @Join_Keys + ' WHERE B.' +
    + substring( @Key_Fields,1,charindex('~',@Key_Fields+~')-1) +
    + ' is null'

EXECUTE (@SQL)
Primary Key Fields

- Derived from Primary Key metadata
  - SQL Server
    - Table_Constraints
    - Key_Column_Usage
  - Oracle
    - All_Indexes
    - All_Ind_Columns

- Tells us the name of the primary key
- Tells us the fields in the primary key
Primary Key SQL

```
SELECT
  stuff(( select '~' + k1.column_name
          from information_schema.key_column_usage k1
          where k1.constraint_name = c.constraint_name
          order by ordinal_position
          for xml path(''"
      ), 1,1,""
    ), 1,1,""
FROM INFORMATION_SCHEMA.Table_Constraints c
WHERE c.table_schema = @Schema1
  AND c.table_name = @Table1
  AND c.constraint_type = 'PRIMAR KEY'
GROUP BY constraint_name

Result for @Key_Fields: Key1~Key2
```
Audit SQL for Add

INSERT INTO Stage.DATA_CHANGES
(Tablename, Action, Key_Fields, Key_Values, Changes)

SELECT 'Sample' tname, 'ADD' action, 'Key1~Key2' kfields,
    convert(varchar,A.Key1) + '~' +
    convert(varchar,A.Key2) kvalues, null

FROM mglasser.Sample A
LEFT OUTER JOIN mglasser.Sample_YDAY B
    ON A.Key1 = B.Key1 and A.Key2 = B.Key2
WHERE B.Key1 is null
## Staging Audit Data

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Action</th>
<th>Key Fields</th>
<th>Key Values</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Delete</td>
<td>Key1~Key2</td>
<td>UMBC~2</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Add</td>
<td>Key1~Key2</td>
<td>UMBC~3</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Update</td>
<td>Key1~Key2</td>
<td>UMBC~1</td>
<td>~Field1…</td>
</tr>
</tbody>
</table>

~~Field1=[Update Me^^Updated]  
~Field2=[Jul 8 2009 12:00AM^^May 1 2009 12:00AM]~

~ field delimiter  
^^ old/new value delimiter
Audit for Update

- Join tables by primary key
- Compare each field in YDAY with same field in today’s table
- String together results of comparison
  - Delimited by ~
- If fields are equal, results are empty
- If not equal, then put fieldname and both values
- In one SQL insert statement
  - Created dynamically from metadata
Comparison Results

• No updates in Sample data
  ~~~

• Updates from Sample data
  ~~Field1~~Field2~~

• Insert into staging table only those records where result has something other than ~
SQL for Comparison

SQL Server

CASE
    WHEN YDAY.Field1 = TODAY.Field1 THEN ' ~ ' 
    ELSE ' Field1 = [ ' + YDAY.Field1 + ' ^ ^ ' + TODAY.Field1 + ' ] ~ ' 
END

Oracle

DECODE( YDAY.Field1, TODAY.Field1, null, 
    ' Field1 = [ ' || YDAY.Field1 || ' ^ ^ ' || TODAY.Field1 || ' ] ~ ' )
SELECT @Changes =
  stuff( (  
    select '+' case when isnull(convert(varchar,A.' + c1.column_name + '),'""') = isnull(convert(varchar,B.' + c1.column_name + '),'""') then "~" else '"' + c1.column_name + ' = ["'+isnull(convert(varchar,A.' + c1.column_name + '),'""') + '"^^" +isnull(convert(varchar,B.' + c1.column_name + '),'""') + '"(~" end '  
  from information_schema.columns c1  
  where c1.column_name not in  
    (  
      'LOAD_DTTM','LASTUPDDTTM','DW_LOAD_DTTM',  
      'UW_LOAD_DTTM','AGE','EFFDT','EFFSEQ'  
    )  
  and c1.table_schema = c2.table_schema  
  and c1.table_name = c2.table_name for xml path('') ) , 1,1,"")
FROM Information_Schema.Columns c2
WHERE table_schema = @Schema1 and table_name = @Table1
GROUP BY c2.table_schema, table_name
### Staging Audit Data

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Action</th>
<th>Key Fields</th>
<th>Key Values</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Delete</td>
<td>Key1~Key2</td>
<td>UMBC~2</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Add</td>
<td>Key1~Key2</td>
<td>UMBC~3</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Update</td>
<td>Key1~Key2</td>
<td>UMBC~1</td>
<td>~Field1…</td>
</tr>
</tbody>
</table>

~~Field1=[Update Me^^Updated]~~

~Field2=[Jul 8 2009 12:00AM^^May 1 2009 12:00AM]~

~~ ~ field delimiter

^^ old/new value delimiter
# Final Audit Data

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Action</th>
<th>Key Fields</th>
<th>Key Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Delete</td>
<td>Key1~Key2</td>
<td>UMBC~2</td>
</tr>
<tr>
<td>Sample</td>
<td>Update</td>
<td>Key1~Key2</td>
<td>UMBC~1</td>
</tr>
<tr>
<td>Sample</td>
<td>Update</td>
<td>Key1~Key2</td>
<td>UMBC~1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Old Value</th>
<th>New Value</th>
<th>DW Load Dttm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field1</td>
<td>Update Me</td>
<td>Updated</td>
<td>4/15/2009 14:30</td>
</tr>
<tr>
<td>Field2</td>
<td>Jul 8 2009 12:00AM</td>
<td>May 1 2009 12:00AM</td>
<td>4/15/2009 14:30</td>
</tr>
</tbody>
</table>
Report the Differences

- The audit is complete
  - The keys for deleted or new records
  - The old and new values for updated fields
- Notify users of summary results
- Allow reporting of details
Email Outputs

Email sent to HR data manager and OIR nightly

<table>
<thead>
<tr>
<th>Records</th>
<th>Add</th>
<th>Delete</th>
<th>Update</th>
<th>Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>1</td>
<td>0</td>
<td>338</td>
<td>Employees_Cur</td>
</tr>
<tr>
<td>52</td>
<td>8</td>
<td>0</td>
<td>602</td>
<td>Jobs_Cur</td>
</tr>
<tr>
<td>23</td>
<td>19</td>
<td>1</td>
<td>3</td>
<td>PS_CITIZENSHIP</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>20</td>
<td>PS_FED_TAX_DATA</td>
</tr>
<tr>
<td>53</td>
<td>8</td>
<td>0</td>
<td>461</td>
<td>PS_JOB</td>
</tr>
<tr>
<td>19</td>
<td>15</td>
<td>0</td>
<td>8</td>
<td>PS_PERS_DATA_EFFDT</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>PS_STATE_TAX_DATA</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>0</td>
<td>6</td>
<td>PS UM_JOB_INFO</td>
</tr>
</tbody>
</table>
### Crystal Report

Comparing to changes made

<table>
<thead>
<tr>
<th>Table: PS_JOB</th>
</tr>
</thead>
</table>

| Key: Emplid-Empl_Rcd 100000808-1 |

<table>
<thead>
<tr>
<th></th>
<th>Old:</th>
<th>New:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR_Status</td>
<td>A</td>
<td>I</td>
</tr>
<tr>
<td>Empl_Status</td>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>Action</td>
<td>PAY</td>
<td>TER</td>
</tr>
<tr>
<td>Action_Dt</td>
<td>Jun 27 2008 12:00AM</td>
<td>Apr 22 2009 12:00AM</td>
</tr>
<tr>
<td>Action_Reason</td>
<td>NLY</td>
<td>TMP</td>
</tr>
<tr>
<td>Ben_Status</td>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>Termination_Dt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asgn_End_Dt</td>
<td>Apr 22 2009 12:00AM</td>
<td>Apr 22 2009 12:00AM</td>
</tr>
<tr>
<td>Last_Date_Worked</td>
<td>Apr 22 2009 12:00AM</td>
<td>Apr 22 2009 12:00AM</td>
</tr>
<tr>
<td>Lastupdoprid</td>
<td></td>
<td>DENISE</td>
</tr>
</tbody>
</table>
Potential Alternatives

• Database logs
  – Utility program from vendors that read the logs the database keeps
  – Not available choice for us
  – Costs money (free ones?)
  – Output may not be flexible?

• Smaller views
  – Only audits the fields in the view
  – Can control for specific records
  – Smaller YDAY tables
  – Faster performance
Poor Alternatives

• Separate program for each table
  – Longer development
  – Potentially huge SQL
  – Maintenance for table changes

• Separate dynamic SQL for each field
  – Horrible performance
  – Reads the tables once for each field
Alternative Uses

- Debugging & development
- Table structure changes
  - ALL_TAB_COLS
  - Information_Schema.Columns
- Deltas for data warehouse loads
- Monitor or analyze database actions over time
- ???
Wrap Up

Any Questions?

Mike Glasser
University of Maryland - Baltimore County

mglasser@umbc.edu  (410) 455-3577

Source code is available upon request