

Data Quality and Database Archiving: The Intersection of Two Important Data Management Functions

ABSTRACT

This presentation shows that when database archiving technology is employed for large database applications that have very long data retention periods, the data quality is preserved. It covers a tutorial on basic definitions of database archiving. It shows how keeping data in operational systems for long periods of time creates many opportunities for the data quality to erode. It concludes with a detailed explanation of why robust database archiving implementation prevents erosion from occurring and thus preserves the original quality for all time.

BIOGRAPHY

Jack Olson

Chief Executive Officer
SvalTech



Jack Olson has worked in the commercial software development business for 40 years. His career has mostly consisted of architecting solutions to IT problems in the area of database systems and tools. He spent 17 years in IBM development labs working on such notable products as CICS, IMS, DB2, and AIX. He worked at BMC software as Corporate Architect, as Vice President of Development at Peregrine Systems, and as Chief Technology Officer for Evoke Software and NEON Enterprise Software. He has worked with several other startup companies in recent years as a consultant, advisor, or board member. He is currently an independent consultant. Jack has published two books: “Data Quality: the Accuracy Dimension”, 2003 and “Database Archiving: How to Keep Lots of Data for a Very Long Time”, 2009. Jack has a BS degree in Mathematics from the Illinois Institute of Technology and an MBA from Northwestern University.

2010 MIT Information Quality Industry Symposium

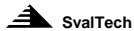
Data Quality and Database Archiving

The intersection of two important Data Management Functions

**“Database Archiving: How to Keep Lots of Data for a Long Time”
Jack E. Olson, Elsevier, 2009**

Jack E. Olson
jack.olson@SvalTech.com
512-565-9584

1



Presentation Roadmap

Definitions


- Database Archiving
- Business Records
- Long Term Data Retention

Data Quality Problems With Single, Operational Database Approach

- Long term loss of clarity of understanding
- Metadata change corruption
- Reference data changes
- Database Consolidation (mergers and acquisitions)








Using Database Archiving for Improved Data Quality

- Education and Awareness
- Early Business Records Capture
- Managing Data and Metadata within Application Segments
- Capture Extended Metadata (become application independent)
- Freeze Reference Data
- Metadata Change Sensitive Data Access

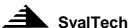
 SvalTech

Database Archiving

The process of removing selected data items from operational databases that are not expected to be referenced again and storing them in an archive database where they can be retrieved if needed.

					
Physical Documents application forms mortgage papers prescriptions	File Archiving structured files source code reports	Document Archiving word pdf excel XML	Multi-media files pictures sound telemetry	Email Archiving outlook lotus notes	Database Archiving DB2 IMS ORACLE SAP PEOPLESOFT 

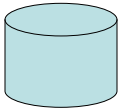
Copyright Jack Olson, 2010 3

 SvalTech

Business Records

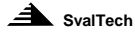
The data captured and maintained for a single business event or a to describe a single real world object.

Databases are collections of business records.



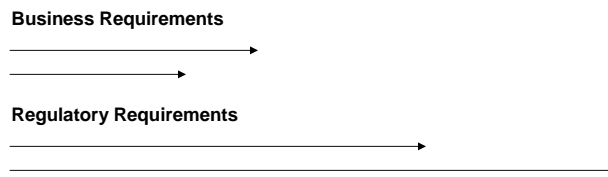
customer
employee
stock trade
purchase order
deposit
loan payment

Copyright Jack Olson, 2010 4



Data Retention

The requirement to keep data for a business object for a specified period of time. The object cannot be destroyed until after the time for all such requirements applicable to it has past.

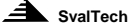


The Data Retention requirement is the longest of all requirement lines.



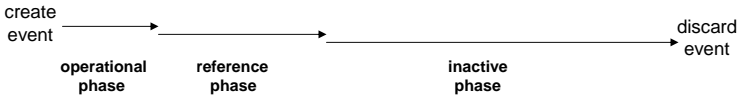
Data Retention

- Retention requirements vary by business object type
- Retention requirements from regulations are exceeding business requirements
- Retention requirements will vary by country
- Retention requirements imply the obligation to maintain the authenticity of the data throughout the retention period
- Retention requirements imply the requirement to faithfully render the data on demand in a common business form understandable to the requestor
- The most important business objects have the longest retention periods
- The data with the longest retention periods tends to be accumulate the largest number of instances
- Retention requirements often exceed 10 years. Requirements exist for 25, 50, 70 and more years for some applications

 SvalTech


Data Time Lines

for a single instance of a data object



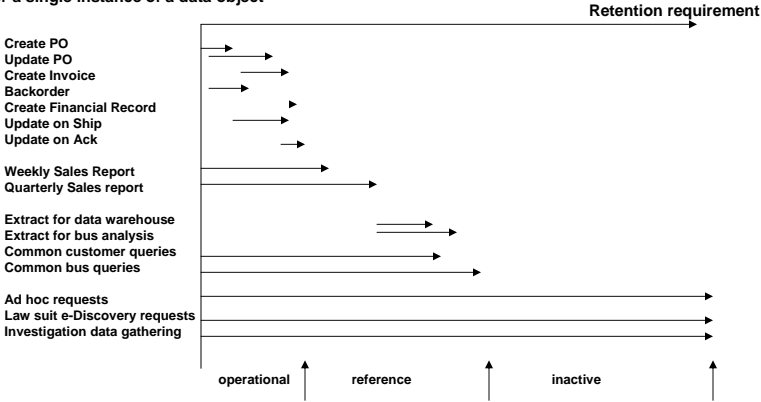
operational phase	can be updated, can be deleted, may participate in processes that create or update other data
reference phase	used for business reporting, extracted into business intelligence or analytic databases, anticipated queries
inactive phase	no expectation of being used again, no known business value, being retained solely for the purpose of satisfying retention requirements. Must be available on request in the rare event a need arises.

Copyright Jack Olson, 2010 7

 SvalTech

Data Process Time Lines

for a single instance of a data object



Create PO Update PO Create Invoice Backorder Create Financial Record Update on Ship Update on Ack Weekly Sales Report Quarterly Sales report Extract for data warehouse Extract for bus analysis Common customer queries Common bus queries Ad hoc requests Law suit e-Discovery requests Investigation data gathering	operational reference inactive
---	--------------------------------------

Copyright Jack Olson, 2010 8



Some Observations

- Some objects exit the operational phase almost immediately (financial records)
- Some objects never exit the operational phase (customer name and address)
- Most transaction data has an operational phase of less than 10% of the retention requirement and a reference phase of less than 20% of the retention requirement
- Inactive data generally does not require access to application programs: only access to ad hoc search and extract tools

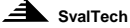


Application Segments

An **application segment** is a set of business objects generated from a single version of an application where all objects in the segment have data consistent with a single metadata definition.

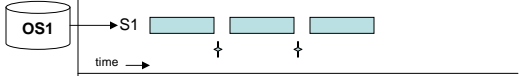
A **metadata break** is a point in the life of the operational database where a change in metadata is implemented that changes the structure of the data or the manner in which data is encoded.

- An application will have many segments over time
- Minor changes in metadata can sometimes be implemented without forcing a segment change
- Major metadata changes will always generate a segment change where data created in the previous segment cannot be recast to the new metadata definition without some compromise in the data
- Application segments can be generated in parallel with one operational implementation using one version of the application at the same time that another operational instance is using a different version of the application

 **SvalTech**

Application Segments

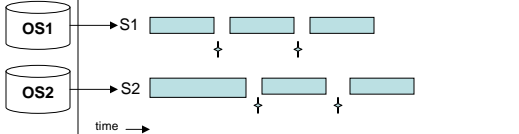
case 1 **Application: customer stock transactions**



time →

Source 1 = Trades – All Stock Trades

case 2 **Application: customer stock transactions**




time →

Source 1 = Stock Trades – North American Division
 Source 2 = Stock Trades – Western Europe Division

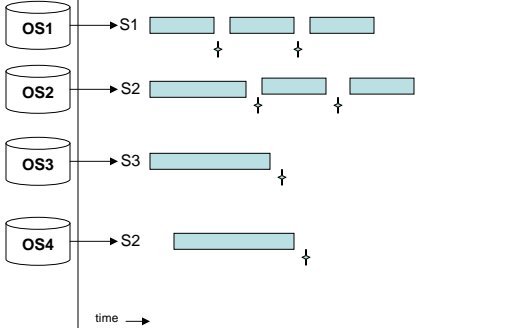
↓ = major metadata break

Copyright Jack Olson, 2010 11

 **SvalTech**

Application Segments

case 3 **Application: customer stock transactions**

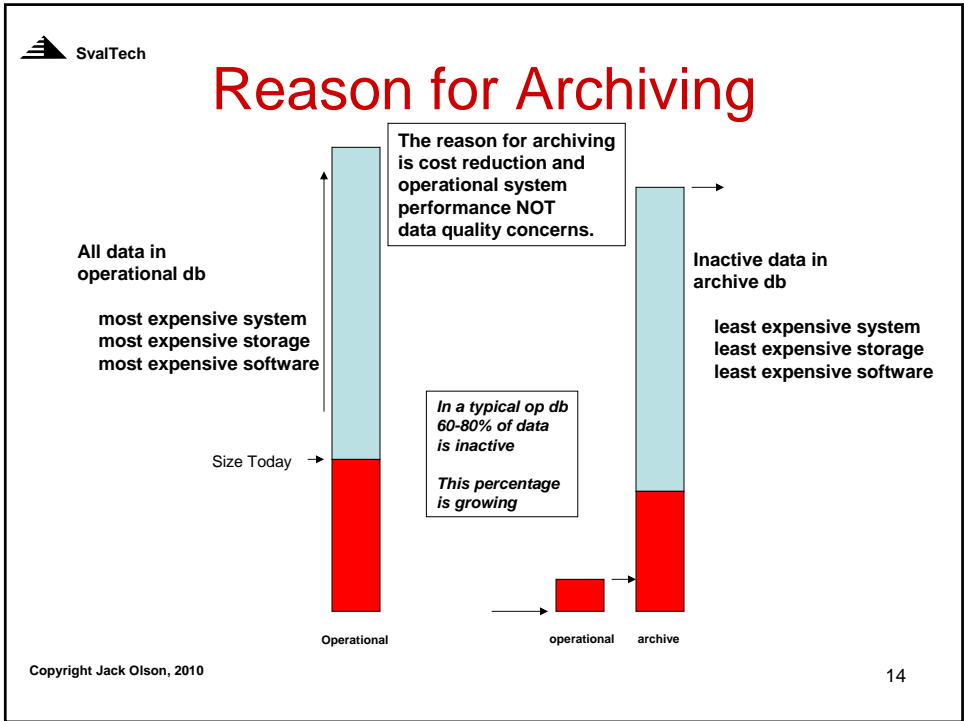
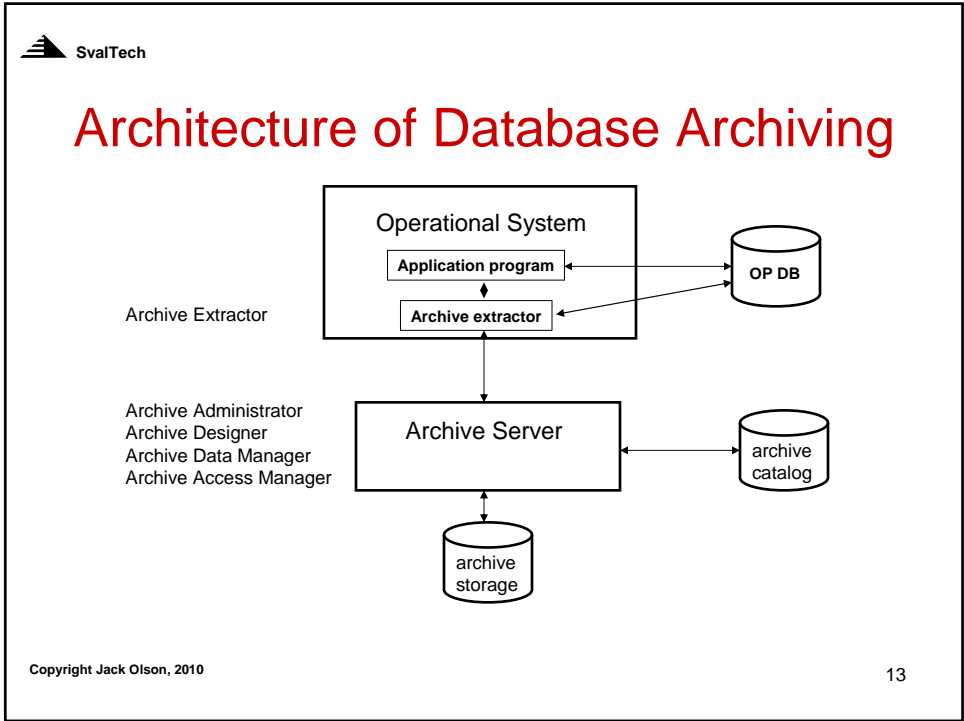


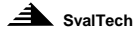
time →

Source 1 = Stock Trades – North American Division – application X
 Source 2 = Stock Trades – Western Europe Division – application Y
 Source 3 = acquisition of Trader Joe: merged with Source 1 on 7/15/2009
 Source 4 = acquisition of Trader Pete: merged with Source 1 on 8/15/2009

↓ = major metadata break

Copyright Jack Olson, 2010 12





Problems with Using a Single Operational Database Approach

Single Operational Database Approach:

Keeping business object data in the single active operational database until the retention period expires and then deleting it.

Operational database contains business objects in all 3 phases

Single objects may exist in the database for decades

This is the most common method for handling long term data retention requirements

Copyright Jack Olson, 2010

15




Root Causes of Problems





1. DBMS products used for operational databases support only ONE definition of a data record. There are no variations, no versions, only ONE definition.
2. Applications sometimes stop being used before the end of the retention periods for the data within the databases. Sometimes the time difference is decades. (**retired applications**)
3. Interpretation of data in a database depends on many factors other than the formal metadata available.

Copyright Jack Olson, 2010


16

 SvalTech

Problem 1: Loss of Clarity of Understanding








<p>Database Structure DDL DBD/PSB</p>		<p>How much do you depend on each of these areas for interpreting data that you see?</p> <p>How accurate are each of these?</p> <p>How complete are each of these?</p>
<p>External Metadata formal metadata repository auxiliary metadata repository copybooks,</p>		
<p>Application externalizations display windows reports</p>		
<p>Knowledge in Employee Heads IT workers business unit workers</p>		

Copyright Jack Olson, 2010 17

 SvalTech

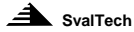
Problem 1: Loss of Clarity of Understanding

While still operational, clarity is maintained.
When application is retired, clarity begins to erode: rapidly

	while operational	after retired
Database Structure		 ☆
External Metadata		 ☆
Application externalizations		
Knowledge in Employee Heads		

☆ only if you remember to save

Copyright Jack Olson, 2010 18



Problem 2: Metadata Change Corruption

The problem with metadata changes is that

the DBMS only supports one version of data definition

which means that old data must be manipulated to conform to the new definition

which often results in data elements being missing or inconsistent

a future user of the data does not know which instances are good and which are not.

When the scope of data in a DBMS covers a short time period the corruption may be acceptable.

The cumulative effect of change corruption over many years can render old data instances highly inaccurate and misleading.



Problem 2: Metadata Change Corruption

Example 1:

Add a column to an existing table. All old rows have value "NULL" inserted for this column. (or worse yet, a single default value that is NOT NULL).

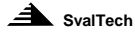
```
ALTER TABLE PERSONNEL ADD COLUMN MILITARY_SERVICE CHARACTER 10
```

10 years later an unknowing user does a query:

```
SELECT NAME FROM PERSONNEL WHERE MILITARY_SERVICE = "NAVY"
```

an answer is returned leaving the user to believe that they have everyone who served in the navy.

the true answer is unknown



Problem 2: Metadata Change Corruption

Example 2:

Increase the length of column COUNTRY from 10 bytes to 15

**This requires use of a special tool such as BMC's DB2 ALTER to execute.
All existing rows are padded with blanks.**

10 years later an unknowing user does a query:

SELECT SUPPLIER_NAME FROM SUPPLIERS WHERE COUNTRY = "SOUTH AFRICA"

an answer is returned leaving the user to believe that they have all supplier names operating in South Africa

the true answer is unknown since before the change any "South Africa" entries were either truncated or abbreviated and the user does not know this

Copyright Jack Olson, 2010

21



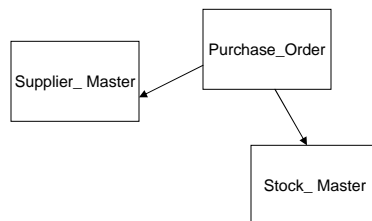
Problem 3: Reference Data Change Corruption

Reference information applies to a transaction as of the time the transaction took place.

Reference information may change over time

Single database solutions do not carry versions of reference information

Thus, years later the reference information may not reveal the truth about the transaction



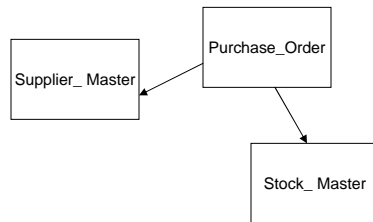
Copyright Jack Olson, 2010

22



Problem 3: Reference Data Change Corruption

- The supplier may change it's name
- The supplier may change its place of business
- The supplier may go out of business
- The supplier may be acquired by another supplier



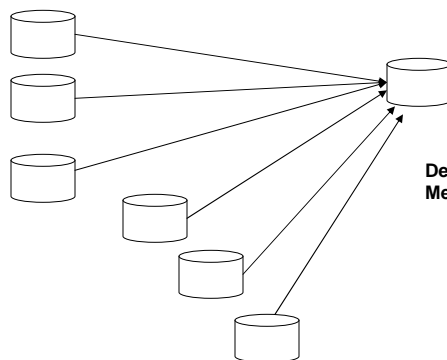
- The part may change its specifications
- The part may stop being used
- The part may change its handling rules
- The part number may be assigned to a different part

Copyright Jack Olson, 2010

23



Problem 4: Database Consolidation



Departmental System Consolidation
Merger and Acquisition Consolidation

Copyright Jack Olson, 2010

24



Problem 4: Database Consolidation

- **Data Corruption**
 - Columns that encode same or similar fact but differently
 - Encoding to a different level of granularity
 - DBMS recording differences
- **Data Fudging**
 - Columns in one database but not the other
- **Application abandonment**
 - Loss of clarity of understanding
- **Staff abandonment (layoffs)**

.....times n

The number of things that change are many times the changes seen by an application in the normal course of business.

And, it happens all at one time



Using Database Archiving for Improved Data Quality

or, at least if you do it right.

poorly designed archiving implementations can make it worse

robust implementations are needed to get the desired outcome

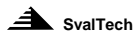


Why Database Archiving Improves Data Quality

- **Captures business record before changes corrupt it**
 - Transaction data
 - Reference data
- **Manages application segments**
 - Homogeneous metadata within application segments
 - Improved metadata within application segments
 - Audit trail of application segment histories
- **Manages access across metadata changes**
 - Only returns data that was original

Copyright Jack Olson, 2010

27



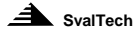
Education and Awareness

Knowledge of Database Archiving Concepts is essential for:

- **Building a database archive that will preserve data in pure state for long periods of time**
- **Designing new applications**
- **Designing database consolidations**
- **Designing changes to be imposed on existing databases**

Copyright Jack Olson, 2010

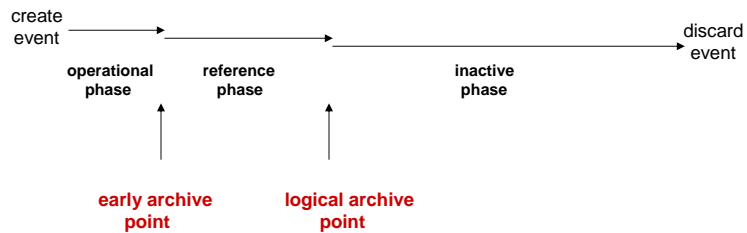
28



Early Business Record Capture

Records should be captured and moved to archive when last updates are made:

for a single instance of a data object

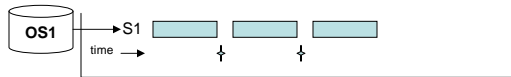


Copyright Jack Olson, 2010

29



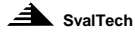
Data and Metadata by Application Segment



- Each application segment should contain
 - All data for archive segment
 - Metadata (should be invariant within segment)
 - Metadata of data source
 - Metadata of archived objects
 - Metadata changes from previous segment
 - Internal indexes
 - External indexes
 - Control information
 - When created
 - Time period data comes from (earliest to latest)
 - Policies used to create it
 - Discard policies

Copyright Jack Olson, 2010

30



Capture Extended Metadata

Application metadata is generally not enough to achieve application independence.

- **Validate metadata for accuracy**
- **Add commentary to fully explain**
 - **Business Record**
 - **Table dependency structures**
 - **Columns**
 - **Column data encoding**
- **Example reports with commented headings**
- **Application program to display business records**

Copyright Jack Olson, 2010

31



Freeze reference Data

Freeze for business records moved to archive at same time.

- **Capture current reference info at archive sweep time**
- **Check to see if ref data has changed within application segment**
- **Add version number column to ref data and to data**
- **Eliminate duplicates in application segment**

- **Check lookup tables for changes**
- **Create versions if different**
- **If only adds, use bigger table**

Copyright Jack Olson, 2010

32



Metadata Sensitive Data Access

Data Quality problems come about in result sets because:

- User does not know which segments to look in
- Access routines do not resolve metadata differences across segment boundaries
- Access routines do not alert query user of potentially missing data or data inconsistencies that might render the result set incomplete at best

Copyright Jack Olson, 2010

33



Access Logic that Should be Used

Query: Select

Based on search arguments, which segments will be needed to satisfy the request

If cannot determine then look at all segments

Who does this?

Will metadata changes between segments invalidate all or some of the answer?
Will it leave doubt about completeness of answer?

Such logic is not possible in operational databases.

Provide answer set for what is possible.

Provide meaningful feedback on segments not used and warnings on potentially compromised results.

Copyright Jack Olson, 2010

34



Final Thought

Failure to address long term data quality erosion issues can lead to archived data being lost, rendered unusable, or meaningless.

A poorly designed strategy can appear to be working smoothly for years while data quality is eroding daily.

When the need for the data arises the consequences of bad design can be costly and an embarrassment to the corporation.

Good design needs to encapsulate application initial design, design change rules, data archiving processes, and on-going data management oversight.



Some Quotes

“When we go back to the archived data we cross our fingers. So far we have been able to get what we want but it is always a major effort. Someday it won’t work.”

“God help us if we have to use that data again.”

**In answer to the question of where do you store your archived data:
“In black plastic garbage bags.”**