



Efficient Reliable Database Migration

A Legacy Innovation Story

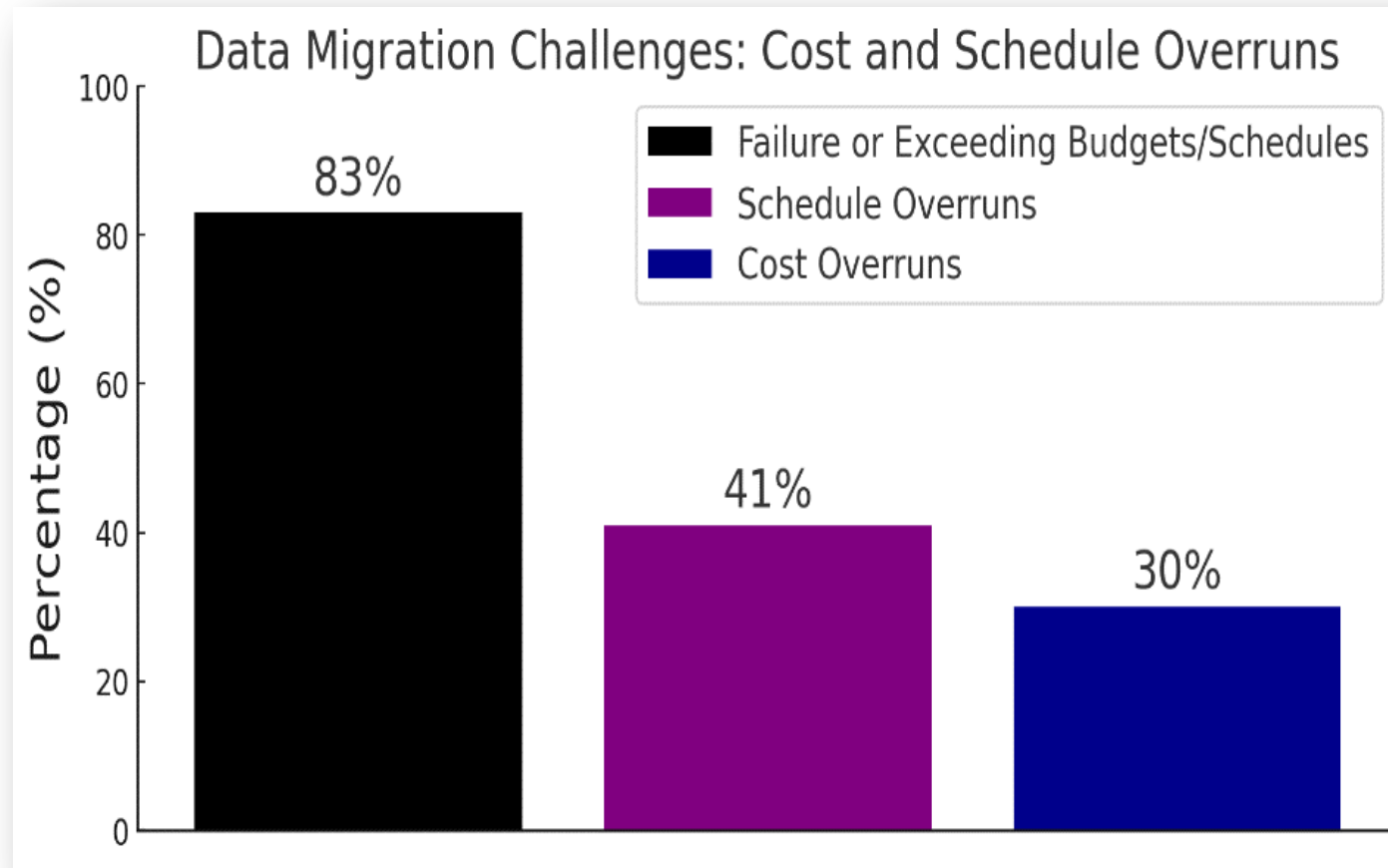
Dave Thomas

CEO Bedarra Corp, Carleton University Canada

 dave@bedarra.com

 www.davethomas.net

THE REALITY OF DATA MIGRATION



ERPrUS HAVE CRITICAL LEGACY PROBLEM!

Leading ERP product for large equipment companies.

Use a legacy network database (LNDB) - was state of art for efficient navigation for the ERP data.

Customers concerns

1. **Improved access to data** and
2. Escaping **lock in to LNDB** on **legacy platforms!**

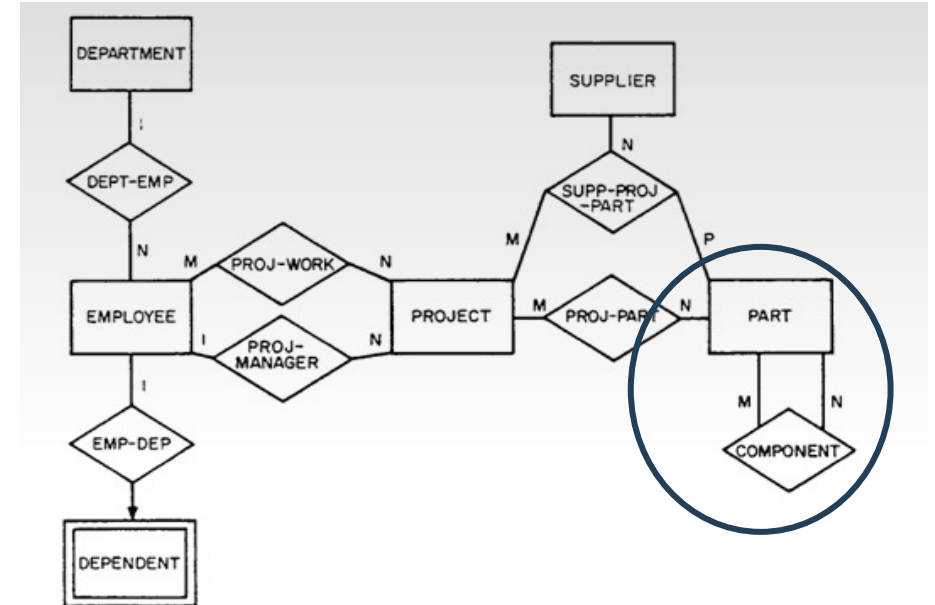


Fig. 11. An entity-relationship diagram for analysis of information in a manufacturing firm

THE COMPANY MAKES BEST EFFORTS TO BETTER REPORTING

- Introduced Agile for Custom reporting and engineering teams
 - Marginal improvements
- Try to develop a generic SQL like query language for LNDB
 - Far too complex and risky for ERP engineering





REPORTING SOLUTION – SQLify!

- **Replicate** LNDB to RDB for standard reporting
- **Implement OEM SQL interface** for critical complex reports
 - Client speaks ODBC
 - Custom server interface to LNDB

CUSTOMERS LIKE AND ADOPT NEAR-TERM SOLUTION



- **Reduced** reporting **backlog**
- **Improved** customer **satisfaction**
- **Enabled** limited SQL development for critical **reports and updates**

THE LONG-TERM MIGRATION CHALLENGE

- ERPrUS is the **only major LNDB customer**
 - LNDB vendor at risk **of going out of business!**
- **Hundreds** of large customers, each with **many on prem** databases!
- Customers demand **migration to standard relational** platform



CONVENTIONAL MIGRATION APPROACH



- Write queries* to retrieve all data in text and write to new SQL database

Challenges

- Difficult to write exhaustive queries and SQL updates
- Testing is very challenging
- Slow due to huge data volumes
- Migration needs to run at customer site
 - Can't handle customer extensions

* In some cases, one can use logs instead of writing queries but this is also much slower.

INNOVATIVE SOLUTION - INSPIRATIONS

- Physical Access => Speed and Simplicity
 - Binary + Batch => Performance
 - Sort Merge => Simplicity and Reliability
- The Humble Linker => Efficient
Serialization



LOGICAL AND PHYSICAL MODEL - LET'S GET PHYSICAL !

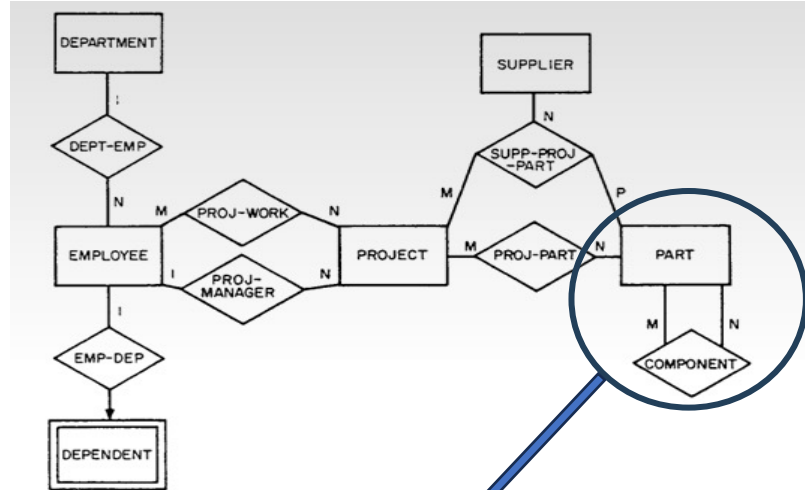
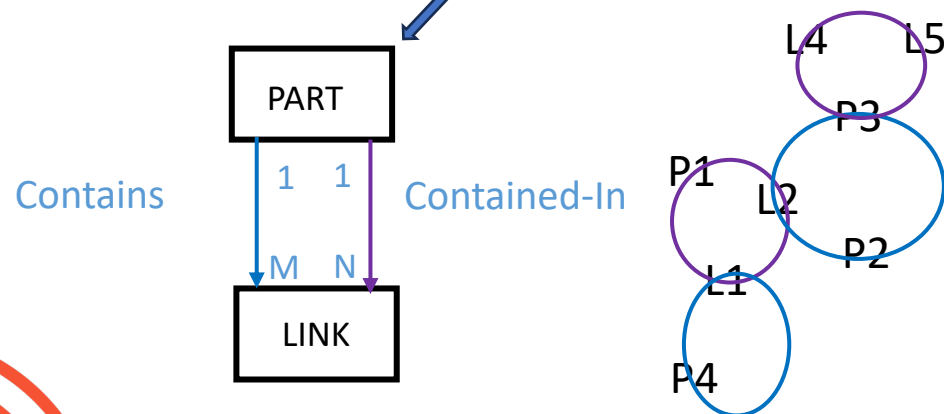
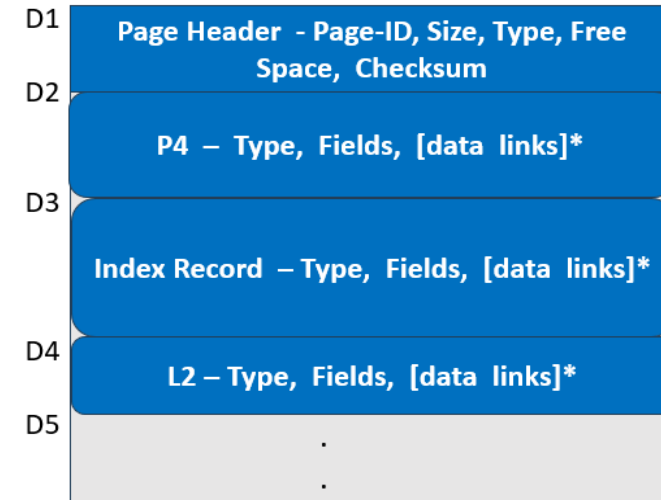


Fig. 11. An entity-relationship diagram for analysis information in a manufacturing firm



Aspect	LNDB	SQL Server
Database Model	Network	Relational (RDBMS)
Use Case	Designed for networked data models	Optimized for relational OLTP/OLAP
Data Access Model	Navigational, pointer-based traversal of network	SQL-based querying with relational model
Page Size	2, 4, 8K	8K Physical Page
Storage Organization	Pages containing records and pointers to related records (sets)	Pages containing table data, indexes, and metadata
Page Structure	Header, data pointers, record data, free space	Header, row data, slot array, free space
Indexing	Navigational, B-tree indexing	B-tree indexes



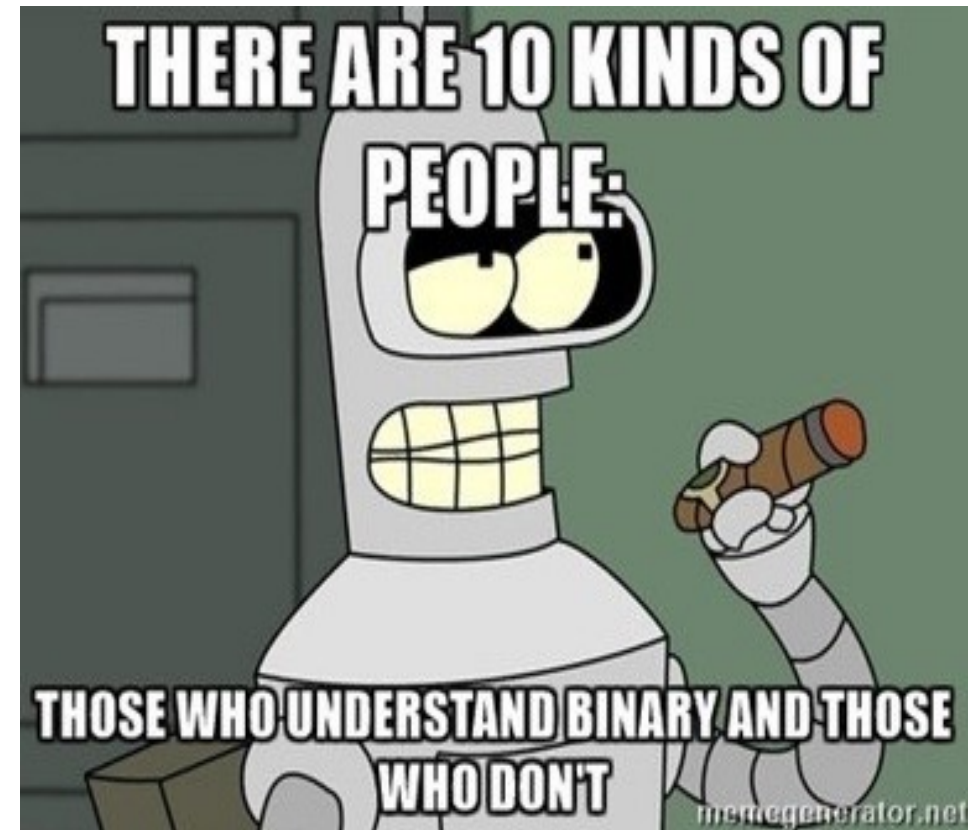
ESSENCE OF APPROACH

1. Migrate LNDB pages to RDB pages
2. Capture record mapping
 - LNDB, RDB types
 - LNDB, RDB disk addresses
3. Capture pointer mapping
 - LNDB, RDB types
 - LNDB, entity and index pointers
4. Sort -merge 2 & 3 to get RDB fixups and apply fixups. (join tables, indexes..)

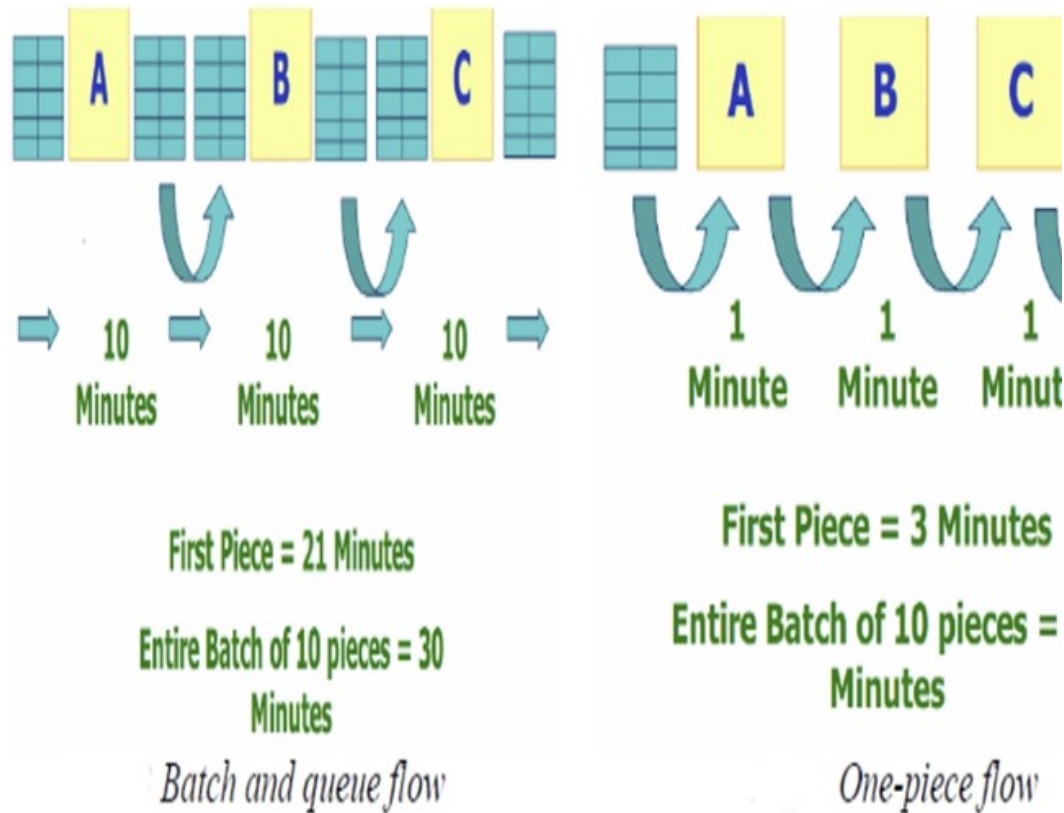


BINARY IS BETTER

- 3-5 times faster than JSON
- Uses (50-86%) less space
- Smaller faster for date/times, decimal ..

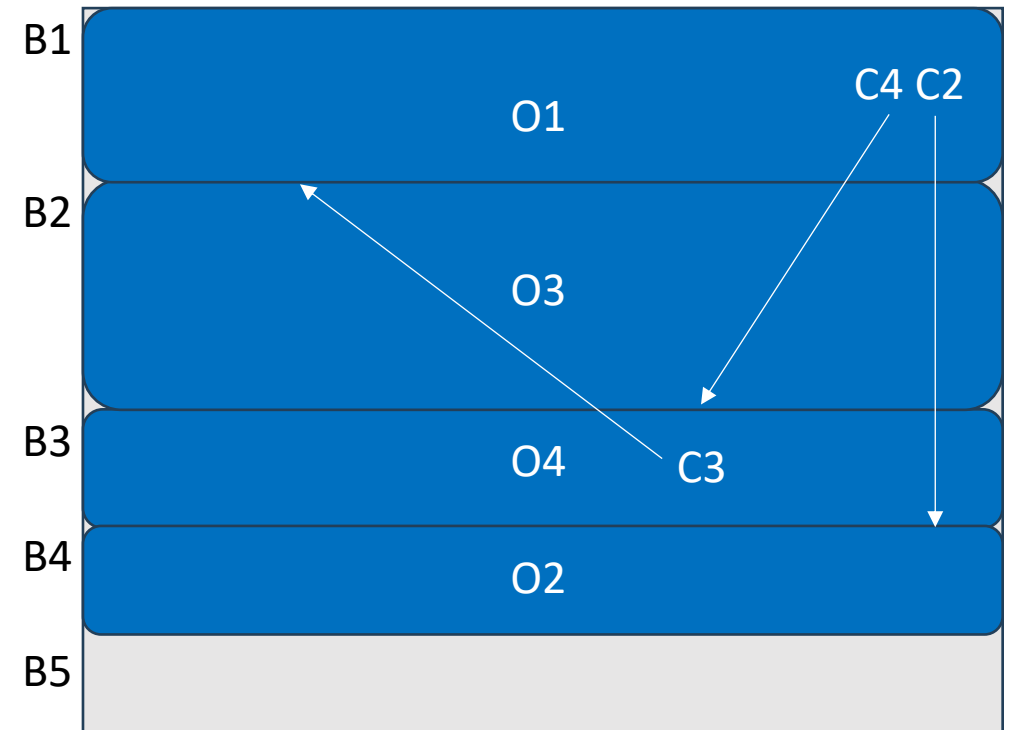


BATCH IS BETTER



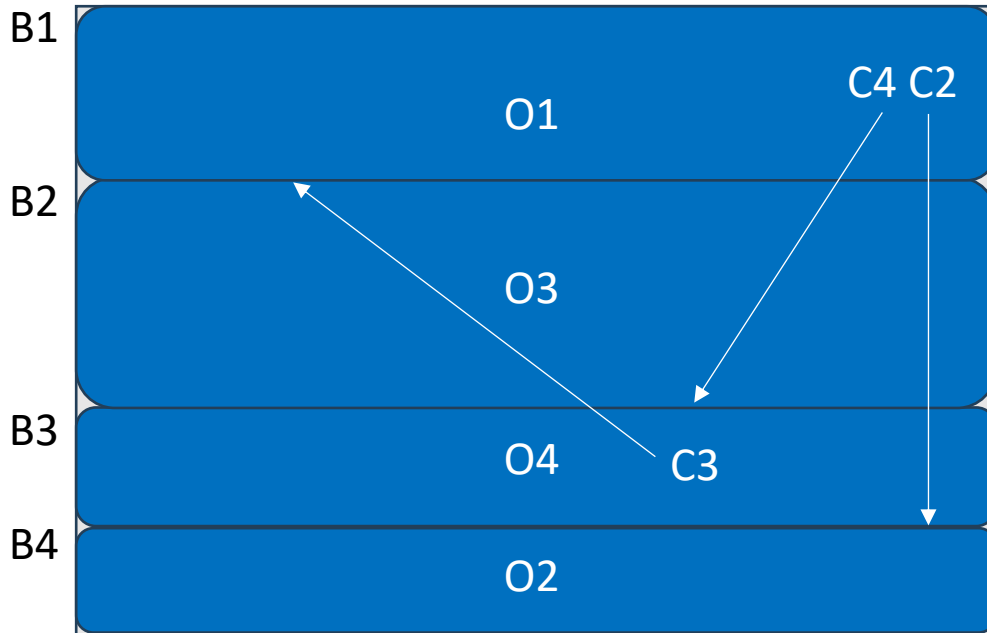
- Fast to read/write data in large sequential blocks
- Favors memory read ahead
- Compression happens in parallel to query/processing
- Batches of transactions reduce latency

EFFICIENT SERIALIZATION (SEND)

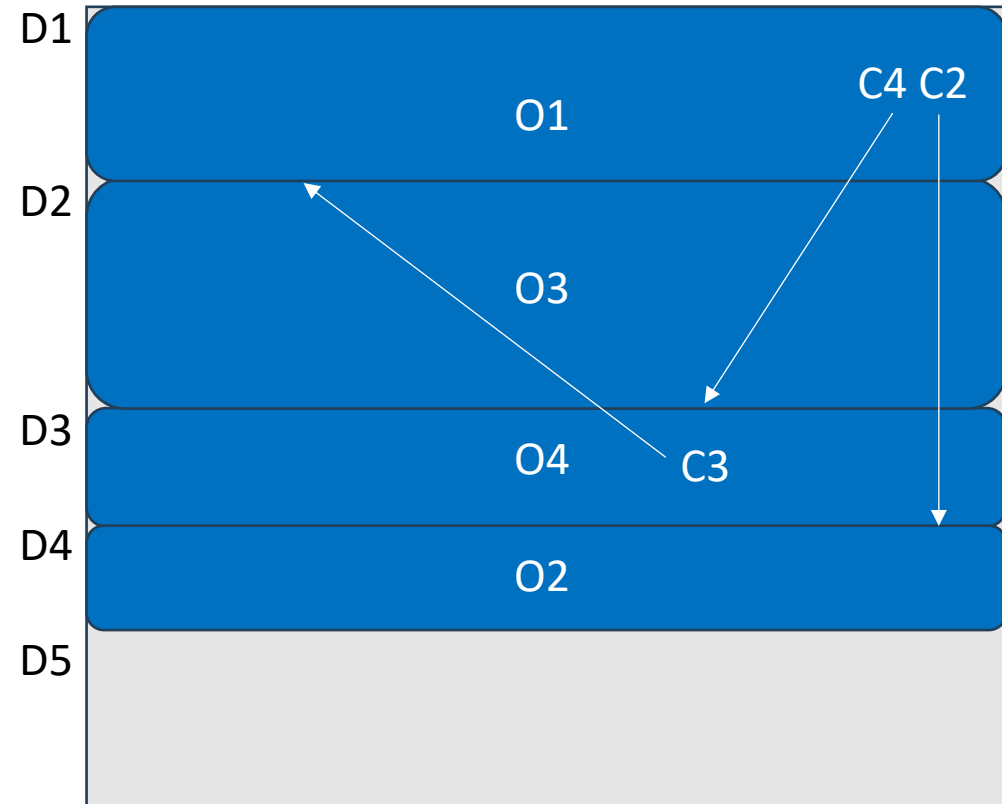


Where Cs are relative offsets from B1
 $C4 = B3 - B1$; $C2 = B4 - B1$; $C3 = B2 - B1$

EFFICIENT SERIALIZATION (RECEIVE)



Where Cs are relative offsets from B1
 $C4 = B3 - B1$; $C2 = B4 - B1$; $C3 = B2 - B1$



Where Ds are 64 bit addresses
 $C4 = B3 + D1$; $C2 = B4 + D1$; $C3 = B2 + D1$

CONSERVATIVE IMPLEMENTATION STRATEGY TO REDUCE RISK



- Algorithm **specified using decision tables** for straightforward verification and implementation
- **Prototyped quickly** validated using a **functional array** language
- Migration high-level **source code generated** from source and target schemas provides:
 - Efficient source/target execution using systems language on platform
 - Straightforward source code review
 - Enables custom data conversions
 - Bit, Byte Packed Fields, Decimal
 - Refactoring – add, remove fields, tables, indexes

Design Verification

Review

Conditions	KeyA	KeyB	A=B	Dups	Actions
A, B Same	Yes	Yes	Yes	No	
KeyA not in B	Yes	No	-	No	
KeyB not in A	No	Yes	-	No	
KeyA < KeyB	Yes	Yes	No	No	
KeyA > KeyB	Yes	Yes	No	No	
EOFA, not B	No	Yes	-	No	
EOFB, not A	Yes	No	-	No	
Dups in A or B	Yes	Yes	Yes	Yes	
Process Batch	Yes	Yes	-	No	
Error	-	-	-	-	

Prototype

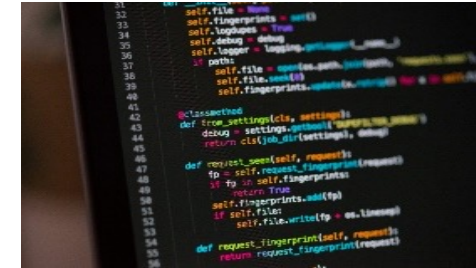
```
locate = [: (j,-/1) 4 $, [: $, -&0*
angles = [: # [: -, 1 {1} [: $, 0 -,- ]
visible = [: angles*1 [ -/- ]

partA = >./ counts =, visible ] asteroids =, locate grid

station =: asteroids {~ counts i. partA
phi =: 1&{1 @ *. @*80j.1
vaporize =: (198 i.- [: +/\ [: -: phi) { (/: phi)

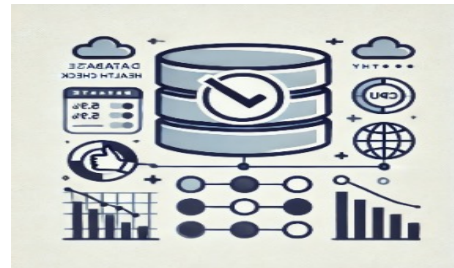
partB =: (vaporize &. (-&station)) asteroids -. station
```

Review

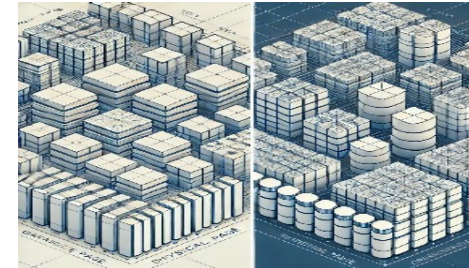


Implementation Testing

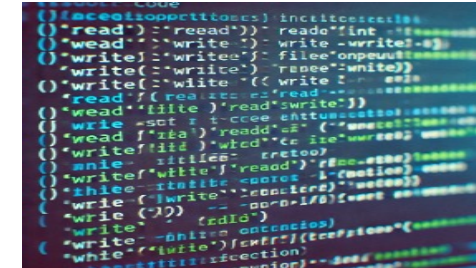
RxCheck



Test



Review



Migration Validation

Validation



READERS AND WRITERS?

It is great to be able to efficiently migrate the data to a modern platform, **but what about the code that uses the LNDB database?**



READERS AND WRITERS?

Approach

- Analysis of legacy code finds all LNDB IO operations and the effected data.

Solution (*No Silver Bullet**)

- Simple reads and writes are straight forward rewrite in SQL
- Complex traversal queries and updates are carefully reviewed and rewritten using SQL (potentially embedded cursor SQL)

*Complexity of automated conversion not justified





Innovative Approaches Can Solve Complex Legacy Challenges!

Thanks

Questions?

Concept Graph -> Knowledge Base
Properties in GraphDB-> Property DB