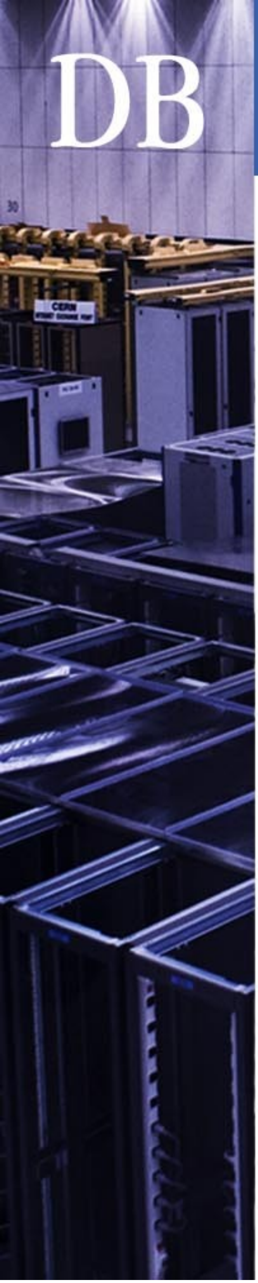


Data Lifecycle Activities

Distributed Database Operations Workshop
November 16th 2010

Marcin Blaszczyk, IT-DB
marcin.blaszczyk@cern.ch

Agenda

Outline

- Data/Information Lifecyle Management
- Motivations for DLM / ILM
- Techniques
- Examples
- Conclusions

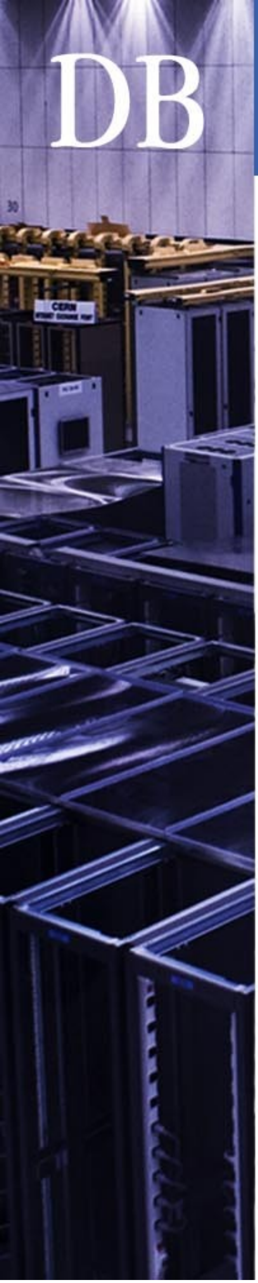
Data Lifecycle Management

„Information Lifecycle Management (ILM) is concerned with everything that happens to data during its lifetime„

„Data life cycle management (DLM) is a policy-based approach to managing the flow of an information system's data throughout its life cycle"

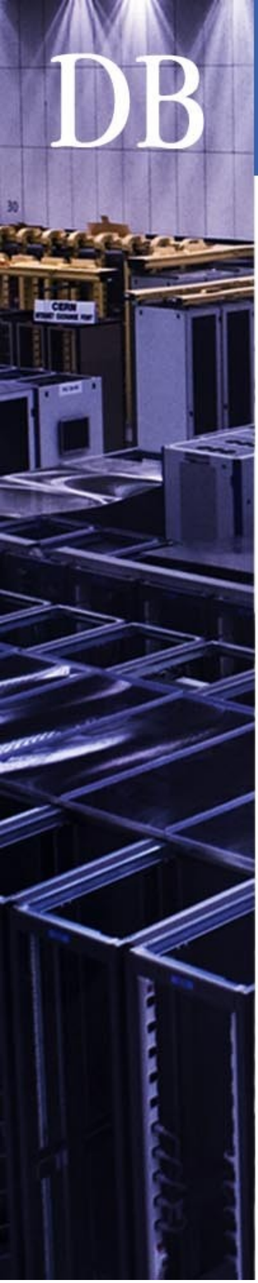
Data Lifecycle Management

- Main challenge:
  - Understand how data evolves
  - Determine how it grows
  - Monitor how its usage change
  - Decide how long it should survive
- General data flow model:



Motivations for DLM

- Why Data Lifecycle Management?
  - Large amounts of data collected and stored for several years
  - Different requirements on performance and SLA can often be found for ‘current’ and ‘old’ data sets
  - To ease maintanace and reduce potential risk of managing very large data sets in the area of:
    - Administration
    - Performance
    - Cost
  - To keep track/identify no longer needed or redundant data



Motivations for DLM

- Data Life Cycle policies cannot be easily implemented from the DBA side only
  - Not all applications can be ‘optimized’ to fit into DLM policy
  - require data model change and application modification
  - It’s ongoing effort
- Close collaboration with application developers and application owners is essential:
  - To reduce amount of data produced
  - To allow DB structure for implementing archiving
  - To define data availability agreements for online data and archive
  - To identify how to leverage Oracle features

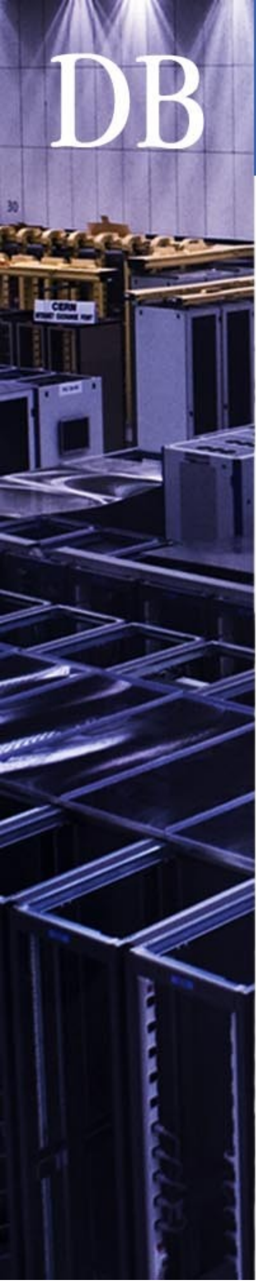
Techniques

No 'out of the box' solution available

- Attack the problem where possible
  - Applications
  - Oracle and DB features
  - HW architecture
- Application layer:
  - Focus on discussing with developers
  - Build life cycle concepts in the applications
- Oracle layer
  - Leverage partitioning and compression
  - Movement of data to an external 'archival DB'

Active Dataset

- Many Physics applications are structured as write-once read-many
  - At a given time typically only a subset of data is actively used
  - Natural optimization: having large amounts of data that are set read only
  - Can be used to simplify administration
  - Replication and backup can profit too



Time Organized Data

- Several key database tables are naturally time organized
  - This leads to range-based partitioning
  - Other solution is 'manual split' i.e. multiple similar tables in different schemas
- Advantages
  - Partitions can be treated as separate tables for bulk operations
  - Full scan operation, if they happen, do not span all tables

Oracle partitioning

- Range partitioning on timestamp attributes
- unique indexes and local partitioning
  - Indexes need to be partitioned locally
  - Partitioning key must be part of index
- Partitions for 'future time ranges'
  - Currently pre-allocated
  - 11g new feature interval partitioning
- Performance benefits:
  - Partition prunning
  - Local indexes
  - table/index partition is separate segment

‘Manual’ Partitioning

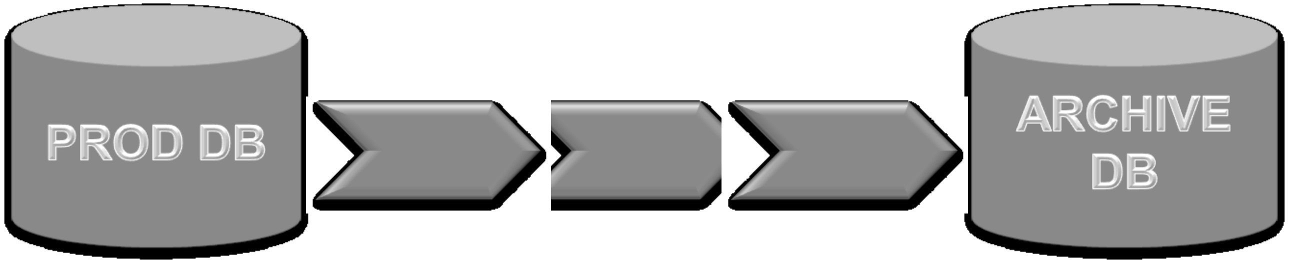
- Range partitioning obtained by creating multiple schemas and sets of tables
  - Flexible, does not require partitioning option
    - And is not subject to partitioning limitations
  - More work goes into the application layer
    - Application needs to keep track of ‘catalog’ of partitions
- CERN Production examples
  - PVSS (commercial SCADA system)
  - COMPASS (custom development at CERN)

Compression

- Compressing segments in Oracle 10g
  - For non-active (read-mostly) data
  - Can Save disk space but
    - Compression factor depends on data
  - Compressed segments need less blocks so:
    - Less physical IO required for full scan
    - Less logical IO space occupied in buffer cache
    - Beware compressed segments consume more CPU
  - Additional cost associated with DML operations on compressed tables

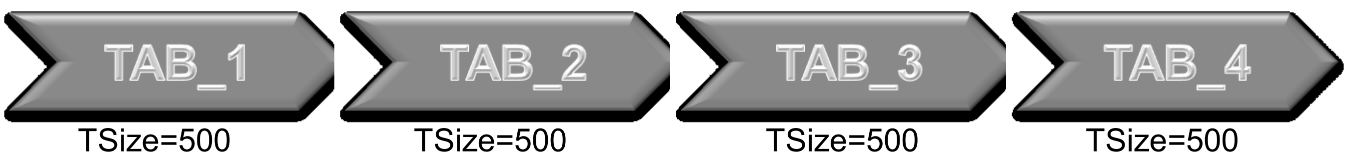
Archive DB

- In production since the end of 2009
  - It's an additional DB service to archive pieces of applications
  - Data in archive DB mainly for read-only workload (with a lower performance)
  - Archive DB is sized for capacity instead of IOPS
  - Reduces impact of production DB growth
  - Less critical for HA than production



PVSS for ATLAS

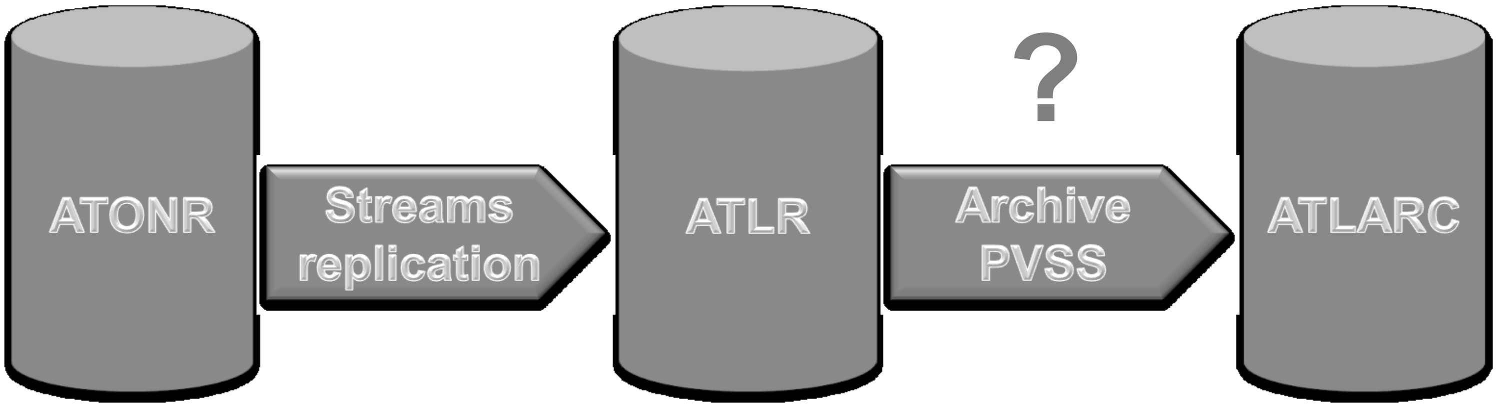
- Example of ‘manual’ partitioning
  - PVSS is consisted of set of schemas
  - Each schema has a set of ‘event tables’ created on regular basis defined by size of a table or time range
    - Partitioning by size



    - Partitioning by time

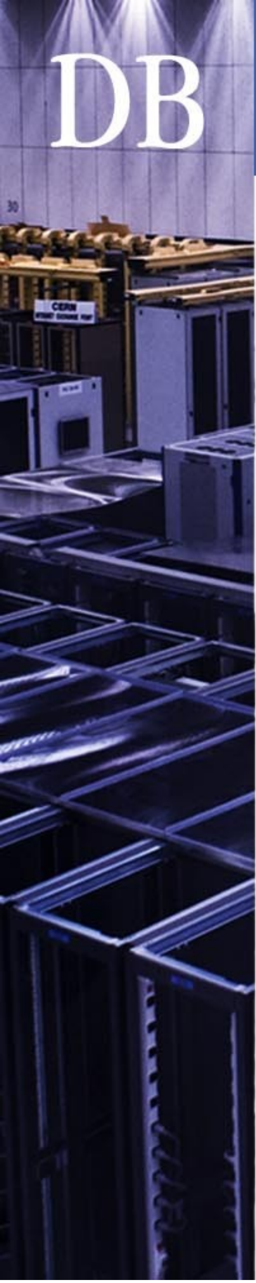
PVSS for ATLAS (2)

- Online needs: one year sliding window
  - Drop PVSS data older than 1 year from ONLINE
  - Do not replicate drop to OFFLINE
- Offline:
  - One tablespace per each PVSS schema per year
  - Current size (Atlas offline): 5.75 TB
  - Possible move old data to ATLARC in the future.



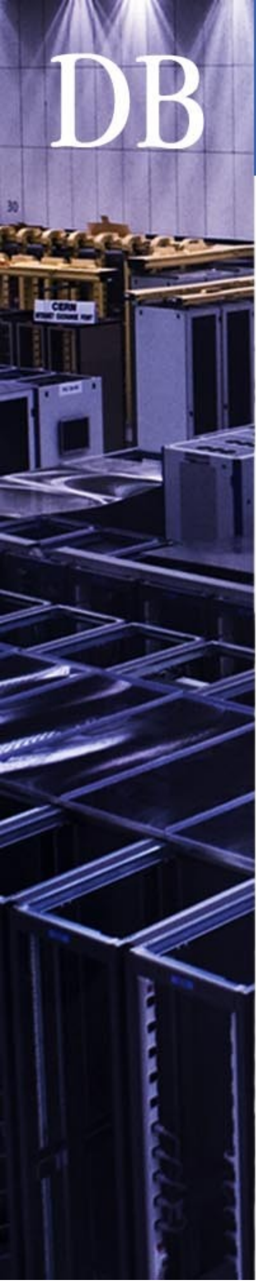
PANDA Archive

- 'Jobsarchived' table consolidates many smaller tables previously in MySQL
  - Historical data coming from production
  - Oracle range partitioning by time
  - Since November 2010 one partition per 3 days of data (instead of monthly partitions)
  - One tablespace per year



PANDA Archive (2)

- Lessons learned and techniques:
  - partition pruning vs. index access
  - smaller partitions (2-3 days) to profit from partition pruning
  - all indexes are local
  - application modifications to add time range in all queries



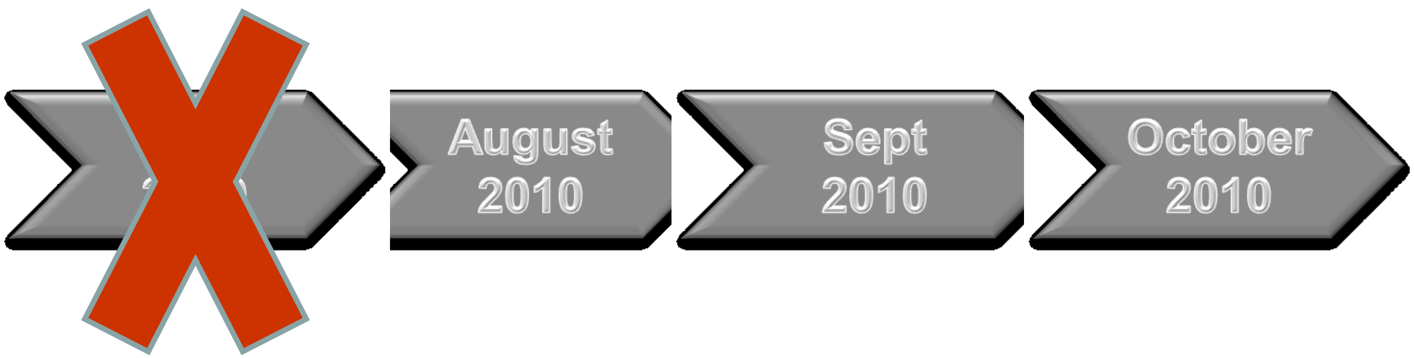
Examples

LCG SAME and GRIDVIEW

- Critical tables are partitioned
  - Range partitioned using timestamp
  - 1 partition per month
  - Contain live and historical data
  - All indexes are local
- Current size
  - 2.1 TB for LCG_SAME and 2.2 TB for GRIDVIEW

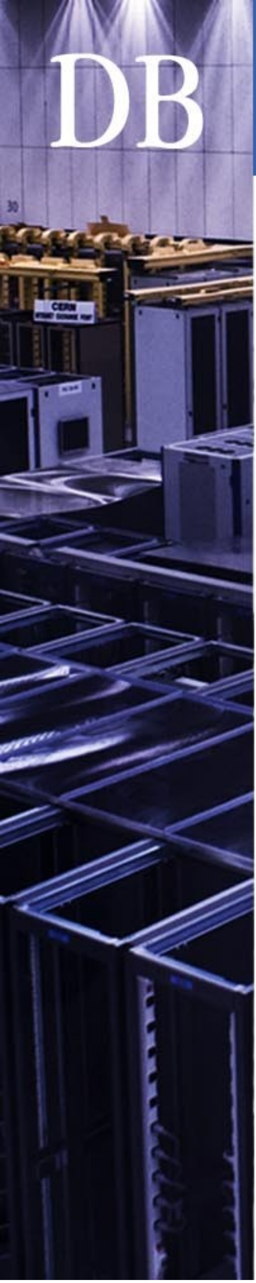
LCG SAME and GRIDVIEW (2)

- Delete old partitions:
  - Every month
  - Partitions older than 3 months



Archive DBs

- Archive DBs run on more powerful hardware since 2010
- High capacity disks (60-100TB of raw space)
- Quadcore servers with 24GB of memory
- Currently being used in ATLAS and CMS for:
  - Snapshots of CMS conditions data
  - ATLAS TAGs data
  - PVSS & COOL 'archived'
  - … ?



Conclusions

- Data Life Cycle Management is worth the effort:
  - Proactively address issues of growing DBs
  - Manageability
  - Performance
  - Cost
- Understanding of data flow, requirements and hardware limitations is a key to success
- Involvement of application owners is fundamental
- Successfully implemented by several applications
- Techniques within Oracle that can help
  - Partitioning
  - Archival DB service
  - Compression

Acknowledgements

- *Luca Canali*
- *Jacek Wojcieszuk*
- *Gancho Dimitrov*
- *Dawid Wojcik*
- *Florbela Tique Aires Viegas*

http://phydb.web.cern.ch/phydb/

Q&A

Thank You!

Questions?

phydb.support@cern.ch

marcin.blaszczyk@cern.ch