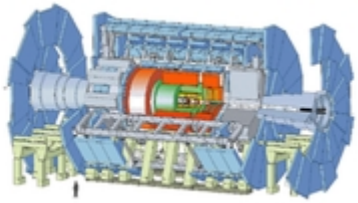
the ATLAS Experiment



Enhance the ATLAS database applications by using the new Oracle 11g features

Gancho Dimitrov

Outline

- The ATLAS databases based on the Oracle backend
- Some interesting facts about the ATLAS databases content at CERN
- New Oracle database 11g features for development that I consider applicable for ATLAS
- Conclusions

The ATLAS databases

At the T0 site

- ATONR - database for the ATLAS detector online datataking
- ATLR - database for the offline (post datataking) analysis
- ADCR - database for the grid jobs and files distribution managent
- ATLARC - database for the ATLAS event metadata (TAGS) and also a place for archiving the old data from all database accounts

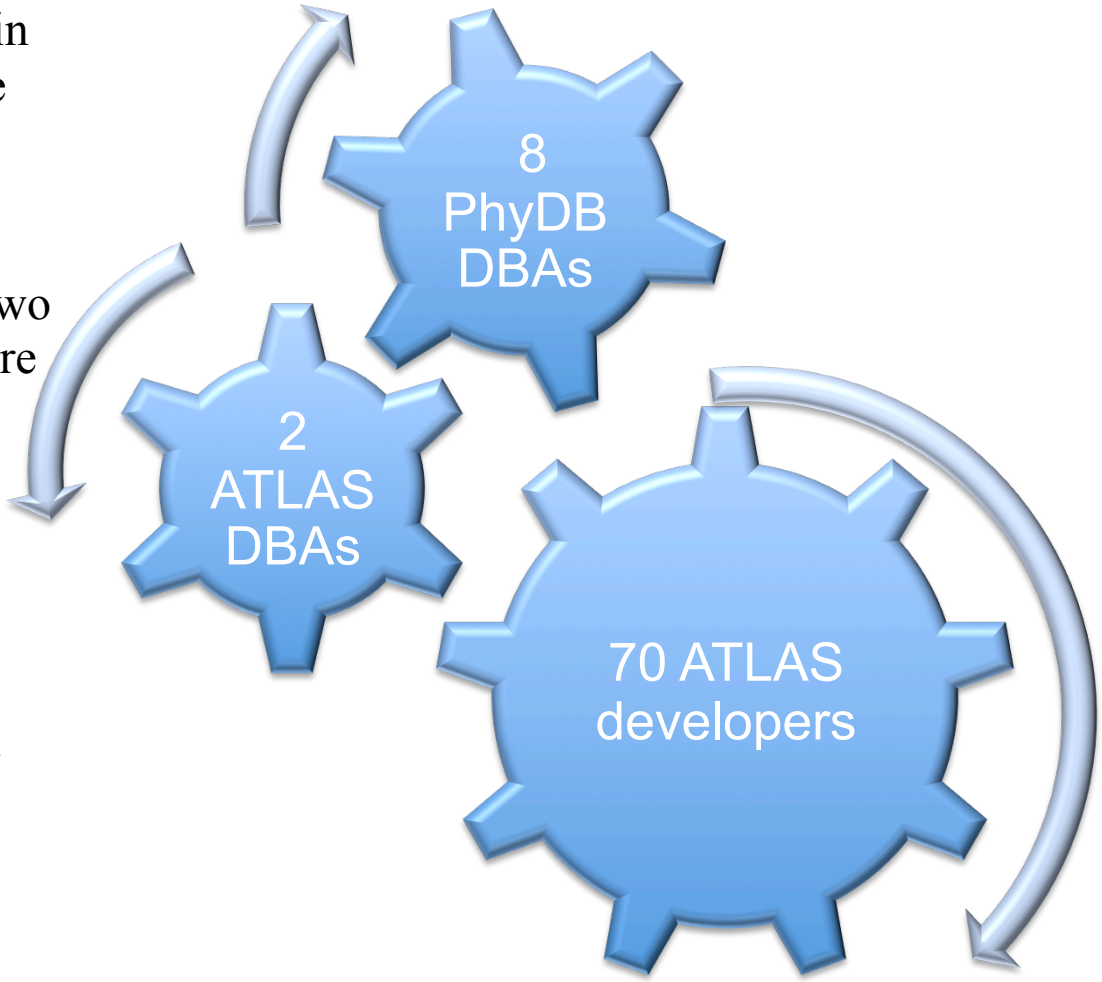
Interaction database admininstrators <=> developers

- The database developers community in ATLAS is a large one. About 70 people are responsible for different database applications.

- To help them, ATLAS has a team of two DB application administrators with more privileges on the databases level.

- The mentioned above people rely on the database service provided from the PhyDB DBAs team.

- And for having a success, we all need to work interactively



The ATLAS databases at CERN (facts)

Database / Statistic	ATONR	ATLR	ADCR	ATLARC
# DB schemes	69	151	10	49 (incl 23 arch)
# tables - Non partitioned - Partitioned	 37045 384	 51019 665	 364 21	 8209 5595
Volume - Table segments - Index segments - LOB segments	 2,14 TB 2,15 TB 0,3 TB	 4 TB 5,2 TB 0,4 TB	 4,1 TB 2,1 TB (incl. IOTs) 0,3 TB	 3,1 TB 7,5 TB -
Largest table in # rows	1 billion rows in a range partitioned table (65 GB)	1,2 billion rows in non-partitioned table (18 GB)	2,1 billion rows in a range partitioned table (650 GB)	160 million rows in partitioned table (20 GB) Note: the data is compressed

Toward the newest Oracle DB version …

- The ATLAS databases need to support different type of applications and access patterns and they do it well, but still there are many annoying issues that need to be addresed.

  Some of them are :

  - Unaccurate or stale statistics on the tables cause non-optimal and resource wasting access plans (some of the large tables cannot get even analized)

  - Sudden and undesirable execution plan changes

  - Lack of intrumentation in changes validation against a real production workload on a test system

- I hope we really be happier with 11g and in the next slides will mention some of the 11g features that sound promising

(1) Object statistics gathering

1. Incremental for the partitioned tables

In 10g we have problems with statistics gathering for large partitioned tables. We had to lock the stats for some of them as practically the process was never ending (MDT_DCS, ProdSys and DQ2 tables)

2. Configurable stale percent threshold

In 10g it is fixed to 10 % and this causes for the large tables, the stats gathering kick in to happen more and more saldom (PanDa, DQ2, PVSS and other large systems)

3. Multicolumn statistics: Oracle will be aware of the logical relation on a set of coulumns.

In 10g the lack of such was causing not optimal data set joins to take place

4. Statistics can be published manually after an ‘approval’ from the DBA.

(1) Object statistics gathering (cont.)

5. Statistics on expressions

For example on the UPPER (dataset_name) from the DQ2 system

6. The issue with stale statistics for the max value on columns of type DATE or TIMESTAMP is real problem for which there seem to be not a solution in 11g.

Workaround: In ATLAS we will stick to our home made solution on updating the relevent statistics values regularly via a daily job.

(2) SQL plan management

1. Bind-aware peeking

Oracle may construct and sustain several execution plans that reflect the bind values

In 10g it was a single plan which in many cases was not appropriate, because the data in is not evenly distributed (a lot of problems with the DQ2 system)

2. SQL plan baselines (the idea is to avoid SQL plan degradation)

I do not believe that we will set automatic capture, evaluation and evolving of the plans (the default setting in 'manual') as this sounds like bringing more instability to the system.

For some particular use cases we could use this new feature for manually fixing the execution plan (with other words, forbid plan evolving for good)

(3) Automatic partitions creation

1. Automatic partition creation for tables with range based partitioning on columns of type DATE/TIMESTAMP or NUMBER

In 10g, the partitions had to be pre-created from the developer and had to keep in mind to create new ones with the time

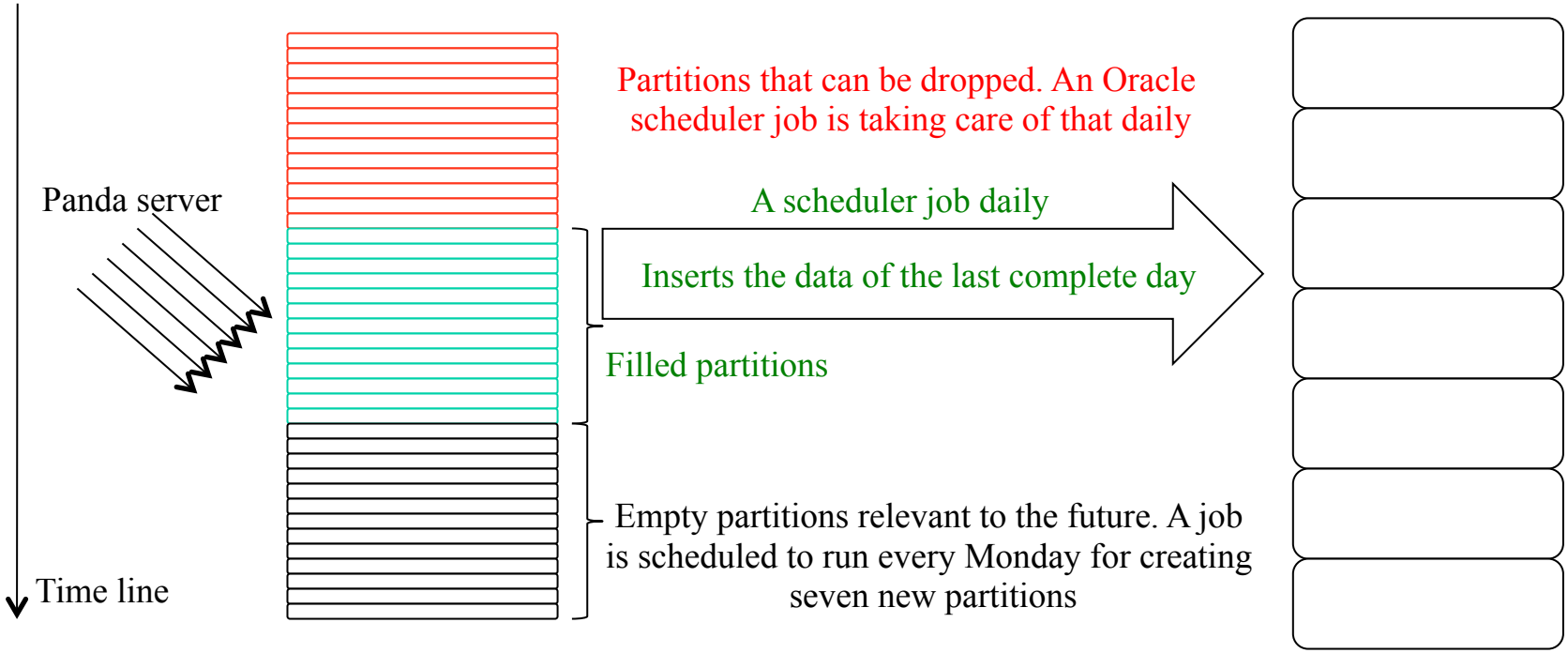
(the MDT_DCS, PanDA, ProdSys, Dq2 and other systems)

Currently for the PanDa schema, the sliding window with daily partitions is home made solution of mine using DBMS_SCHEDULER jobs (details in the next slide)

The PanDA (jobs submission system) 'live' and 'archive' data

ATLAS_PANDA.JOBSARCHIVED4 table:
partitioned on a 'modificationtime' column.
Each partition covers a time range of a day
(the same is applied for other tables as well)

ATLAS_PANDAARCH.JOBSARCHIVED:
range partitioned based on the
'modificationtime' column.



Important: For being sure that job information will be not erased without being copied in the PANDAARCH schema, a special verification PLSQL procedure is taking place before the partition drop event!

(4) Result set caching

1. Caching of the result set of queires, sub-queries and PL/SQL functions. A quite appealing new feature.

However I do not see it as being activated for all quieries, but rather per specific use cases. A particular interest could for the DQ2, PanDa and TAGs systems

As you know, the best metric to consider when tuning queries, is the number of Oracle block reads. Queries for which result has been gotten from the result cache, would show ‘block reads = 0’ ☺

(5) Less space wastage

1. Disk space can be re-claimed by setting index partitions as unusable

Now in the production we have couple of use cases with partitioned indexes on DATE or TIMESTAMP columns that are there only to satisfy searches in the most recent rows.

By setting the older partitions as unusable, Oracle scratches the relevent segment and frees the space in the tablespace

2. Data compression.

Can be activated for the normal (conventional) insertion (OLTP). Would be benficial for systems as PVSS, PanDA, DQ2,, Alert log messing systems and few others …

The advantage is not just saving space, but the real one is saving IO by having fewer block read when quering the data.

In 10g, compression was only possible for bulk insert operation

(6) Flashback data archive

1. Complete history of the changes can be kept in an archive with a desired retention. Thus, the client can have a data snapshot representation from any given time in the past.

A good candidate for using this is the ATLAS authorship qualification list and probably DQ2 systems

(7) And few more …

1. Invisible indexes (‘invisible’ because the Optimizer does not see it)

The importance of an index (used or not used at all) can be verified by playing with its visibility from the Oracle Oprimizer

2. Read only tables

3. Virtual Columns (computed on a fly from a user defined expression)

4. Resuming suspended statements

Conclusions

- From developer's point of view there are couple of new features in the Oracle 11g that sound appealing.
- Plus, there is:
  - More flexibility
  - More instrumentation to intervene
  - More sophisticated monitoring and testing tools
- The next months would show us what from those we can really put in production in 2012.
- Would be good to know the plans of the other DB groups and experiments and the experience they have got so far with 11g!