

# ATLAS Detector Description Database

Vakho Tsulaia  
University of Pittsburgh

3D workshop, CERN  
14-Dec-2004

- The purpose and logical organization of the ATLAS Detector Description Database (DDDB)
- Physical architecture
- Tools for data access
- Database distribution issues
- Conclusions

## Logical organization of the data

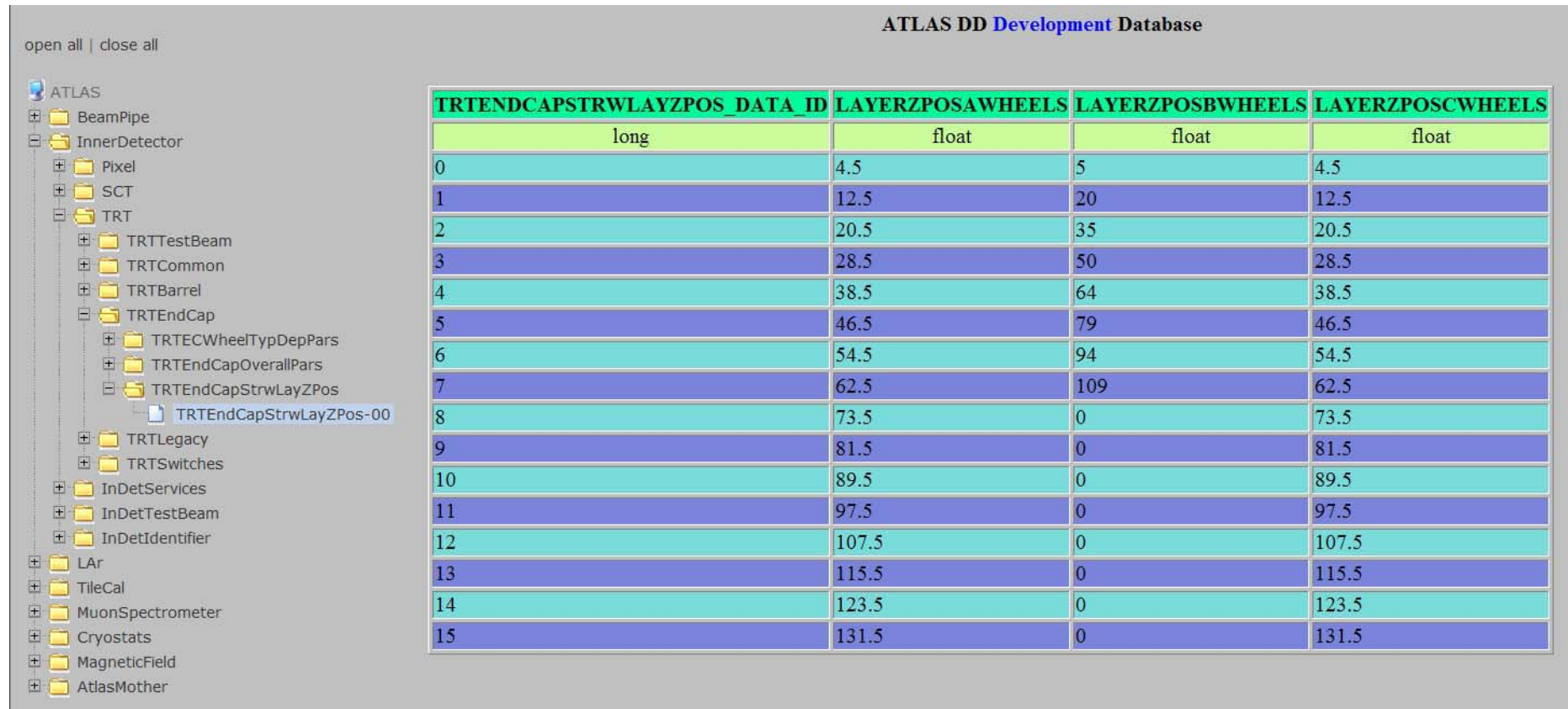
---

- The main purpose of the ATLAS DDDB is to store in one common place all primary numbers for ATLAS subsystem geometries together with the configuration information (tags, switches)
- The primary numbers for ATLAS Detector Description are organized in DDDB into a set of Data Tables
  - Each data table describes some particular piece of ATLAS geometry
  - ~150 Data Tables at the moment
- Data tables are logically grouped into HVS (Hierarchical Versioning System) node tree, where
  - **Leaf HVS Nodes** correspond to the Data Tables
  - **Branch HVS Nodes** are pure logical entities supposed to group child nodes and build the tree hierarchy

# DDDB Web browser (by node hierarchy)

open all | close all

ATLAS DD [Development Database](#)



The screenshot shows the ATLAS DD Development Database web browser. On the left is a tree view of the database hierarchy. The 'TRTEndCapStrwLayZPos-00' node is selected. On the right is a table with 4 columns: 'TRTENDCAPSTRWLAYZPOS\_DATA\_ID', 'LAYERZPOS AWHEELS', 'LAYERZPOS BWHEELS', and 'LAYERZPOS CWHEELS'. The table contains 16 rows of data, with the first row (ID 0) having a 'long' data type and the others having a 'float' data type.

TRTENDCAPSTRWLAYZPOS_DATA_ID	LAYERZPOS AWHEELS	LAYERZPOS BWHEELS	LAYERZPOS CWHEELS
0	4.5	5	4.5
1	12.5	20	12.5
2	20.5	35	20.5
3	28.5	50	28.5
4	38.5	64	38.5
5	46.5	79	46.5
6	54.5	94	54.5
7	62.5	109	62.5
8	73.5	0	73.5
9	81.5	0	81.5
10	89.5	0	89.5
11	97.5	0	97.5
12	107.5	0	107.5
13	115.5	0	115.5
14	123.5	0	123.5
15	131.5	0	131.5

# Logical organization of the data

---

- HVS nodes can be tagged
  - **Leaf HVS Node Tag** consists of a set of corresponding Data Table records
  - **Branch HVS Node Tag** consists of its children tags
- HVS tags can be locked
  - Tag is locked usually after successful validation of corresponding data records
  - The validation procedure involves the usage of the data by ATLAS detector description applications
- Tag locking means
  - All daughter HVS tags are locked recursively
  - Locked tags cannot be renamed or deleted
  - If a record in some data table corresponds to any locked tag, this record cannot be updated or deleted anymore
- In particular, locking the root ATLAS node tag closes the entire ATLAS geometry version

# DDDB Web browser (by tag hierarchy)

ATLAS DD Development Database

Set root node:

open all | close all

ATLAS Tags

- ATLAS-00
  - BeamPipe-00
  - Cryostats-00
  - InnerDetector-00
    - InDetServices-00
    - Pixel-00
    - SCT-00
    - TRT-DC2-00
      - TRTBarrel-00
      - TRTCommon-00
      - TRTEndCap-00
        - TRTECWheelTypDepPars-00
        - TRTEndCapOverallPars-00
        - TRTEndCapStrwLayZPos-00
      - TRTLegacy-00
      - TRTSwitches-02
      - InDetIdentifier-00
    - LAr-00
    - MagneticField-DC2-00
    - MuonSpectrometer-00
    - TileCal-00
    - AtlasMother-00
  - ATLAS-01

TRTENDCAPSTRWLAYZPOS_DATA_ID	LAYERZPOS AWHEELS	LAYERZPOS BWHEELS	LAYERZPOS CWHEELS
long	float	float	float
0	4.5	5	4.5
1	12.5	20	12.5
2	20.5	35	20.5
3	28.5	50	28.5
4	38.5	64	38.5
5	46.5	79	46.5
6	54.5	94	54.5
7	62.5	109	62.5
8	73.5	0	73.5
9	81.5	0	81.5
10	89.5	0	89.5
11	97.5	0	97.5
12	107.5	0	107.5
13	115.5	0	115.5
14	123.5	0	123.5
15	131.5	0	131.5

## Physical architecture - location

---

- Presently there are two Oracle accounts serving the ATLAS DDDB at CERN
  - **Development (DEVDB)**. Is used to enter and validate new data
  - **Production (PDB)**. Starting from recently is default account for the ATLAS Detector Description applications. Contains just a subset of DEVDB account data (locked tags)
- The main reasons for supporting two different accounts are:
  - It is foreseen to replicate DDDB to remote Oracle servers and also transfer its contents to MySQL
  - The strategy for these activities is not yet clear
  - ... so we have to worry about consistency of distributed data
- The present situation with these two accounts is going to be revised in the near future

## Physical architecture - user roles

---

- We have introduced three user accounts for the ATLAS DDDB (both DEVDB and PDB)
- **Administrator** account (DEVDB, PDB). The owner of the database with all privileges
- **Writer** account with SELECT and INSERT privileges
  - **DEVDB**. Is used directly by the ATLAS subsystem responsible users to put new data into database
  - **PDB**. Should be used by the data publishing tools only to transfer the data corresponding to some locked tag from DEVDB to PDB
- **Reader** account (DEVDB, PDB). Only SELECT privilege is granted. Is used by the ATLAS Detector Description applications and also by the Web Browser for read-only data access



- Write:
  - Schema modifications (adding new data tables), by database administrators only
  - Filling the contents of data tables through direct usage of very simple SQL scripts
  - Configuration management tasks (node tagging, tag collecting, tag locking etc.) through interactive PHP-based web tool
- Read:
  - Reading of primary numbers by ATHENA-based applications using a dedicated ATHENA service
  - PHP-based web browser
- Data publishing from the development to the production account
  - Using a dedicated utility

We have developed a PHP-based interactive web tool offering various functionalities for HVS node management

Example 1. Leaf HVS node tag collecting

Node: **DUMMYPARAMS**

Tag: **DummyParams-00**

	DUMMYPARAMS_DATA_ID	XPOS	YPOS	DESCRIPTION
COLLECT	long	double	double	string
<input checked="" type="checkbox"/>	0	11.1	22.2	First record
<input checked="" type="checkbox"/>	1	33.3	44.4	Second record

Example 2. Branch HVS node tag collecting and locking

Node: **InnerDetector**

Tag: **InnerDetector-DC1-00**

Child tags:

Pixel	Pixel-DC1-00	<a href="#">UPDATE</a>
SCT	SCT-DC1-00	<a href="#">UPDATE</a>
TRT	TRT-DC1-Final-00	<a href="#">UPDATE</a>
InDetServices	InDetServices-00	<a href="#">UPDATE</a>
InDetIdentifier	InDetIdentifier-02	<a href="#">UPDATE</a>
InDetTestBeam	--	<a href="#">UPDATE</a>

Node: **InnerDetector**

Tag: **InnerDetector-DC1-00 UNLOCKED**

Child tags:

Pixel	Pixel-DC1-00	<b>UNLOCKED</b>
SCT	SCT-DC1-00	<b>UNLOCKED</b>
TRT	TRT-DC1-Final-00	<b>UNLOCKED</b>
InDetServices	InDetServices-00	<b>LOCKED</b>
InDetIdentifier	InDetIdentifier-02	<b>UNLOCKED</b>
InDetTestBeam	--	--

- Primary numbers and configuration switches in DDDB are presently validated by building geometries of various ATLAS subsystems and using them in Simulation/Reconstruction
- The primary numbers in DDDB are accessed by ATHENA applications through RDBAccessSvc
  - The service was developed based on POOL Relation Access Layer, which provides a common interface to the data in different Relational Database Management Systems (RDBMS)
  - The concrete RDBMS is chosen at run time by loading the appropriate plug-in
- The primary numbers in all DDDB Data Tables are presented to clients of RDBAccessSvc in uniform way through Recordset objects
  - Recordset is a snapshot of data table records corresponding to the given tag
  - The records can be retrieved from Recordset by index or using the iterator
  - The actual primary numbers are retrieved from records by field names

## General strategy for data publishing on PDB

---

- The main principle: **Only the data corresponding to locked tags should be published on the PDB side** using the dedicated tool
- We have developed a first version of the data publishing tool based on POOL RAL
  - Transfers locked tags from DEVDB to PDB
  - Uses transactions
  - Still needs some improvements, especially for configurability
  - Using this program we have successfully published two ATLAS tags on PDB
- Any subsequent updates of the PDB database content should not affect the existing data, just new locked tags will be added
- PDB database should be the default storage of DD primary numbers for ATHENA applications, providing just the read-only access

- The Production DB account should be used as a single source for any further replication of DDDDB contents to remote Oracle servers and for translation to MySQL
- The strategy and tools for these activities have not been chosen yet. We foresee two possible scenarios:
  1. **Usage of a generic replication tool.** (For example: Octopus replicator)
    - Using Octopus we have successfully transferred present contents of PDB account to MySQL and have built ATLAS geometry based on MySQL
    - We need too keep the present situation with two Oracle accounts if Octopus is chosen
  2. **Usage of a HVS-aware tool.** (For example: our data publishing tool)
    - Using this tool we could not make Oracle-to-MySQL transfer because of problems with RAL MySQL plug-in related to data inserts
    - This bug was fixed in the last internal release of POOL so we can try it now
    - If a HVS-aware tool is chosen we can merge two Oracle accounts into one (on PDB)

- The ATLAS Detector Description Database has already become an essential part of ATLAS DD applications, in particular the ATLAS Geometry Versioning System
- DDDDB configuration management and publishing tools have already been used quite effectively, they need some further developments though
- We are ready to distribute the database following the distribution strategy which has to be decided by the ATLAS database management team