

FairRoot Virtual Database

Status Report

Denis Bertini

GSI - Scientific Computing

December 9, 2013

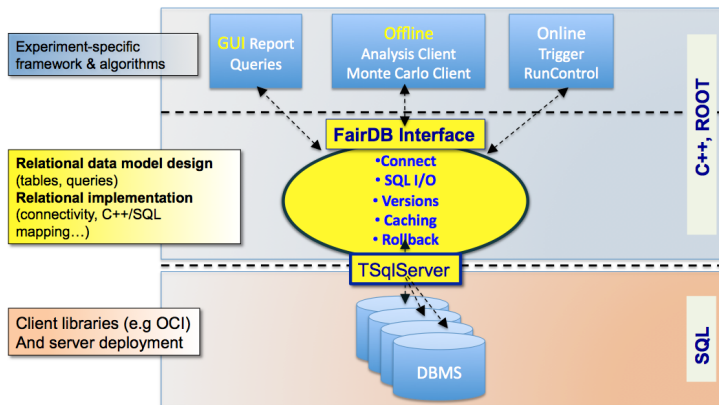




- 1 FairDB
 - Design
 - FairRoot Integration
- 2 Features
 - Connectivity
 - Versioning
 - SQL I/O
 - **Parameter Class**
 - **Error Handling**
- 3 Getting started
- 4 **User Manual (PDF)**
- 5 Conclusion

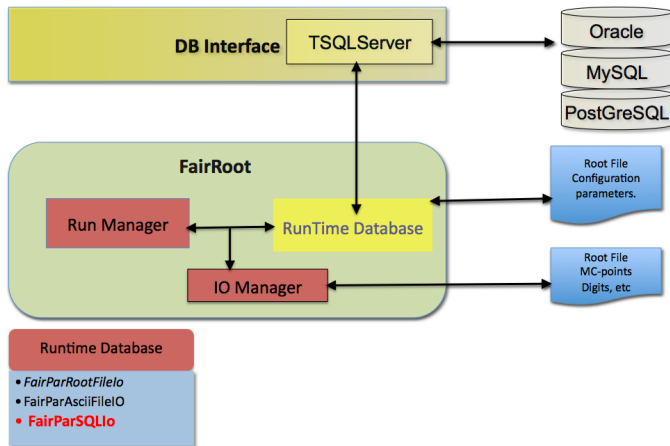


FairDB Design



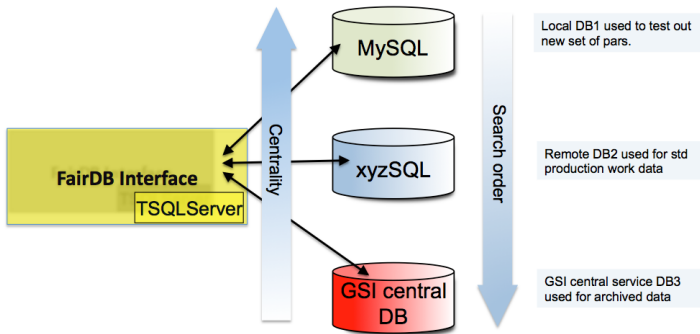


FairDB within FairRoot





Connectivity





Configuration

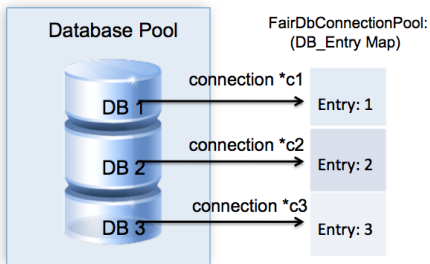
- **JDBC style:** defines a list of DB URLs, a user name and a password
- Configuration via environment variables of type FAIRDB_TSQL_*

Configuration example:

```
export FAIRDB_TSQL_URL="      mysql://localhost/r3b;  
                          pgsql://localhost:5432/R3B"  
  
export FAIRDB_TSQL_USER="    mysql_r3b";"postgres"  
export FAIRDB_TSQL_PSWD="    password1";"password2"
```



Multiple Connections



```

// SQL-IO Interface
FairParTSQLIo* inp = new FairParTSQLIo();

// Get access to the connector class
const FairDbConnectionPool& fConn = inp->GetConnections();

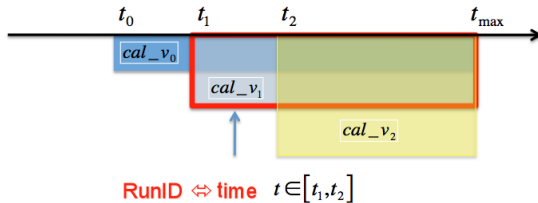
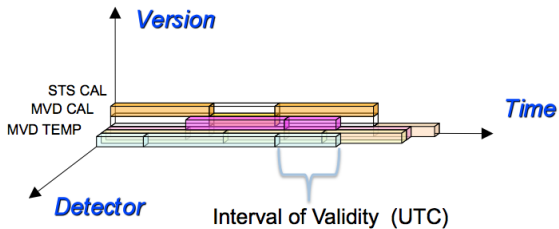
// Get number of connected databases
Int_t n_db = fConn.GetNumDb();

//Loop over all connections
for(Int_t i=0;i<n_db;++i){
  // create SQL statement
  auto_ptr<FairDbStatement> fStmt(fConn.CreateStatement(i));
  fStmt->Commit("DROP TABLE IF EXISTS FAIRDBTUTPAR");
  ...
}

```

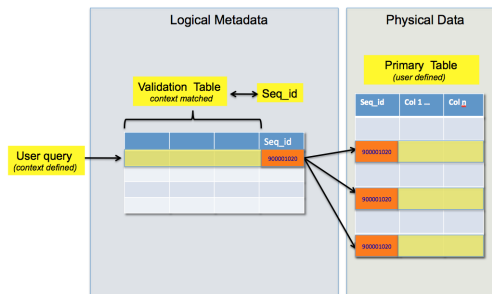


Concepts





Version Management



Query process:

- 1 Condition (timestamp, detector, version) as primary key
- 2 Condition converted to a unique **seq_id**
- 3 **seq_id** used as key to access all rows in the main table
- 4 System gives user access of all such rows



I/O Components

• Data Organization

- **Condition:** event date, time, version, detector
- **Interval:** a condition extended to a time window
- `ObjToTableMap`: Object mapped to single row in a table
- **Single or Composite:** (a set of `ObjToTableMap`: sharing a Interval)

• I/O process

• Write by Interval of Validity (IoV)

- Use case: *do calibrate channel in crate , estimate interval of validity which it remains valid and store in the DB*

• Read using a Condition

- Use case: *given a condition, get all calibration constants for every channel in every crate.*
- **Caching mechanism:** cache owns query results for reuse, caller just gets const pointer.



Templated SQL I/O

```
// Writing by IOV
const ValInterval iov;

// Create Template I/O
FairDbWriter<MyParam> writer(iov)

// Fill writer with rows of Agg
MyParam par0, par1, ...;

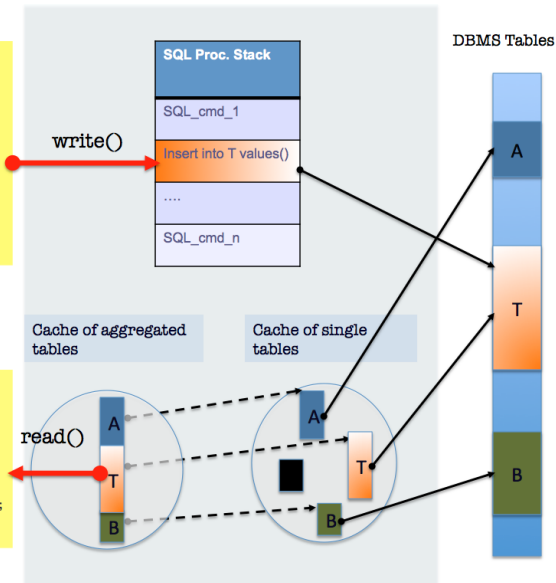
writer<< par0<<par1<< ...;

// Commit
writer.Close()
```

```
// Reading with Condition
const ValCondition cond;

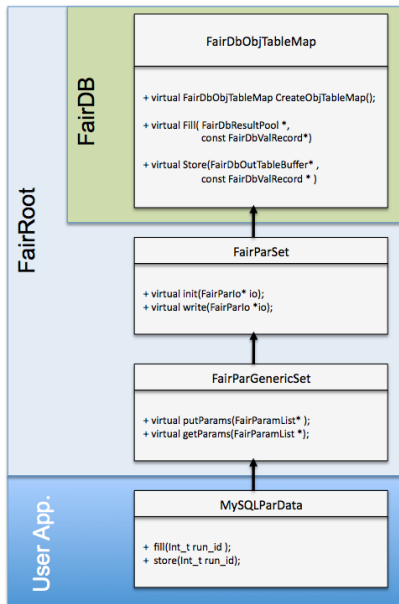
// Set up the templated reader
FairDbReader<MyParam> reader(cond);

// retrieve all rows
const MyParam* row0 = reader.GetRow(0);
const MyParam* row1 = reader.GetRow(1);
```



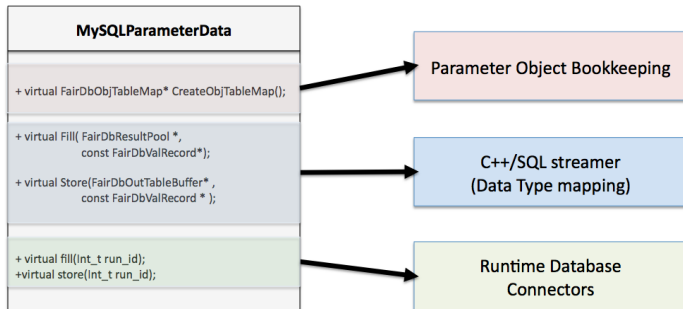


FairParSet Redesigned



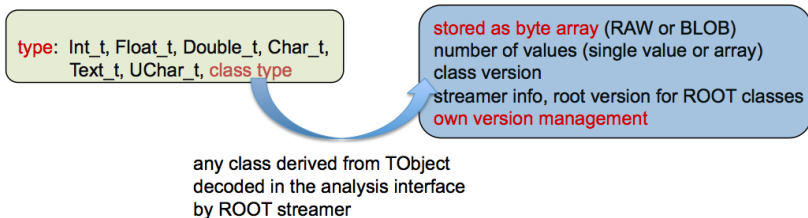


SQL-Aware Parameter Class





Data Types





Connection Info

```

fairdbinfo.log - emacs
[2013/11/28 09:02:14] -I- FairDb: FairDbTableInterfaceStore.cxx::SetLoggingStreams():[439] $ FairDb Logging service: opened.
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::FairDbConnection():[37] $ Creating a DB connection
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Open():[152] $ Successfully opened connection to: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::FairDbConnection():[40] $ successfully opened connection to: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::FairDbConnection():[78] $ this client, and MySQL server (MySQL 5.5.19) does support prepared statements.
[2013/11/28 09:02:14] -I- FairDb: FairDbStatement.cxx::ExecuteQuery():[63] $ Server:r3b SQL:Select * from FAIRDBGLOBALSEQNO where ID=0
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Close():[192] $ closed connection: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbConnectionPool.cxx::FairDbConnectionPool():[112] $

FairDbConnectionPool:: Server Status :
Status URL: Closed (auth) mysql://localhost/r3b

[2013/11/28 09:02:14] -I- FairDb: FairDbTableInterfaceStore.cxx::SetLoggingStreams():[439] $ FairDb Logging service: opened.
[2013/11/28 09:02:14] -I- FairDb: FairDbStatement.cxx::ExecuteQuery():[63] $ Server:r3b SQL:select * from FAIRDBGLOBALSEQNO;
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Open():[152] $ Successfully opened connection to: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Close():[192] $ closed connection: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbTableInterfaceStore.cxx::GetTableInterface():[229] $ create a FairDbTableInterface 0x7ffff5f8
FairDbTableInterface proxyname# FAIRDBTUTPAR::FairDbTutPar
[2013/11/28 09:02:14] -I- FairDb: FairDbTableMetaData.cxx::FairDbTableMetaData():[28] $ create for table # FAIRDBTUTPAR
[2013/11/28 09:02:14] -I- FairDb: FairDbTableMetaData.cxx::FairDbTableMetaData():[28] $ create for table # FAIRDBTUTPARVAL
[2013/11/28 09:02:14] -I- FairDb: FairDbProxy.cxx::CreateMetaData():[531] $ Create meta-data for table: FAIRDBTUTPAR
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Open():[152] $ Successfully opened connection to: mysql://localhost/r3b
--:-- fairdbinfo.log Top L12 (Fundamental)-----

```



SQL Info

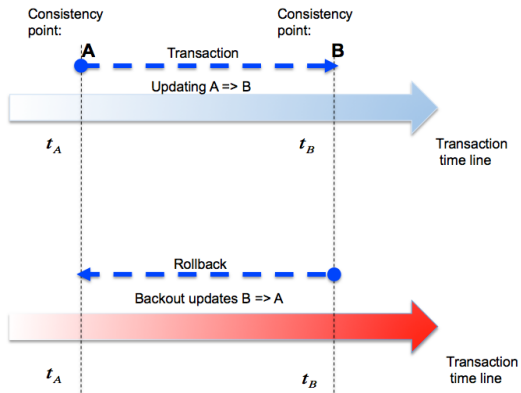
```

[2013/11/28 09:02:14] -I- FairDb: FairDbTableInterface.cxx::Query():[85] $ Query fulltimewindow 1
[2013/11/28 09:02:14] -I- FairDb: FairDbCache.cxx::Search():[165] $ Primary cache search of table: FAIRDBTUTPAR for: [ Gfil Dat
[2013-10-22 15:00:00.00000000Z] with Version: 0
[2013/11/28 09:02:14] -W- FairDb: FairDbCache.cxx::Search():[170] $ Primary cache search failed :: (derived-cache-1) is empty
[2013/11/28 09:02:14] -I- FairDb: FairDb.cxx::SetTimeWindow():[130] $ Set time window: 864000 for: FAIRDBTUTPAR
[2013/11/28 09:02:14] -I- FairDb: FairDb.cxx::GetTimeWindow():[34] $ Return time window 864000 for FAIRDBTUTPAR
[2013/11/28 09:02:14] -I- FairDb: FairDb.cxx::GetTimeWindow():[34] $ Return time window 864000 for FAIRDBTUTPAR
[2013/11/28 09:02:14] -I- FairDb: FairDbProxy.cxx::QueryValidity():[293] $ db_id: 0 SQL query: select * from FAIRDBTUTPARVAL where
TimeStart <= '2013-11-01 15:00:00' and TimeEnd > '2013-10-12 15:00:00' and DETID & 8 and DATAID & 1 and Version = 0 order by T
TIMEINCR desc;
[2013/11/28 09:02:14] -I- FairDb: FairDbStatement.cxx::ExecuteQuery():[63] $ Server:r3b SQL:select * from FAIRDBTUTPARVAL where
TimeStart <= '2013-11-01 15:00:00' and TimeEnd > '2013-10-12 15:00:00' and DETID & 8 and DATAID & 1 and Version = 0 order by TI
MEINCR desc;
[2013/11/28 09:02:14] -I- FairDb: FairDbConnection.cxx::Open():[152] $ Successfully opened connection to: mysql://localhost/r3b
[2013/11/28 09:02:14] -I- FairDb: FairDbValRecord.cxx::Fill():[159] $ FairDbValRecordord for row: 0: ldet_id 0x0008|data_id 0x0001|
2013-10-30 13:22:33.00000000Z
2038-01-19 03:14:07.00000000Z
origin: Added Conditions: seq_id: 90000111 combo_id: -1 version_id: 0
[2013/11/28 09:02:14] -I- FairDb: FairDbValRecord.cxx::Fill():[159] $ FairDbValRecordord for row: 1: ldet_id 0x0008|data_id 0x0001|
2013-10-28 11:12:07.00000000Z
2038-01-19 03:14:07.00000000Z
origin: Added Conditions: seq_id: 90000109 combo_id: -1 version_id: 0
[2013/11/28 09:02:14] -I- FairDb: FairDbValRecord.cxx::Fill():[159] $ FairDbValRecordord for row: 2: ldet_id 0x0008|data_id 0x0001|
2013-10-28 11:00:33.00000000Z
--- fairdbinfo.log 13% L59 (Fundamental)-----

```




Rollback



```
FairParTSQLIo inp_io;
```

```
// Global
inp_io.SetRollback("2013-10-01","**")
```

```
// Selective
inp_io.SetRollback("2013-10-01","CALTOF**")
inp_io.SetRollback("2013-08-01","GFI**")
```



Availability

- External packages **April 2013 or higher release**
 - Root with **DB plugins**
 - FairRoot installation (`svn` or `cvmfs`)
- Use the **trunk** version of FairRoot base (or 12/2013 release)

```
https://subversion.gsi.de/fairroot/fairbase/trunk/base
```

- Add the new example directory containing the FairDB tutorial15

```
https://subversion.gsi.de/fairroot/fairbase/trunk/example
```

- adapt the FairDB config script **dbconfig.sh**



FairDbTutPar

```
class FairDbTutPar : public FairParGenericSet
{
public : // ....
```

private:

```
    // Strip Parameters
    Double_t fTopPitch; // Strip pitch on top wafer side
    Double_t fTopAnchor; // Anchor point of top strip#0
    Int_t fTopNrFE; // NFE attached to top wafer side
    string fFeType; // Frontend type name
    ...
```

// I/O member functions

```
    virtual void Fill(FairDbResultPoli* ,
                    const FairDbValRecord*);
    virtual void Store(FairDbOutTableBuffer* ,
                    const FairDbValRecord* ) const;
    virtual void fill(UInt_t rid);
    virtual void store(UInt_t rid);

    // Validity frame definition
    virtual ValCondition GetContext(UInt_t rid) {
        return ValCondition(Detector::kGfi,
                          Datatype::kData,
                          ValTimeStamp(rid));
    }
    ... }
```



sql_param_write.C

```

FairRuntimeDb* db = FairRuntimeDb::instance();
cout << "-I- FairRuntimeDb created ----> " << db << endl;

// Create in memory the relevant container
FairDbTutPar* p1 = (FairDbTutPar*)(db->getContainer("TUTParDefault"));
FairDbTutPar* p2 = (FairDbTutPar*)(db->getContainer("TUTParAlternative"));

// Set the Ascii IO as first input
FairParAsciiFileIo* inp1 = new FairParAsciiFileIo();

TString work = getenv("VMCWORKDIR");
TString filename = work + "/example/Tutorial5/macros/ascii-example.par";
inp1->open(filename.Data(), "in");
db->setFirstInput(inp1);

// Set the SQL based IO as second input
FairParTSQLIo* inp2 = new FairParTSQLIo();
inp2->open();
db->setSecondInput(inp2);

// <INIT> containers from Ascii input
// with assigned RunId

db->initContainers(runId);
p1->Print();
p2->Print();

// <WRITE> back containers to the user-defined
// Database using the SQL based IO of the
// second input.

db->setOutput(inp2);
db->writeContainers();

```



Data and Metadata Tables

1 • select * from FAIRDBTUTPAR;

100% 27.1

Filter: Q

SEQNO	ROW_ID	TOPPITCH	TOPANCHOR	TOPNRFE	FETYPE
900000001	1	0.01	-3	10	APV25
900000002	1	0.0075	-2	14	APV22
900000003	1	0.01	-3	10	APV25
900000004	1	0.0075	-2	14	APV22
900000005	1	0.01	-3	10	APV25
900000006	1	0.0075	-2	14	APV22

1 • select * from FAIRDBTUTPARVAL;

100% 1.2

Filter: Q

SEQNO	TIMESTART	TIMEEND	DETID	DATAID	VERSION	COMPOSITEID	TIMEINCR	TIMETRANS
900000001	2013-10-22 ...	2038-01-19 ...	8	1	0	-1	2013-10-22 ...	2013-10-22 14:27:54
900000002	2013-10-22 ...	2038-01-19 ...	8	1	1	-1	2013-10-22 ...	2013-10-22 14:27:54
900000003	2013-10-22 ...	2038-01-19 ...	8	1	0	-1	2013-10-22 ...	2013-10-22 14:27:57
900000004	2013-10-22 ...	2038-01-19 ...	8	1	1	-1	2013-10-22 ...	2013-10-22 14:27:57
900000005	2013-10-23 ...	2038-01-19 ...	8	1	0	-1	2013-10-22 ...	2013-10-23 07:03:14
900000006	2013-10-23 ...	2038-01-19 ...	8	1	1	-1	2013-10-22 ...	2013-10-23 07:03:14
900000007	2013-10-23 ...	2038-01-19 ...	8	1	0	-1	2013-10-22 ...	2013-10-23 07:07:28
900000008	2013-10-23 ...	2038-01-19 ...	8	1	1	-1	2013-10-22 ...	2013-10-23 07:07:28
900000009	2013-10-23 ...	2038-01-19 ...	8	1	0	-1	2013-10-22 ...	2013-10-23 07:08:47



sql_param_read.C

```

// Create a Runtime Database singleton.
FairRuntimeDb* db = FairRuntimeDb::instance();

// Set the SQL IO as first input
FairParTSQLIo* inp = new FairParTSQLIo();
inp->open();
db->setFirstInput(inp);

// Create the container via the factory if not already created
FairDbTutPar* p1 = (FairDbTutPar*)(db->getContainer("TUTParDefault"));
FairDbTutPar* p2 = (FairDbTutPar*)(db->getContainer("TUTParAlternative"));

```

```

// Create a dummy runID using date in UTC from which
// corresponding parameters will be initialised

ValTimeStamp tStamp(2013,04,08,08,17,00);
UInt_t runId = tStamp.GetSec();
cout << "-I- looking for parameters at runID# " << runId << endl;
cout << "-I- corresponding time in runID (UTC) " << tStamp.AsString("c") << endl;

```

```

// Use the generated RunID to initialise the parameter
// using the SQL-based IO input
db->initContainers(runId);

```

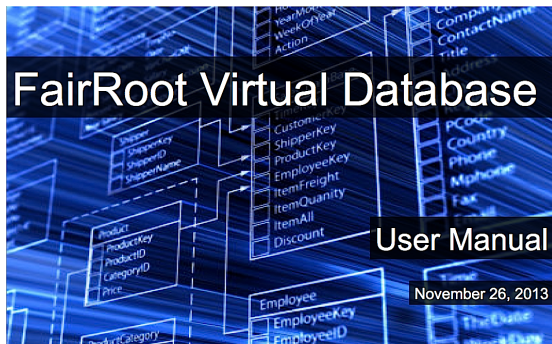
```

pp1->Print();
pp2->Print();

```



User Manual





Contents

Contents

I Basic concepts	1
1 The Virtual Database	2
1.1 Introduction	2
1.2 Problem statement and motivation	3
1.2.1 Flits versus database	3
1.3 Database in practice	4
1.3.1 Work overheads	5
1.3.2 Selecting a database	5
1.4 New database technologies	7
1.4.1 The CAP theorem	7
1.4.2 Eventual-consistency	7
1.5 Design	9
1.5.1 Interface as compromise	9
1.5.2 Architecture	10
1.5.3 FairRoot integration	11
1.6 Parameter data types	12
1.7 Mapping data objects to tables	13
1.8 Version management	17
1.8.1 Temporal database	17
1.8.2 FairRoot initialization scheme	18
1.8.3 Validity time interval	19
1.8.4 Validation basic rules	20
1.8.5 Relational model implementation	21
1.9 Persistency scheme	24
1.9.1 SQL processing	24
1.9.2 Disk cache	25
1.10 Connection Pooling	26
II Database Guidelines	28
2 Handling a Database	29
2.1 Installing	29
2.1.1 Local @ GSI	29
2.1.2 Standalone system	29
2.1.3 MySQL	30
2.1.4 PostgreSQL	33
2.1.5 Oracle	34
2.2 Handling a database server	37
2.2.1 Managing accounts	39
2.2.2 Server Checks	40

ii	Contents
III User Manual	42
3 Basic Settings	43
3.1 Setting up the Environment	43
3.2 The SQL-IO Interface	44
3.3 Handling Connections	46
3.3.1 Multiple Connections	46
3.3.2 Checking Connections	47
3.3.3 Holding Connections	48
3.3.4 Closing Connections	49
3.4 Caching	49
3.5 Rollback	50
3.5.1 Rollback Mechanism	50
3.5.2 Rollback Configuration	51
3.6 Ordered Queries	52
3.7 Error Handling	53
3.7.1 Exceptions Logging	54
3.7.2 Output Log File	54
4 C++ Parameter Object to Table Mapping	57
4.1 Creating SQL-aware Parameter Container	57
4.1.1 FairRoot Tutorials	57
4.1.2 Parameter Class Ownership	60
4.2 Creating the Object Table	61
4.2.1 Naming a Table	61
4.2.2 Describing a Table	62
4.2.3 Transient Table	64
4.3 Parameter Intrinsic SQL I/O	65
4.3.1 C++/SQL Data Types Mapping	66
4.3.2 Data Representation	66
4.3.3 Data Conversion	67
4.3.4 Single Data types	68
4.3.5 User-Defined Data types	69
4.3.6 Storing Large Object in a Database?	71
4.4 Advanced SQL I/O Features and Optimizations	72
4.4.1 Table Schema Evolution and Context Selection	74
4.4.2 Caching Activation	75
4.4.3 Ordering Rows	76
5 Versioning Management	77
5.1 Validation Table	77
5.2 FairPartSQLio Responsibilities	79
5.2.1 Run Numbers versus Timestamps	79
5.2.2 Runtime Database and SQL I/O Interface	80
5.3 Templated I/O	83
5.3.1 FairDbWriter Template	83
5.3.2 FairDbReader Template	86
5.4 Condition data	88
5.4.1 Data Members Structure	88
5.4.2 Template Instantiation	89



Conclusion

- A new FairRoot database interface is available
- It does not fix the DBMS technology
 - MySQL
 - Oracle
 - PostGreSQL
 - ... NoSQL ?
- **User Manual** (available in PDF format)
- Ongoing work
 - Finalizing PostGreSQL integration
 - Realistic tests (R3B/Epics, R3B/LandO2 ...)
- User feedback needed !