Status of Database Interface and Crestdb

Wenhao Huang
Xingtao Huang

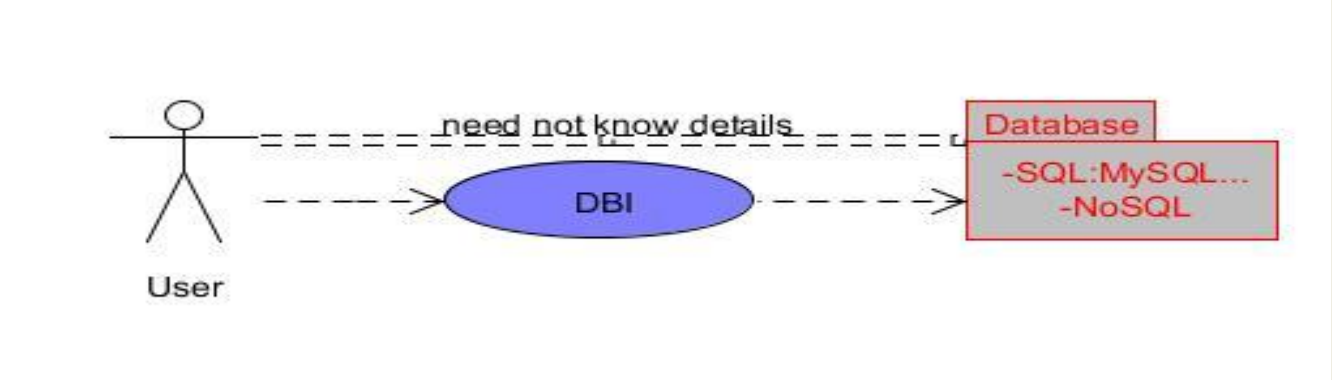
2018/05/10-2018/05/15

Outline

- Database Interface(DBI)
  - *Motivation*
  - *Main Components*
  - *Usage*
- Crestdb
  - *Introduction*
  - *Crestdb-Server*
  - *Crestdb-Client*
- Summary

Database Interface--Motivation

- Between user's code and specific database.



- Robust high-level interface
  - *DBITableRow*
    - Abstract interface to Table Row object. Each database table consists of a collection of Table Row objects.
    - Create SQL queries automatically when accessing or filling tables.

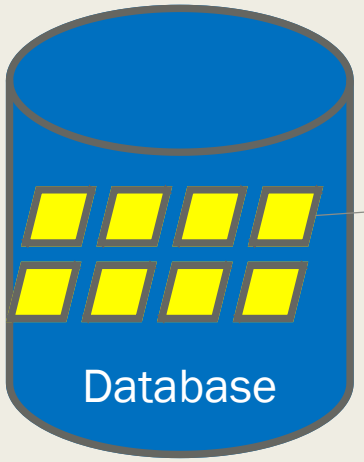
Database Interface--Motivation

- Advanced functions
  - *DBIRollbackDates*
    - Data could be rolled back to a certain version by hands.
- Data stored in databases
  - *Detector related information*
    - PMT placement geometry
    - Calibration tables or constant
  - *Tabular static data*
    - Optical parameters
  - *Others suitable for DB*

Database Interface—Main Components

- Main components of DBI implemention
  - *Context(timestamp)*
    - Tag information of real data
  - *DBITableRow Class*
    - Base object of accessing or filling tables
  - *Accessing tables by DBIResultPtr*

Context and range table



Context range table

Main data table

Tag information, decided by context

```
mysql> desc MaterialProperty;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| sftVer | varchar(16) | YES  |     | NULL    |       |
| name   | varchar(256)| YES  |     | NULL    |       |
| keyId  | int(11)     | NO   | PRI | NULL    |       |
| notes  | varchar(256)| YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

```
mysql> desc MaterialPropertyVld;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| keyId      | int(11)  | NO   | PRI | NULL    | auto_increment |
| TimeStart  | datetime | NO   | MUL | NULL    |                |
| TimeStop   | datetime | NO   | MUL | NULL    |                |
| VersionDate| datetime | NO   |     | NULL    |                |
| InsertDate | datetime | NO   |     | NULL    |                |
+------------+----------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Context is a set of times. Could be changed according to needs.

Data

DBITableRow Class(Usage)

DBI would generate the class automatically .

```
class MaterialProperty:public DBITableRow
{
 MaterialProperty(char* sftVer, char* name, int keyId, char* notes);
 virtual ~MaterialProperty();

 char* fsftVer;
 char* fname;
 int fkeyId;
 char* fnotes;

 void SetsftVer(char* sftVer){fsftVer = sftVer;}
 void Setname(char* name){fname = name;}
 void SetkeyId(int keyId){fkeyId = keyId;}
 void Setnotes(char* notes){fnotes = notes;}

 char* GetsftVer(){return fsftVer;}
 char* Getname(){return fname;}
 int GetkeyId(){return fkeyId;}
 char* Getnotes(){return fnotes;}

 ClassDef(MaterialProperty, 1)
};
```

DBIResultPtr(Usage)

- Accessing existing tables

```
TimeStamp tmstamp(2018, 1, 5, 10, 3, 1);
Context vc(tmstamp);
DBIResultPtr dbptr("DatabaseSvc", vc);
vector<DBIDemoData> res;
res << dbptr; //data has been stored in res
for(auto tem : res)
{
 //...
 //do operations for each element
 //...
```

- Filling tables

```
TimeStamp tmstamp(2018, 1, 5, 10, 3, 1);
TimeStamp tmsop(2018, 1, 6, 10, 5, 1);
ContextRange vc(tmstamp);
DBIResultPtr dbptr("DatabaseSvc", vc);
vector<DBIDemoData> res;
//...
//do operations filling res
//...
res >> dbptr; //data has been inserted into db
~
```

Performance test (linear)

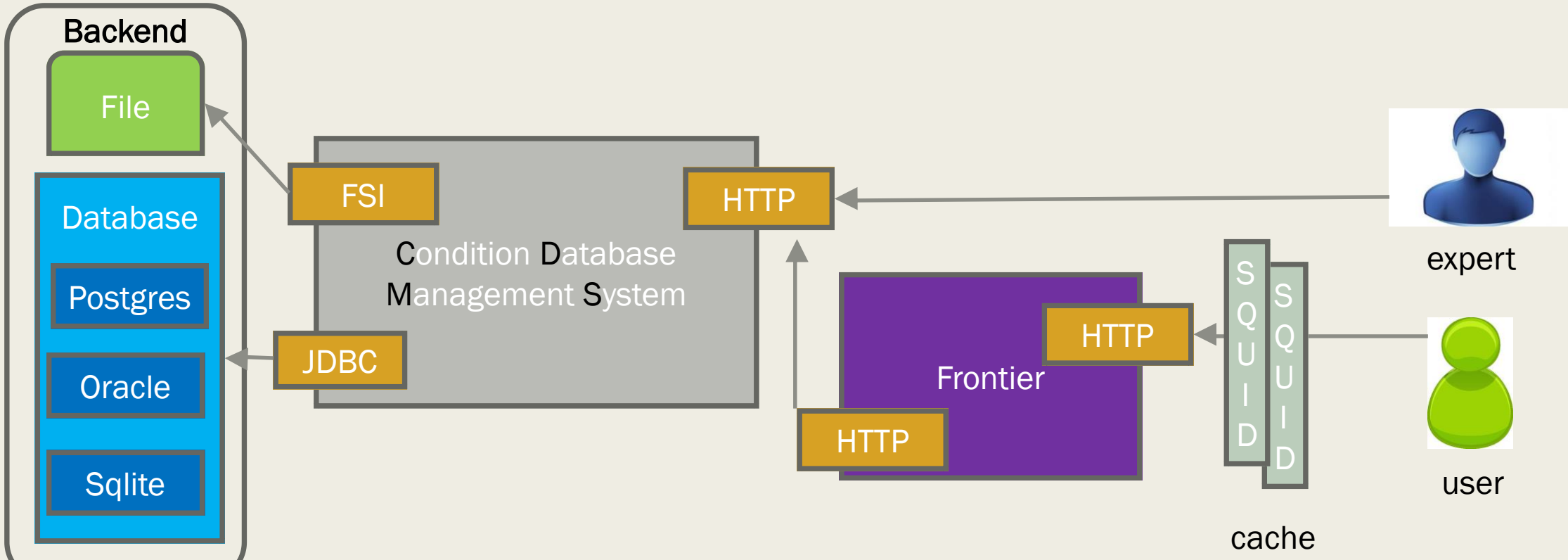
Inserting: 10,000 rows ~ 8s
100,000 rows ~ 79s
1,000,000 rows ~ 760s

Reading: 1,000,000 rows ~ 4.5s
2,000,000 rows ~ 8s

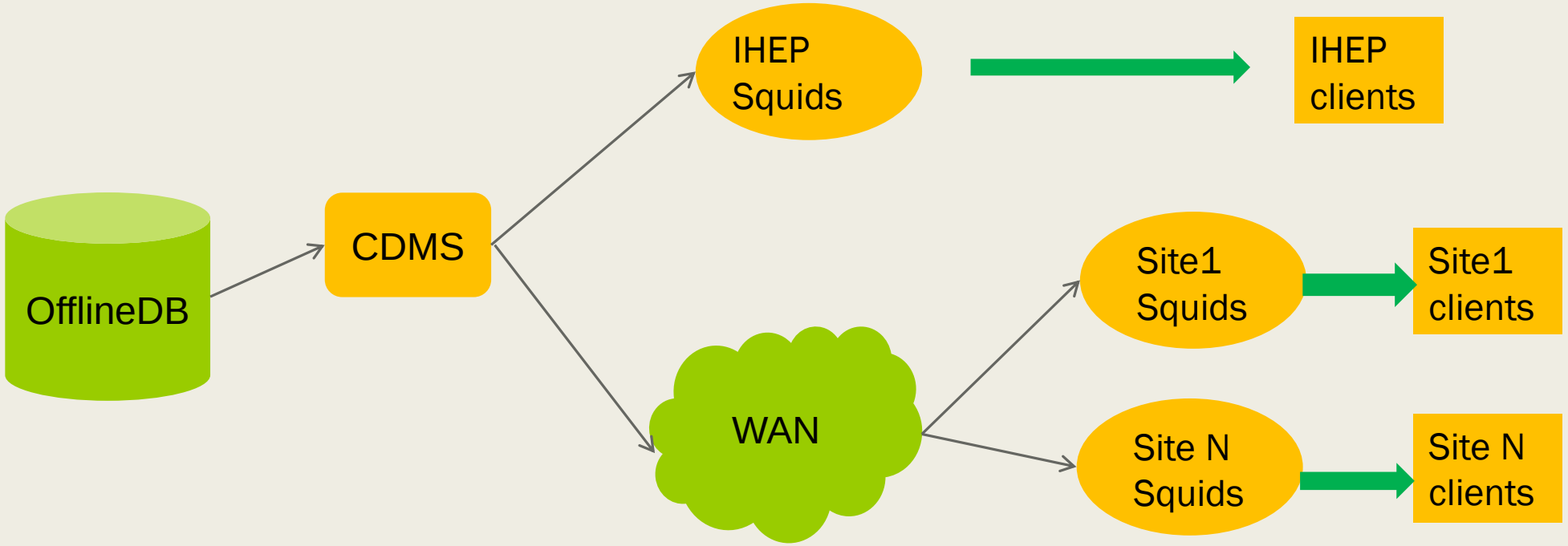
Crestdb--Introduction

- Test project for the implementation of a generic purpose conditions database for physics experiment. This server was generated by the swagger-codegen project
- Condition database used for condition data. Usually data always changed.
  - *Condition data*
    - Configuration parameters
    - Detector Control System
    - Detector and system monitoring information
    - Detector calibration data

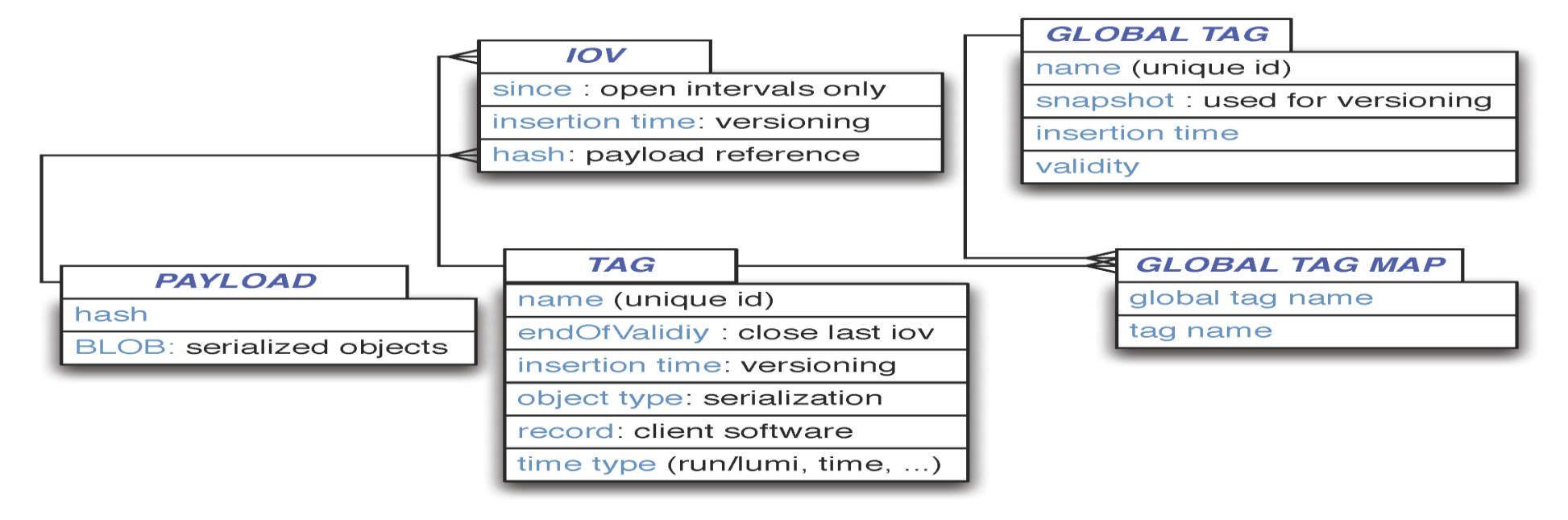
Archetype of condition database



Condition database management of JUNO(development)



Data model


IOV
since : open intervals only
insertion time: versioning
hash: payload reference
GLOBAL TAG
name (unique id)
snapshot : used for versioning
insertion time
validity
PAYLOAD
hash
BLOB: serialized objects
TAG
name (unique id)
endOfValidiy : close last iov
insertion time: versioning
object type: serialization
record: client software
time type (run/lumi, time, ...)
GLOBAL TAG MAP
global tag name
tag name

Data model

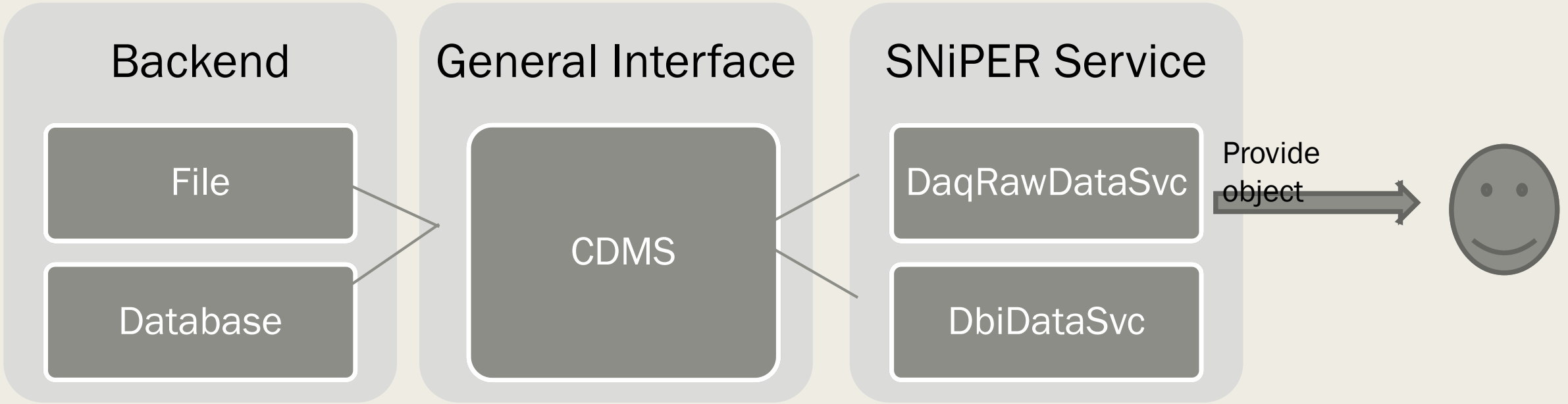
- Explanations of data model
  - ***Global Tag****: top-level configuration, resolves to a particular system* ***Tag*** *via the* ***Global Tag Map*** *table*
  - ***Global Tag Map****: corresponding* ***Global Tag*** *and* ***Tag***
  - ***Tag:*** *consisting of much iov information*
  - ***Iov:*** *intervals of validity or entries*
  - ***Payload****: conditions data uniquely identified by a hash*
- Advantage of data model
  - *Global Tag and Tag play roles as validity tables*
  - *Payload consists of condition data as a blob type, ignoring data differences*

Crestdb--client

- Curl is the base way to access the Crest server.

- CERN provides python API and client .
- C or Cpp API for Crest Server are developing
  - *A Cpp client and API was generated by Swagger.*
    - Based on cpprest(Microsoft)
    - C++11
    - Gcj (Cpp and Java cross-compile)
  - *Wrapping Curl directly is under consideration*

JUNO users accessing



We retain a general database interface for non-condition database accessing.

Summary

- Database Interface
  - *Accessing database via DBI*
  - *Based on official database connector*
- Crestdb
  - *Server has been setup on test machine*
    - MySQL, SQLite supported
  - *Client*
    - Using Crestdb via python API(official)
    - Developing C or Cpp API
    - Cache needed