Polyglot data system

applied to

Francesco Foresi | 508971
Pier Vincenzo De Iellis | 512521

M8

Data Management Problems

01 Various types of data model

02 No scalable architecture

03 No common view of data resources

Data Management Solution

Multiple Data stores

Query System

Heterogeneous Storage Engine

“One size does not fit all”

—State of Art

Federated vs Polystore

- Unique Query Language for all DBs
- Simple implementation
- Individual storage engines are independent

Federated

VS

- Native Query Language for DBs
- Hard implementation
- Storage engines are managed together as an integrated set

Polystore

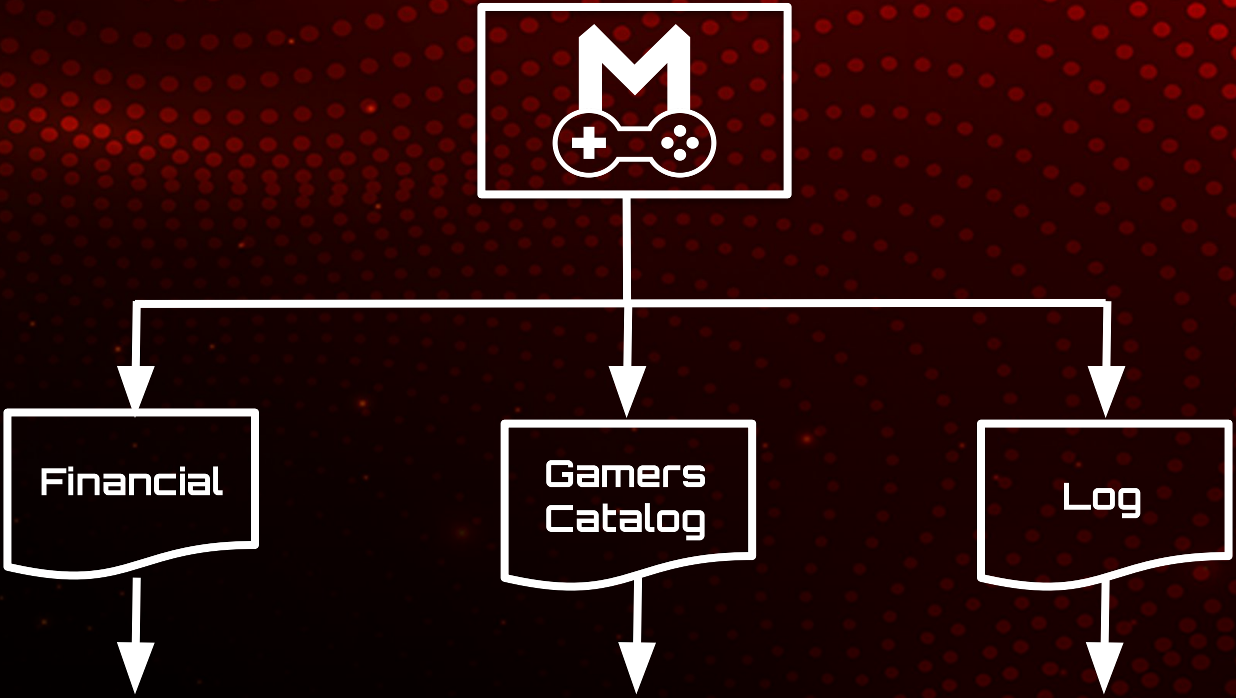
BigDAWG vs Polypheny

- Different way of organizing data and making queries
- Different data migration
- Different Architecture

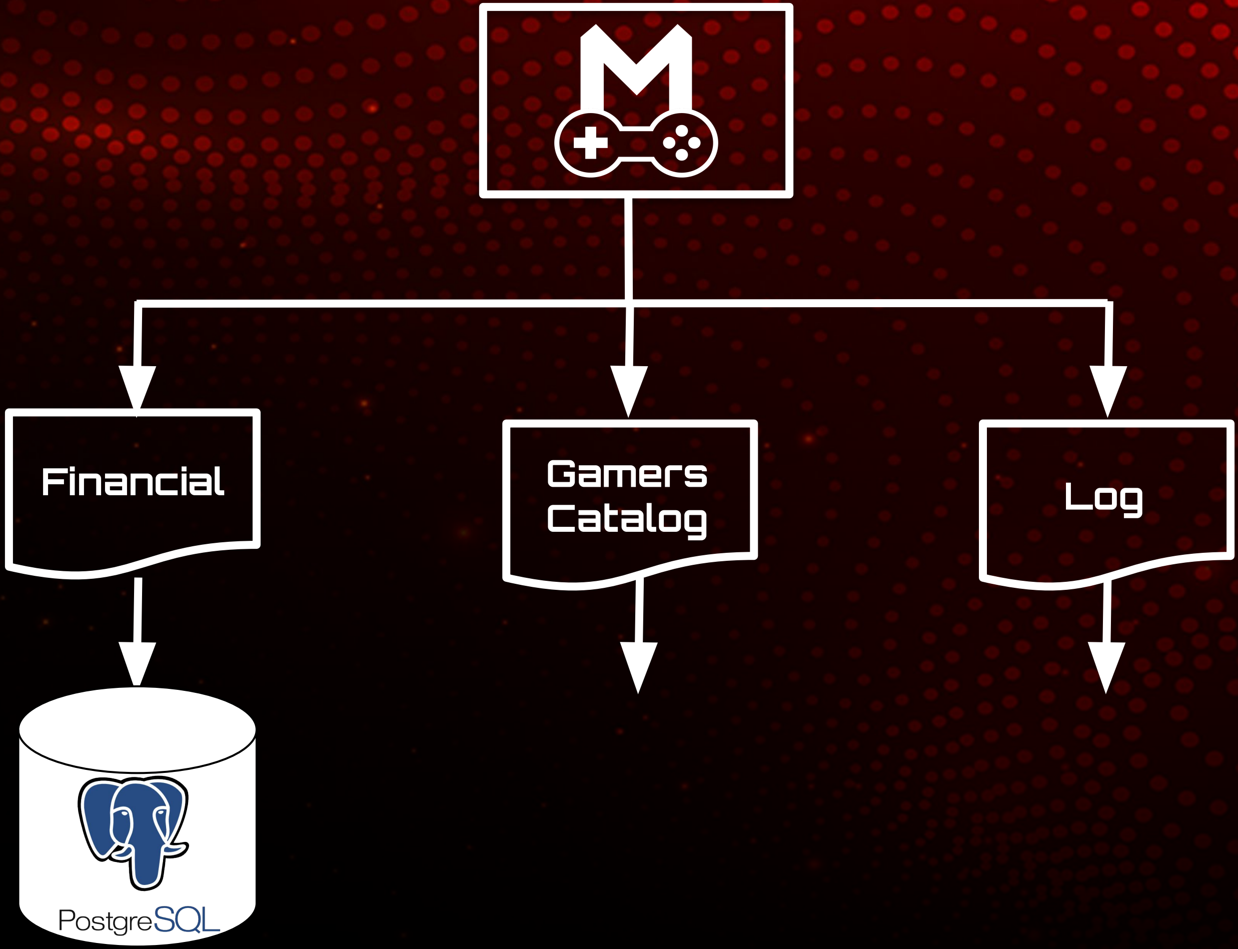
Our choice: BigDAWG

- BigDAWG is based on 3 fundamental concepts:
    - **Islands** (data model and programming model)
    - **Shim** (model translate)
    - **Cast** (data migration)

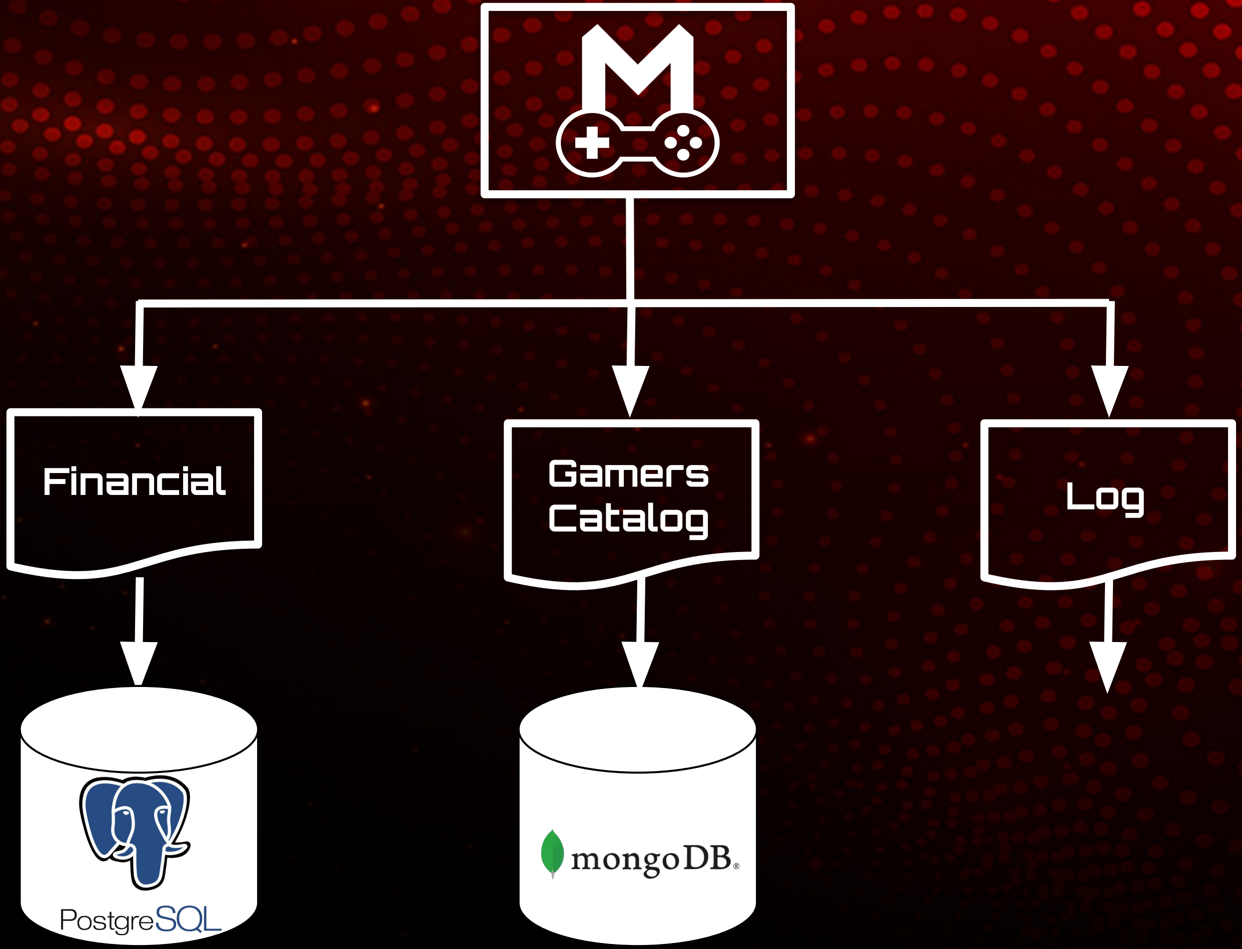
M8 Polystore



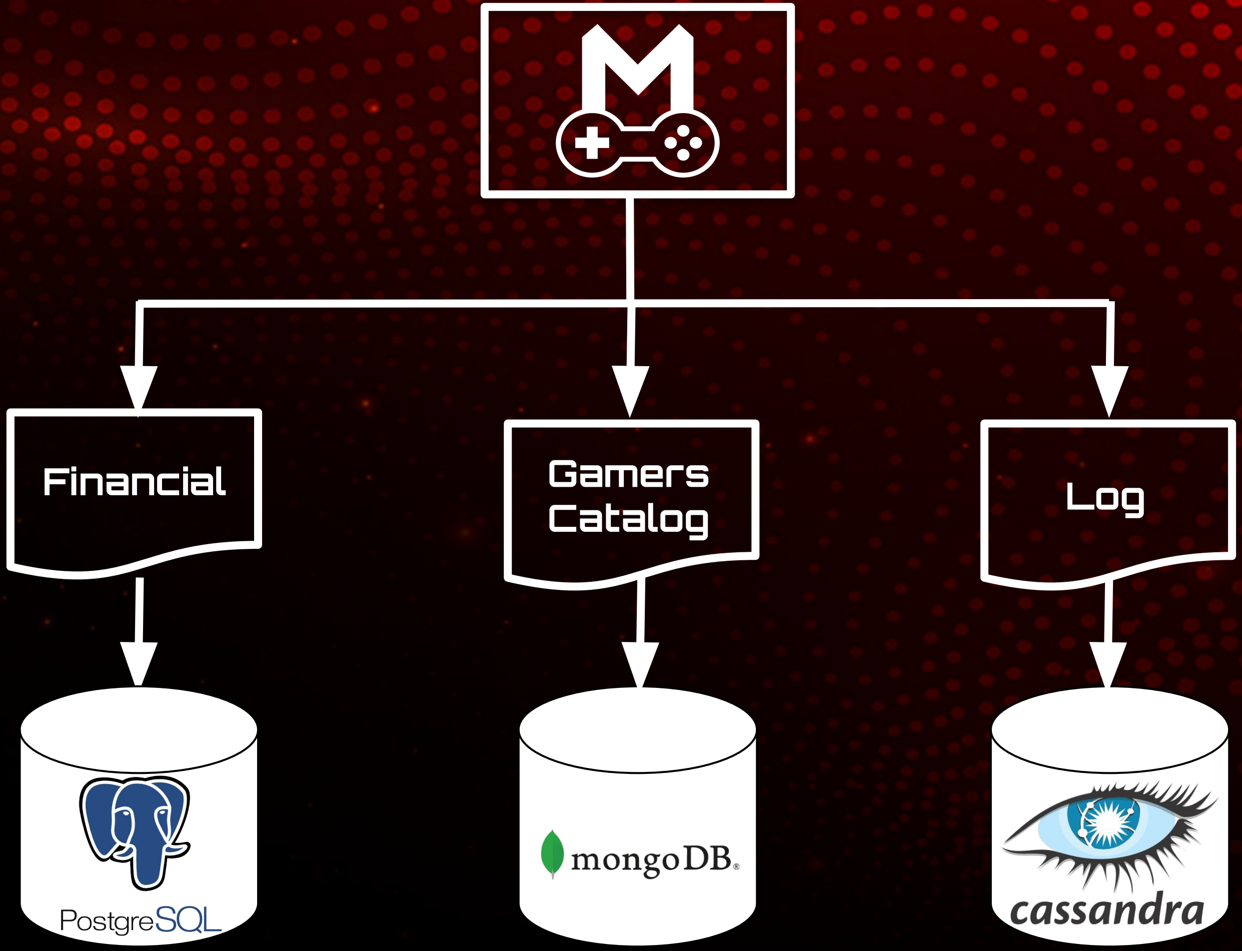
M8 Polystore



M8 Polystore



M8 Polystore



Our Implementation

M8 Client BigDAWG

- Spring Boot
- BigDAWG
- PostgreSQL, SciDB, Accumulo

Our Implementation

Dataset

- **MIMIC II** - *Multiparameter Intelligent Monitoring in Intensive Care II*

- **3 types of data:**
    - **Clinical data** (Relational)
    - **Time-series waveform data** (Array)
    - **Textual medical reports** (Key-Value)

Our Implementation

Execution environment

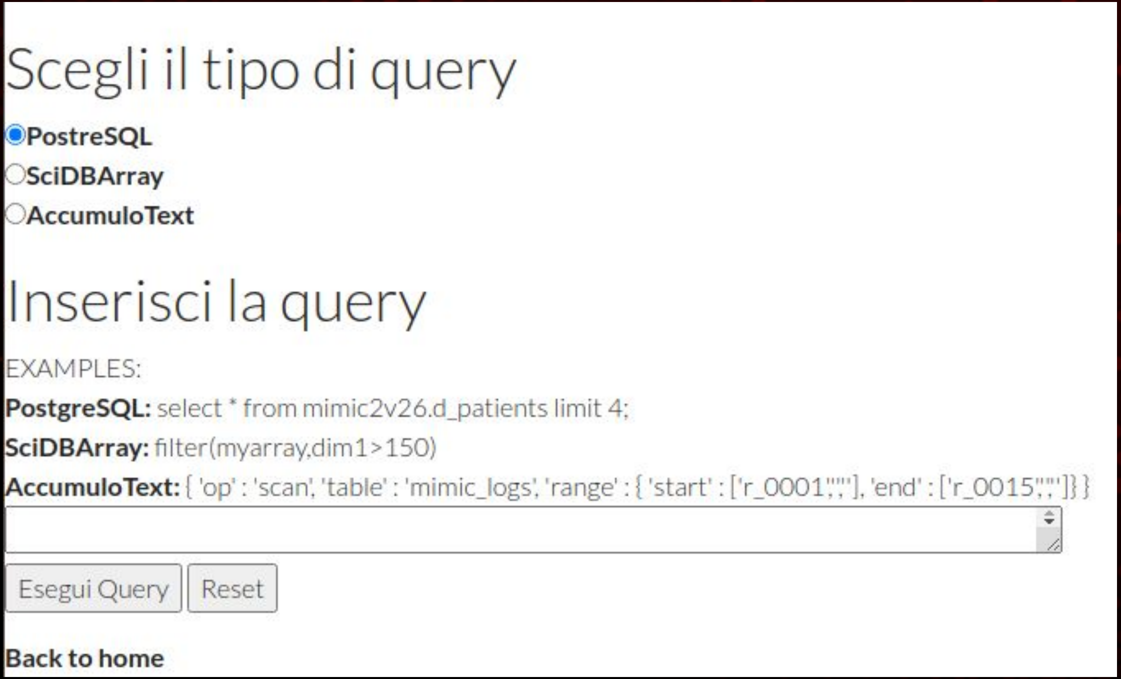
- Oracle Virtual Box - Ubuntu 18.04
- Docker
  - postgres-catalog
  - postgres-data1
  - postgres-data2
  - scidb
  - accumulo-containers (x4)

Our Implementation

Function: Query

- **Executes relational, array or textual queries**
- **BigDAWG syntax:** query token
    - **bdrel(** <SQL query content> **)**
    - **bdarray(** <SciDB query content>**)**
    - **bdtext(** <Accumulo query content>**)**

Demo: Query function



Scegli il tipo di query

[✓ PostreSQL]
SciDBArray
AccumuloText

Inserisci la query

EXAMPLES:

PostgreSQL: select * from mimic2v26.d_patients limit 4;

SciDBArray: filter(myarray,dim1>150)

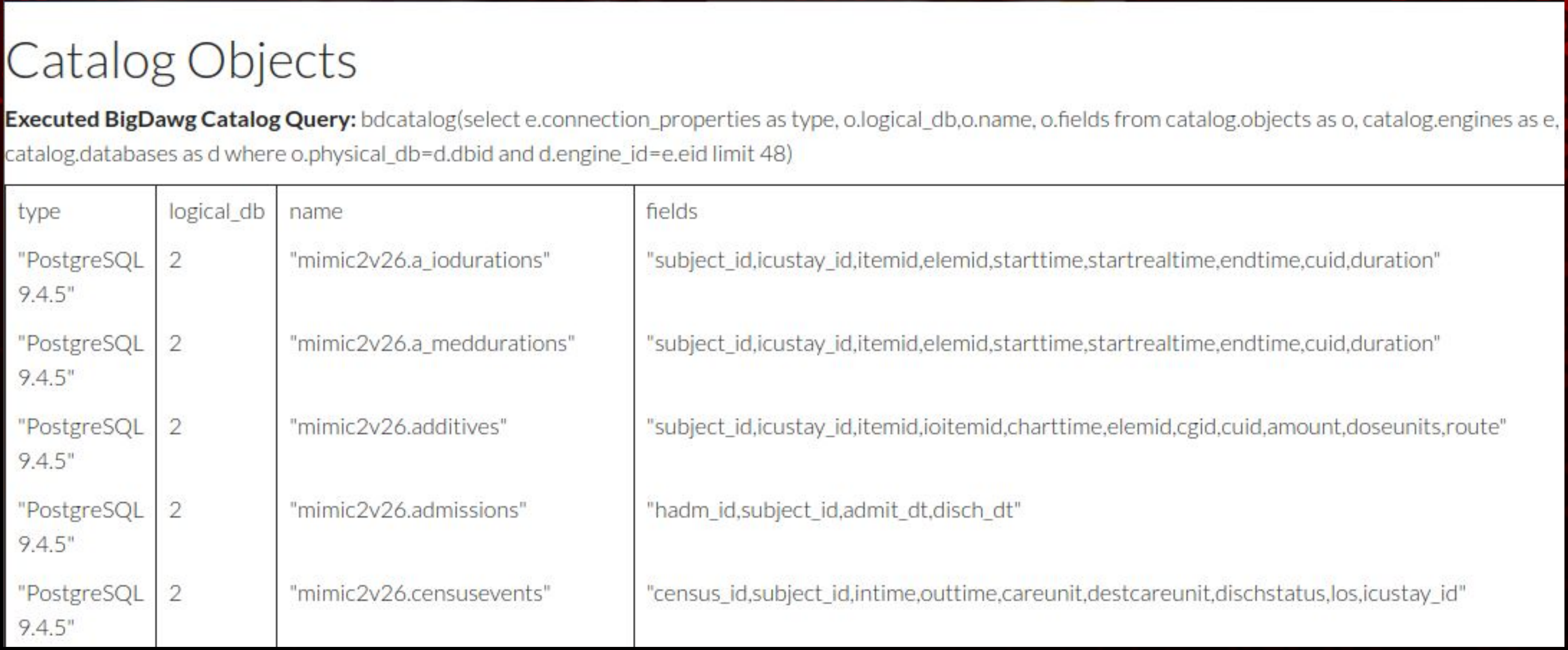
AccumuloText: { 'op' : 'scan', 'table' : 'mimic_logs', 'range' : { 'start' : ['r_0001','',''], 'end' : ['r_0015','','']} }

Our Implementation

Function: Catalog

- **Interrogate catalog for meta-data knowledge**
- **BigDAWG syntax:** catalog token
  - **bdcatalog(** <SQL query content> **)**
  - **objects, engines, databases, shims**

Demo: Catalog function



Catalog Objects

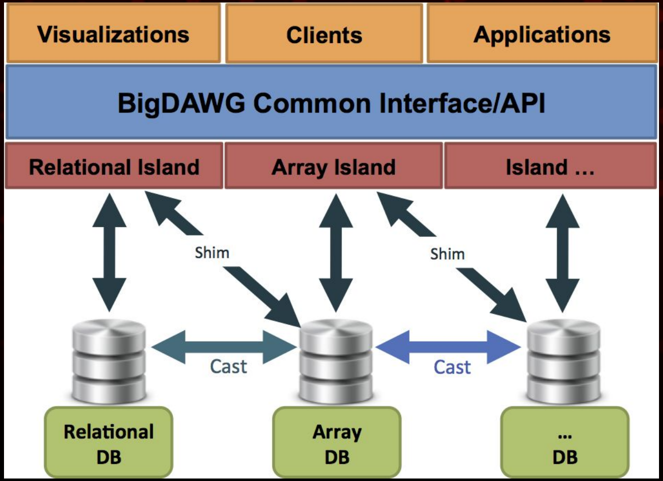
Executed BigDawg Catalog Query: bdcatalog(select e.connection_properties as type, o.logical_db,o.name, o.fields from catalog.objects as o, catalog.engines as e, catalog.databases as d where o.physical_db=d.dbid and d.engine_id=e.eid limit 48)

type	logical_db	name	fields
"PostgreSQL 9.4.5"	2	"mimic2v26.a_iodurations"	"subject_id,icustay_id,itemid,elemid,starttime,startrealtime,endtime,cuid,duration"
"PostgreSQL 9.4.5"	2	"mimic2v26.a_meddurations"	"subject_id,icustay_id,itemid,elemid,starttime,startrealtime,endtime,cuid,duration"
"PostgreSQL 9.4.5"	2	"mimic2v26.additives"	"subject_id,icustay_id,itemid,ioitemid,charttime,elemid,cgid,cuid,amount,doseunits,route"
"PostgreSQL 9.4.5"	2	"mimic2v26.admissions"	"hadm_id,subject_id,admit_dt,disch_dt"
"PostgreSQL 9.4.5"	2	"mimic2v26.censusevents"	"census_id,subject_id,intime,outtime,careunit,destcareunit,dischstatus,los,icustay_id"

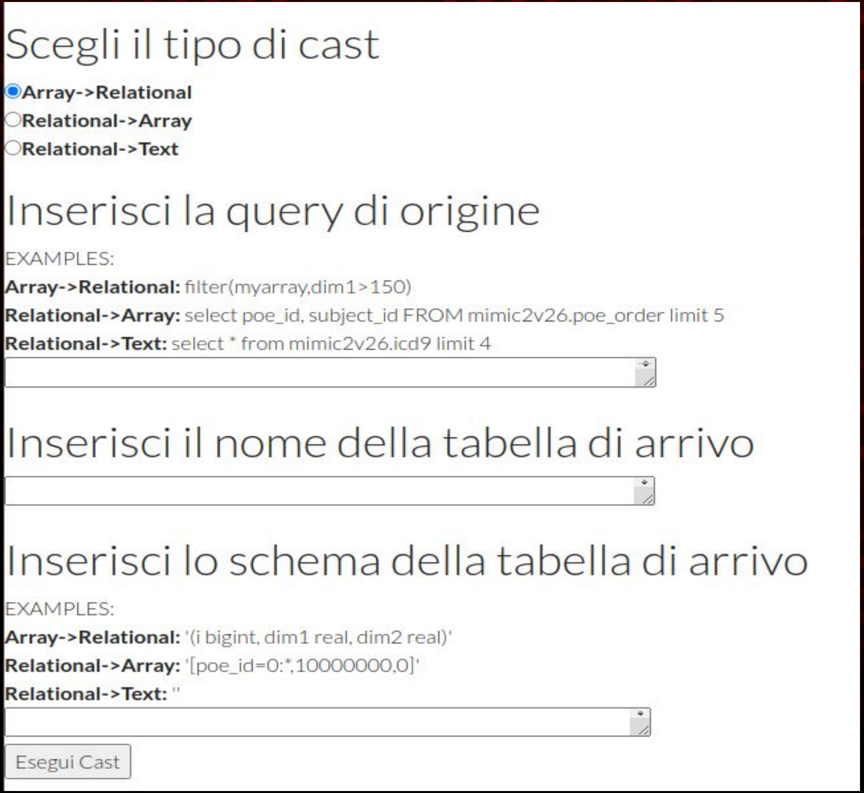
Our Implementation

Function: Cast

- **Executes cast function**
- **BigDAWG syntax:** annidate cast token **bdcast(...)**
  - Array to Relational
  - Relational to Array
  - Relational to Textual



Demo: Cast function



Scegli il tipo di cast

[✓ Array->Relational]
Relational->Array
Relational->Text

Inserisci la query di origine

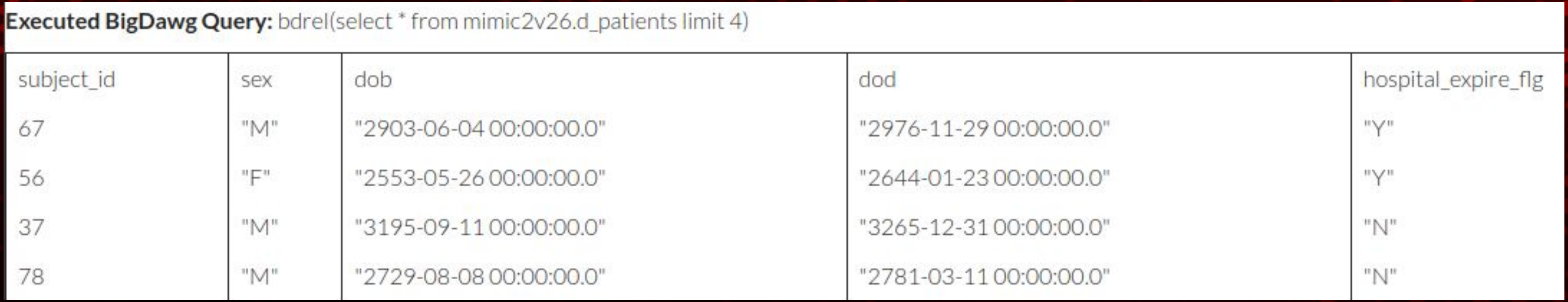
EXAMPLES:
Array->Relational: filter(myarray,dim1>150)
Relational->Array: select poe_id, subject_id FROM mimic2v26.poe_order limit 5
Relational->Text: select * from mimic2v26.icd9 limit 4

Inserisci il nome della tabella di arrivo

Inserisci lo schema della tabella di arrivo

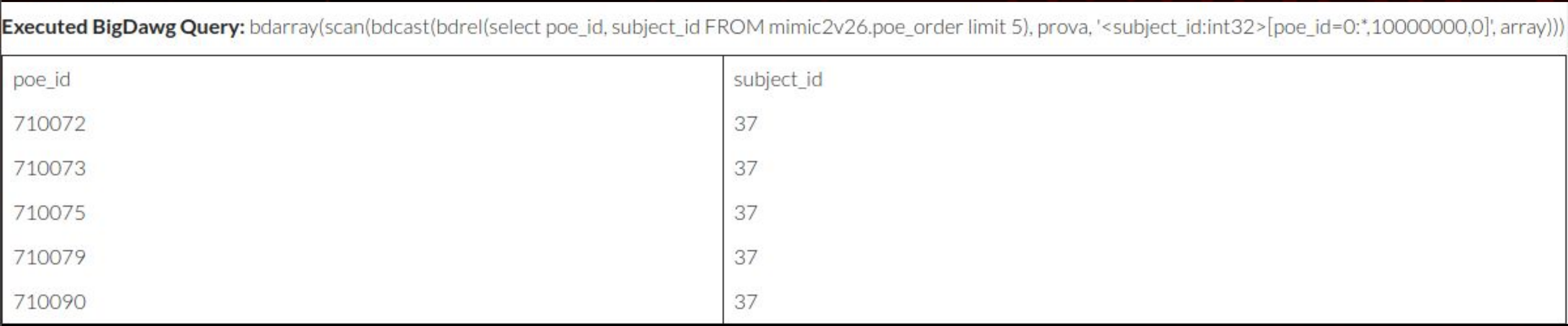
EXAMPLES:
Array->Relational: '(i bigint, dim1 real, dim2 real)'
Relational->Array: '[poe_id=0:*,10000000,0]'
Relational->Text: ''

Demo: Query/Cast results



Executed BigDawg Query: bdrel(select * from mimic2v26.d_patients limit 4)

subject_id	sex	dob	dod	hospital_expire_flg
67	"M"	"2903-06-04 00:00:00.0"	"2976-11-29 00:00:00.0"	"Y"
56	"F"	"2553-05-26 00:00:00.0"	"2644-01-23 00:00:00.0"	"Y"
37	"M"	"3195-09-11 00:00:00.0"	"3265-12-31 00:00:00.0"	"N"
78	"M"	"2729-08-08 00:00:00.0"	"2781-03-11 00:00:00.0"	"N"



Executed BigDawg Query: bdarray(scan(bdcast(bdrel(select poe_id, subject_id FROM mimic2v26.poe_order limit 5), prova, '<subject_id:int32>[poe_id=0:*,10000000,0]', array)))

poe_id	subject_id
710072	37
710073	37
710075	37
710079	37
710090	37

Future developments

- Integrate MongoDB and Cassandra engines
- Introduce analytic layer for batch processing

Thanks

Thanks for your attention!

pie.delellis@stud.uniroma3.it | 512521
fra.foresi@stud.uniroma3.it | 508971