Obevo

NY Java SIG

You can find me at:

Github: shantstepanian

Twitter: @shantstepanian

shant.p.stepanian@gmail.com

Obevo: At a Glance

Database Deployment Tool handling Enterprise Scale and Complexity

Open-sourced by Goldman Sachs in 2017

http://github.com/goldmansachs/obevo

DB2, Oracle, PostgreSQL, SQL Server, Sybase, MongoDB

Agenda: Product Overview

Overview of database deployment problem space

Overview of tooling in the market

How Obevo helps

Agenda: Hands-on Demo

Perform a simple database deployment

Explore Java integrations: ORM and in-memory testing

Reverse-engineering existing applications

DB Deployment Problem Space

Areas under Database Deployment

Production deployments

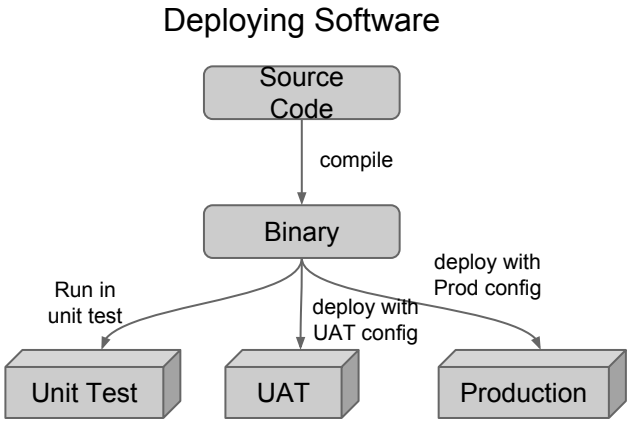
Non-production deployments (testing)

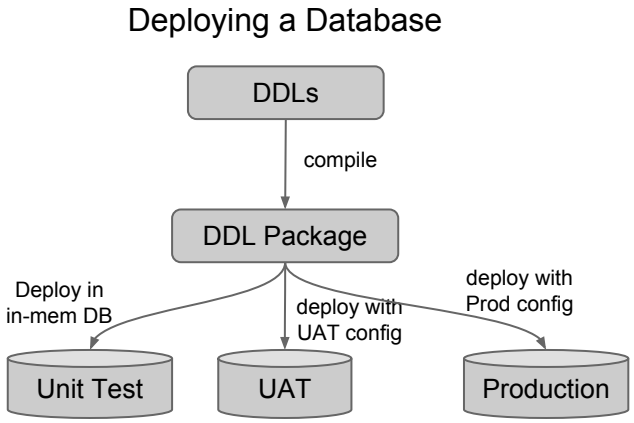
Code maintenance and organization

Solving existing messes □

DB Deploy Within SDLC

Deploy your database code as you would deploy your application code (DevOps / IaC)





What goes into a DB Deployment

Stateful Scripts

(incremental definitions)

table DDLs

data migrations

sequences

...

Stateless Scripts

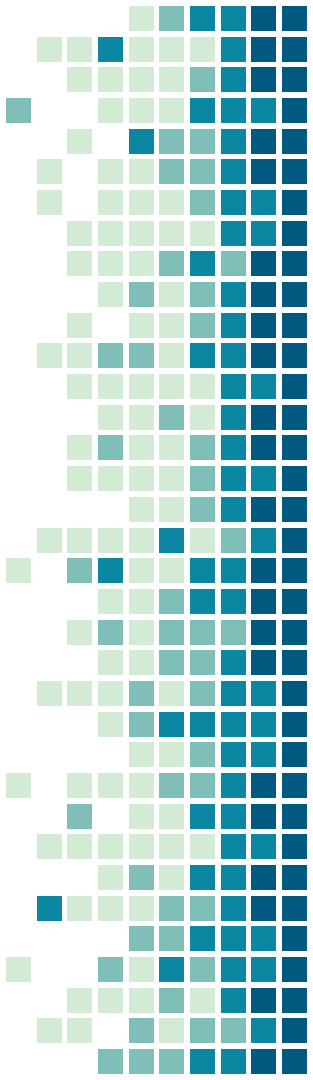
(full / rerunnable definitions)

views

stored procedures

code table data updates

...



Stateful Script Deployments

```
create table Employee (
    id bigint,
    name VARCHAR(32),
    status INT,
    PRIMARY KEY (id)
)
```

```
ALTER TABLE employee
ADD department VARCHAR(32)
```

```
ALTER TABLE employee
ADD salary BIGDECIMAL
```

```
CREATE TABLE employee (
    id BIGINT,
    name VARCHAR(32),
    status INT,
    department VARCHAR(32),
    salary BIGDECIMAL,
    PRIMARY KEY (id)
)
```

Note: full table DDL is never executed in the DB

Stateless Script Deployments

views

```
CREATE OR REPLACE VIEW
v_emp AS
SELECT * FROM employee
```

```
CREATE OR REPLACE VIEW
v_emp AS
SELECT * FROM employee
WHERE status = 0
```

```
CREATE OR REPLACE VIEW
v_emp AS
SELECT * FROM employee
WHERE status = 1
```

Note: full object definition in DB is represented in code

code table data updates

```
DEPT_ID,DEPT_NAME,TYPE
1,Finance,A
2,IT,A
3,Operations,B
```

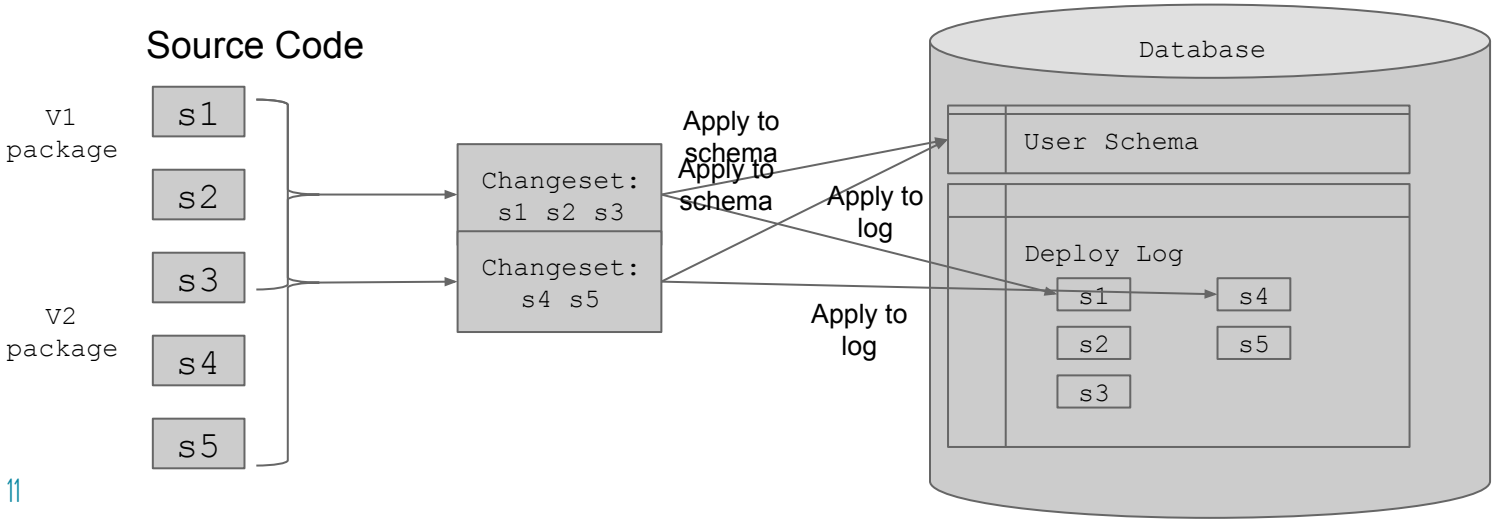
```
DEPT_ID,DEPT_NAME,TYPE
2,IT,A
3,Operations,B
4,Tax,B
5,Research,C
```

```
DEPT_ID,DEPT_NAME,TYPE
2,IT,A
3,Operations,C
4,Tax,B
5,Research,C
6,Engineering,D
```

How DB Deploy Tools Work

Scripts modeled as entries in a deploy log

Tool applies changes not yet in the deploy log



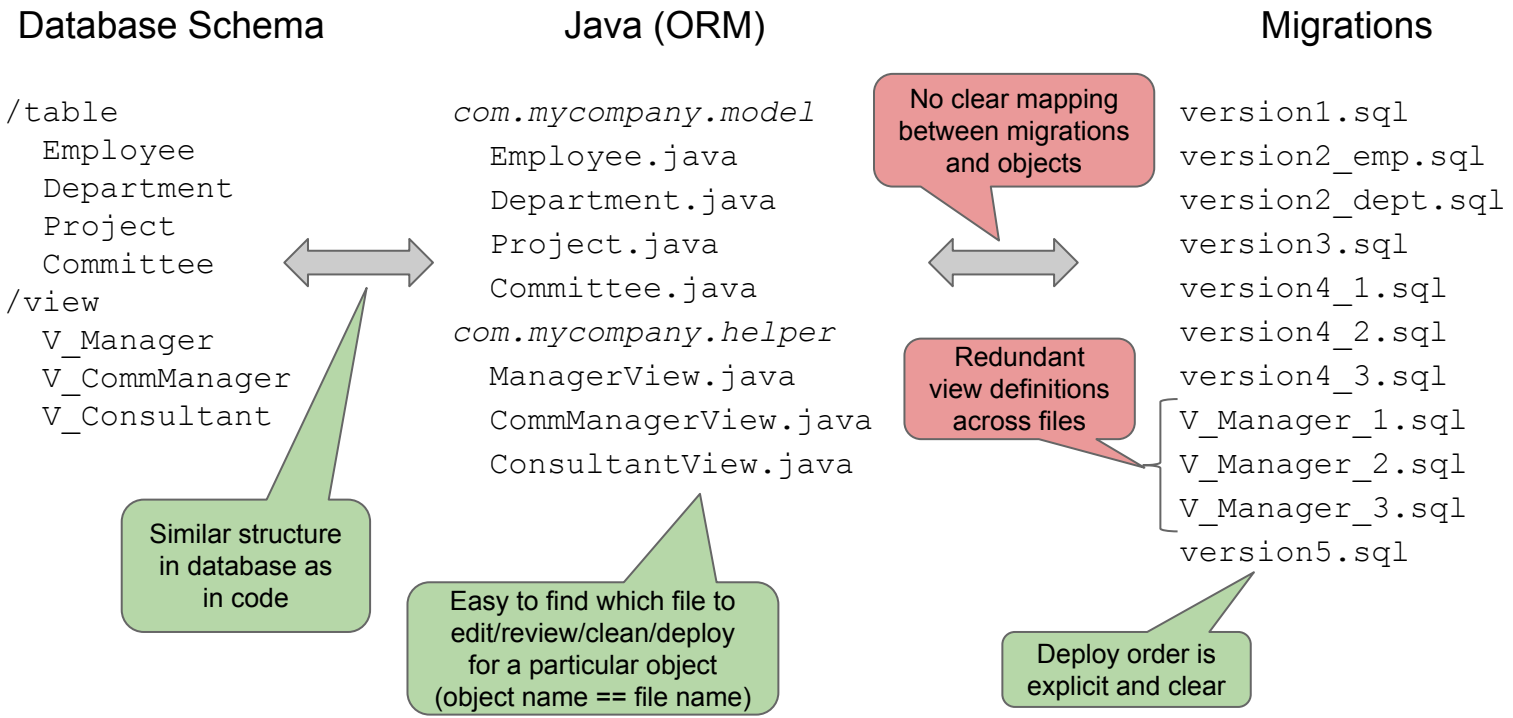
Migration Script Representation

DB Deploy tools will mostly differ on the following aspects:

- How to representation migrations: migrations per file? denoting subsections within a file?
- How to order migrations: use a naming convention on the file? use a separate "command file" to list the order?
- Rerunnability/mutability of scripts under certain circumstances

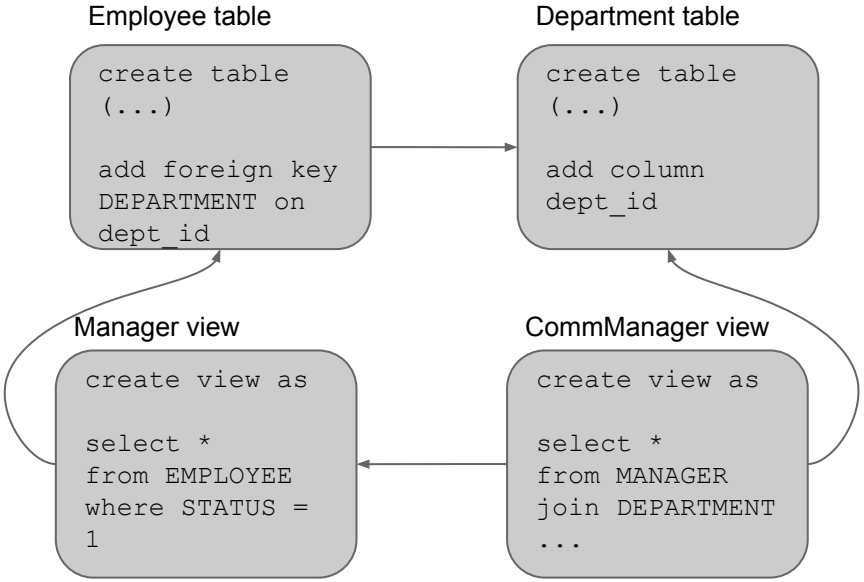
Problems with current tooling?

Migration File Maintenance



Change Ordering

So why not just arrange scripts by file?



How to handle multiple scripts for stateful objects?

How to order scripts?

How to discover dependencies?

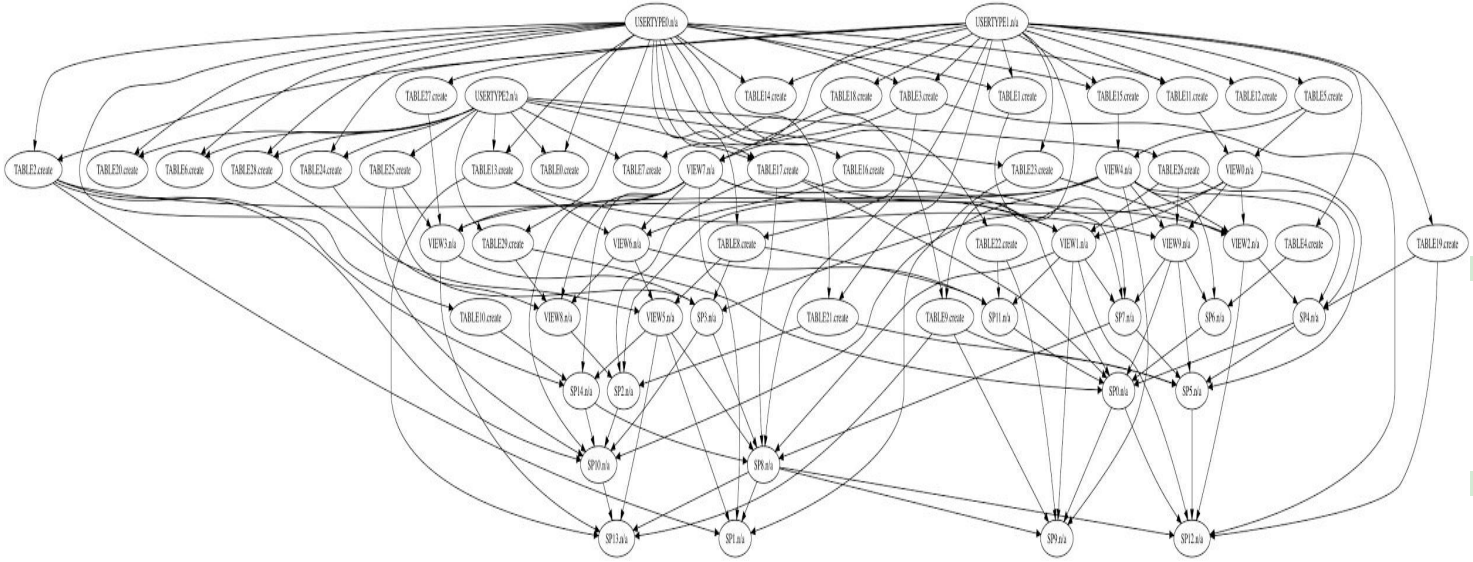
At Scale and Complexity

Various object types (tables, SPs, views, packages, ...)

Many developers, many releases

Hundreds and thousands of objects

At Scale and Complexity



At Scale and Complexity

Too complex! Why bother?

Because systems live on and still need development.

Can we work through this?

Obevo Overview

Obevo Approach

Let's solve problems for all kinds of systems

New applications	Long-lived systems
Tens/hundreds of objects	Hundreds/thousands of objects
Tables only	Tables, views, procedures, and more
Unit testing with in-memory databases	Integration testing with regular databases

Technical Problems to Address

1) To facilitate editing, reviewing, and deploying objects for both simple and complex projects

2) To integrate well with unit-testing and integration-testing tools

3) To onboard existing production systems with ease

Object-Based Code Organization

Overview of the Benefits

Database Schema

```
/table
  Employee
  Department
  Project
  Committee
/view
  V_Manager
  V_CommManager
  V_Consultant
```

Obevo File Representation

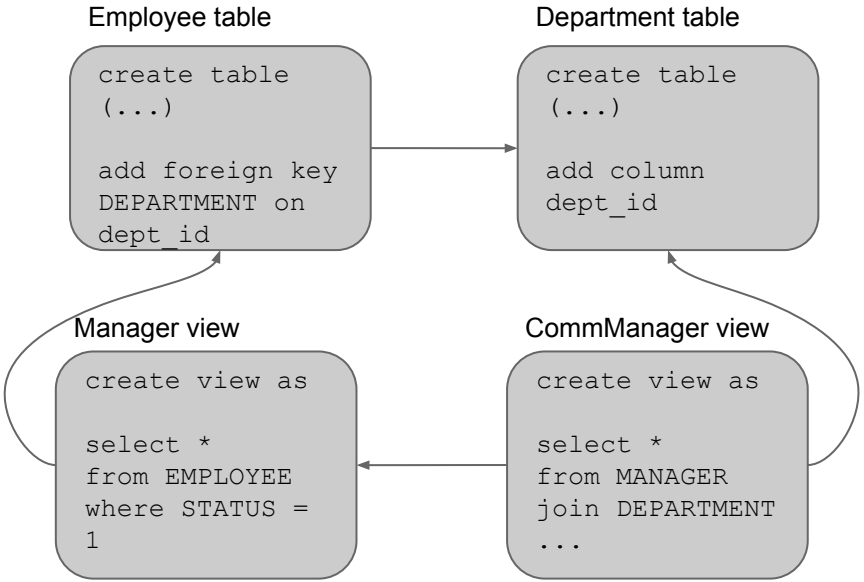
```
/table
  Employee.sql
  Department.sql
  Project.sql
  Committee.sql
/view
  V_Manager.sql
  V_CommManager.sql
  V_Consultant.sql
```

Similar structure in database as in code

Easy to find which file to edit/review/clean/deploy for a particular object (object name == file name)

Can selectively deploy subset of schema for unit/integration testing

Revisiting the Challenges



How to handle multiple scripts for stateful objects?

How to order scripts?

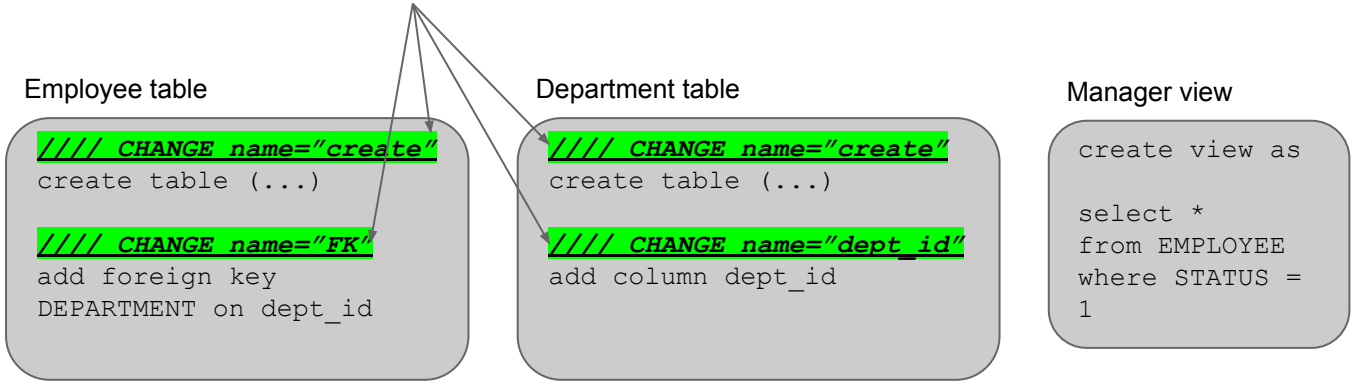
How to discover dependencies?

Handling Stateful Objects

Break up stateful files into multiple sections

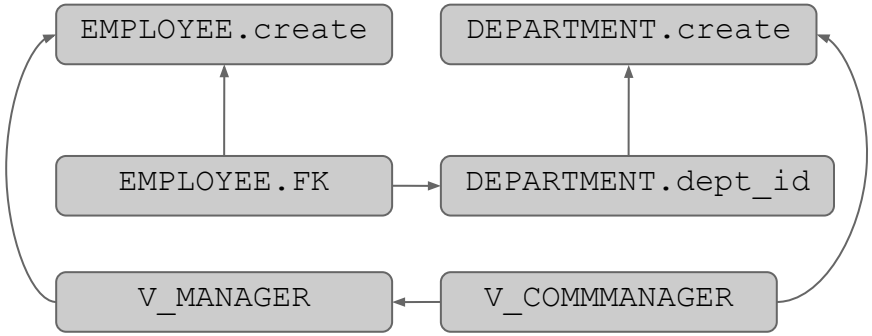
(Stateless files can remain as is)

Change Key = Object Name + Change Name



Topological Sorting for Ordering

Algorithm for ordering nodes in a graph that respect edge dependencies



Great! Now how do we actually determine these dependencies?
I have to parse my DBMS syntax!

Acceptable Orderings:

One example:

1. DEPARTMENT.create
2. DEPARTMENT.dept_id
3. EMPLOYEE.create
4. EMPLOYEE.FK
5. V_MANAGER
6. V_COMMMANAGER

Another example:

1. EMPLOYEE.create
2. V_MANAGER
3. DEPARTMENT.create
4. DEPARTMENT.dept_id
5. V_COMMMANAGER
6. EMPLOYEE.FK

... and many more

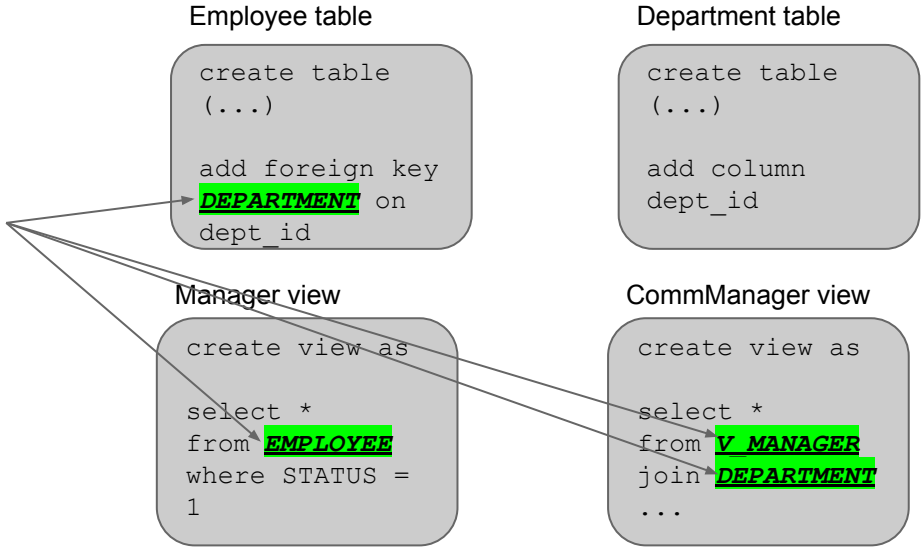
Dependency Discovery

Low-tech solution: text-search for object names

Allow overriding false positives via annotations

Object Names:

1. EMPLOYEE
2. DEPARTMENT
3. V_MANAGER
4. V_COMMMANAGER

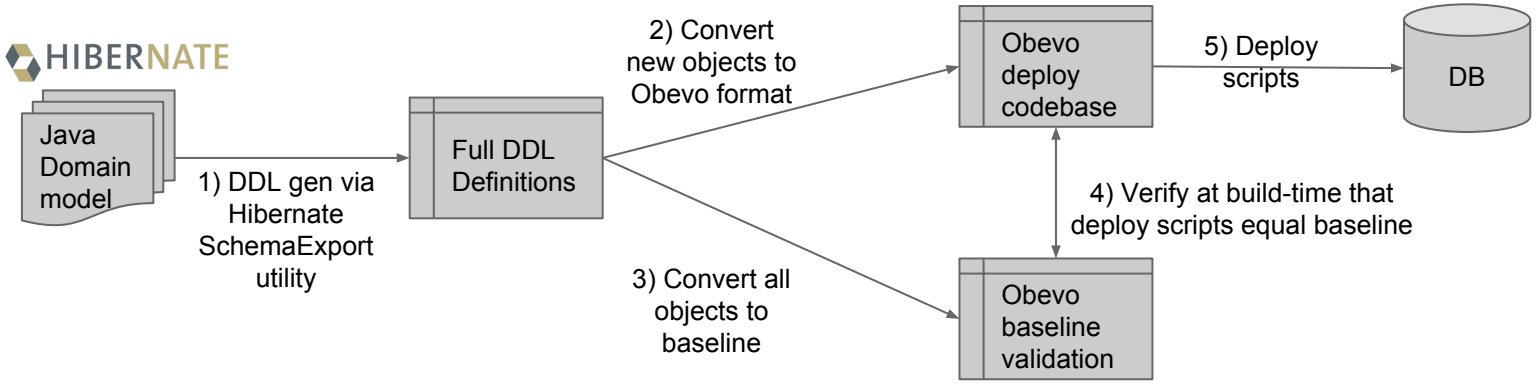


Developer Integrations

ORM Integration

ORMs can generate latest view of DDLs, but more difficulty with migration scripts

Rely on Obevo for deployment and verification that deployed DDLs match your ORM model

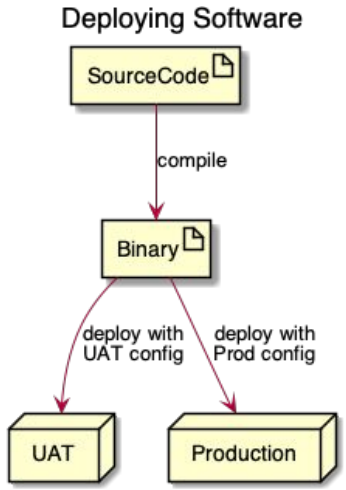


In-memory Testing

In-memory databases have grown in popularity for unit testing

How to test table DDLs in unit tests?

- Separate DDLs for in-memory DBs (but original DBs are not tested)
- Use another language from SQL for migrations (but lose out on SQL ecosystem)
- What about a translation layer?



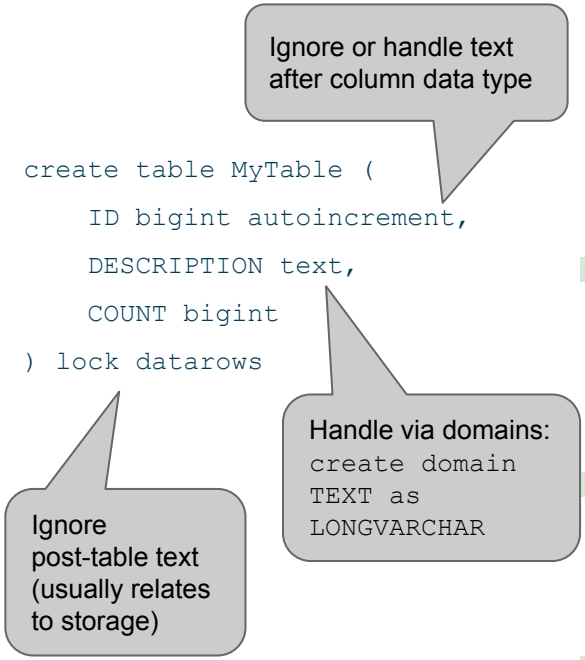
In-memory DB Translation Layer

Focus on preserving the main object structure, avoid DBMS-specific values

Use ASTs to extract clauses from SQL; handle or ignore irrelevant sections

Limited to certain object types (e.g. tables, views, but no procedures)

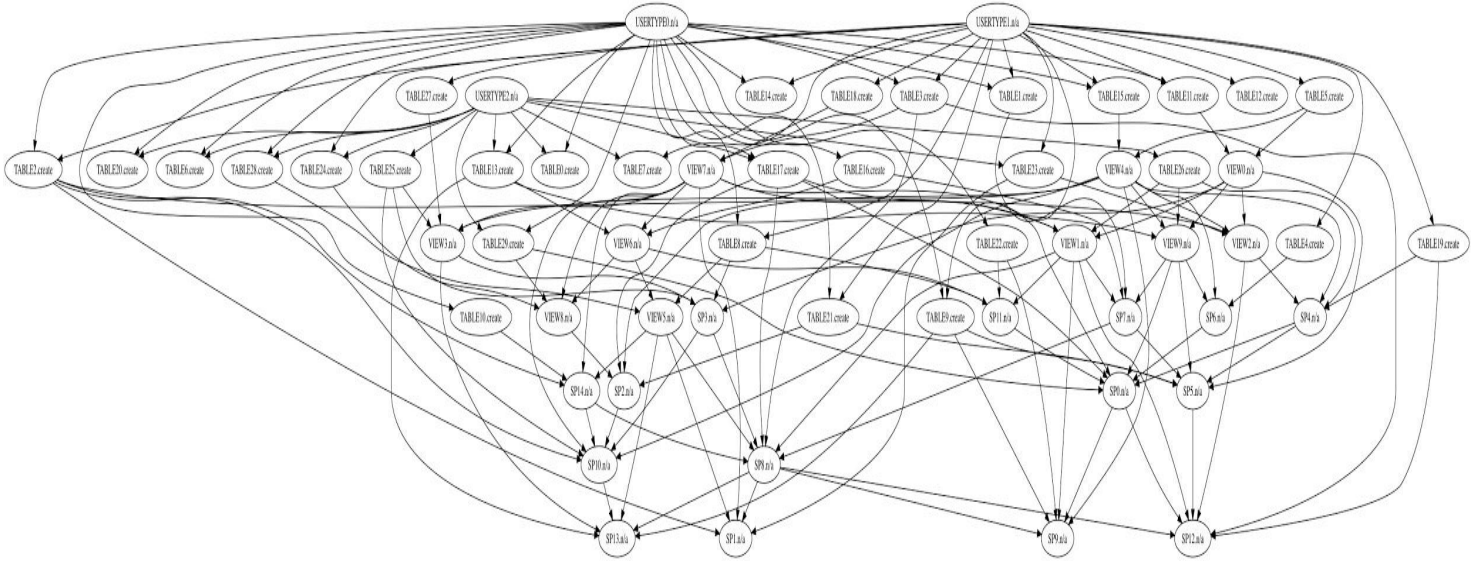
Users can fall back to custom SQL if needed



Onboarding Existing Systems

Reverse Engineering

How to onboard this system?



Reverse Engineering

Prefer to reverse-engineer all objects in a schema

No strong standard API available that can generate object definitions across DBMS types

- JDBC Metadata is inconsistently implemented
- SchemaCrawler API is a good start, but not at a sufficient level for reverse engineering

Reverse Engineering – Approach

Obevo leverages vendor-provided APIs, e.g. pg_dump, DB2LOOK, Oracle DBMS_METADATA

Most APIs only provide text output

Obevo has text-parsing logic to convert that output to the Obevo folder structure

- This is an easier problem to solve than to try a Java-based metadata API

Other Topics (beyond scope of talk)

Centralized permission management and cleanup

Rollback

Phased deployments

Long-running deployments and index creation

DB2 REORG handling

… and more …

Code Kata

Onto the Kata!

https://github.com/goldmansachs/obevo-kata/

THANKS!

Any questions?

You can find me at:

@shantstepanian

shant.p.stepanian@gmail.com

CREDITS

Special shout-outs to some useful open-sourced products leveraged along the way:

- SchemaCrawler for DB API access
- JGraphT for graph algorithm implementations
- SlidesCarnival for this presentation template