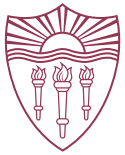
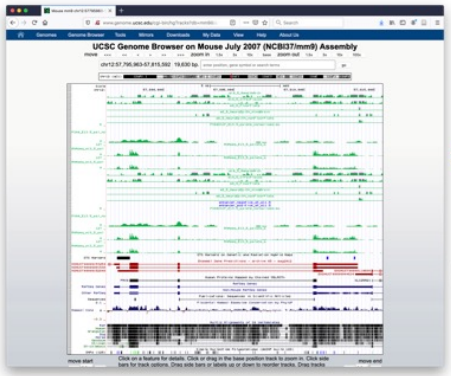


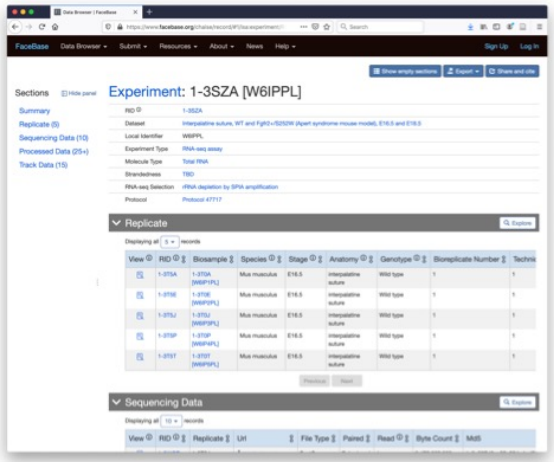
MANAGING DATABASE-APPLICATION CO-EVOLUTION IN A SCIENTIFIC DATA ECOSYSTEM

Robert Schuler and Carl Kesselman
USC Information Sciences Institute
IEEE eScience 2022, Salt Lake City, Utah, USA
October 13, 2022

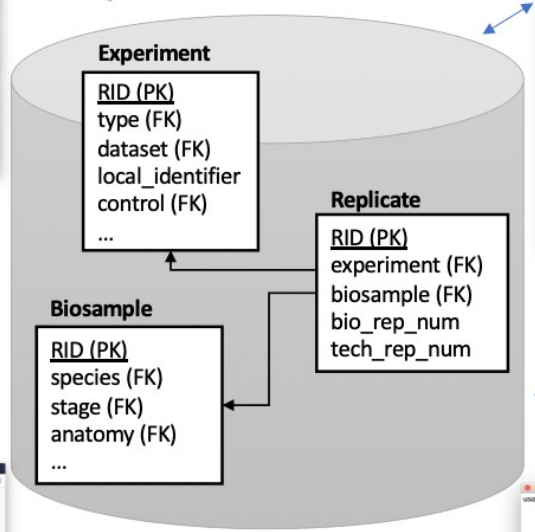
A Typical Data Ecosystem for Science Involves Numerous Database-Oriented Applications

Genomic visualization services

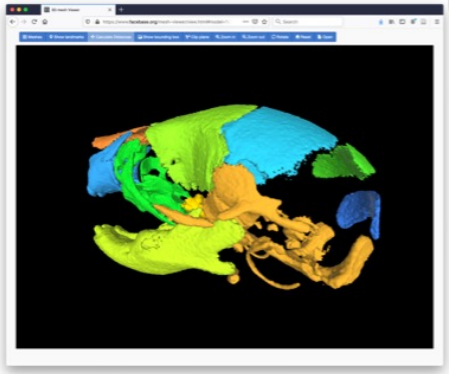


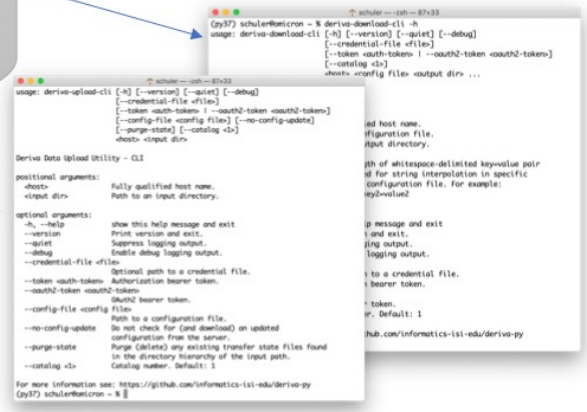


Data entry forms and search interfaces



Anatomic visualization clients





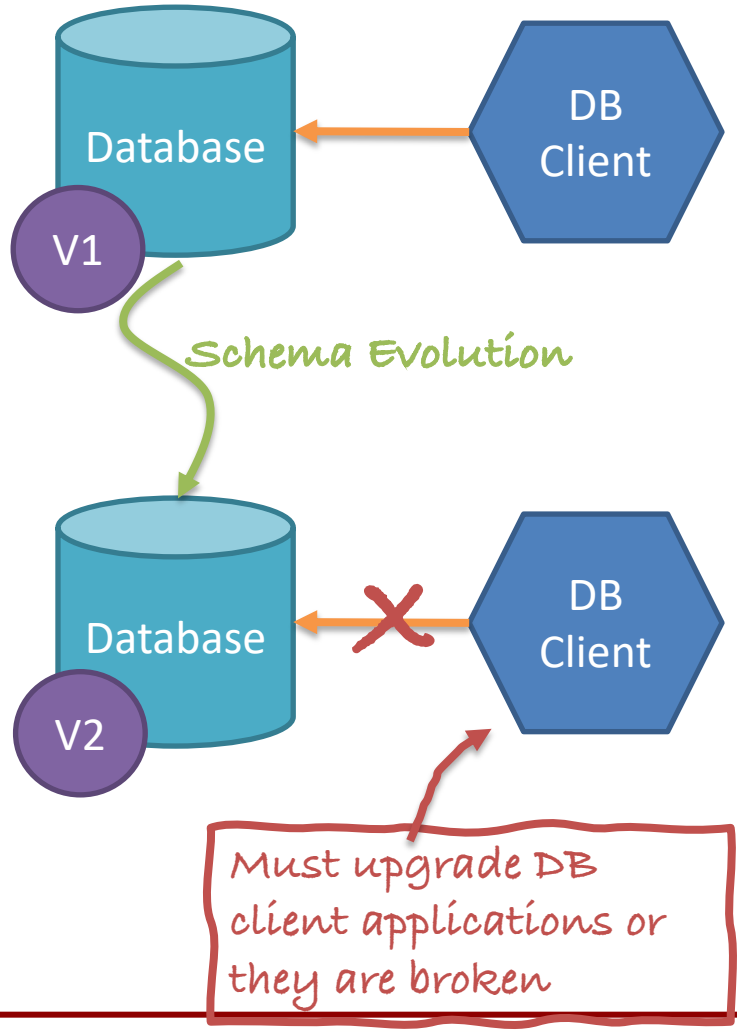
Bag (.zip)

Bulk data upload and downloads

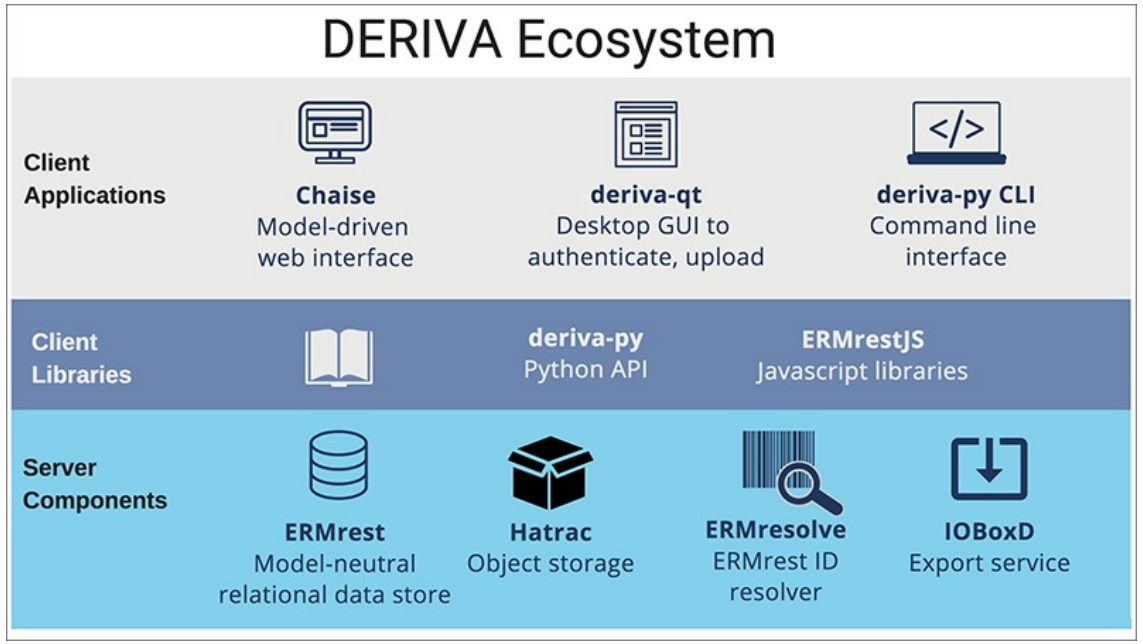
Data exchange bundles

Database Evolution *Tied* to Application Evolution

- *Science is rapidly changing, new requirements force update to database schema and applications*
- By some estimates, initial database designs become ***outdated in months***
- Databases typically do not exist in a vacuum. Rather they support an ***ecosystem of applications***
- …this leads to the ***application-database co-evolution problem***:
  - The evolution of databases is intrinsically tied to the evolution of applications; applications must be upgrade or database integrity must be compromised



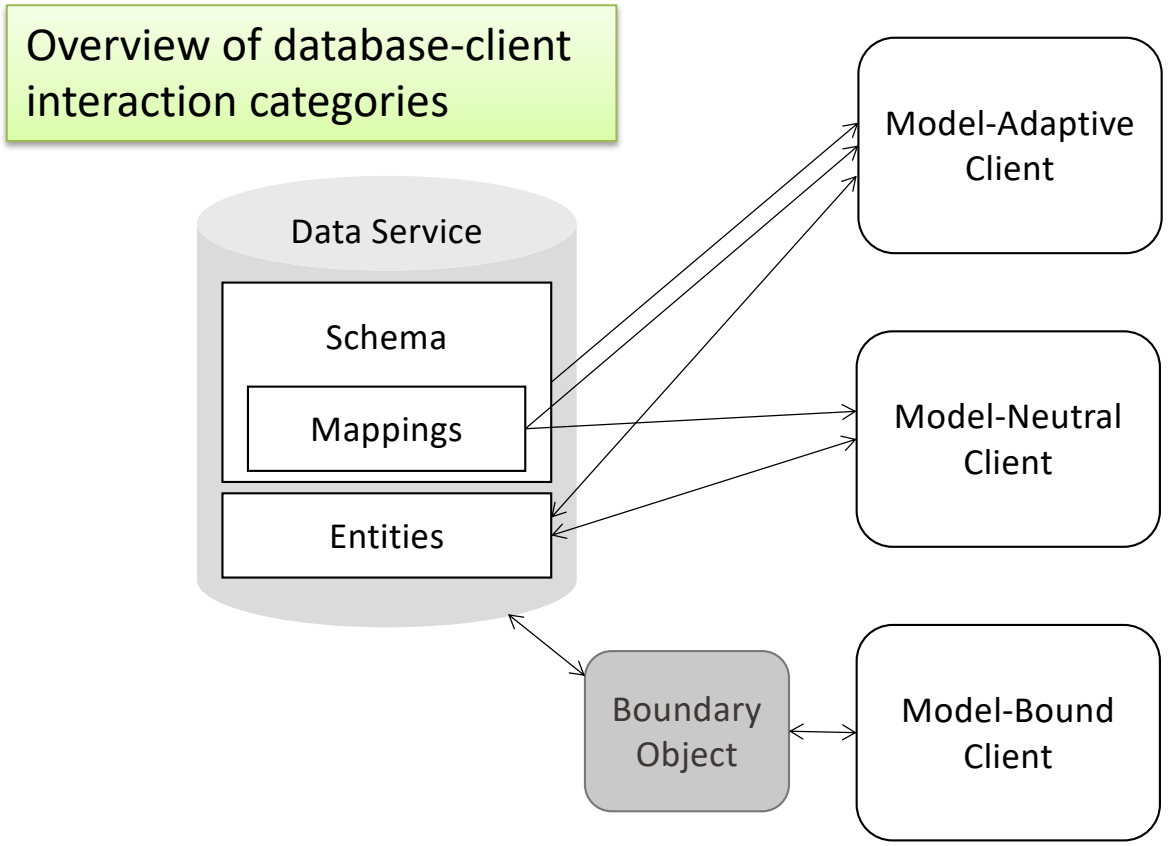
DERIVA: Platform for Scientific Data Ecosystems To Support Full Lifecycle of Scientific Inquiry



https://deriva.isi.edu

- In development and usage since 2014 on mid- to large-scale scientific applications
- Relational model-based approach to managing scientific "assets" (i.e., data objects)
- Components adapt to the model, designed for rapid schema changes
- *Scientists and research engineers are intended users (not only DBA)*

Characteristics of Database-Client Interactions W.R.T. Schema Dependencies (in DERIVA)

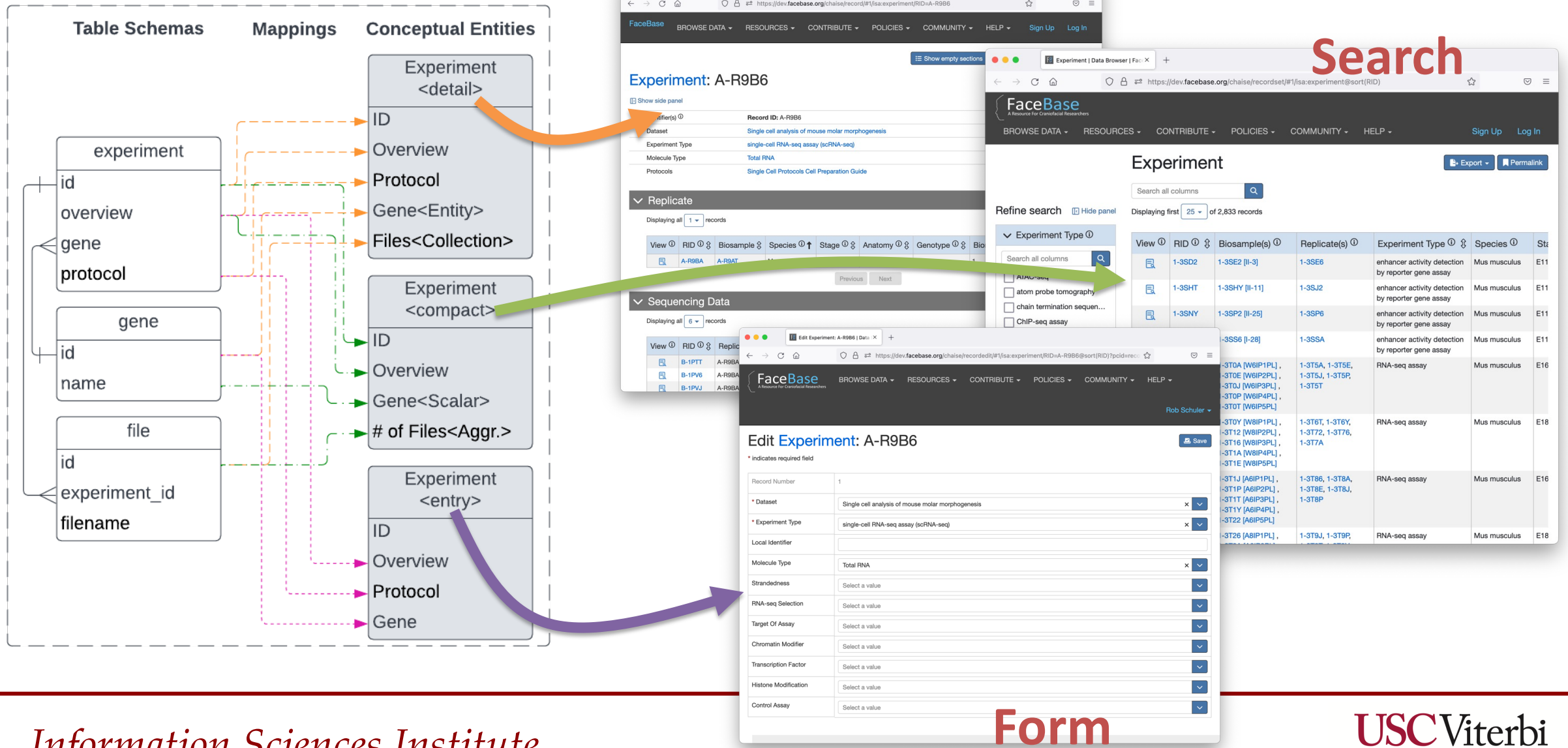


- **Model *mappings***: specification of how to convert data from one model to another; somewhat analogous to Object-Relational Mapping (ORM)

- Database interaction styles**:**
  - **Model-*adaptive***: use model mappings to determine how to query & update
  - **Model-*neutral***: use query templates to insulate client from schema changes
  - **Model-*bound***: fixed schema and need an intermediary layer

Schuler et al., *SSDBM*, 2020.

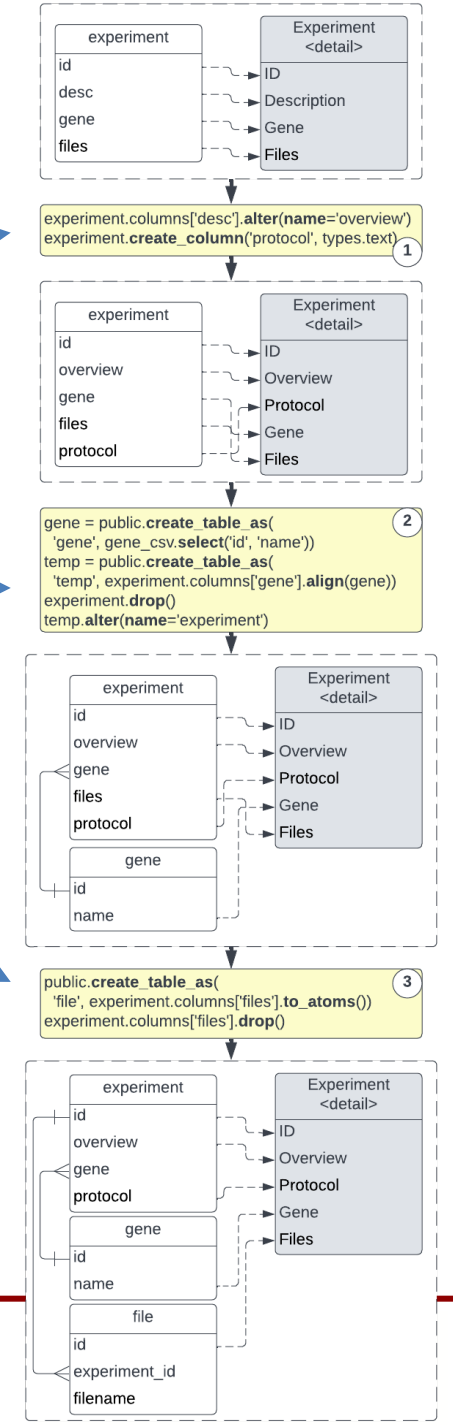
Schema is Mapped to Multiple Usage "Contexts"



Evolution of Schema Must be Coupled with Evolution of the Model Mappings

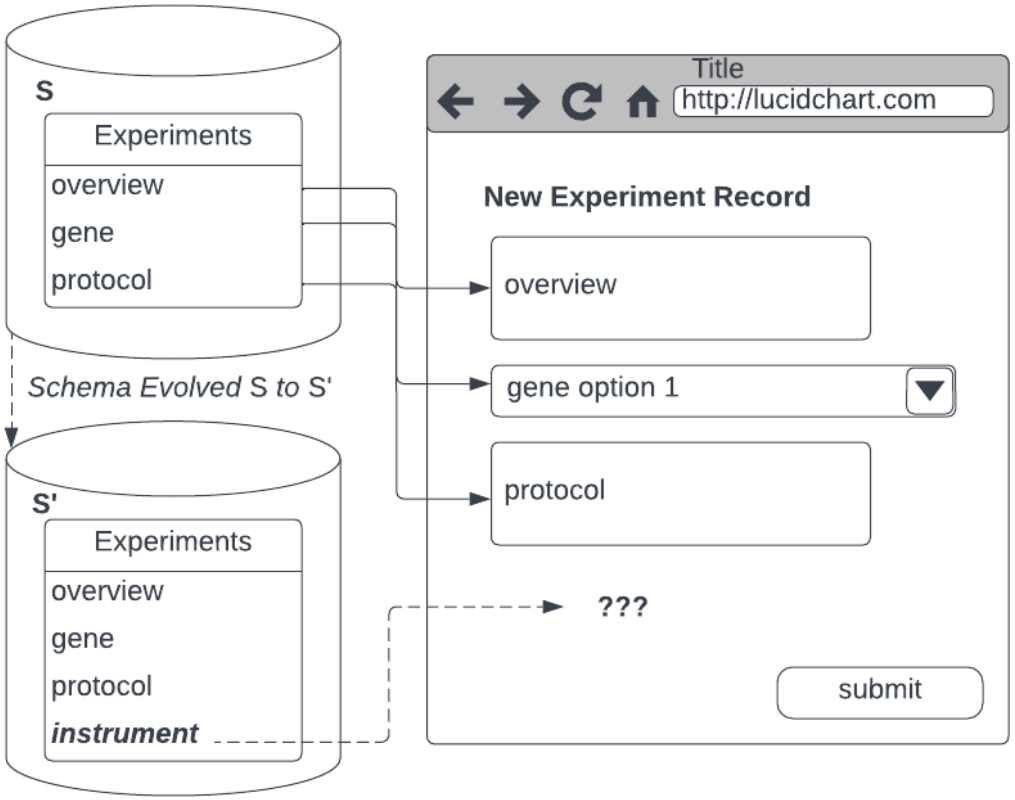
Example schema evolution:

1. Adding a new column (Protocol);
2. Refactoring a text column (Gene) into a table of controlled vocabulary terms;
3. Normalizing an array column (Files) into a separate table with row per filename entry.



Each of these schema changes must result in application updates to (1) view a new field, (2) drop down term picker in forms, (3) view and navigate to a related table, etc.

Need *Coordinated* Changes to Model Mappings



- Objective is not only to preserve existing mapping from the schema to the application model
- Objective is complementary changes to the application model in a coordinated manner with changes to the schema
- Simple Example: add *instrument* attribute to Experiment table → need *instrument* field in data entry form

Current Approaches to “Model Management”

- Model Management Operator (MMO): a class of operators for manipulating models and their mappings. Example MMOs:
  - Match: infer mapping
  - Compose: combine mappings
  - Invert: reverse a mapping
  - Diff: of model not part of mapping
  - Merge: combine schemas
  - ModelGen: change meta-model

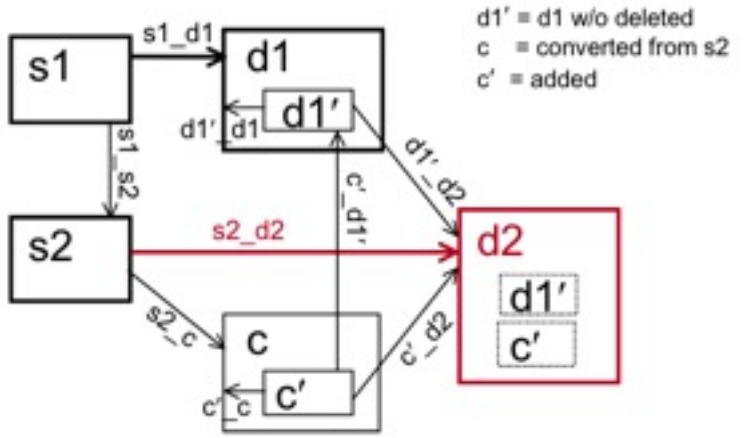


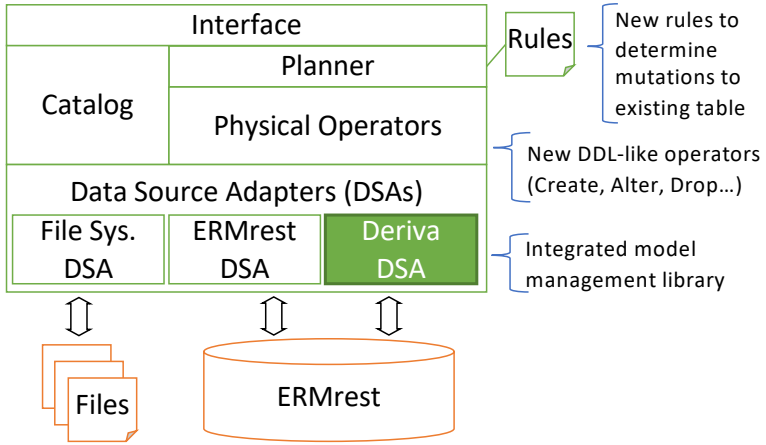
Figure 2: Schematic representation of a solution for change propagation scenario of Figure 1

– *(figure)* Melnik et al., *SIGMOD*, 2003.

- Assume schema is versioned *out-of-band*

Approach: Integrate Model Management in a Database Evolution Language for Coupled Evolution of DB and Apps

CHiSEL: a *user-oriented framework* for schema evolution, based on an algebra of relational schema modification operators, geared toward usage by scientists and research engineers.



CHiSEL architecture enhanced with MMOs

- **DEL**: *database evolution language* sibling to other SQL dialects (DQL, DML, DDL, etc.)
- **SMO**: *schema modification operator* in a DEL that performs a transformation of schema *and* data migration
- **MMO**: *model management operator* for manipulating model mappings following schema changes

Declarative Mappings from a Source Model (e.g., DB Schema) to a Target Model (e.g., Application Concepts)

- Schema Annotations
  - Semantics about the schema that go beyond the traditional ERM
- Schema Mappings
  - Declarative expressions for translating from a set of table definitions to an application model
- Mapping Fragments
  - A.k.a., model correspondences; Expressive equivalent to a *CQ* (conjunctive query) language

example.org/ermrest/catalog/1/schema/isa/table/dataset

```
…
"tag:isrd.isi.edu,2016:visible-columns": {
  "entry": [
   ["isa", "dataset_project_fkey"],
   "accession",
   "title", …],
  "compact": [
   ["isa", "dataset_RID_key"],
   "title",
   { "entity": false,
     "source": [
      {"inbound": ["isa", "dataset_experiment_…fkey1"]},
      {"outbound": ["isa", "dataset_experiment_…_fkey2"]},
     "name"],
   },…
```

Potentially complex paths

More Formal Semantics for Such a *Mapping System*…

- *S*: schema of database *D*
- *T*: conceptual (application) model
- *M*: schema mapping from *S* to *T;* set of mappings *v* from table schema to conceptual entity
- *v*: mapping from table schema to conceptual entity in T; as a set of mapping fragments q
- *q*: mapping fragment for a single attribute value represented as set of symbols *s*
- *s*: symbolic representation of a column or (foreign) key constraints

example.org/ermrest/catalog/1/schema/isa/table/dataset

```
…
"tag:isrd.isi.edu,2016:visible-columns": {
   "entry": [
    ["isa", "dataset_project_fkey"],
    "accession",
    "title", …],
   "compact": [
    ["isa", "dataset_RID_key"],
    "title",
    { "entity": false,
      "source": [
       {"inbound": ["isa", "dataset_experiment_...fkey1"]},
       {"outbound": ["isa", "dataset_experiment_..._fkey2"]},
      "name"],
    },…
```

What are the Complementary MMOs for a DEL?

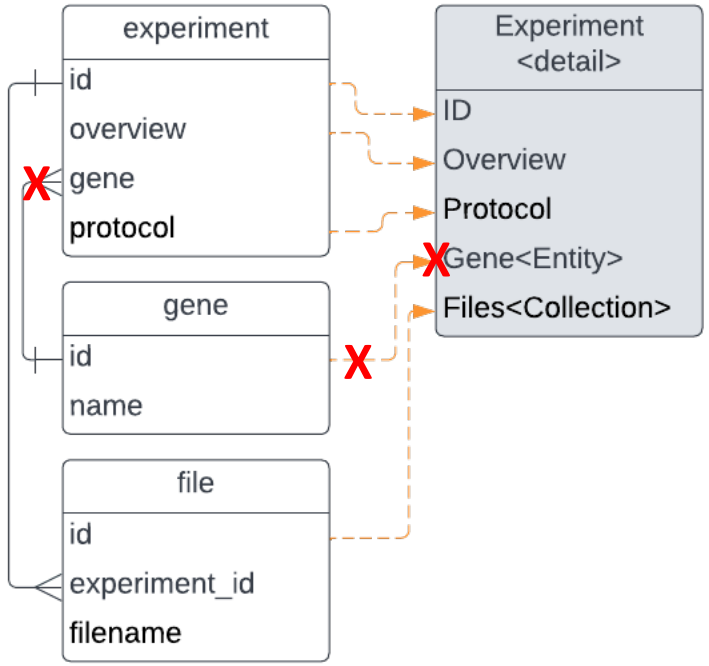
MMO	Complements	Semantics
Union	Create Table	Combines set of mappings
Difference	Drop Table	Difference of sets of mappings
Graft	Add a column or constraint	Introduce mapping fragment into a mapping
Prune	Drop column or constraint	Removes any mapping fragment in model where symbol is present
Replace	Rename column or constraint	Swaps a new symbol name for a replaced symbol throughout all mapping fragments in model

Prune Example

1) **Drop** foreign key on gene (id)

2) Mapping fragment experiment → gene invalidated

3) **Prune** mapping and Gene<Entity> from conceptual entity



What is the MMO Complement for an SMO Expression?

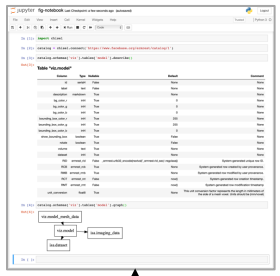
- *SMO expressions* equivalent to "**CREATE TABLE … AS …**" statements; i.e., defining a new table based on an expression

- CHiSEL SMOs include *Select*, *Project*, *Join*, *Union*, …
- Unlike conventional relational algebra, *SMOs output a full schema definition* for the relation

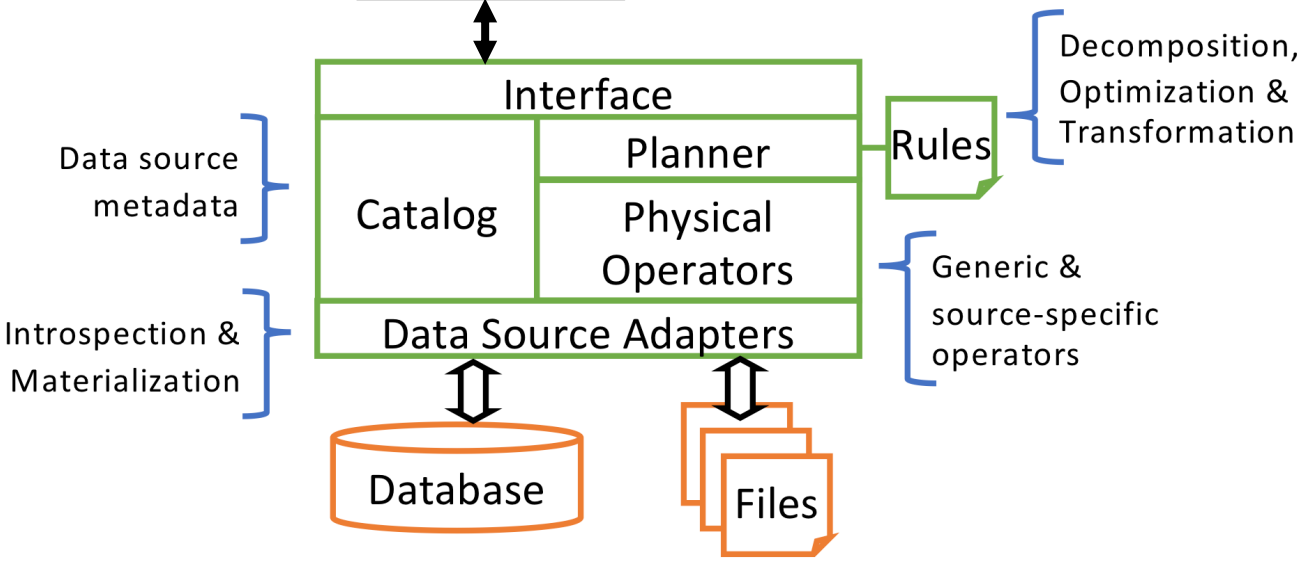
- **Morph**: a general transformation algorithm to complement SMO expressions:

```
Input: relation r, attributes A
1. Identify all columns C in r
   satisfied by A
2. Preserve all key K and f. key F
   constraints in r satisfied by C
3. Replace mappings for all renamed
   columns C and constraints (K, F)
4. Prune mappings for all dropped
   columns C and constraints (K, F)
Output: (C, K, F, mappings)
```

CHiSEL Is Delivered as a Python Library That Interfaces With Underlying Data Sources



Interactive Environment (Jupyter Notebook)



High-Level Architecture of CHiSEL (green)

Schuler and Kesselman, *SSDBM*, 2019.

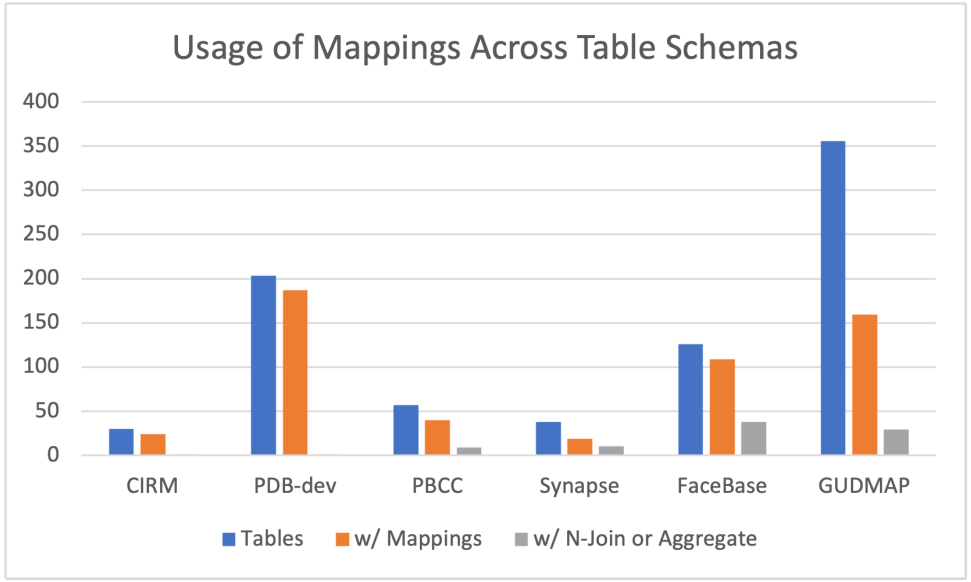
- Reduce learning curve by **embedding in Python** programming environment as a domain specific language
- **Interactive** environment support (e.g., Jupyter Notebook)
- Data source **adapters**
  - Relational and semi-structured
- Data source **federation**
  - Expressions over multiple sources
- For **efficiency**, push down operators to sources, fall back on core set of algorithms for data manipulation

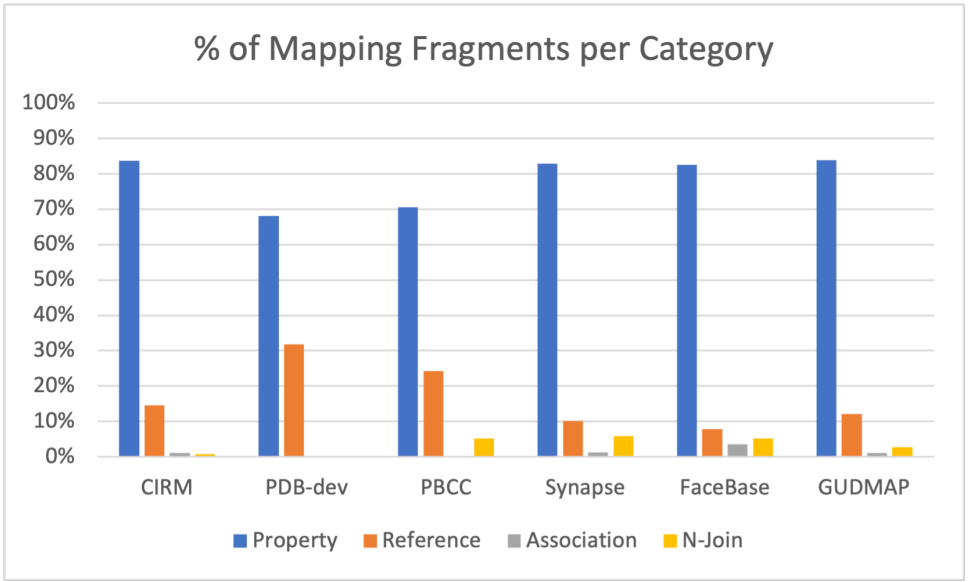
Evaluation of Model Mappings in DERIVA Deployments

- Examined the usage and characteristics of model mappings in several production deployments of DERIVA that support ongoing scientific applications
- Breadth of usage, Type of mappings, Complexity of mappings, Mapping errors

Deployment	Description
CIRM	A microscopy Core Facility supporting regenerative medicine particularly with respect to kidney disease.
PDB-dev	A prototype archiving system for structural models collected by the Protein DataBase (PDB).
PBCC	A research consortium centered on pancreatic betacell modeling.
Synapse	A collaborative investigation seeking to map the whole synaptome of a model organism.
FaceBase	The data resource for an international craniofacial research.
GUDMAP	The data resource for the Genito-Urinary Molecular Atlas Project (GUDMAP) and Re-Building a Kidney (RBK) consortia.

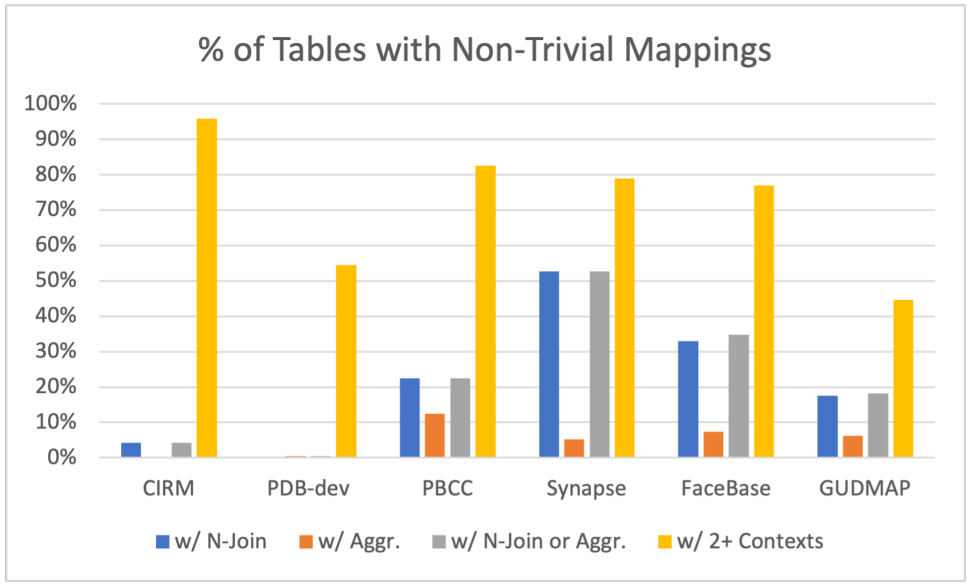
Large Proportion of Tables Mapped to a Display, Edit, Search, or Other Application Context

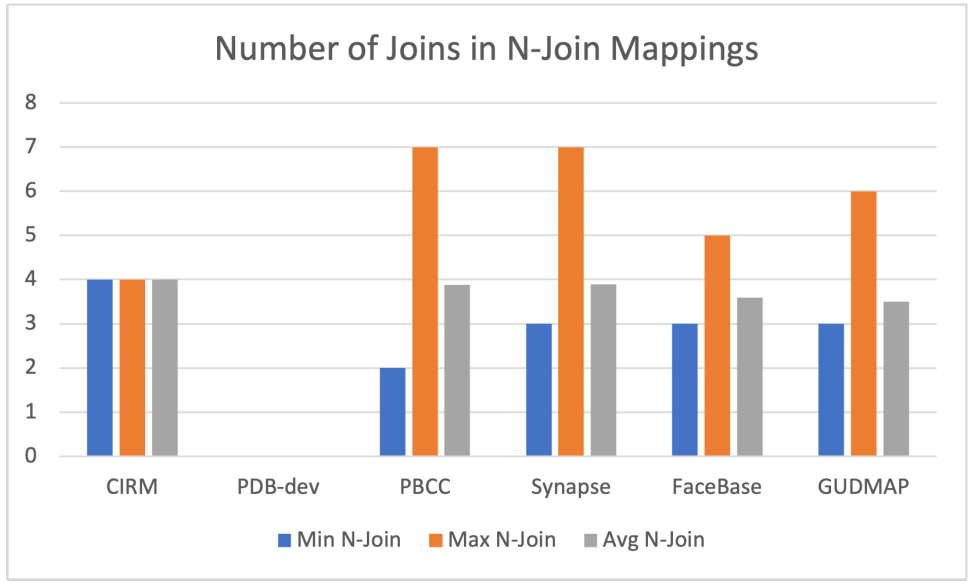




While many “mapping fragments” involve simple table column to application attribute mapping, a significant proportion involve inbound/outbound references, association table traversal, or multiple table joins to map the desired attributes.

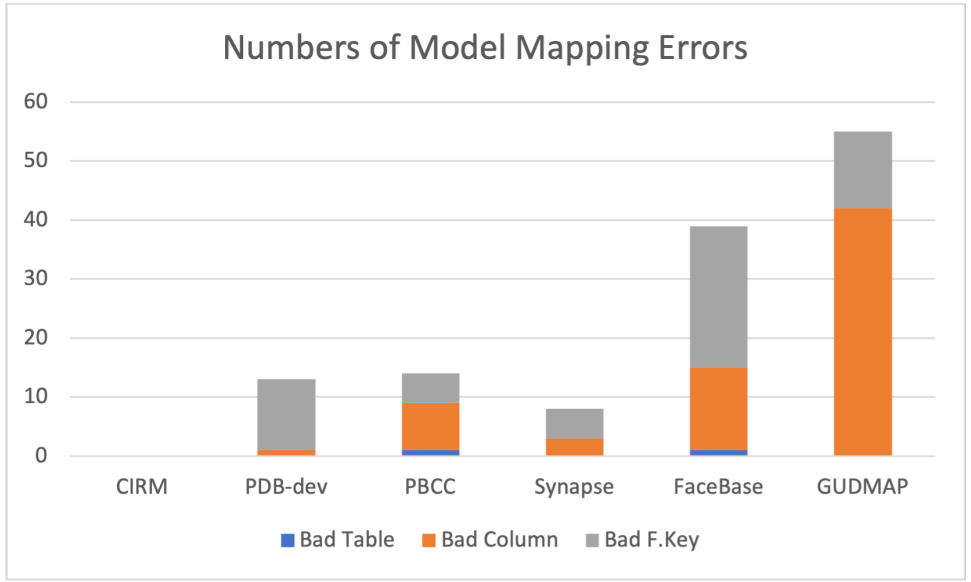
Tables Generally Mapped to Multiple “Contexts” with Substantial Usage of Non-Trivial Mappings





Most tables are mapped to more than a single application “context” (e.g., edit, browse, search) and many mapping specifications rely on non-trivial joins involving multiple foreign key relationship traversal.

Mapping Errors Found in Most Deployments



- Some human error attributed to hand crafted mapping specifications in schema annotations
- Other errors involve schema changes prior to usage of integrated SMO+MMO approach
- No errors in CIRM deployment attributed to a relatively older and static database

Comparison between CHiSEL and EF Core Migrate

- Extended from motivating example – realistic but hypothetical evolution of a DB
- `Migrate` generally resulted in data loss for most scenarios without user data migration code
  - Complex changes involving multiple tables or relationships also not supported
  - Even simple changes (e.g., table rename) would result in data loss
  - Limited semantics about changes

TABLE VI
EVALUATION OF SCHEMA (**S**), DATA (**D**), AND MODEL (**M**) EVOLUTION BETWEEN CHISEL AND EF CORE MIGRATE.

Type of Operation	CHiSEL			EF Core		
	S	*D*	*M*	*S*	*D*	*M*
1. Create Table	x	n/a	x	x	n/a	x
2. Rename Table	x	x	x	*	*	x
3. Rename Column	x	x	x	x	x	x
4. Add Column	x	n/a	x	x	n/a	x
5. Reify	x	x	x	*	-	x
6. Change Cardinality	x	x	x	*	-	x
7. Normalize	x	x	x	*	-	x
8. Create Vocabulary	x	x	x	x	-	x
9. Align to Vocabulary	x	x	x	*	-	x

Code available:
https://drive.google.com/file/d/1BIK5-lz02Z21QLaKNZRzh71uU65PtcnB

(x) supported
(-) data loss
(*) incomplete

Pilot Project for Earthquake Forecast Model Use Case

- Southern California Earthquake Center (SCEC) use case for Epidemic-Type Aftershock Sequence (ETAS) models
- Research Staff Engineer, not a DBA or DB developer
- Iterative development of DERIVA catalog
- Developed in familiar Jupyter Notebook environment
- Other trials currently underway

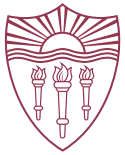
chisel-etas-gist.py — Raw

```python
"""Minimal example of a possible rendering of the ETAS starter model.
"""
from deriva.core import DerivaServer, get_credential
from deriva.chisel import Model, Schema, Table, Column, Key, ForeignKey, builtin_types, tag

# Connect to server and catalog ----------------------------------------------------------------#

hostname = 'TBD.derivacloud.org'    # this is a dev server for prototype work (change to TBD)
catalog_id = 'TBD'                  # this was a catalog used to test this script (change to TBD)

model = Model.from_catalog(
    DerivaServer('https', hostname, credentials=get_credential(hostname)).connect_ermrest(catalog_id)
)


# Cleanup --------------------------------------------------------------------------------------#

if 'ETAS' in model.schemas:
    # Purge anything so we can "do over" repeatedly at this point
    model.schemas['ETAS'].drop(cascade=True)


# ETAS schema ----------------------------------------------------------------------------------#

# Create the "schema" to organize tables in the catalog "model"
model.create_schema(Schema.define('ETAS'))


# ETAS Forecast --------------------------------------------------------------------------------#

# Create the tables at the "Forecast" level of the table hierarchy

# ETAS_Forecast
model.schemas['ETAS'].create_table(Table.define(  # <--- 'Table.define(...)' defines a general-purpose table
    'ETAS_Forecast',                                                    # table name
    column_defs=[                                                       # column definitions
        Column.define('Forecast_Name', builtin_types.text, nullok=False),   # column for the name of the forecast
        Column.define('Forecast_Config_JSON', builtin_types.text, annotations={  # this is for DERIVA "schema annoations" to
            tag.asset: {                                                # the "asset" annotation for enabling web upload
                'url_pattern': '/hatrac/ETAS/ETAS_Forecast/WebUpload_{{{_Forecast_Config_JSON.md5_hex}}}/{{{_Forecast_Config
                'filename_ext_filter': ['.json']
            }
        })
    ],
    key_defs=[                                                          # key definitions
        Key.define(['Forecast_Name'])                                   # each key specifies its list of columns of 1+ c
    ],
)
```

https://gist.github.com/robes/3d6153d77e3832fc4628342aee9ecb48

Concluding Remarks

- Databases and software ***exist in ecosystems*** *not in isolation*
- Co-evolution is an unavoidable reality for data-centric ecosystems
- Built on the DERIVA design approach that emphasizes *database-client interactions* around models and mappings
- Definition and implementation of a *novel set of model management operators* designed to be coordinated with schema change
- Introduced a **database evolution language** *integrated with model management operators*
- Implemented in the DERIVA platform and evaluated in real-world pilot and ongoing projects



THANK YOU!