Program synthesis for database-driven web applications

Rushi Shah
Turing Class of 2020
UT Program Analysis Research Group (UToPiA)

Motivation

- Automating web application developer tasks created when the application's underlying database schema is refactored

Programmer Tasks

- Code migration (PLDI'19)

- Data migration (VLDB'20)

Code Migration

- Migrator
  - Programming Languages Design & Implementation ‘19
  - Yuepeng Wang, James Dong, Rushi Shah, Isil Dillig

PLDI
Phoenix 2019

Motivating Example

- Instructor
  - Instructor ID
  - Instructor Name
  - Instructor Picture
- TA
  - TA Id
  - TA Name
  - TA Pic

```
update addInstructor(id, name, pic)
     INSERT INTO Instructor
          VALUES (id, name, pic)
```

Motivating Example

- Instructor
  - Instructor ID
  - Instructor Name
  - Instructor Picture
- TA
  - TA Id
  - TA Name
  - TA Pic

- Instructor
  - Instructor ID
  - Instructor Name
  - Picture ID
- TA
  - TA Id
  - TA Name
  - Picture ID
- Picture
  - Picture ID
  - Picture

```
update addInstructor(id, name, pic)
    INSERT INTO Instructor
        VALUES (id, name, pic)
```

Motivating Example

- Instructor
  - Instructor ID
  - Instructor Name
  - Instructor Picture
- TA
  - TA Id
  - TA Name
  - TA Pic

→

- Instructor
  - Instructor ID
  - Instructor Name
  - Picture ID
- TA
  - TA Id
  - TA Name
  - Picture ID
- Picture
  - Picture ID
  - Picture

```
update addInstructor(id, name, pic)
     INSERT INTO Instructor
          VALUES (id, name, pic)
```

→

```
update addInstructor(id, name, pic)
          ???
```

Motivating Example

- Instructor
  - Instructor ID
  - Instructor Name
  - Instructor Picture
- TA
  - TA Id
  - TA Name
  - TA Pic

→

- Instructor
  - Instructor ID
  - Instructor Name
  - Picture ID
- TA
  - TA Id
  - TA Name
  - Picture ID
- Picture
  - Picture ID
  - Picture

```
update addInstructor(id, name, pic)
     INSERT INTO Instructor
          VALUES (id, name, pic)
```

→

```
update addInstructor(id, name, pic)
     INSERT INTO Instructor
          VALUES (id, name, UID0)
     INSERT INTO Picture
          VALUES (UID0, pic)
```

Synthesis Methodology

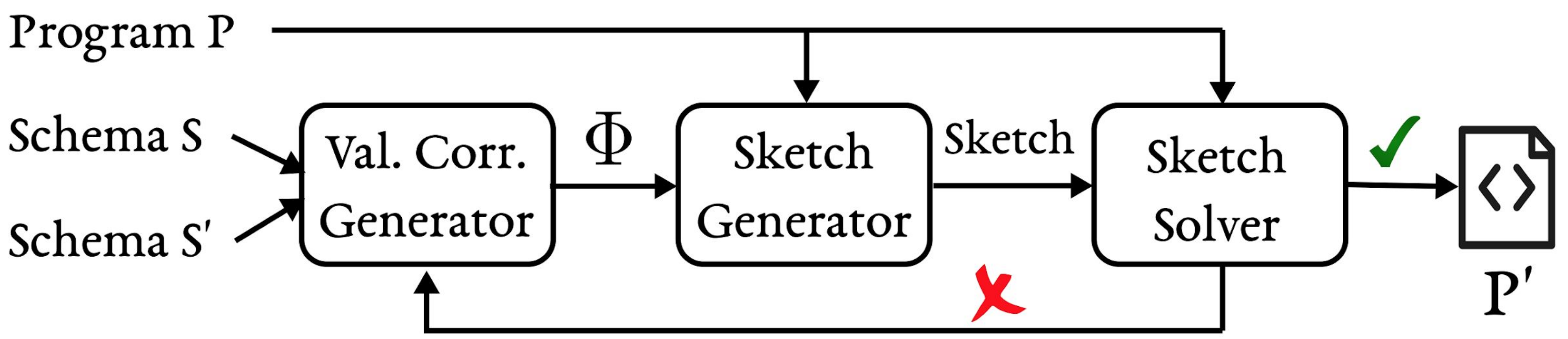


Figure 1. Synthesis methodology.

Sketch Generation

- Instructor
  - Instructor ID
  - Instructor Name
  - Instructor Picture
- TA
  - TA Id
  - TA Name
  - TA Pic

- Instructor
  - Instructor ID
  - Instructor Name
  - Picture ID
- TA
  - TA Id
  - TA Name
  - Picture ID
- Picture
  - Picture ID
  - Picture

```
update addInstructor(id, name, pic)
     INSERT INTO Instructor
          VALUES (id, name, pic)
```

```
update addInstructor(id, name, pic)
     INSERT INTO ??
          VALUES (id, name, pic)
where ?? is the choice of which tables
to insert into
```

Sketch Completion

- Naive solution :'(
- Minimum Failing Inputs (MFI)
  - Efficiently prune search space

Sketch Completion - Evaluation

Reimplemented our synthesis solution into state-of-the-art tool called "Sketch"

(My main contribution!)

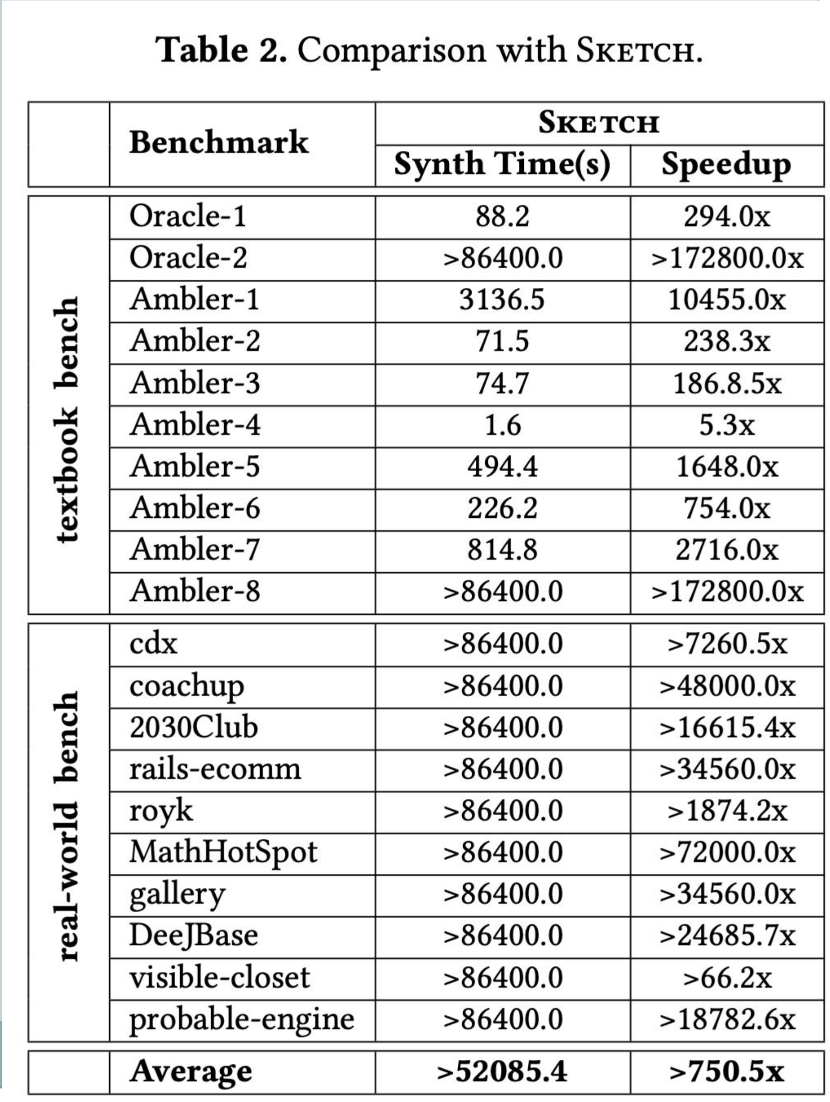
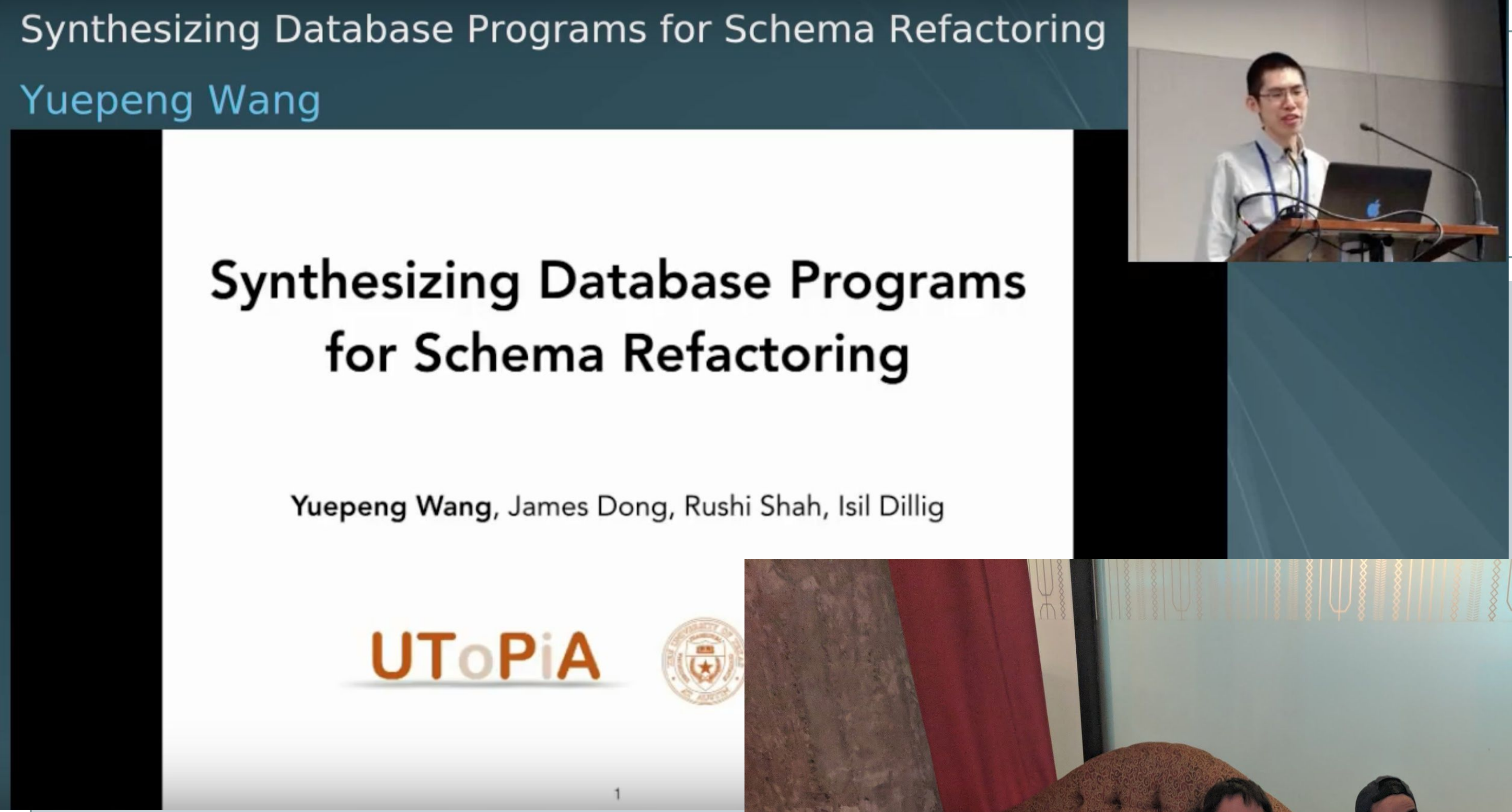


Table 2. Comparison with Sketch.

	Benchmark	Sketch	
		Synth Time(s)	**Speedup**
textbook bench	Oracle-1	88.2	294.0x
	Oracle-2	>86400.0	>172800.0x
	Ambler-1	3136.5	10455.0x
	Ambler-2	71.5	238.3x
	Ambler-3	74.7	186.8.5x
	Ambler-4	1.6	5.3x
	Ambler-5	494.4	1648.0x
	Ambler-6	226.2	754.0x
	Ambler-7	814.8	2716.0x
	Ambler-8	>86400.0	>172800.0x
real-world bench	cdx	>86400.0	>7260.5x
	coachup	>86400.0	>48000.0x
	2030Club	>86400.0	>16615.4x
	rails-ecomm	>86400.0	>34560.0x
	royk	>86400.0	>1874.2x
	MathHotSpot	>86400.0	>72000.0x
	gallery	>86400.0	>34560.0x
	DeeJBase	>86400.0	>24685.7x
	visible-closet	>86400.0	>66.2x
	probable-engine	>86400.0	>18782.6x
	Average	**>52085.4**	**>750.5x**

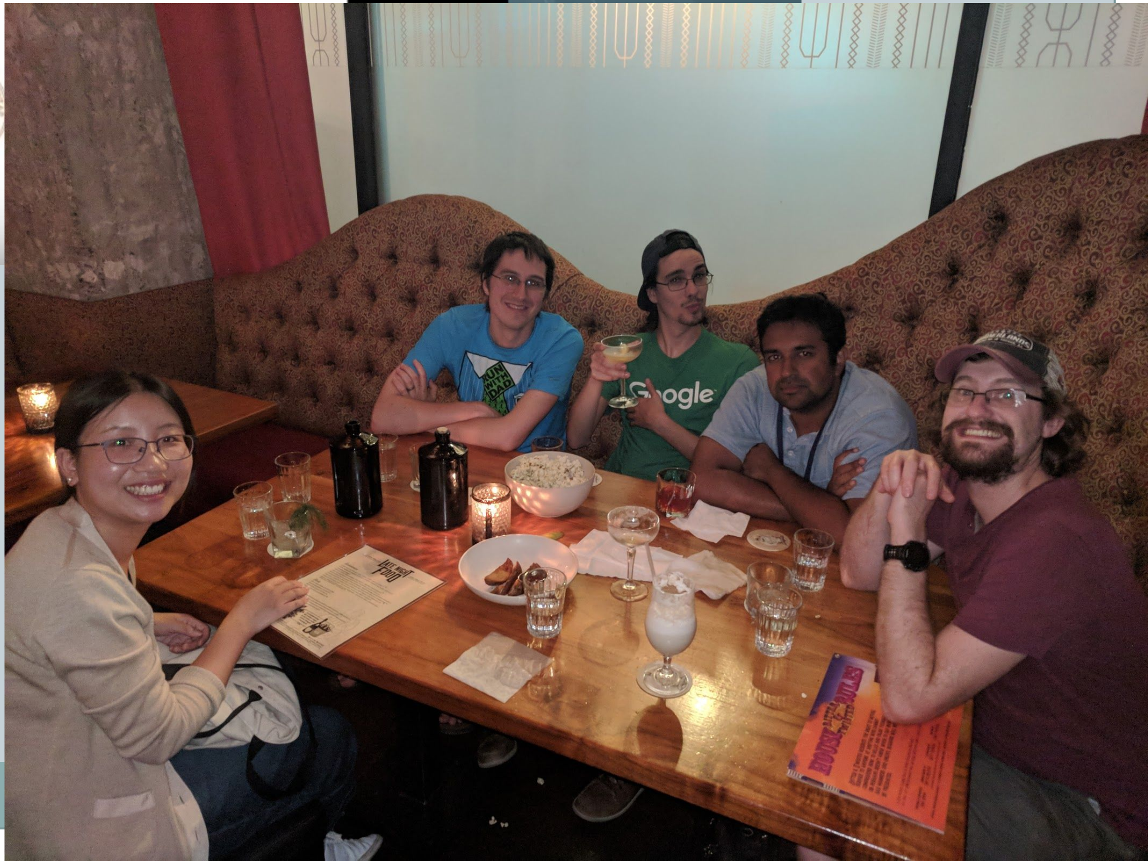


Synthesizing Database Programs for Schema Refactoring

Yuepeng Wang

Synthesizing Database Programs for Schema Refactoring

Yuepeng Wang, James Dong, Rushi Shah, Isil Dillig



Data Migration

- Dynamite
  - Very Large Data Bases ‘20
  - Yuepeng Wang, Rushi Shah, Abby Criswell, Isil Dillig

Motivating Example

```
"businesses": [
  {
    "name": "Quiznos",
    "checkins": [
      "Mon-0:1", "Fri-21:1",
      "Thu-22:1", "Mon-23:1"
    ],
    [...]
  },
  {
    "name": "My Thai",
    "checkins": [
      "Fri-17:1", "Wed-22:1"
    ],
    [...]
  }
]
```

```
Business(name, checkin_id, [...])
"Quiznos", "fk_quiznos", [...]
"My Thai", "fk_thai", [...]

Checkins(business_id, checkin)
"fk_quiznos", "Mon-0:1"
"fk_quiznos", "Fri-21:1"
"fk_quiznos", "Thu-22:1"
"fk_quiznos", "Mon-23:1"
"fk_thai", "Fri-17:1"
"fk_thai", "Wed-22:1"
```

Datalog Background

- Facts
  - `parent(rahil, arjun)`
  - `parent(arjun, divya)`
- Rules
  - `child(Y, X) :- parent(X, Y)`

Datalog for Data Migration

- Example
  - `parent(bill, mary)`
  - `parent(mary, john)`
  - `child(Y, X) :- parent(X, Y)`
- Rules represent the migration relationship
  - LHS from target schema, RHS from source schema
- Facts are rows in the database to be migrated

Motivating Example

- Input
  - Source and target database schema
    - `Source: parent(adult, kid)`
    - `Target: child(kid, adult)`
  - Input/Output Example
    - `parent(rahil, arjun)` should give `child(arjun, rahil)`
  - Source database instance
    - `parent(rahil, arjun)`
    - `parent(arjun, divya)`
    - `[...]`
- Output
  - Target database instance
    - `child(arjun, rahil)`
    - `child(divya, arjun)`
    - `[...]`

Dynamite Overview

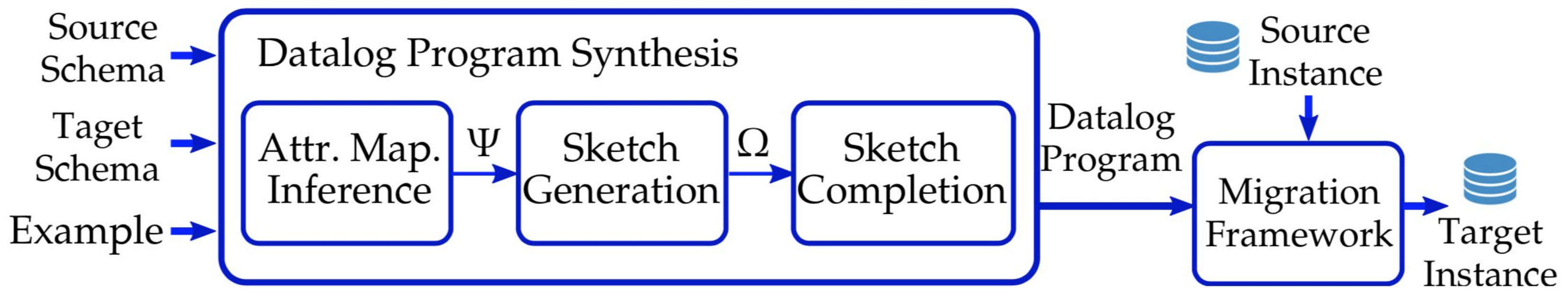


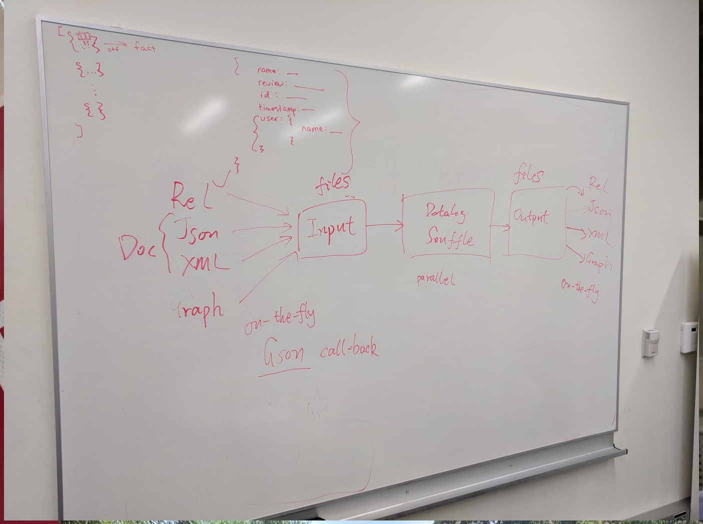
Figure 1: Schematic workflow of DYNAMITE.

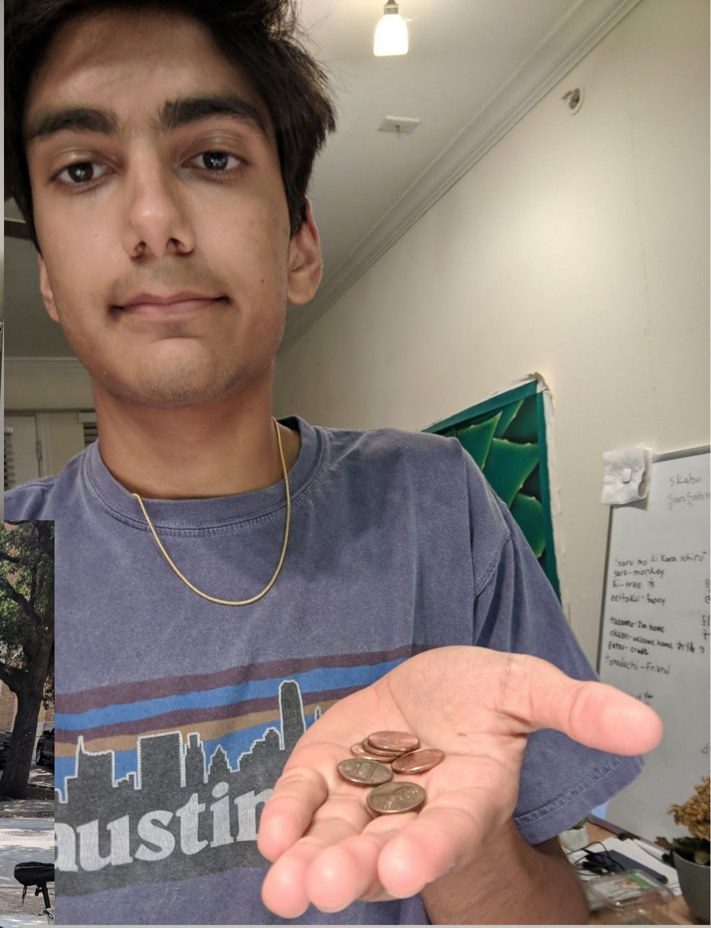
Datalog Program Synthesis

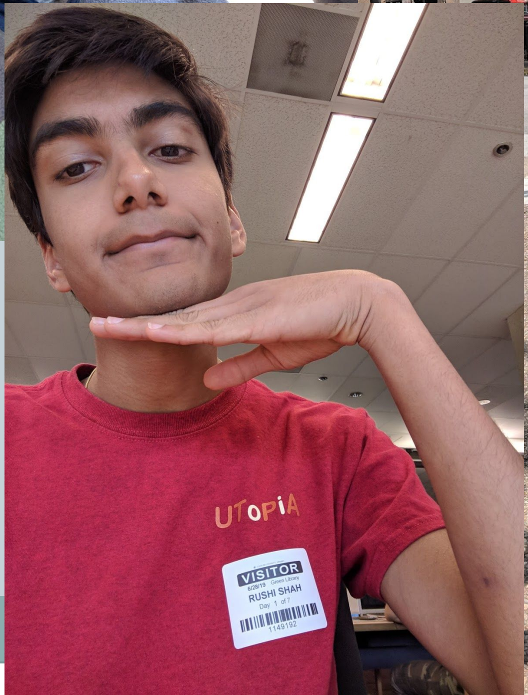
- Similar synthesis methodology
  - Generating Datalog program sketch
  - Sketch completion
    - Minimal Distinguishing Projection (MDP)
      - Efficiently prune search space
      - Basically “minimum failing inputs” from Migrator

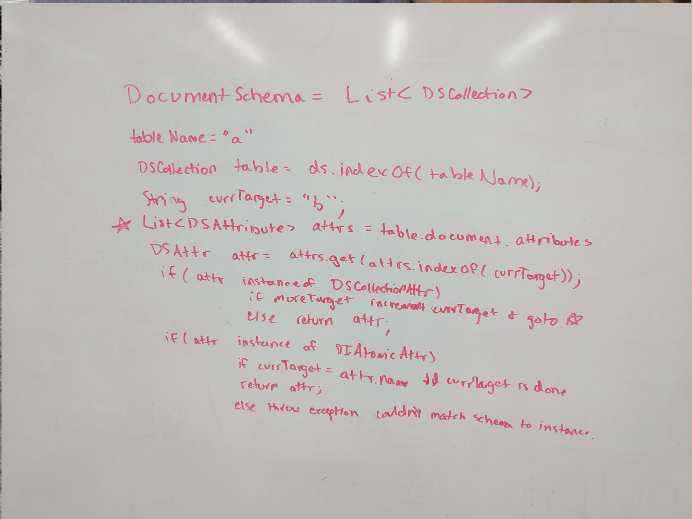
Migration Framework

- (My main contribution!)
- Actually performing the migration using the Datalog program on real-world databases is non-trivial
  - IMDB, Yelp, DBLP, etc. are "V E R Y   L A R G E"
    - Documents were recursively nested
  - Supporting relational, document, & graph database formats
    - Database instance to Datalog facts and vice versa









```
Document Schema = List< DSCollection>

table Name = "a"
 DSCollection table = ds.indexOf( table Name);
 String currTarget = "b";
☆ List<DSAttribute> attrs = table.document.attributes
  DSAttr attr = attrs.get(attrs.indexOf( currTarget));
   if ( attr instanceof DSCollectionAttr)
          if moreTarget increment currTarget & goto ☆
          else return attr;
   if ( attr instance of DIAtomicAttr)
          if currTarget = attr.name && currTarget is done
          return attr;
         else throw exception couldn't match schema to instance.
```

Conclusion

- Failed previous projects
  - SyPet -> Pythagoras
  - Database Model Checker
- Next steps
- Hook ‘em 🤘🏽