ILLINOIS INSTITUTE
OF TECHNOLOGY

IIT DBGroup

Research Profile

Boris Glavic

Outline

1) **Who am I?**
2) Provenance
3) Next-generation DB Engines
4) Foundations for Data Science

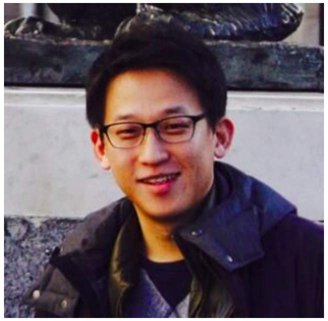
Boris Glavic

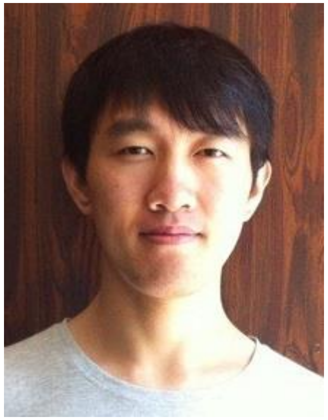
I am a **database** guy!

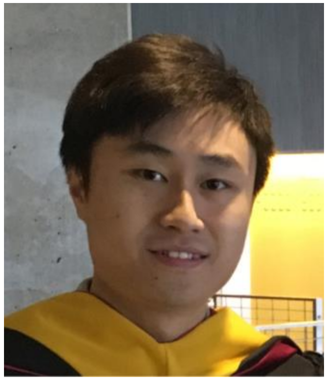
Hi, I am **Boris Glavic,**
Professor in
CS

IIT DBGroup

- **My research group**
  - http://www.cs.iit.edu/~dbgroup/

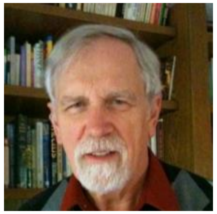


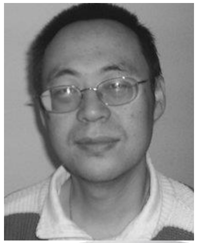




IIT DBGroup

- **Collaborators**
  - Duke
  - SUNY Buffalo
  - UIUC
  - Oracle
  - NYU
  - Free University Bolzano
  - Northeastern University
  - Università della Basilicata
  - EURECOM
  - INSA Centre Val de Loire
  - University of Zurich
  - IIT
  - DePaul
  - **You?**





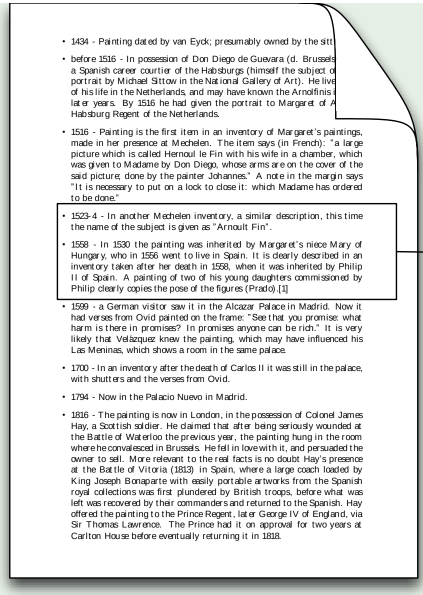
Outline

1) Who am I?
2) **Provenance**
3) Next-generation DB Engines
4) Foundations for Data Science

What the heck is provenance?

- **Provenance in art**
  - Record of the ownership (history) of a piece of art

Jan Van Eyck - Arnolfini Portrait



- 1434 - Painting dated by van Eyck; presumably owned by the sitt
- before 1516 - In possession of Don Diego de Guevara (d. Brussels a Spanish career courtier of the Habsburgs (himself the subject of portrait by Michael Sittow in the National Gallery of Art). He live of his life in the Netherlands, and may have known the Arnolfinis i later years. By 1516 he had given the portrait to Margaret of A Habsburg Regent of the Netherlands.
- 1516 - Painting is the first item in an inventory of Margaret's paintings, made in her presence at Mechelen. The item says (in French): "a large picture which is called Hernoul le Fin with his wife in a chamber, which was given to Madame by Don Diego, whose arms are on the cover of the said picture; done by the painter Johannes." A note in the margin says "It is necessary to put on a lock to close it: which Madame has ordered to be done."
- 1523-4 - In another Mechelen inventory, a similar description, this time the name of the subject is given as "Arnoult Fin".
- 1558 - In 1530 the painting was inherited by Margaret's niece Mary of Hungary, who in 1556 went to live in Spain. It is clearly described in an inventory taken after her death in 1558, when it was inherited by Philip II of Spain. A painting of two of his young daughters commissioned by Philip clearly copies the pose of the figures (Prado).[1]
- 1599 - a German visitor saw it in the Alcazar Palace in Madrid. Now it had verses from Ovid painted on the frame: "See that you promise: what harm is there in promises? In promises anyone can be rich." It is very likely that Velázquez knew the painting, which may have influenced his Las Meninas, which shows a room in the same palace.
- 1700 - In an inventory after the death of Carlos II it was still in the palace, with shutters and the verses from Ovid.
- 1794 - Now in the Palacio Nuevo in Madrid.
- 1816 - The painting is now in London, in the possession of Colonel James Hay, a Scottish soldier. He claimed that after being seriously wounded at the Battle of Waterloo the previous year, the painting hung in the room where he convalesced in Brussels. He fell in love with it, and persuaded the owner to sell. More relevant to the real facts is no doubt Hay's presence at the Battle of Vitoria (1813) in Spain, where a large coach loaded by King Joseph Bonaparte with easily portable artworks from the Spanish royal collections was first plundered by British troops, before what was left was recovered by their commanders and returned to the Spanish. Hay offered the painting to the Prince Regent, later George IV of England, via Sir Thomas Lawrence. The Prince had it on approval for two years at Carlton House before eventually returning it in 1818.

1523-4 In another Mechelen inventory, a similar description, this time the name of the subject is given as "Arnoult Fin".

1558 In 1530 the painting was inherited by Margaret's niece Mary of Hungary, who in 1556 went to live in Spain. It is clearly described in an inventory taken after her death in 1558, when it was inherited by Philip II of Spain. A painting of two of his young daughters commissioned by Philip clearly copies the pose of the figures (Prado).[1]

What the heck is provenance?

- **Provenance in databases**

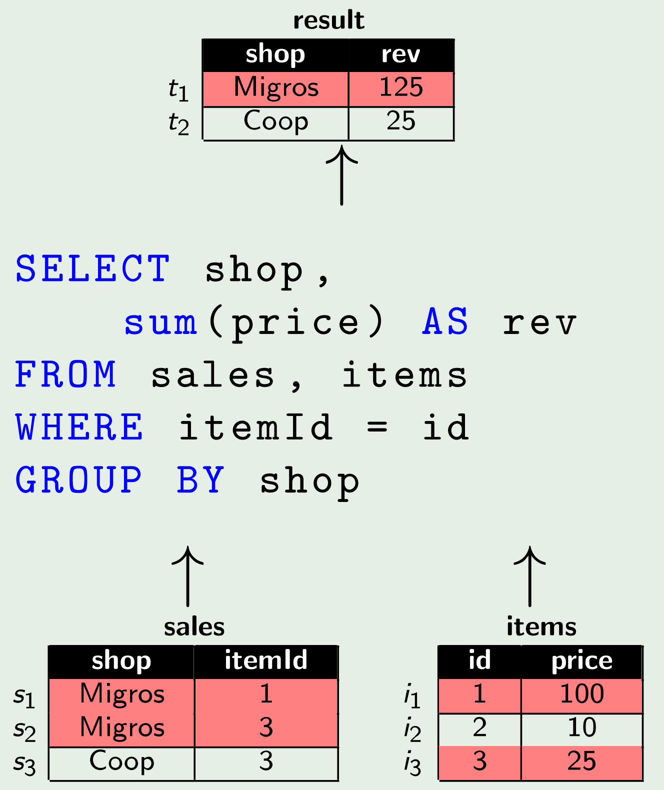
Given a piece of data

- How do we know . . .
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - . . .

Definition (Data Provenance)

Information about the origin and creation process of data.

Example



result

	shop	rev
t_1	Migros	125
t_2	Coop	25

```
SELECT shop,
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

Why should I care?

- **Auditing + Forensics**
- **Trust + Tracebility**
- **Proof of ownership**
- **Debugging**
- **Tracking uncertainty/probability**
- **Understanding data**
- **Relevance-based data management**
- …

My provenance research

- **Tracking provenance for updates+queries**
  - **reenactment**
- **Optimizing provenance capture and management**
- **Unifying provenance and missing answers**
- **Relevance-based data management**
- **GProM => Long term systems effort**

Provenance for updates

- **Databases are typically are not static, but updated over time**
  - **Need to understand how …**
    - current state of the data was derived from old state
    - which operations affected the data
    - **=> provenance for updates!**

Motivation

- **Understand how data is derived by concurrent transactions**
  - **Debugging transactions**
    - Akin to debugging concurrent programs
  - **Auditing data and Forensics**
    - Prove how this row in my database came to be?
    - Which operations of which transactions affected it?
  - **Data Curation and Integration**
    - From which source is this value derived?
    - Can I trust this result?
    - How did my cleaning heuristic interacts with my manual modifications
- **… this is post mortem provenance for transactional histories**
  - **Data dependencies** – this row is derived from these rows
  - **Operations** – these operations did affect this row
  - **Intermediate states** – state of relation R visible to operation

Challenges

- **C1 - What is the right model for transaction provenance?**
  - Model data dependencies across database versions
  - Model control dependencies at granularity of updates
  - Account for low isolation levels
- **C2 - How to compute provenance for transactions efficiently?**
  - Efficient retrieval of provenance for a transactional history (or parts thereof)
  - Low overhead for operations if no provenance is requested
  - Tolerable storage requirements
  - Non-invasive possible (no changes to workload + system)?
- **C3 - How to represent and query provenance?**
  - Represent provenance in a human readable format
  - Provide powerful query capabilities for provenance

Our approach

- **C1 – MV-semirings – a provenance model for queries and transactions**
  - A provenance model for transactions
  - Each row **is annotated with a symbolic expression** that encodes
    - From which previous row versions it was derived
    - How these row versions were combined by the derivation
    - Which updates of which transactions were involved in the derivation
  - Backwards compatible to **provenance semirings for queries**
    - the “most general” provenance model for a large class of queries

Our approach

$$C^1_{T_6,16}(U^1_{T_6,12}(C^1_{T_1,4}(I^1_{T_1,2}(x_1))))$$
$$C^2_{T_5,14}(U^2_{T_5,11}(C^2_{T_1,4}(I^2_{T_1,3}(x_2))))$$
$$C^3_{T_5,14}(U^3_{T_5,13}(U^3_{T_5,11}(C^3_{T_2,3}(I^3_{T_2,2}(x_3)))))$$

Account

cust	typ	bal
Alice	Checking	-1100
Alice	Savings	1100
Peter	Savings	5390

Our approach

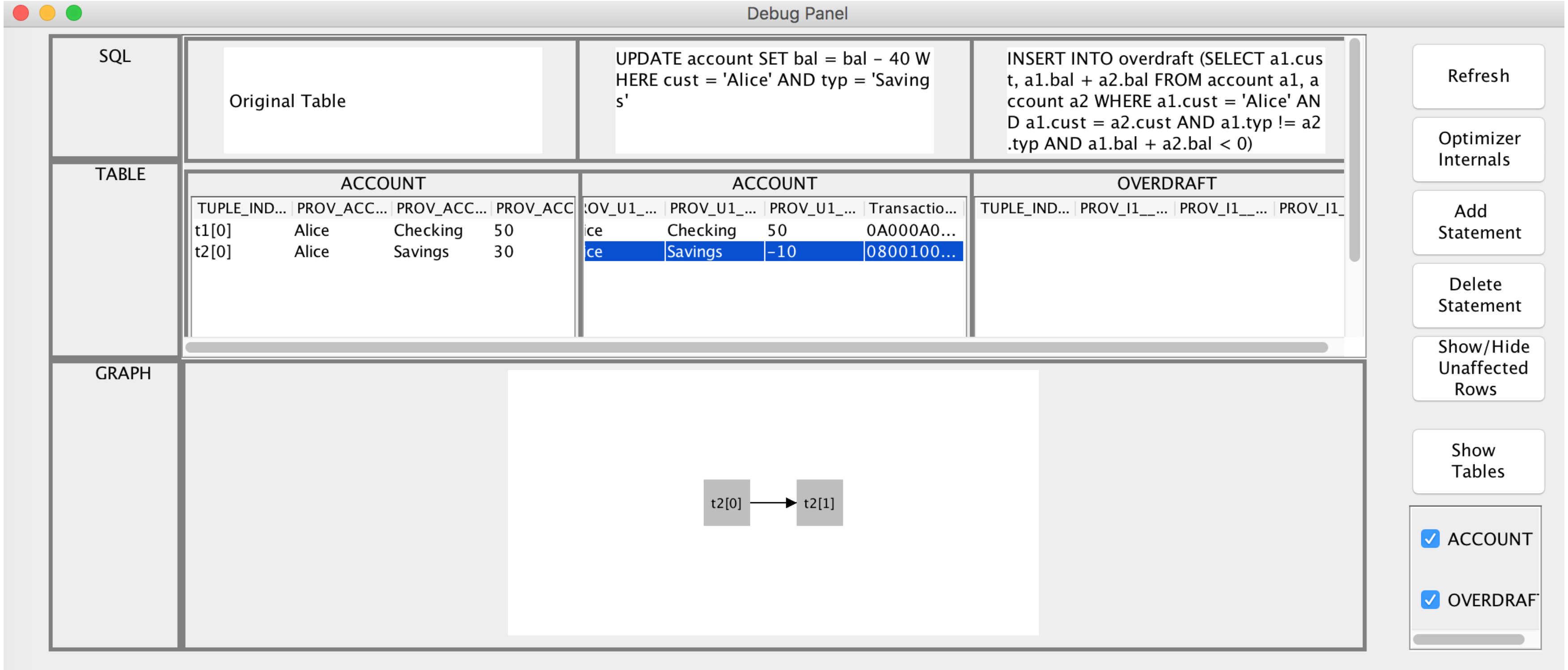
- **C2 – Reenactment – replaying transactional workloads using queries**
  - Translate a **transactional history** (or parts thereof) into a **query** that is **equivalent under annotated semantics** (MV-semirings)
    - Produces the **same result**
    - Has the **same provenance** (data and control dependencies)
    - **Simulates interactions among transactions**
- **C3 – Implementing reenactment though SQL queries**
  - User-friendly **relational encoding** of our provenance model
  - Compile a **reenactment query into an SQL query** that produces this encoding
    - Uses **audit logging** and **time travel** (natively supported by many DBMS)
  - Make provenance requests an **SQL query feature**

Advantages

- **Non-invasive:** requires no modifications to
  - DBMS
  - Transactional workload
- **Low overhead for your workload**
  - No provenance materialized upfront
  - Only overhead payed is ~20% for write only workloads
    - Time travel
    - Audit log
- **Efficient retroactive reconstruction of provenance (or hypothetical scenarios)**
  - One query -> optimize across statements and transactions
  - Replay does not result in disk writes
- **Fully integrated with SQL querying**
  - SELECT count(*)
  - FROM PROVENANCE OF TRANSACTION ‘1234454234’
  - WHERE ….

Example application

- **Post-mortem debugger for transactions**



Debug Panel

SQL

Original Table

```
UPDATE account SET bal = bal − 40 WHERE cust = 'Alice' AND typ = 'Savings'
```

```
INSERT INTO overdraft (SELECT a1.cust, a1.bal + a2.bal FROM account a1, account a2 WHERE a1.cust = 'Alice' AND a1.cust = a2.cust AND a1.typ != a2.typ AND a1.bal + a2.bal < 0)
```

TABLE

ACCOUNT

TUPLE_IND...	PROV_ACC...	PROV_ACC...	PROV_ACC
t1[0]	Alice	Checking	50
t2[0]	Alice	Savings	30

ACCOUNT

:OV_U1_...	PROV_U1_...	PROV_U1_...	Transactio...
ice	Checking	50	0A000A0...
ce	Savings	-10	0800100...

OVERDRAFT

TUPLE_IND...	PROV_I1__...	PROV_I1__...	PROV_I1_

GRAPH

https://github.com/IITDBGroup/GProMTransactionDebugger

GProM

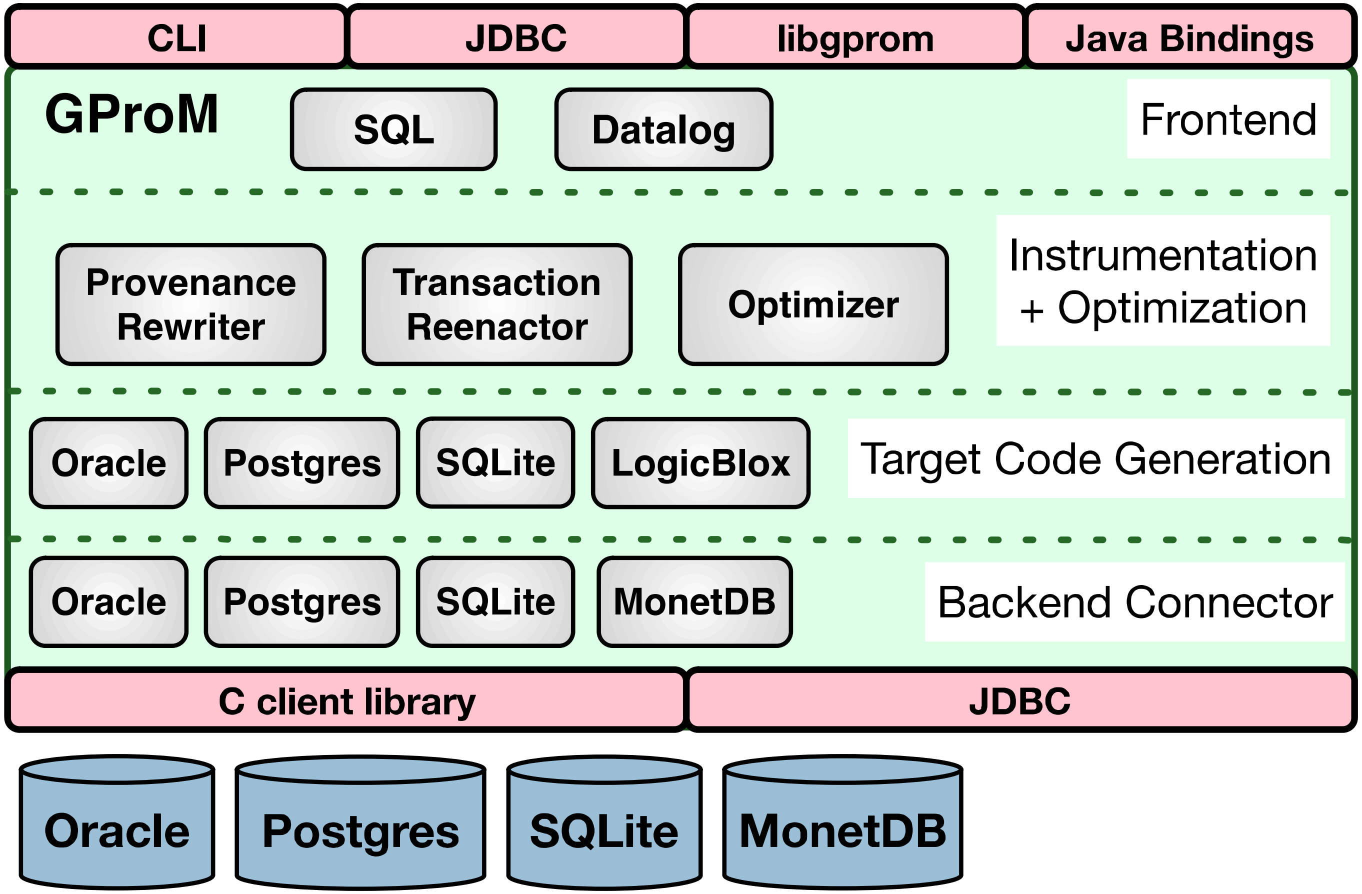
- **Generic middleware for implementing provenance functionality on top of DBMS**
  - **Query-to-query compiler**
    - Take queries with provenance features and compile them into SQL
  - **Modular system**
    - Pipelines consisting of multiple compilation steps
  - **Versatile**
    - All features mentioned in the beginning implemented in GProM
    - Useful beyond provenance: uncertainty, temporal queries …

GProM - Architecture

PROVENANCE OF (SELECT * FROM ...

Q(X) :- R(X,Y). WHY(Q(Peter))

https://github.com/IITDBGroup/gprom



Outline

1) Who am I?
2) Provenance
3) **Next-generation DB Engines**
4) Foundations for Data Science

Building Next-gen DB Engines

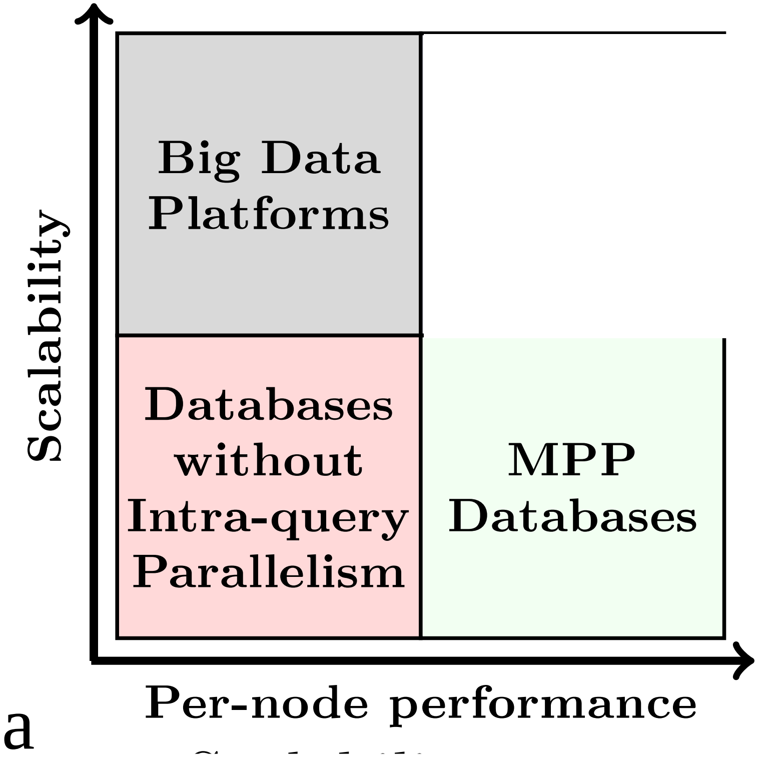
- **How to make databases more scalable without sacrifizing performance?**
  - **HRDBMS**
    - In-house distributed database
    - Scales like Big Data platforms, performs like databases
- **How to maximize performance?**
  - **NautDB** – Generality + Performance
    - Custom dataflow engine / kernel hybrid for compiled queries
  - Combine specialization techniques from the OS and DB community

Scalable SQL Processing

- **Petabyte datawarehouses are becoming common**
  - – eBay: 6.5PB (Greenplum) [2009]
  - – eBay: 2.5PB (Teradata) [2009]
  - – Facebook: 2.5PB (Hive) [2010]
  - – Walmart: 2.5PB (Teradata) [2008]
  - – Bank of America: 1.5PB (Teradata) [2008]
- **Meanwhile**
  - – Traditional MPP DBMS are approaching their scalability limits
    - e.g., Teradata (1024 nodes), Greenplum (1000 nodes), DB2 (1024 nodes)

Dilemma

- **Big Data platforms provide scalability**
  - Hive
  - Spark SQL
  - Flink, AsterixDB, …
- **DBMS provide per-node performance**
  - DB2, Oracle SQL server
  - Teradata, Greenplum, Netazza
  - Hana, MonetDB, …



Scalability

Big Data Platforms

Databases without Intra-query Parallelism

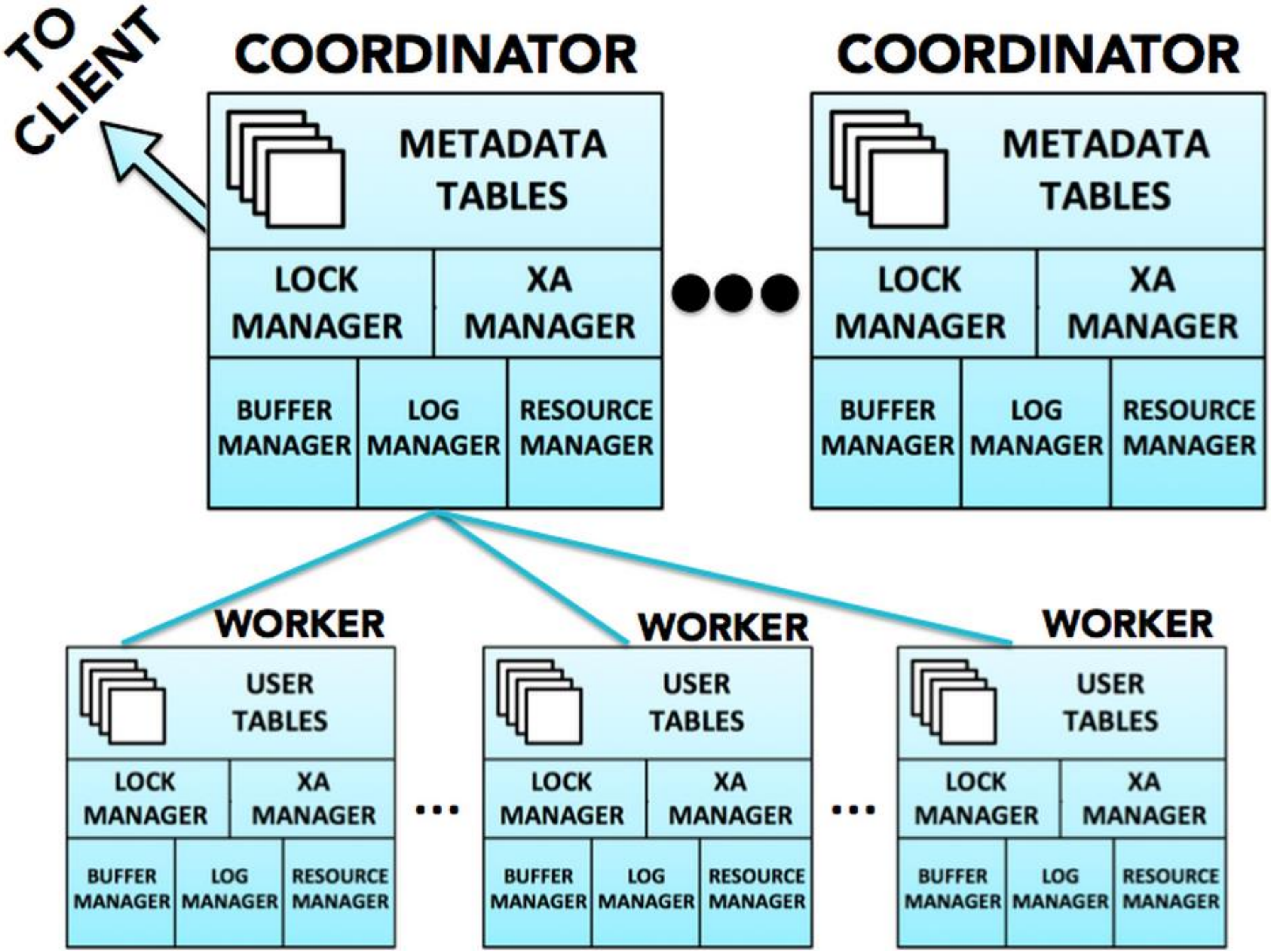
MPP Databases

Per-node performance

What is HRDBMS?

- **Distributed shared-nothing database**
  - Supports standard SQL queries
  - Support for serializable transactions
  - Runs on commodity hardware
  - Written in Java (~165K LOC)
- **Design goals**
  - **Scale** like an elephant (Hadoop)
    - i.e., Big data platform
  - **Perform** like a fruit (Greenplum)
    - i.e., traditional DBMS

HRDBMS Architecture



Approach – Pick the best

- **Combine the best of traditional disk-based DBMS and Big data platforms**
  - Distributed, asynchronous dataflow execution engine
  - Page-oriented storage
  - Buffer manager for in-memory cache
  - Pipelining
  - Intra and inter operator parallelism
  - Cost-based optimization based on statistics
  - Disk-resident index structures
  - Strong consistency (transactions)

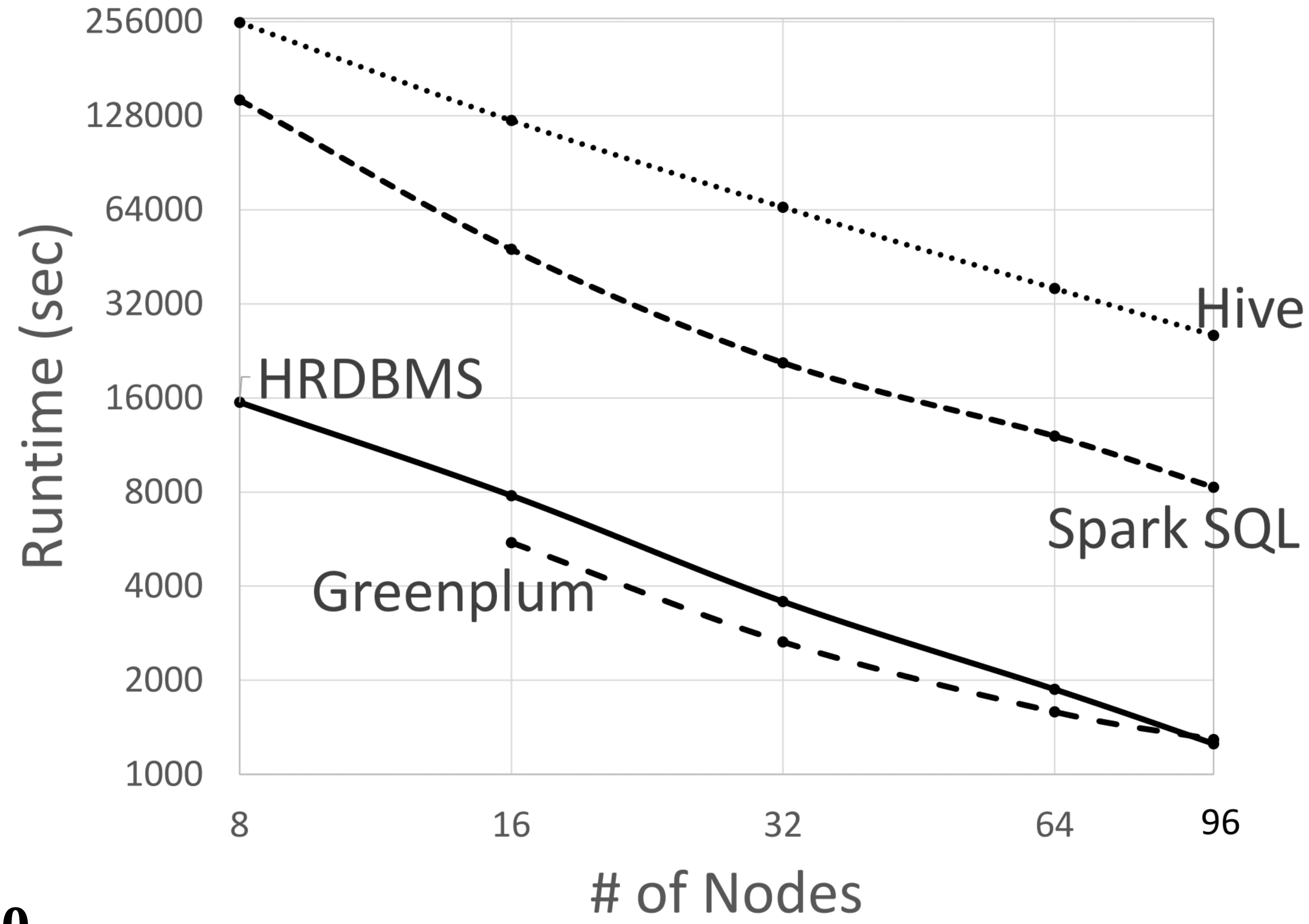
Approach – Improve!

- **Improve over both to avoid scalability and performance pitfalls**
  - Dedicated relational operator implementation in dataflow engine
  - Non-blocking shuffle with optional sort
  - Enforce scalable communication patterns
    - Limit #open-connections per node
  - Enforce data locality
  - Predicate cache for page-skipping

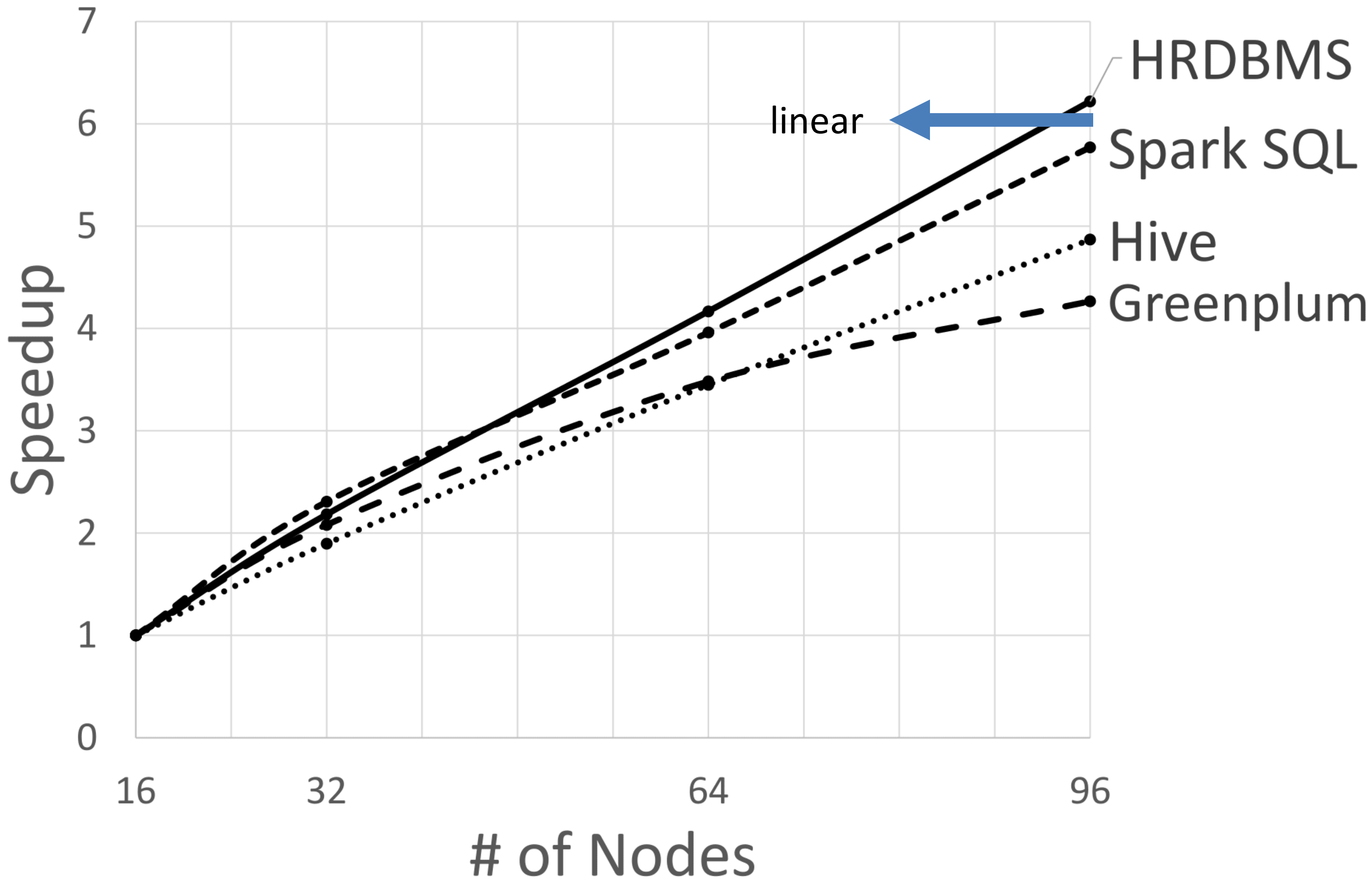
Experiments

- **Workload**
  - 1 TB TPC-H instance (SF=1,000)
- **Setup**
  - **Cooley cluster at Argonne (8 to 96 nodes)**
    - GPFS file system
  - **Nodes**
    - 2 x Intel Haswell E5-2620 v3 (12 cores total)
    - 386GB RAM (we use only 24GB)
- **Competitors**
  - Hive V1.2.1, Spark SQL V1.6.0, Greenplum V4.3.99

TPC-H Total Runtime



Scalability – relative to 16 nodes



Conclusions

- **Goal**
  - Scale like an elephant, perform like a fruit!
- **Assessment**
  - Goal achieved!
- **Take-away message(s)**
  - Don’t disregard traditional methods
    - Stealing = good
  - Don’t be afraid of new ideas
  - Systems research does the trick!
    - Build and evaluate large-scale systems!

Questions?

- **Homepage:**

Jason: http://www.cs.iit.edu/~dbgroup/people/jarnold.php

Boris: http://www.cs.iit.edu/~glavic/

Ioan: http://www.cs.iit.edu/~iraicu/

- **HRDBMS Project**

http://www.cs.iit.edu/~dbgroup/research/hrdbms.php

- **Github**

https://github.com/IITDBGroup/HRDBMS

Outline

1) Who am I?
2) Provenance
3) Next-generation DB Engines
4) **Foundations for Data Science**

Data Science (what you think)

- **1. Collect data**
- **2. Ask questions**
- **3. Get answers (and more questions)**
- **4. Goto 2**

Data Science (real world)

- **1. Collect data**
- **2. Spend weeks curating your data**
- **3. Discover data bugs (goto 2 until clean)**
- **3. Ask questions**
- **4. Discover more data bugs (goto 2)**
- **5. Get answers (and more questions)**
- **6. Realize that you cannot trust the answers because you are not sure whether curation introduced new errors => goto 2?????**

Data Science (real world)

- **80% of time spend on data curation/collection/preparation**
- **This process is currently not supported as much as it should be**
- **Current tools do not automate subtasks (fix certain types of problems)**
- **Current tools do not help users to collaborate and track problems**

My research

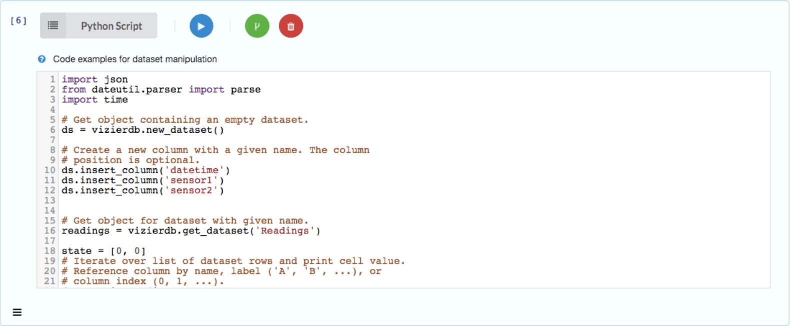
- **Data cleaning + integration**
  - There are many automated tools for data cleaning and curation
    - Help to evaluate them with **BART** and **iBench**
- **Uncertainty management**
  - How to track how uncertainty inherent in data or introduced by (semi-)automated curation through an analysis
- **Vizier**
  - Data curation and exploration platform

Vizier

- **http://www.vizierdb.info/**
- **Data exploration and curation platform**
- **Multi-model Interface**
- **Best-Effort Data Ingest**
- **Data Debugging + Tracebility + Uncertainty Management**
- **Versioned Notebook**
- **Collaboration**
- **Scalable and Compatible**

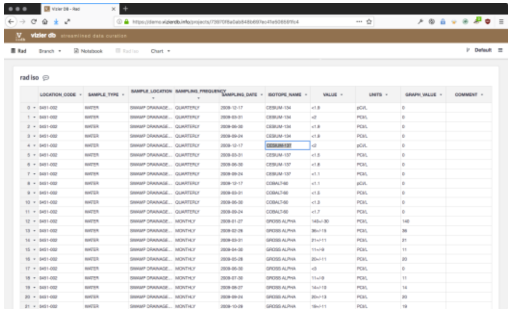
Multi-modal Interface

- **Notebook (think Jupyter)**
  – Python
  – SQL
- **Spreadsheet**
- **Graphs**



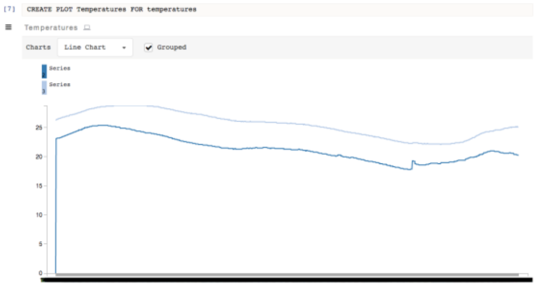
Code examples for dataset manipulation

```
import json
from dateutil.parser import parse
import time

# Get object containing an empty dataset.
ds = vizierdb.new_dataset()

# Create a new column with a given name. The column
# position is optional.
ds.insert_column('datetime')
ds.insert_column('sensor1')
ds.insert_column('sensor2')


# Get object for dataset with given name.
readings = vizierdb.get_dataset('Readings')

state = [0, 0]
# Iterate over list of dataset rows and print cell value.
# Reference column by name, label ('A', 'B', ...), or
# column index (0, 1, ...).
```



rad iso

	LOCATION_CODE	SAMPLE_TYPE	SAMPLE_LOCATION	SAMPLING_FREQUENCY	SAMPLING_DATE	ISOTOPE_NAME	VALUE	UNITS	GRAPH_VALUE	COMMENT
0	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-12-17	CESIUM-134	<1.8	pCi/L	0	
1	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-03-31	CESIUM-134	<2	PCI/L	0	
2	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-06-30	CESIUM-134	<1.8	PCI/L	0	
3	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-09-24	CESIUM-134	<1.9	PCI/L	0	
4	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-12-17	CESIUM-137	<2	pCi/L	0	
5	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-03-31	CESIUM-137	<1.5	PCI/L	0	
6	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-06-30	CESIUM-137	<1.8	PCI/L	0	
7	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-09-24	CESIUM-137	<1.1	PCI/L	0	
8	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-12-17	COBALT-60	<1.1	pCi/L	0	
9	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-03-31	COBALT-60	<1.5	PCI/L	0	
10	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-06-30	COBALT-60	<1.3	PCI/L	0	
11	0451-002	WATER	SWAMP DRAINAGE...	QUARTERLY	2008-09-24	COBALT-60	<1.7	PCI/L	0	
12	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-01-27	GROSS ALPHA	140+/-30	PCI/L	140	
13	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-02-28	GROSS ALPHA	36+/-18	PCI/L	36	
14	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-03-31	GROSS ALPHA	21+/-11	PCI/L	21	
15	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-04-30	GROSS ALPHA	11+/-9	PCI/L	11	
16	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-05-28	GROSS ALPHA	20+/-11	PCI/L	20	
17	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-06-30	GROSS ALPHA	<3	PCI/L	0	
18	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-07-30	GROSS ALPHA	11+/-9	PCI/L	11	
19	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-08-27	GROSS ALPHA	14+/-10	PCI/L	14	
20	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-09-24	GROSS ALPHA	20+/-13	PCI/L	20	
21	0451-002	WATER	SWAMP DRAINAGE...	MONTHLY	2008-10-28	GROSS ALPHA	19+/-11	PCI/L	19	



[7] `CREATE PLOT Temperatures FOR temperatures`

Versioned Notebook + Collaboration

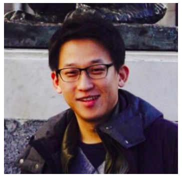
- **Full history of your notebook**
- **Branching, checkpointing, and time travel**
- **Share and release versions with full edit history**
- **Annotate data**
  - Annotations travel along with the data!

Uncertainty Tracking

- **Automated curation is often heuristic in nature**
  - – E.g., impute missing values by replacing them with values predicted by a classifier (ML model)
- **Heuristics make analysis results inherently untrustworthy!**
- **Vizier knows how automated curation methods introduce uncertainty**
  - – Uncertainty is propagated through operations

Conclusions

- **Brief overview of my research**
  - Want to know more? Just reach out
  - I will share these slides



- **Links**
  - https://github.com/IITDBGroup/
  - http://www.cs.iit.edu/~dbgroup/research/
  - http://cs.iit.edu/%7edbgroup/people/index.html
  - http://cs.iit.edu/%7edbgroup/publications/index.php