Systems for Big Data

Mike Carey
Information Systems Group
Computer Science Department
UC Irvine

Plan For Today's Talk

- Raising the level: towards *declarative* tools
  - On saying *what*, not *how*!
- Systems for declarative *data management*
  - Database management systems
  - Structured query language (SQL)
- Moving from data to *Big Data*
  - Definition and challenges
  - Current systems (SQL, NoSQL, data analytics platforms)
- A bigger picture: the *data lifecycle*
  - From ingestion to insights and/or production

Declarative or Imperative?

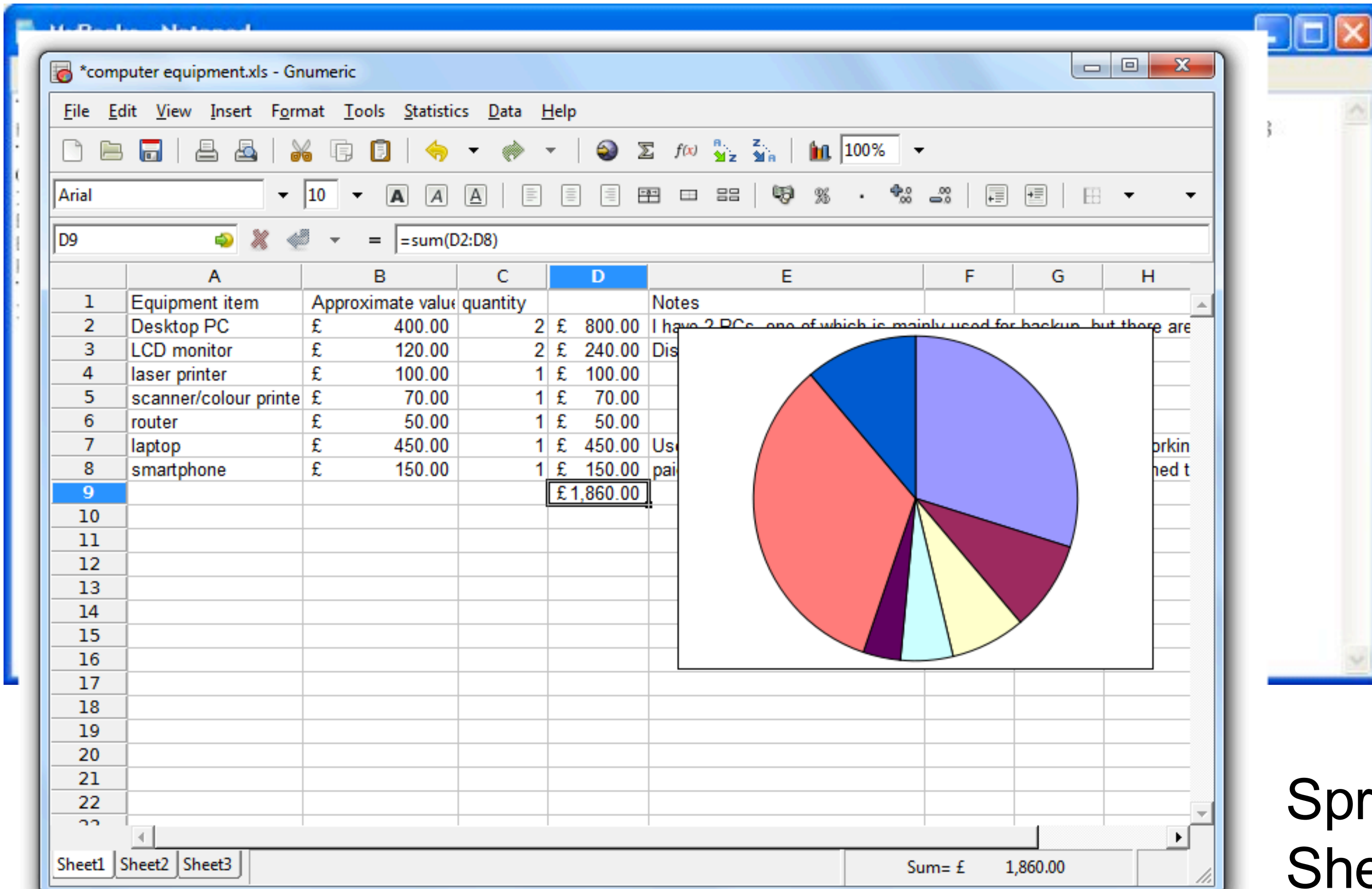
- Suppose we wanted to make a pizza:
- *Imperative* instructions might say…
  1. Get a 3″ ball of pizza dough.
  2. Using a rolling pin, flatten the ball until it is 12″ in diameter.
  3. Open and spread 4 3oz cans of pizza sauce over the dough.
  4. Hand grate 3 oz of mozzarella cheese evenly over the dough.
  5. Starting slightly inside the dough, encircle the pizza with evenly spaced 1″ pepperoni slices; repeat again and again, moving inwards, until you reach the center of the pizza.
  6. Preheat the oven to 350F.
  7. When the oven is ready, bake the pizza for 25 minutes.

Declarative or Imperative? *(cont.)*

- Suppose we wanted to make a pizza:
- *Declarative* instructions would say…
  1. Create a pizza with a 12″ diameter crust,
  2. covered with a 12oz layer of pizza sauce,
  3. a 3oz layer of mozzarella cheese,
  4. and a layer of 1″ pepperoni slices,
  5. and baked for 25 minutes at 350F.

- Notice how we said ***what*** we wanted this time, but didn't have to specify ***how*** to make it…!

Small Data "Management"



Text Files

Spread Sheets

What *is* a Database System?

- So what's a *database*?
  - A (very) large, integrated collection of data
- Often a model of a *real-world enterprise* or a history of *real-world events*
  - *Entities* (*e.g.*, students, courses, Facebook users, …)
  - *Relationships* (*e.g.*, Susan is taking CS 234, Susan is a friend of Lynn, Mike filed a grade change for Lynn, …)
- What's a *database management system* (DBMS)?
  - A software system designed to store, manage, and provide access to one or more such databases

Evolution of DBMS

Files

Manual Coding
• Byte streams
• Majority of application development effort goes towards building and then maintaining data access logic

CODASYL/IMS

Early DBMS Technologies
• Records and pointers
• Large, carefully tuned data access programs that have dependencies on physical access paths, indexes, etc.

Relational

Relational DB Systems
• **Declarative approach**   • Tables + views bring "data independence"   • Details left to system
• Designed to simplify data-centric application development

New Data

???
• …
• …

Why Use a DBMS?

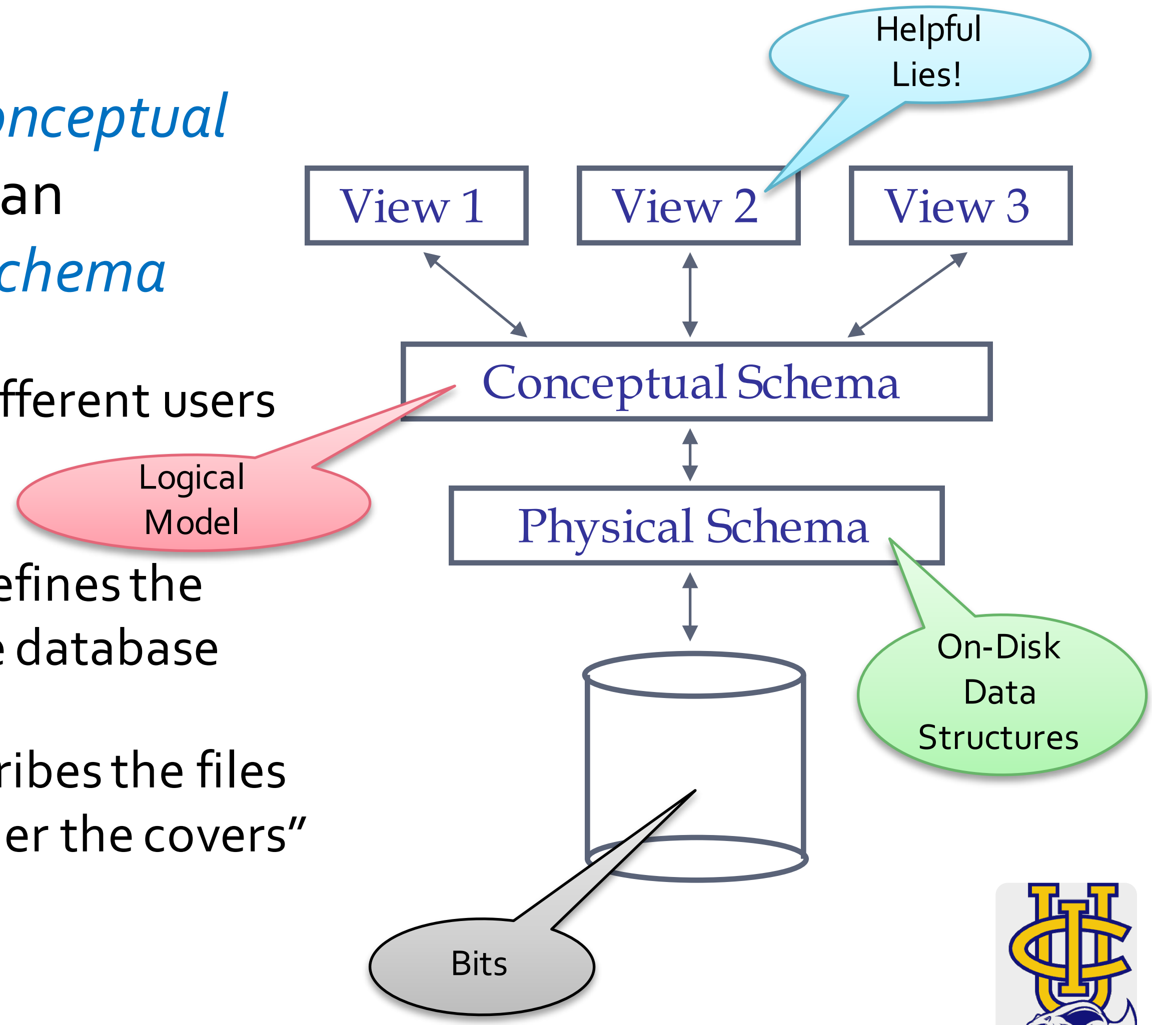
- Data independence
- ***Efficient (automatic) data access***
- ***Reduced development time***
- ***Data integrity and security***
- Uniform data administration
- Concurrent access and recovery from crashes

Data Models

- A *data model* is a collection of concepts for describing data (to one another, or to a DBMS)
- A *schema* is a description of a particular collection of data, using a given data model
- The ***relational model*** is the most widely used data model today
  - *Relation* – basically a **table** with rows and (**named**) columns
  - *Schema* – describes the tables and their columns

Levels of Abstraction

- Many *views* of one *conceptual (logical) schema* and an underlying *physical schema*
  - ***Views*** describe how different users or groups see the data
  - ***Conceptual schema*** defines the logical structure of the database
  - ***Physical schema*** describes the files and indexes used "under the covers"



Example: University DB

- Conceptual schema (*a.k.a.* stored tables):
  - *Students(<u>sid</u>: string, name: string, login: string, age: integer)*
  - *Courses(<u>cid</u>: string, cname: string, credits: integer)*
  - *Enrolled(<u>sid</u>: string, <u>cid</u>: string, grade: string)*
- Physical schema (*a.k.a.* storage and indexing):
  - Tables each stored internally as unordered files
  - Have *indexes* on first and third columns of *Students*
- External schema (*a.k.a.* views):
  - *CourseInfo(<u>cid</u>: string, cname: string, enrollment: integer)*

Example: University DB *(cont.)*

Courses

cid	cname	credits
CS 101	Programming I	6
CS 102	Programming II	6
CS 103	Computer Games	4
Stat 101	Statistics 1	4

Students

sid	name	login	age
22	Dustin	dusty@aol.com	22
29	Brutus	bbrute@gmail.com	19
31	Rusty	rust@hotmail.com	23
32	Andrew	andyman@aol.com	18
58	Suzy	susan@yahoo.com	22
71	Rosie	flower@fb.com	20

Enrolled

sid		grade
22	CS 101	B-
22	CS 103	A-
58	Stat 101	A
29	CS 101	C+
71	CS 103	A+

Example: University DB *(cont.)*

- User query (in SQL, against the view):
  - *SELECT c.cid, c.enrollment*
    FROM CourseInfo c
    WHERE c.cname = 'Computer Games'

Declarative!

- Equivalent query (against the stored tables):
  - *SELECT e.cid, count(e.*)*
    FROM Enrolled e, Courses c
    WHERE e.cid = c.cid AND c.cname = 'Computer Games'
    GROUP BY c.cid

Also Declarative!

- Under the h … (against the phys … chema)
  - Access *Courses* … *name* to find associated *cid*
  - Access *Enr* … se in … count the enrollments

Details…

SQL DB Summary

- User query (in SQL, against the view):
  - *SELECT c.cid, c.enrollment*
    FROM CourseInfo c
    WHERE c.cname = 'Computer Games'

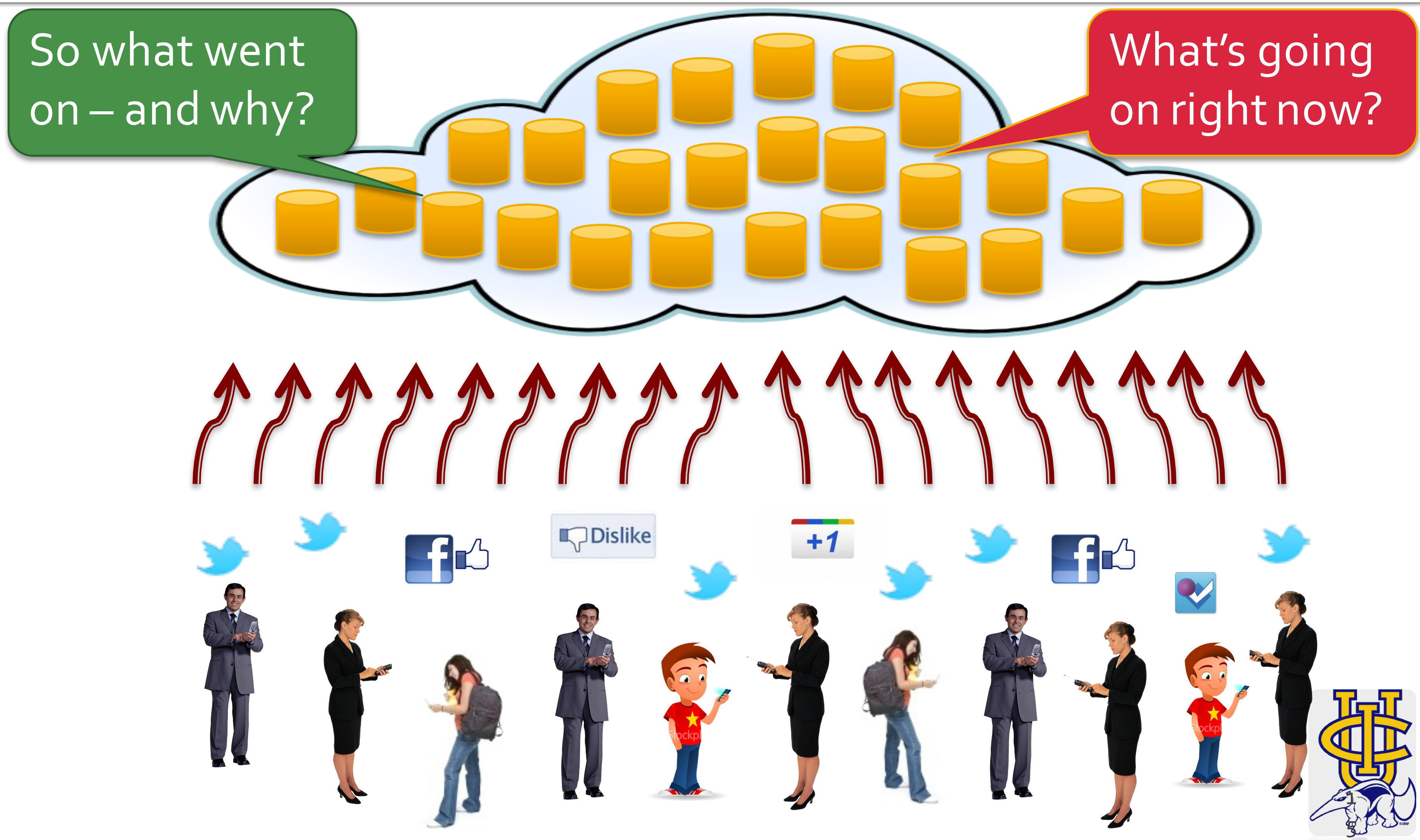
Declarative!

Result

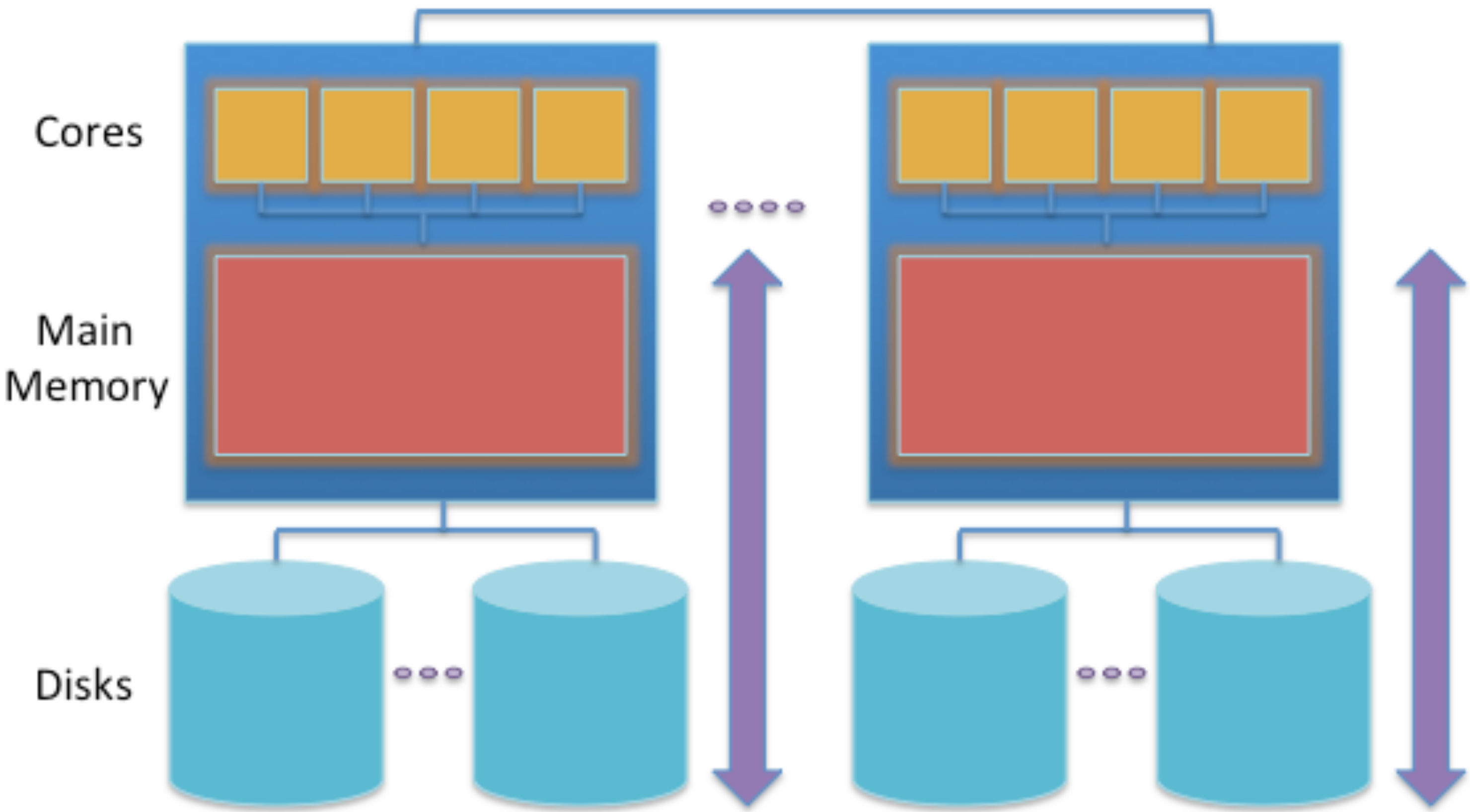
cid	enrollment
CS 103	2

IBM DB2, Oracle, Microsoft SQL Server, MySQL, SQLite, PostgreSQL, and others...

Big Data / Web Warehousing

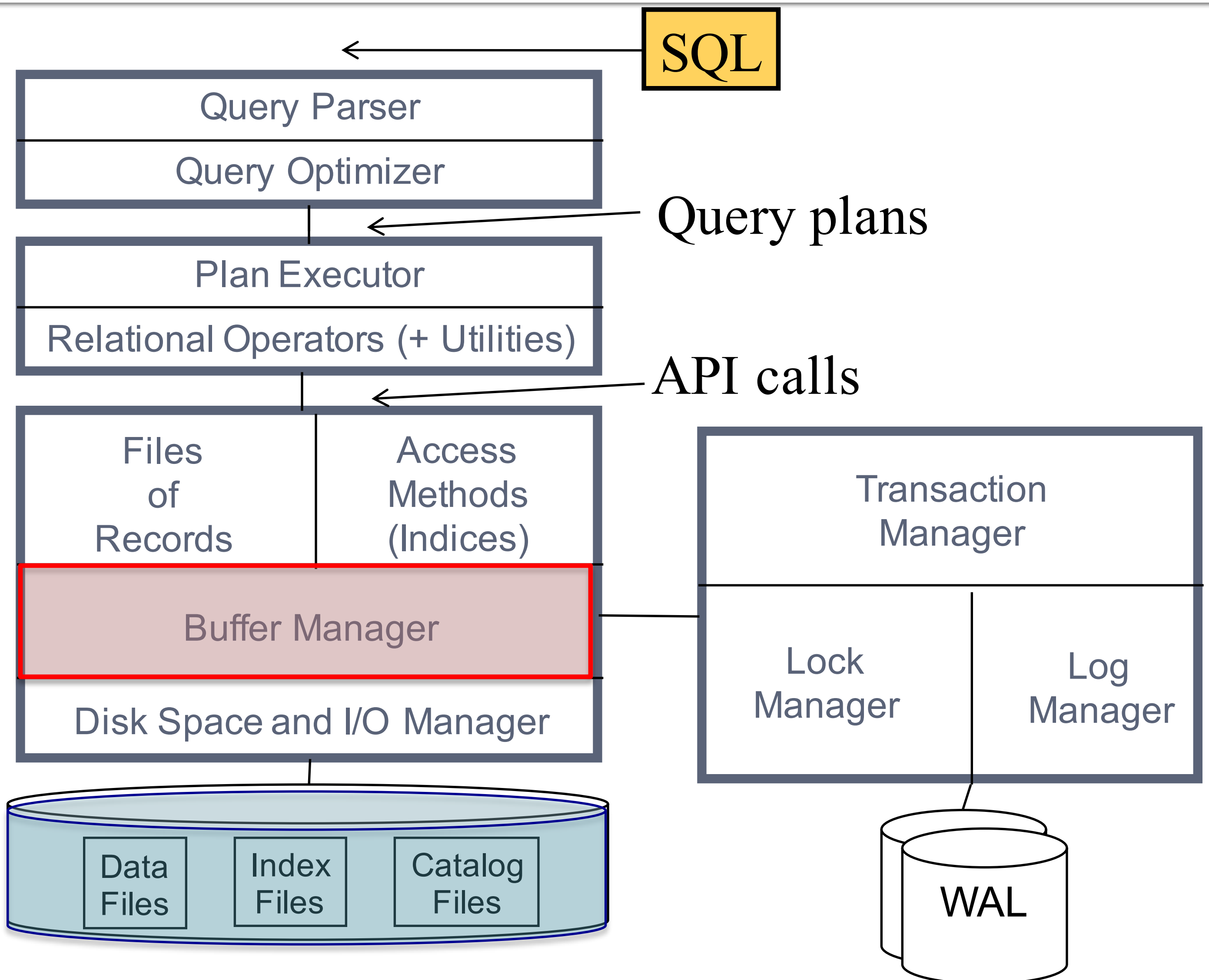


How Big is "Big Data"?



This is
big
data!

DB Systems: Under the Hood

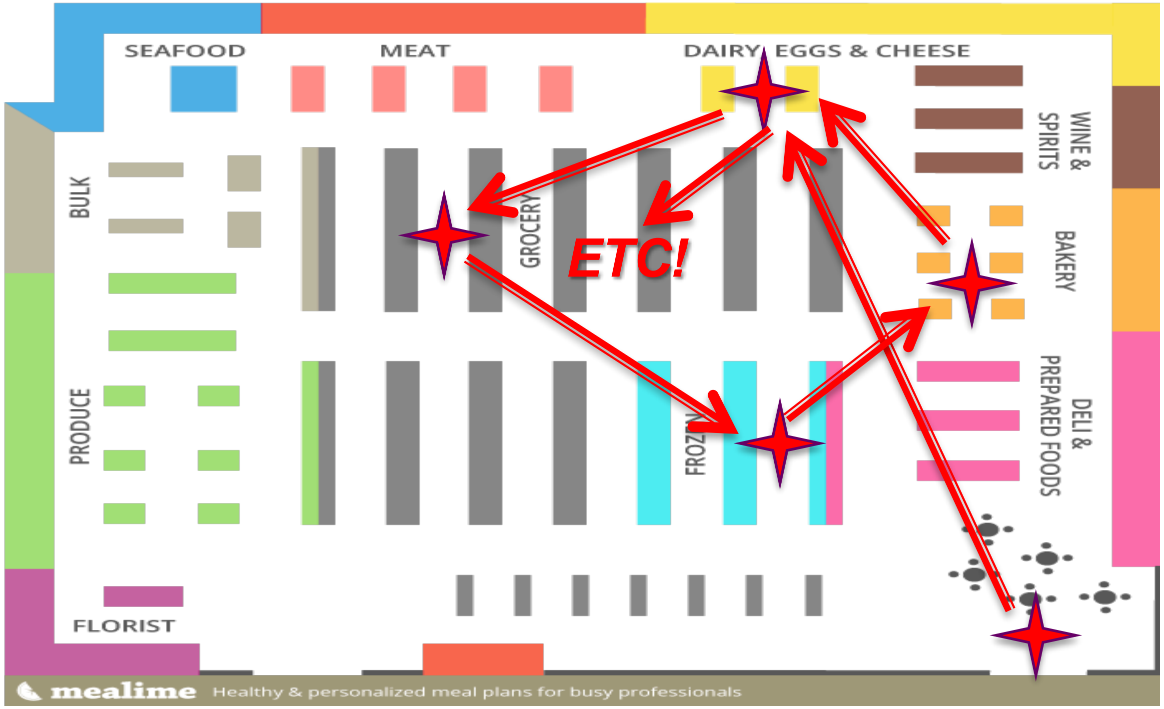


Data on Disk: Algorithms

- Let's consider a grocery shopping analogy:
  - List 1 = {milk, cheerios, ice cream, bread}

Data on Disk: Algorithms *(cont.)*

- Continuing our grocery shopping analogy:
  - List 2 = {milk, cheerios, ice cream, bread, **cream, cat food, chicken, coffee, napkins, coke, jelly, kleenex,**…(87)…, **water**}

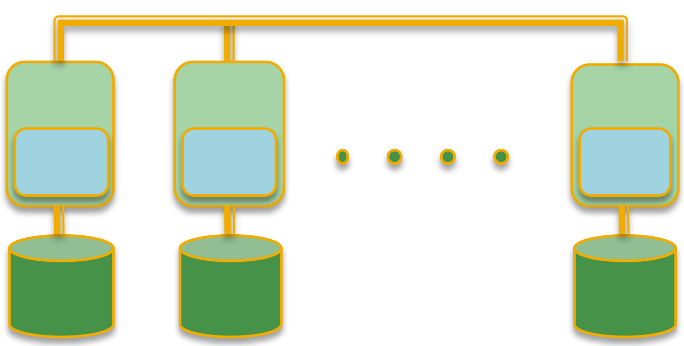


*Think of **disk accesses** as being similar to aisle visits!*

***Sorting by aisle** could help a lot!*

Big Data in the Database World

- Enterprises wanted to store and query historical business data (data warehouses)
  - 1970's: Relational databases appeared (w/SQL)
  - Late 1970's: Database machines based on novel hardware and early (brute force) parallelism
  - 1980's: Parallel database systems based on "shared-nothing" architectures (Gamma, GRACE, Teradata)
  - 2000's: Netezza, Aster Data, *DATAllegro*, Greenplum, Vertica, ParAccel, ... (*Serious "Big $" acquisitions!*)

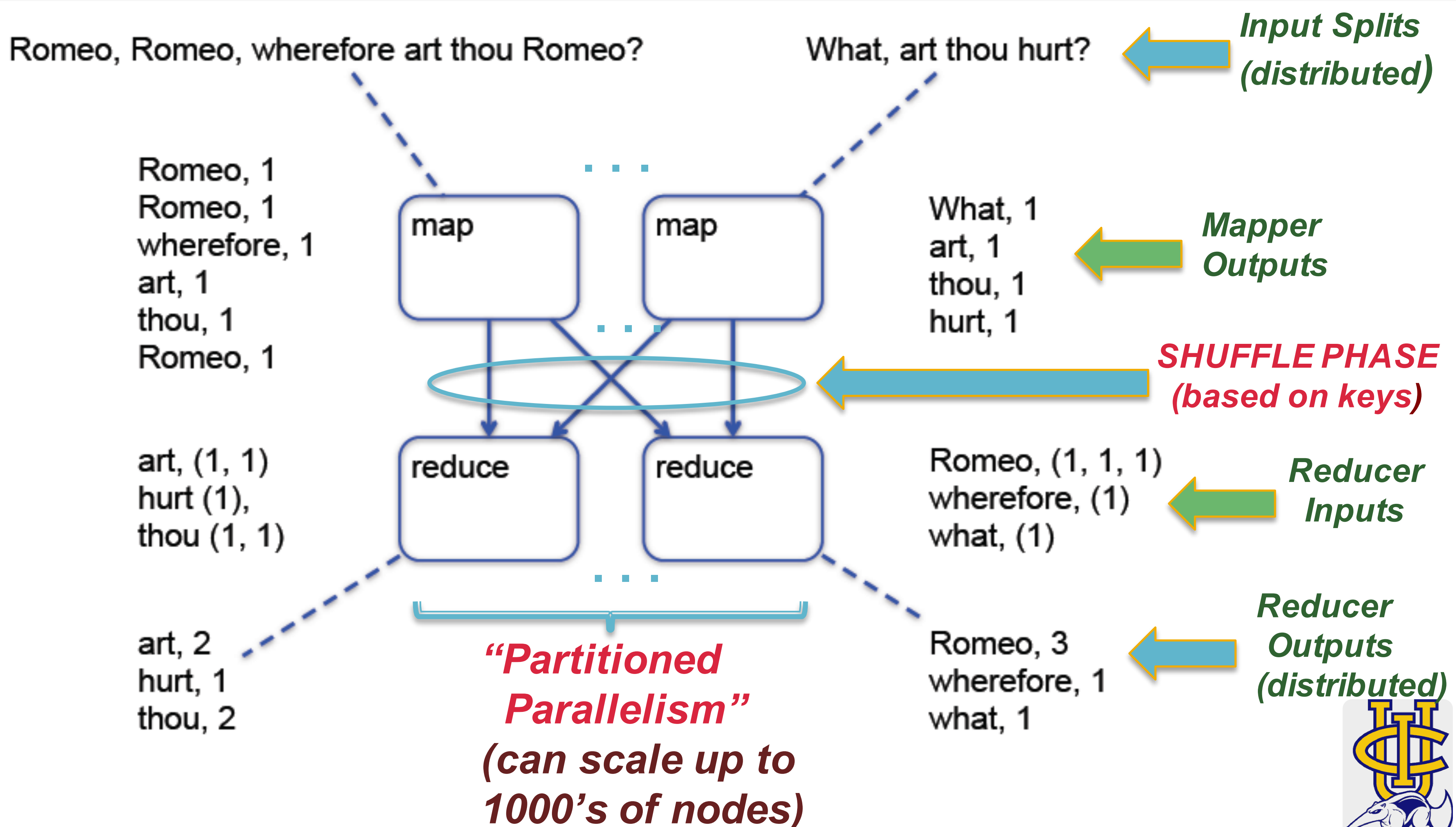


(Each node runs an instance of an indexed database data storage and runtime system)

Big Data in the Systems World

- Late 1990's brought a need to index and query the rapidly exploding content of the Web
  - SQL-based databases didn't fit the problem(s)
  - Google, Yahoo! *et al* had to do something
- Google responded by laying a new foundation
  - Google File System (GFS)
    - OS-level byte stream files spanning **1000's** of machines
    - 3-way replication for fault-tolerance (and high availability)
  - MapReduce (MR) programming model
    - User writes just two simple functions: ***Map*** and ***Reduce***
    - ***"Parallel programming for dummies"*** – MR runtime does all the heavy lifting (using partitioned parallelism)

MapReduce: A Quick Example



MapReduce Programming model

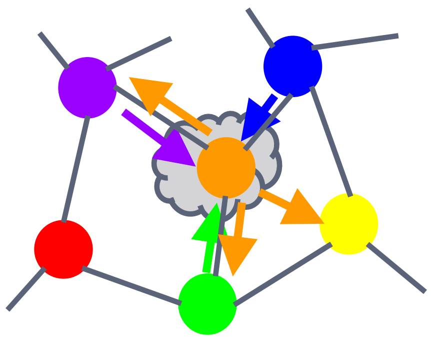
- Inputs and outputs are sets of key/value pairs
- Programmers simply provide two functions
  - **map**(K1, V1) -> list(K2, V2)
    - Produces list of intermediate ***key/value pairs*** for each input key/value pair
  - **reduce**(K2, list(V2)) -> list(K3, V3)
    - Produces a list of result values for all intermediate values that are associated with the ***same intermediate key***
- In our word count example, notice that
  - The keys were the words and the counts were the values
  - ***We never had to think about parallelism!***

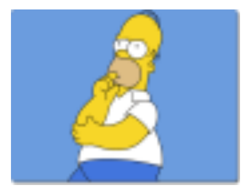
Soon a Star Was Born…

- Yahoo!, Facebook, and friends cloned Google's "Big Data" infrastructure from papers
  - GFS → Hadoop Distributed File System (HDFS)
  - MapReduce → Hadoop MapReduce
  - Widely used for Web indexing, click stream analysis, log analysis, information extraction, some machine learning
- Tired of puzzle-solving with just two moves, higher-level languages were developed to "hide" MR
  - *E.g.*, Pig (Yahoo!), Hive (Facebook), Jaql (IBM)
- Similar happenings at Microsoft
  - Cosmos, Dryad, and SCOPE (which powers Bing)

Other Up-and-Coming Platforms

- Bulk Synchronous Programming (**BSP**) platforms, e.g., Pregel, Giraph, GraphLab, ..., for doing Big* Graph analysis



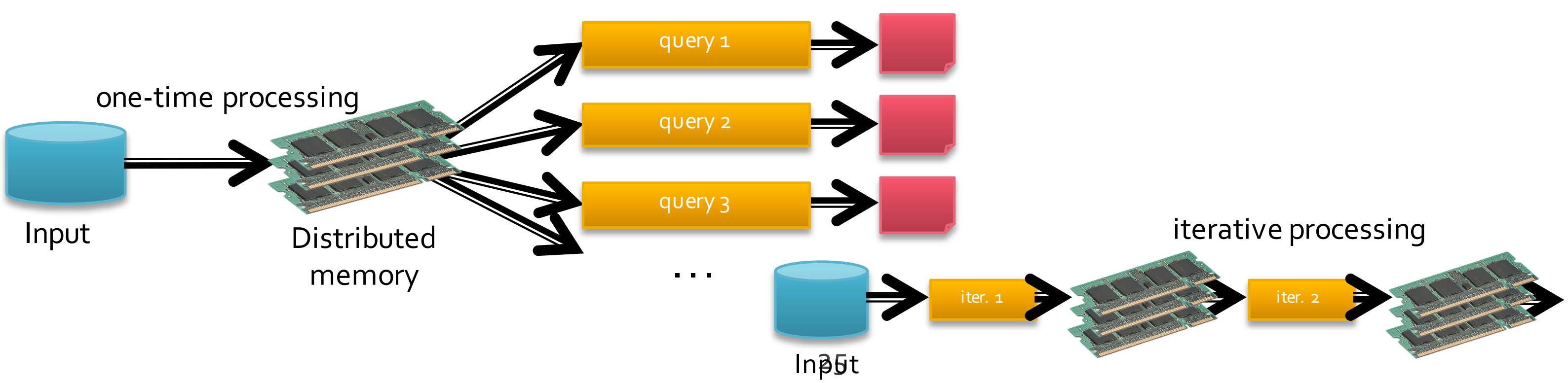


"Think Like a Vertex"

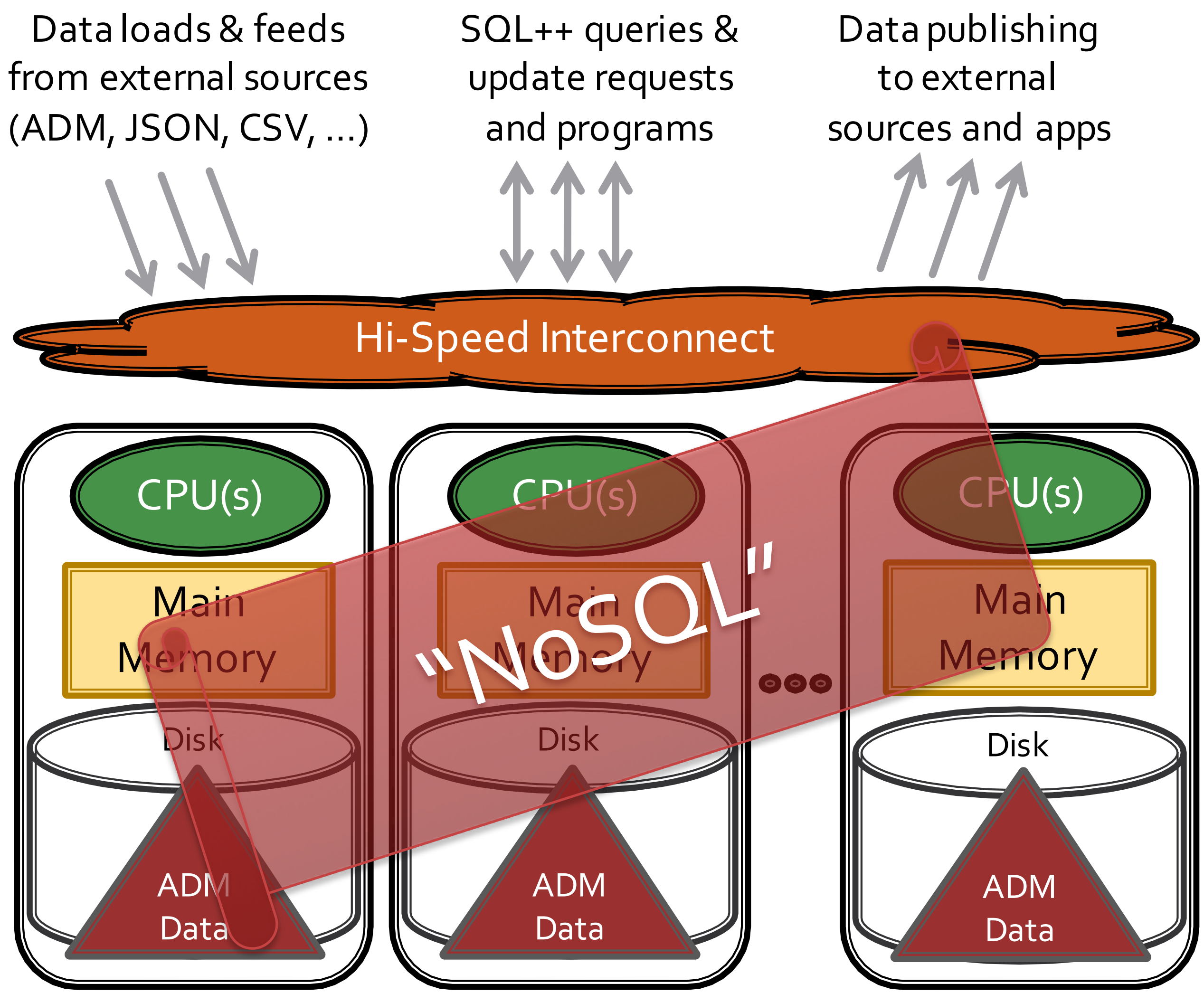
- Receive messages
- Update state
- Send messages

(Big is the platform's problem)*

- **Spark** for in-memory cluster computing – for repetitive data analyses, iterative machine learning tasks, ...



AsterixDB System (UCI / UCR)



ASTERIX Goal: *Ingest, digest, persist, index, manage, query, analyze, and publish massive* quantities of semi-structured information...

(**ADM** = ASTERIX Data Model, **SQL++** = ASTERIX Query Language)

ASTERIX Data Model (ADM)

```
CREATE DATAVERSE TinySocial;
USE TinySocial;

CREATE TYPE GleambookUserType AS {
    id: int,
    alias: string,
    name: string,
    userSince: datetime,
    friendIds: {{ int }},
    employment: [EmploymentType]};

CREATE TYPE GleambookMessageType
AS {
        messageId: int,
        authorId: int,
        inResponseTo: int?,
        senderLocation: point?,
        message: string
    };
```

```
CREATE TYPE EmploymentType AS {
    organizationName: string,
    startDate: date,
    endDate: date?
};

CREATE DATASET GleambookUsers
                        (GleambookUserType)
PRIMARY KEY id;

CREATE DATASET GleambookMessages

(GleambookMessageType)
PRIMARY KEY messageId;
```

"NoSQL" characteristics include...

- Objects can be nested
- Fields can be multivalued (plural)
- Content can vary from object to object
- Schemas are not mandatory

(*Other examples:* MongoDB, Couchbase,...)

UCR

Ex: Gleambook Users Data

{"id":1, "alias":"Margarita", "name":"MargaritaStoddard", "nickname":"Mags",
"userSince":datetime("2012-08-20T10:10:00"), "friendIds":{{2,3,6,10}},
"employment": [{"organizationName":"Codetechno", "startDate":date("2006-08-06")},
{"organizationName":"geomedia" , "startDate":date("2010-06-17"),
"endDate":date("2010-01-26")}],
"gender":"F"
},

{"id":2, "alias":"Isbel", "name":"IsbelDull", "nickname":"Izzy",
"userSince":datetime("2011-01-22T10:10:00"), "friendIds":{{1,4}},
"employment": [{"organizationName":"Hexviafind", "startDate":date("2010-04-27")}]
},

{"id":3, "alias":"Emory", "name":"EmoryUnk",
"userSince":datetime("2012-07-10T10:10:00"), "friendIds":{{1,5,8,9}},
"employment": [{"organizationName":"geomedia", "startDate":date("2010-06-17"),
"endDate":date("2010-01-26")}]
},

. . . . .

⋮

UCR

ASTERIX Queries (SQL++)

- *Q1:* List the user names and messages sent by Gleambook social network users with less than 3 friends:

```
SELECT user.name AS uname,
        (SELECT VALUE msg.message
         FROM GleambookMessages msg
         WHERE msg.authorId = user.id) AS messages
FROM GleambookUsers user
WHERE COLL_COUNT(user.friendIds) < 3;
```


```
{ "uname": "NilaMilliron", "messages": [  ] }
{ "uname": "WoodrowNehling", "messages": [ " love acast its 3G is good:)" ] }
{ "uname": "IsbelDull", "messages": [ " like product-y the plan is amazing", " like
  product-z its platform is mind-blowing" ] }
. . .
```

AsterixDB Status

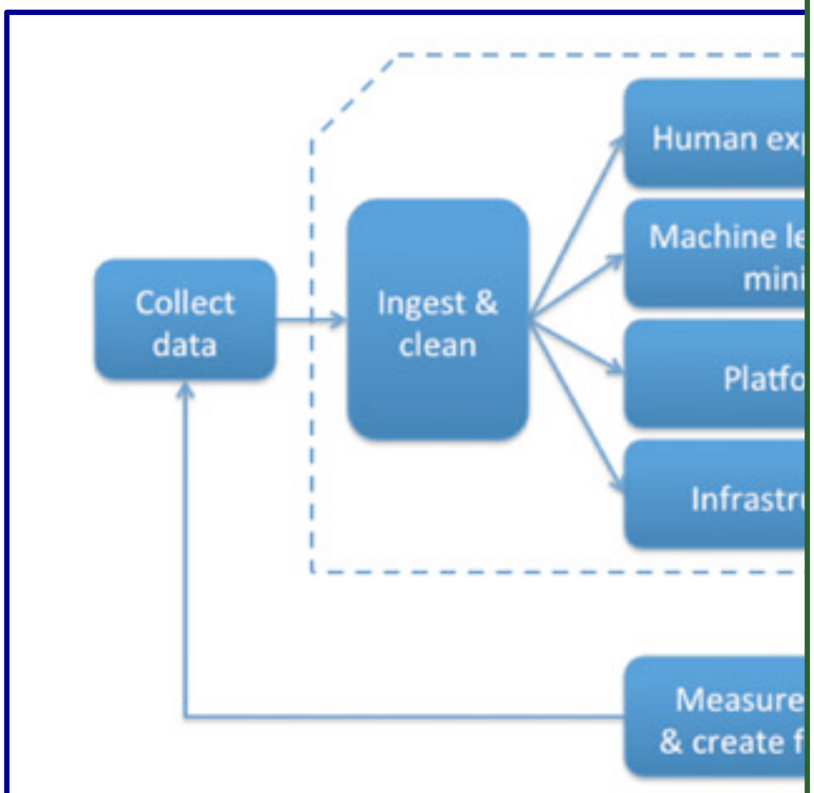
- 4 year initial NSF project (250+ KLOC @ UCI+UCR)
- AsterixDB BDMS! (First shared June 6th, 2013)
  - Semistructured "NoSQL" style data model
  - Declarative parallel queries, inserts, deletes, ...
  - LSM-based storage/indexes (primary & secondary)
  - Internal and external datasets both supported
  - Rich set of data types (including text, time, location)
  - Fuzzy and spatial query processing
  - NoSQL-like transactions (for inserts/deletes)
  - Data feeds and external indexes in next release
- Performance competitive w/parallel relational DBMS, MongoDB, and Hive (see papers)
- *Now in Apache!*

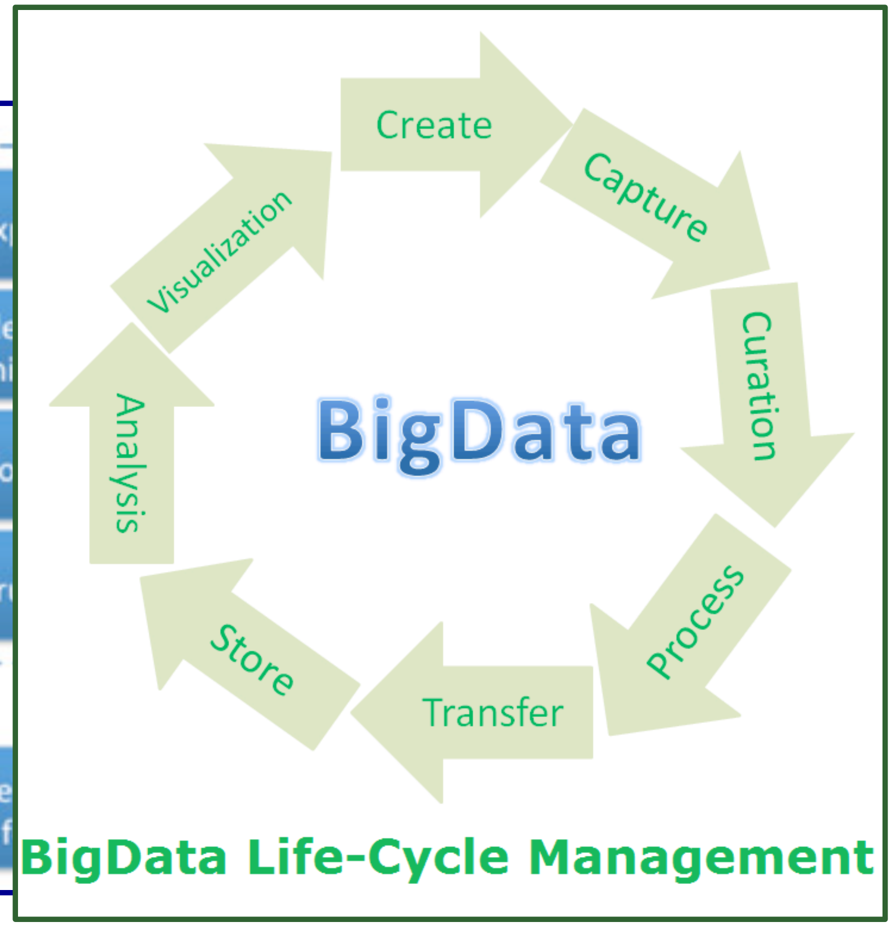
Some Use Case Examples

- Recent or projected use case areas include…
  - Behavioral science (at UCI)
  - Social data analytics
  - Cell phone event analytics
  - Power usage monitoring
  - Public health (joint effort with UCIPT@UCLA)
  - Cluster management log analytics
  - *Your future use cases go here…* (☺)

Big Data Lifecycle

- I've just described one piece of the Data Science "Big Data puzzle"…





What We've Touched On

- Raising the level: towards *declarative* tools
  - It's all about saying *what*, not *how*!
- Systems for declarative *data management*
  - Database management systems
  - Structured query language (SQL) in particular
- Moving from data to *Big Data*
  - Definition of "big" and some of the challenges
  - Current systems (SQL, NoSQL, data analytics platforms)
- The bigger picture: the *data lifecycle*
  - From ingestion to insights and production (and repeat!)

Questions?