COS316
Layers in Database Management Systems

PRINCETON UNIVERSITY

Suppose you're a data scientist...

- Your boss says:
  - Bring me some f*#(@ coffee!
  - Get the average time users spent on the site last year
  - Where's my f*#@$S coffee??!!

What data do we have?

```
[2023-09-21 11:05:41] uid: 123, sid: 923, GET  /blog/how-to-quit-your-job
[2023-09-21 11:05:41] uid: 834, sid: 923, GET  /feed.xml
[2023-09-21 11:05:42] uid: 145, sid: 923, GET  /robots.txt
[2023-09-21 11:05:42] uid: 923, sid: 923, GET  /blog/10-looks-4-the-apocolypse
[2023-09-21 11:05:42] uid: 523, sid: 923, GET  /index.html
[2023-09-21 11:05:43] uid: 915, sid: 923, GET  /blog/how-to-quit-your-job.html
[2023-09-21 11:05:43] uid: 256, sid: 923, GET  /index.html
[2023-09-21 11:05:46] uid: 123, sid: 923, POST /blog/comments
...
```

Millions of these

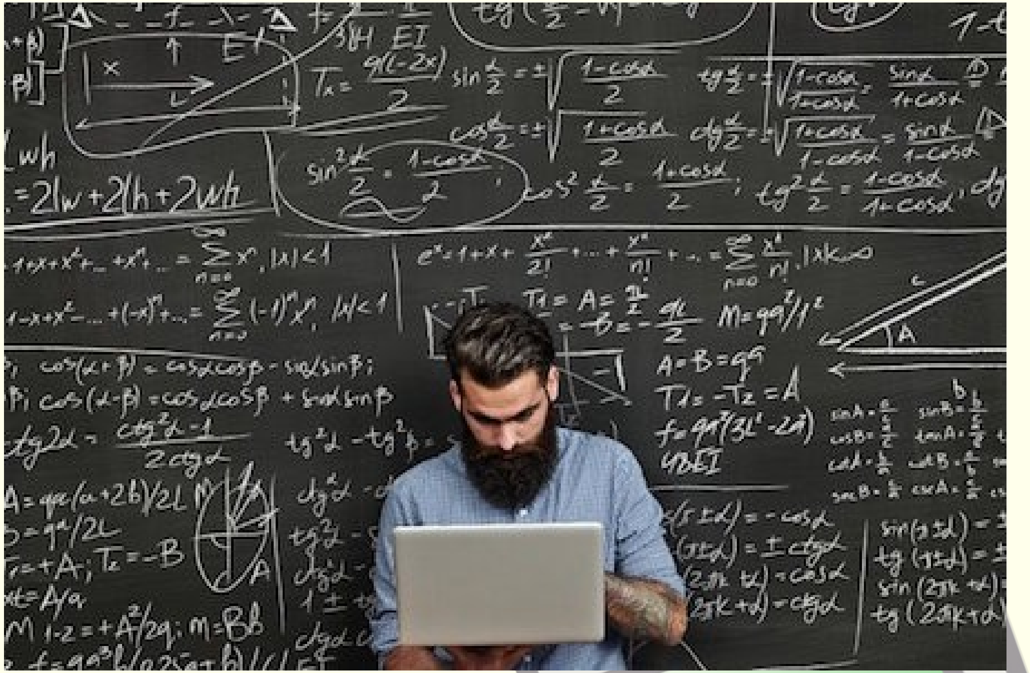
```
sessions = {}

for line in log.lines():
     ts,uid,sid,method,req = line.parse()
     sessions[(uid,sid)] = True

for (uid,sid) in sessions:
     start, end = None, None
     for line in log.lines():
          ts, uidn, sidn, _, _ line.parse()
          if uid == uidn and sid == sidn
               count += 1
               if not start:
                    start = ts
               end = ts

     sessions[(uid,sid)] = (end - start)

return math.average(sessions.values())
```



Ugh...

What's wrong with this?

- Inefficient: scan data for each session
- Ties intention to implementation
  - Is it correct?
- Time consuming to iterate on
  - Small changes in the query might require a rewrite

Database Management Systems

Roles of a DBMS

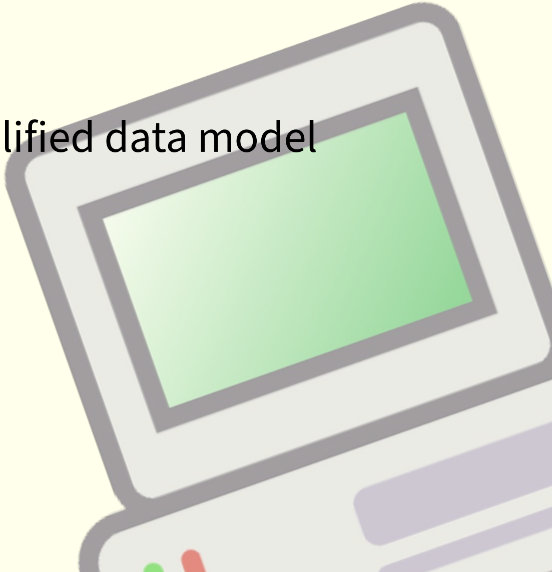
- Store data: durability, availability, cost, performant
- Organize data meaningfully
- Make modifying and querying database as fast and simple as possible

Many kinds of DBMSs

- ***Relational***
- Document
- Graph
- …
- We'll focus on relational databases

DBMS Layers (top down)

- Query parser
  - transforms a declaritive query into relational algebra
- Query planner
  - decides how to best *execute* a query given the underlying data model, storage layout, etc
- Transactional tuple (key-value) store
  - Provides Atomicity Consistency Isolation and Durability (ACID) for a simplified data model
- Page storage
  - Low-level storage mechanism



Why Layers?

- SQL serves as the “narrow waste” of database applications
- Simplicity of each layer
  - Relational algebra is hard enough without having to worry about consistency or storage hardware performance
- Application portability across DBMSs
- Layer reuse
  - PostgresQL query parser and planner used in other DBMSs
  - RocksDB key-value store used for many DBMSs

Relational Model

A relation is an unordered set that contain the relationship of attributes that represent entities.

A tuple is a set of attribute values (also known as its domain) in the relation.

- Values are (normally) atomic/scalar.

Artist(id, name, year)

id	name	year
101	Wu-Tang Clan	1992
102	Notorious BIG	1992
103	GZA	1990

Relational Model: Primary Keys

A relation's primary key uniquely identifies a single tuple.

Some DBMSs automatically create an internal primary key if a table does not define one.

Artist(id, name, year)

id	name	year
101	Wu-Tang Clan	1992
102	Notorious BIG	1992
103	GZA	1990

Relational Model: Select

Choose a subset of the tuples from a relation that satisfies a selection predicate.

- Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.
- Can combine multiple predicates using conjunctions / disjunctions

```
select * from artists
where year = 1992
```

id	name	year
101	Wu-Tang Clan	1992
102	Notorious BIG	1992
103	GZA	1990

Relational Model: Projection

Choose a subset of the tuples from a relation that satisfies a selection predicate.

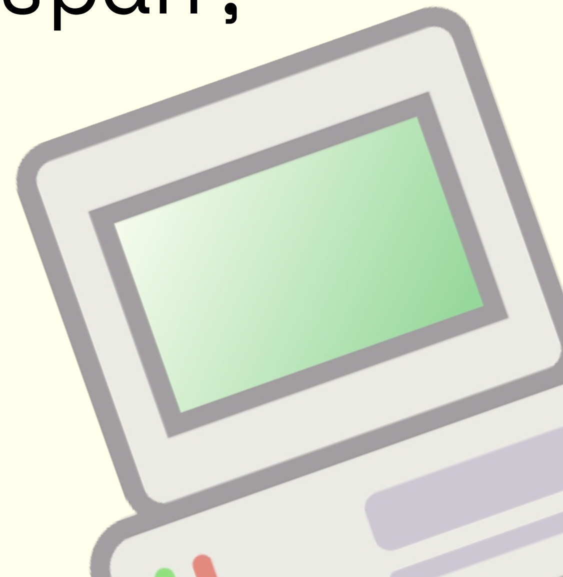
- Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.
- Can combine multiple predicates using conjunctions / disjunctions

```
select name from
    artists
    where year = 1992
```

id	name	year
101	Wu-Tang Clan	1992
102	Notorious BIG	1992
103	GZA	1990

SQL—the standard bearer

```
select avg(span) from (
  select max(ts) - min(ts) from logs
    group by (uid, sid)) as span;
```



Page Storage

- Minimize reads/write from disk
- Store tuples in blocks of data called *pages*
- Common layout called “N-Ary”
  - Each page stores entire rows, layed out sequentially
  - Size of each row is well-known, so the nth row is at offset n * row-size
  - Typically ordered by primary key