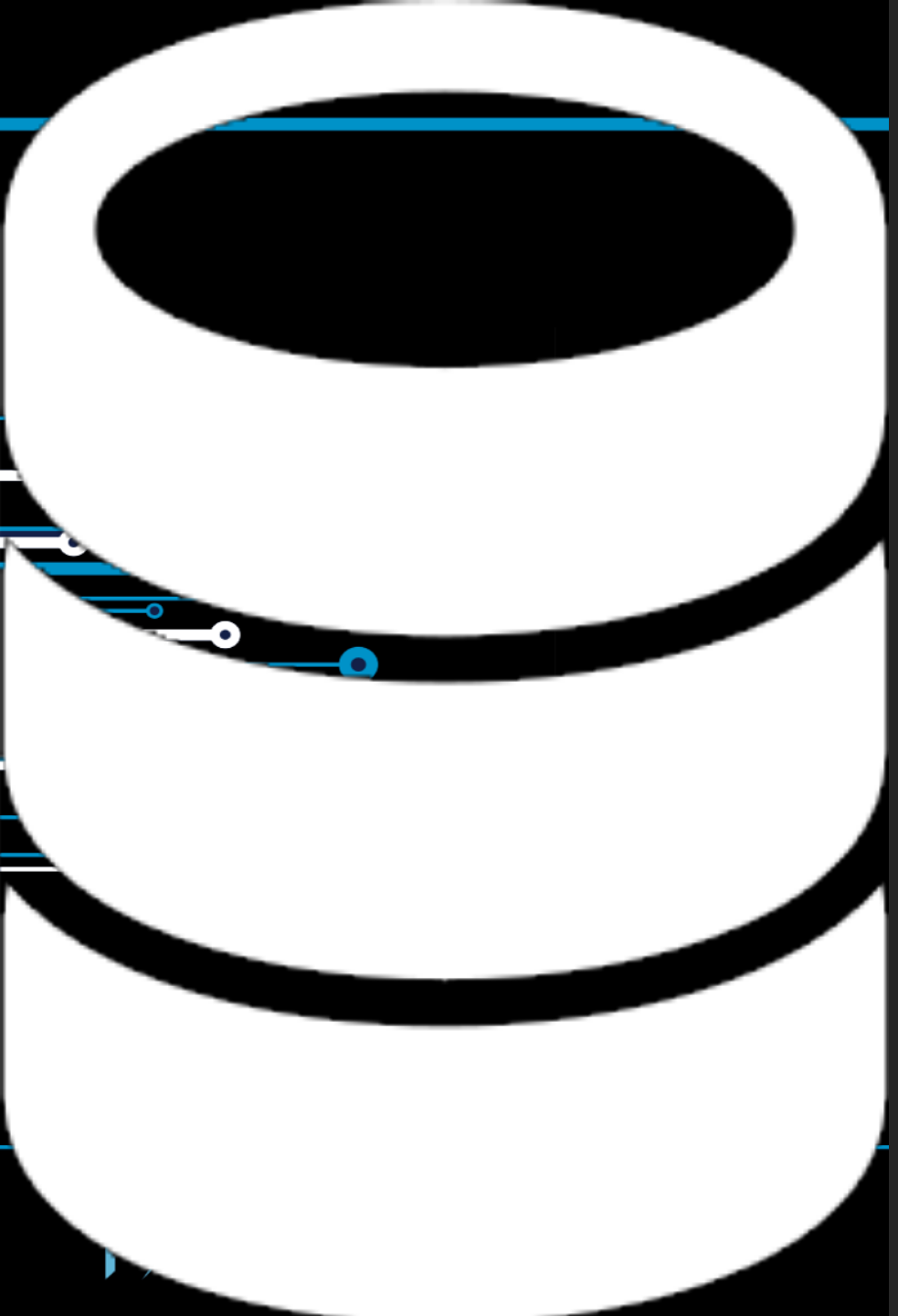


Crash Course on the History of Database Systems

Adapted from *"What Goes Around Comes Around,"*
by Hellerstein & Stonebraker


Why?

DATA MODELS

A data model is collection of concepts for describing the data in a database.

A schema is a description of a particular collection of data, using a given data model.

DATA MODELS

RELATIONAL

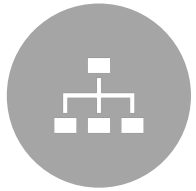
KEY/VALUE

GRAPH

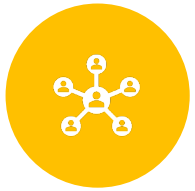
DOCUMENT

COLUMN-FAMILY

ARRAY / MATRIX



HIERARCHICAL



NETWORK

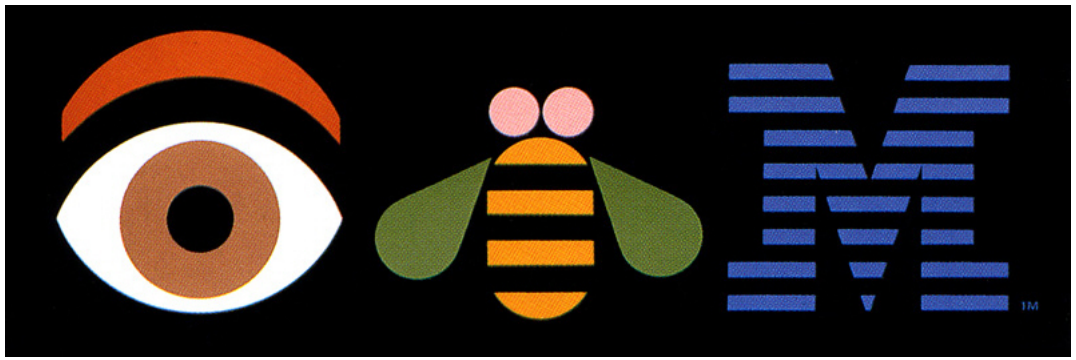
History Repeats Itself

Old database issues are still relevant today.

Many of the ideas in today's database systems are not new.

The "SQL vs. NoSQL" debate is reminiscent of "Relational vs. CODASYL" debate.

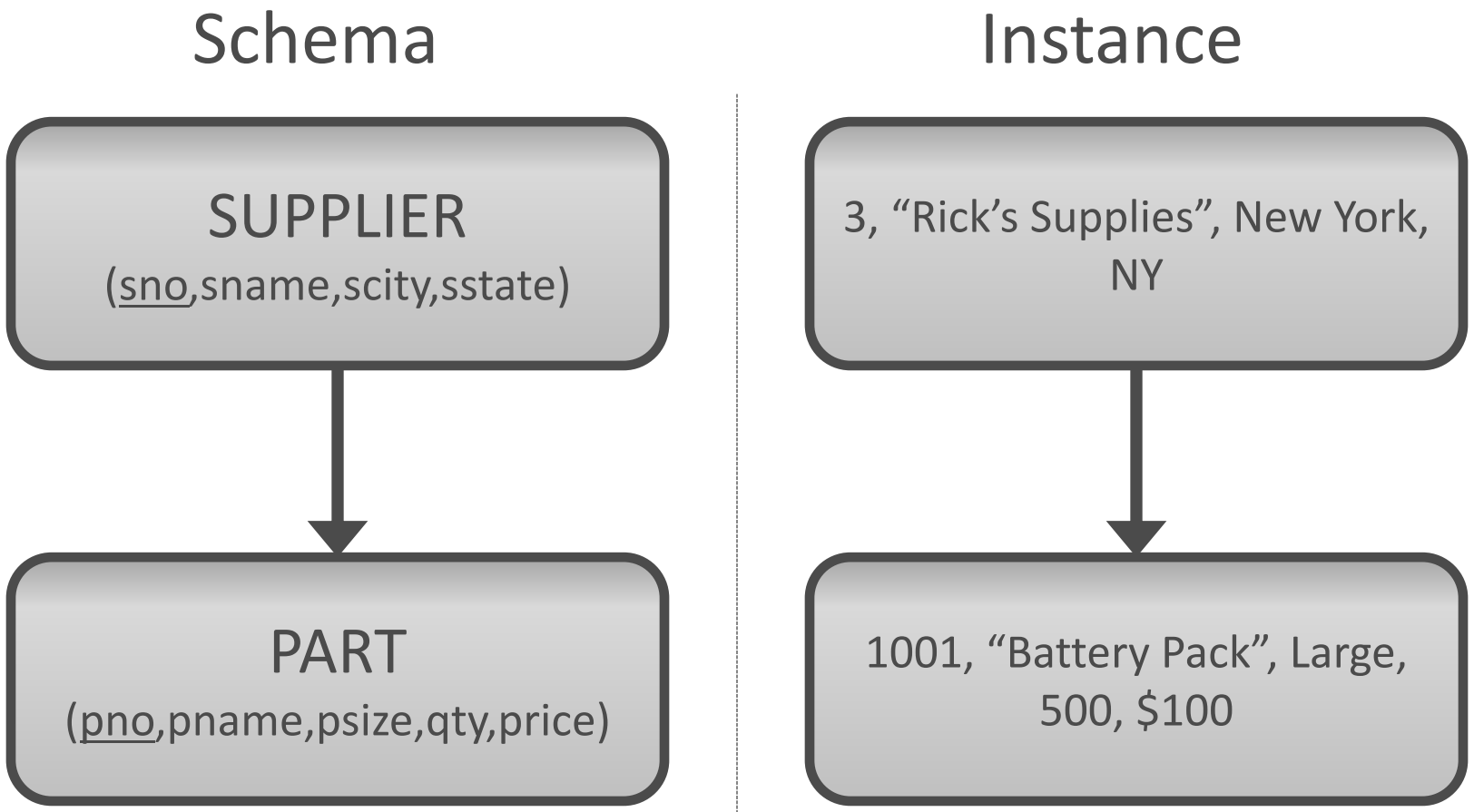
1960s – IBM IMS



- First database system.
- Hierarchical data model.
- Programmer-defined physical storage format.
- Tuple-at-a-time queries.

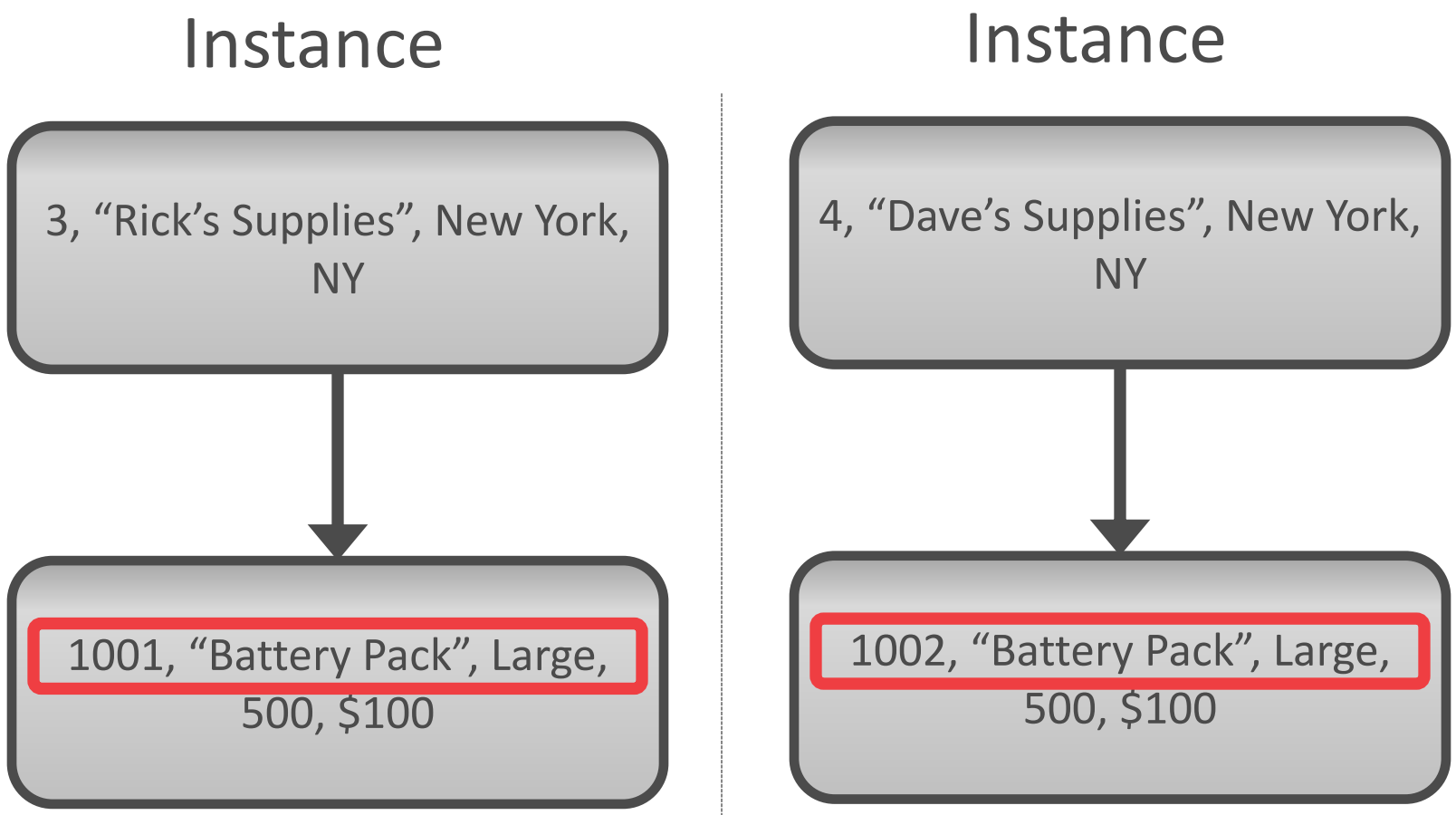
HIERARCHICAL DATA MODELS

Hierarchical Data Model



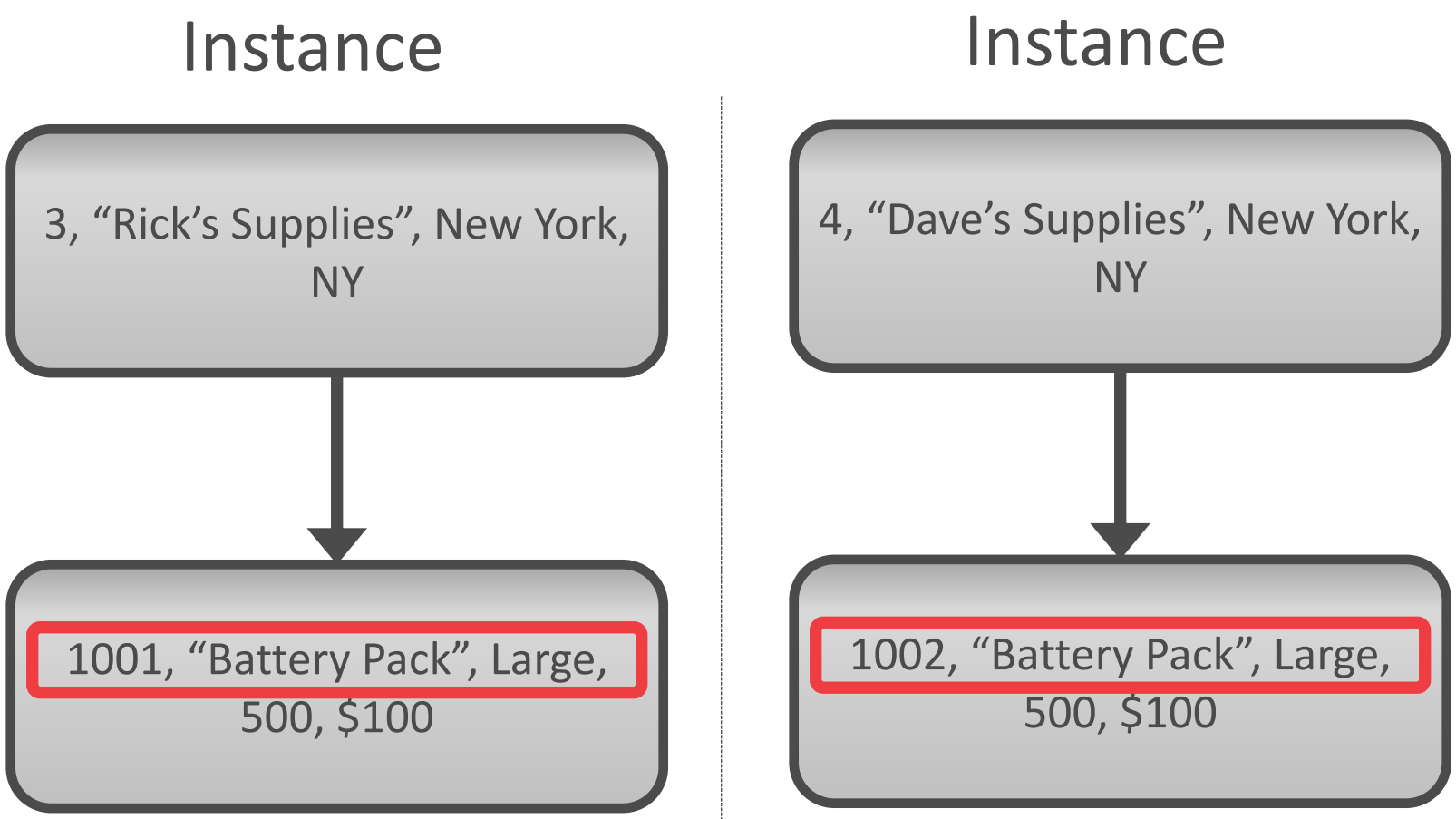
Can you have a part without a supplier?

Hierarchical Data Model



Duplicate data if same part sold by different supplies

Hierarchical Data Model



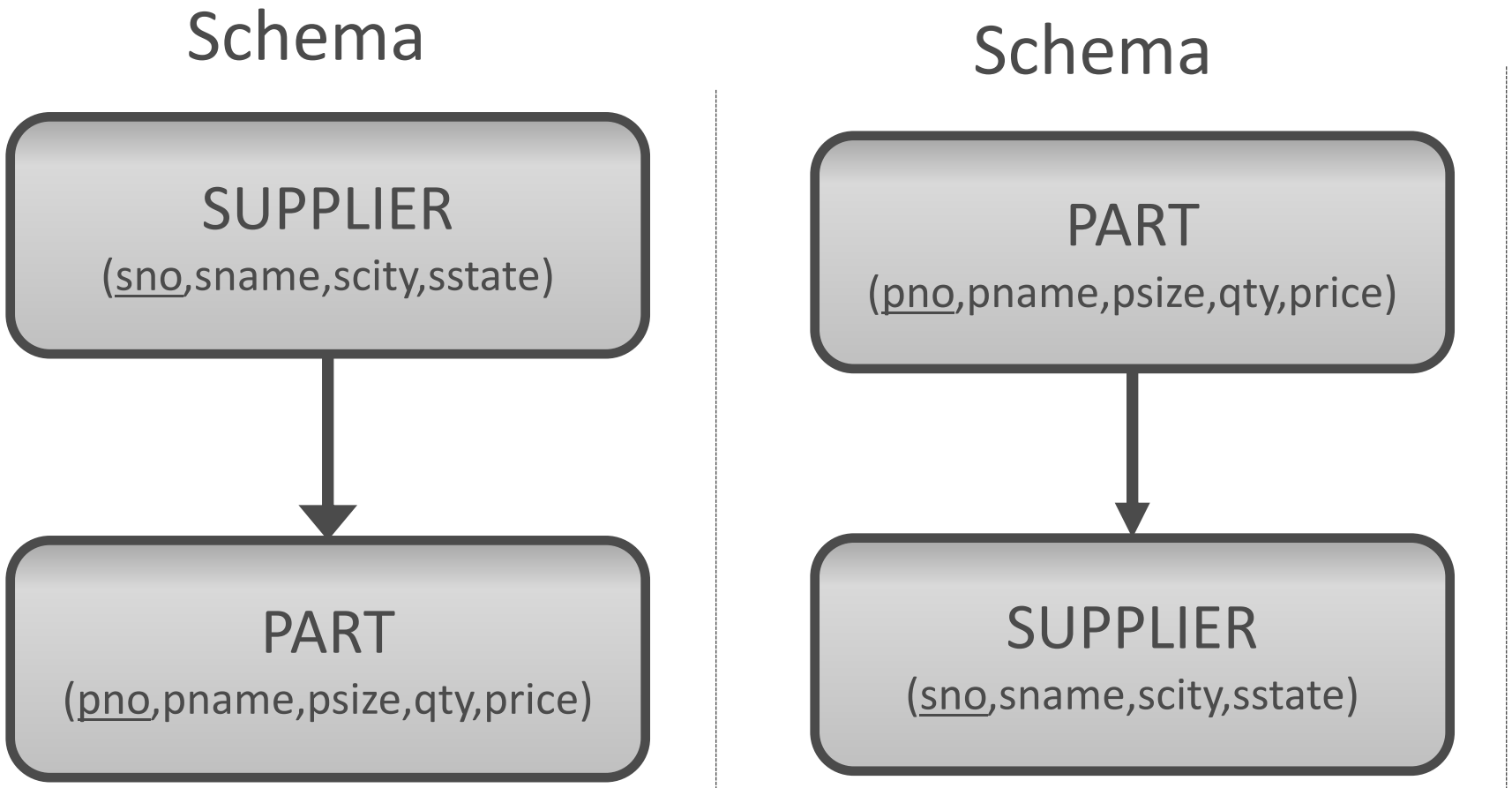
How do you update the price of a battery pack?

Hierarchical Data Model

Record-at-a-time

- Write algorithm to access data
- Iterate through hierarchy
- Breaks if hierarchy changes
- Performance is algorithm & data dependent

Schema impacts code



What happens when the schema changes?

Schema Impacts Code

```
For s in suppliers:
    For p in parts[s]:
        print s, p

For p in parts:
    For s in suppliers[p]:
        print s, p
```

Data Independence

- Data organization can change without programming change
- Important to work through **logical abstractions**

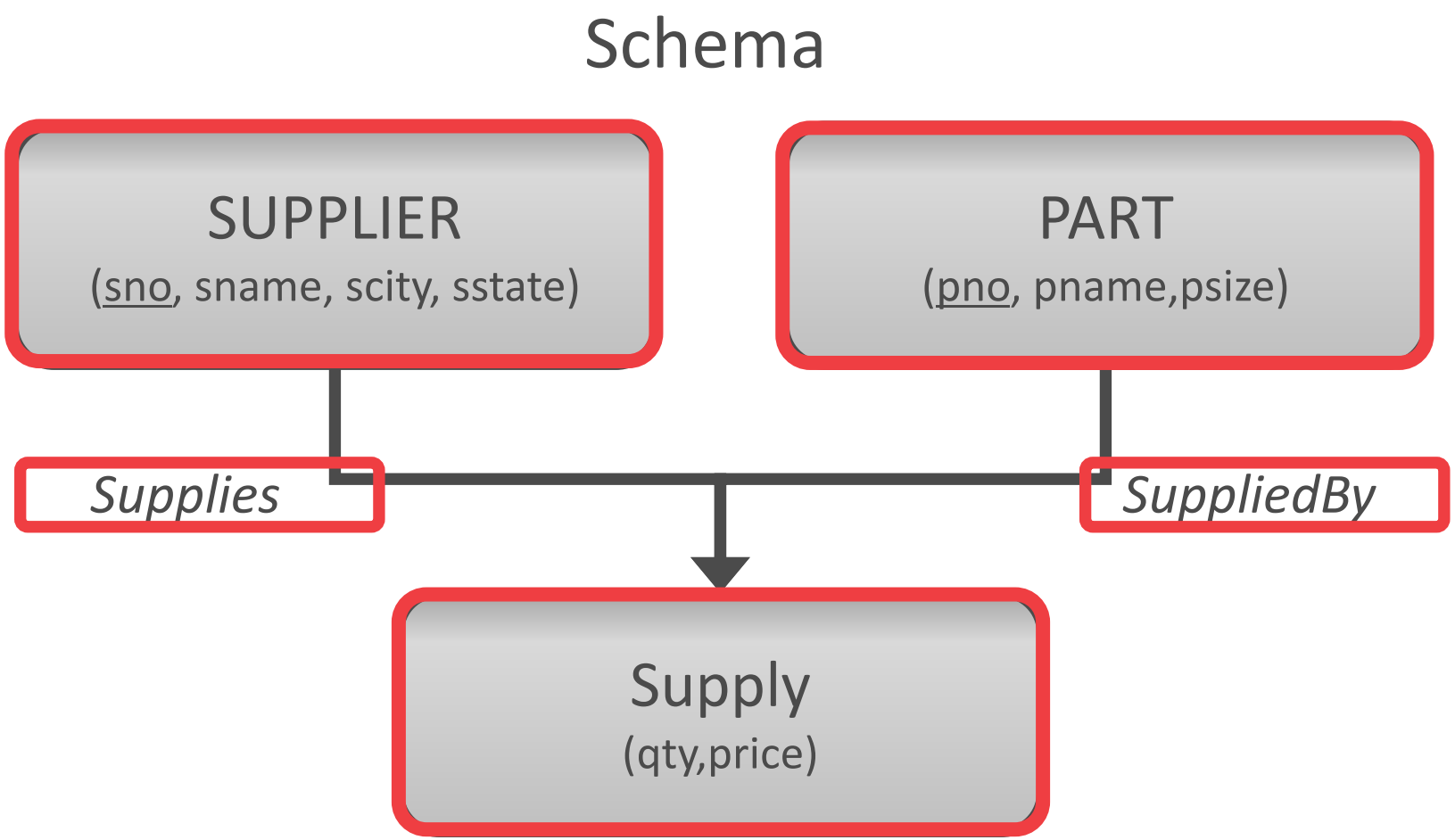
NETWORK DATA MODELS

Bachman

1970s – CODASYL

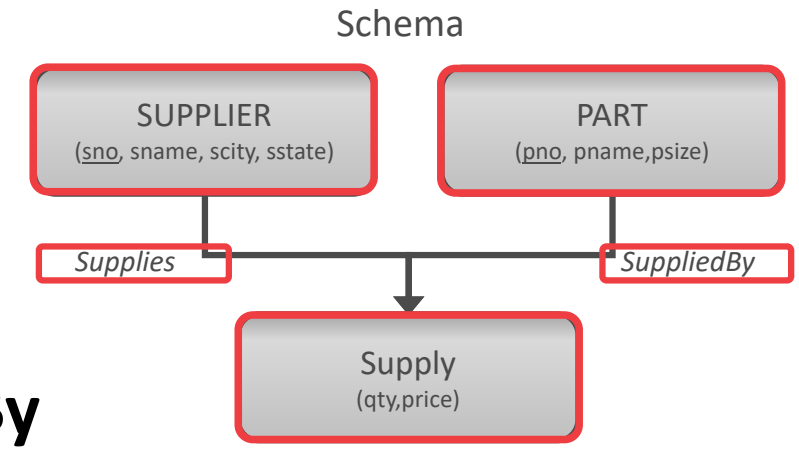
- COBOL people got together and proposed a standard.
- Network data model.
- Record-at-a-time queries.

Network Data Model



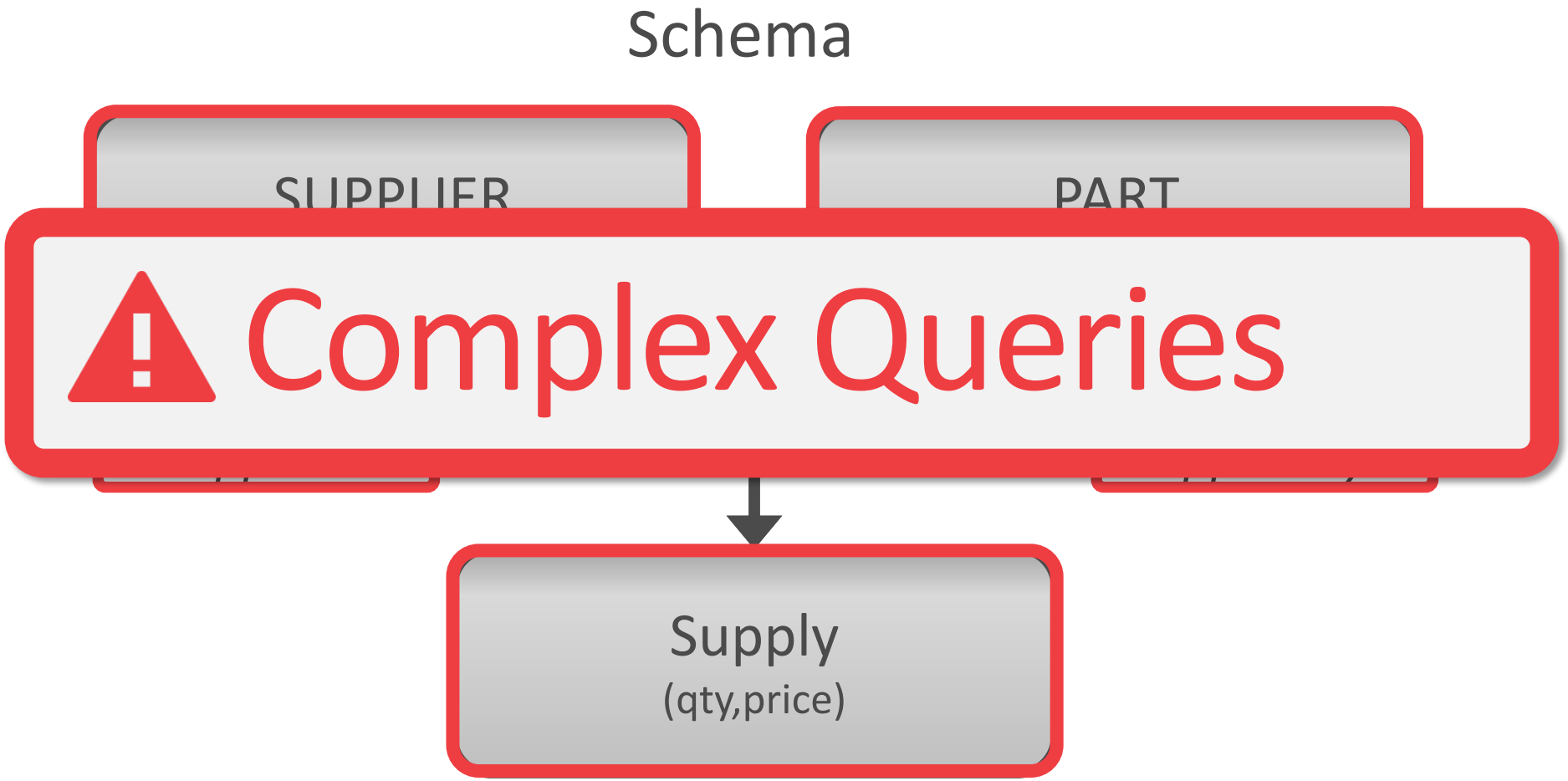
Network Data Model

Find **Supplier** (SNO=16)
Until no-more {
  Find next **Supply** record in **Supplies**
  Find owner **Part** record in **Supplied_By**
  Get current record & check for psize="large"
}



Programming defines how to retrieve data

Network Data Model

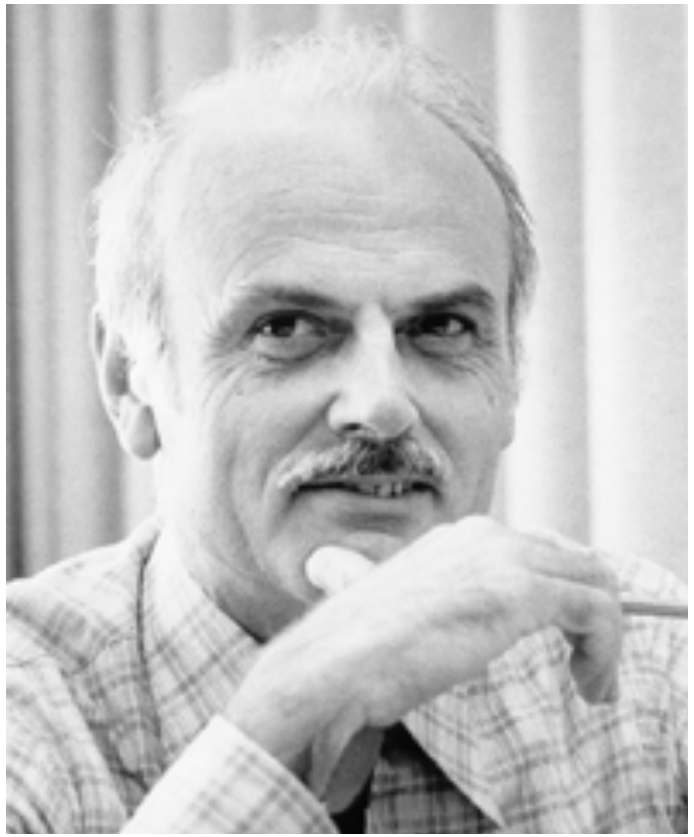


Stonebraker Lessons

- Physical and logical data independence are good.
- Tree-based data models are too restrictive.
- Record-at-a-time forces the programmer to do manual query optimization.

RELATIONAL DATA MODELS

1970s – Relational Model



Edgar F. Codd

- Codd saw the maintenance overhead for IMS/Codasyl.
- Proposes database abstraction based on tables.

Relational Model

Structure: Store database in simple data structures (i.e., tables).

Manipulation: Access it through high-level language (i.e., SQL).

Integrity: Ensure content satisfies constraints

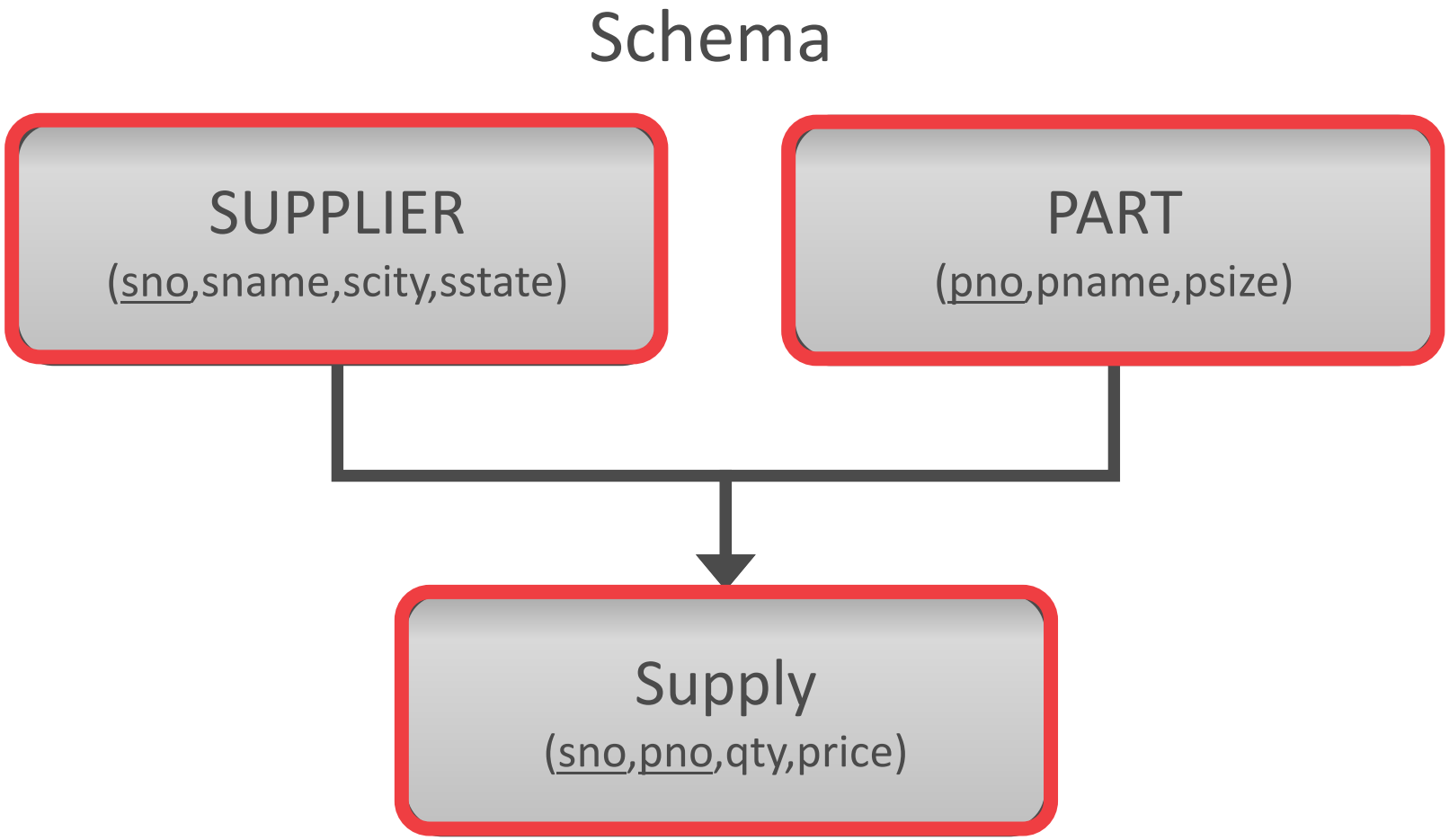
Schema

"A database schema specifies, based on the database administrator's knowledge of possible applications, the facts that can enter the database, or those of interest to the possible end-users."

Tables / relations composed of **attributes / fields**, which are of specific **types** (integer, text).

Imielinski, T.; Lipski, W. "A systematic approach to relational database theory". Proceedings of the 1982 ACM SIGMOD

Relational Data Model



RELATIONAL MODEL

- A relation is 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.
  - The special value **NULL** is a member of every domain.

Artist(name, year, country)

name	year	country
Wu Tang Clan	1992	USA
Notorious B.I.G.	1992	USA
Ice Cube	1989	USA

n-ary Relation

=

Table with *n* columns

RELATIONAL MODEL: PRIMARY KEY

- A relation's primary key is a candidate key that is deemed more "important" than other candidate keys.
- Some DBMSs automatically create an internal primary key for a table if you do not define one.

Artist(id, name, year, country)

id	name	year	country
101	Wu Tang Clan	1992	USA
102	Notorious B.I.G.	1992	USA
103	Ice Cube	1989	USA
104	Ice Cube	2017	India
105	Ice Cube	2017	USA
106	Ice Cube	2017	USA

RELATIONAL MODEL: FOREIGN KEYS

- A foreign key specifies that an attribute from one relation has to map to a tuple in another relation.

Artist(id, name, year, country)

id	name	year	country
101	Wu Tang Clan	1992	USA
102	Notorious B.I.G.	1992	USA
103	Ice Cube	1989	USA

ArtistAlbum(artist_id, album_id)

artist_id	album_id
101	11
101	22
103	22

Album(id, name, year)

id	name	artists	year
11	Enter the Wu Tang	101	1993
22	St.Ides Mix Tape	???	1994

Schema

Primary key (PK) -- unique identifier

Foreign key (FK) – reference to PK

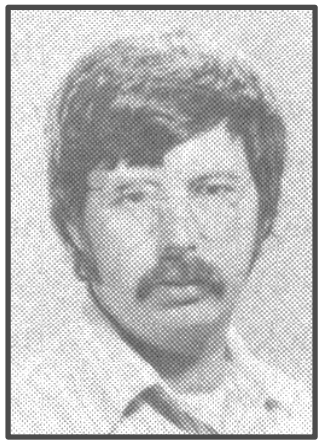
Normalization – reduce redundancy

- Divide info across smaller tables
- Define relationships between tables
- Isolate data so operations can modify one table
- Every table should have a PK

1970s – Relational Model

- System R – IBM Research
- INGRES – Berkeley
- Oracle – Larry Ellison


Gray


Stonebraker


Ellison

1980s – Relational Model

- IBM comes out with DB2.
- SQL becomes the standard.
- Oracle wins marketplace.
- Stonebraker creates Postgres.

Stonebraker

Stonebraker Lessons

Set-at-a-time interface offers better physical data independence.

Database system optimizer is better than manual tuning.

Set-at-a-time interface

Operate on many rows at once

Common operation applied across sets of data

Mathematically easier to follow

Quick Detour SQL

SQL

- Logical language to express database operations
  - Data description language (DDL):
    - create, drop tables / views
  - Data manipulation language (DML):
    - select, insert, update, delete
  - *View: Logical view of tables expressed with a select query*
  - *Index: Fast lookup to data (B-Tree, R-Tree, Hash Table)*

SQL Reviews:

1. https://github.com/swcarpentry/sql-novice-survey/
2. https://www.w3schools.com/sql/sql_join.asp

SQL

- *Create table tweet(attr1 int, attr2 int);*
- *Create view v_tweet_a1 AS*
  select attr1 from tweet;
- *Insert into tweet VALUES (1, 2);*
- *Select attr1, attr2 from tweet;*
- *Update tweet set attr2 = 3 where attr1 = 1;*
- *Delete from tweet where attr1 = 1;*
- *Drop table tweet;*

Semantics

- Set-based
- Order of rows not specified by default
- Set-based operations
  - UNION, INTERSECT, DISTICT, …
- Aggregation operations (group by):
  - Count, sum, average, …
- Can explicitly order results

Optimizer

- Provide DB your SQL query (logical)
- It optimizes the query
  - Method of access (scans, indexes)
  - Determines access path (order data are accessed)
  - Determines how to combine / join multiple tables
  - Creates a physical query plan
    (to see, run EXPLAIN <query>)
  - Uses DB stats and machine resources to optimize
- The optimizer is often smarter than you

Usage: Terminal

- sqlite3 tempdb
- >run sql commands from previous slide
- [for help] >.help
- [to quit] >.q

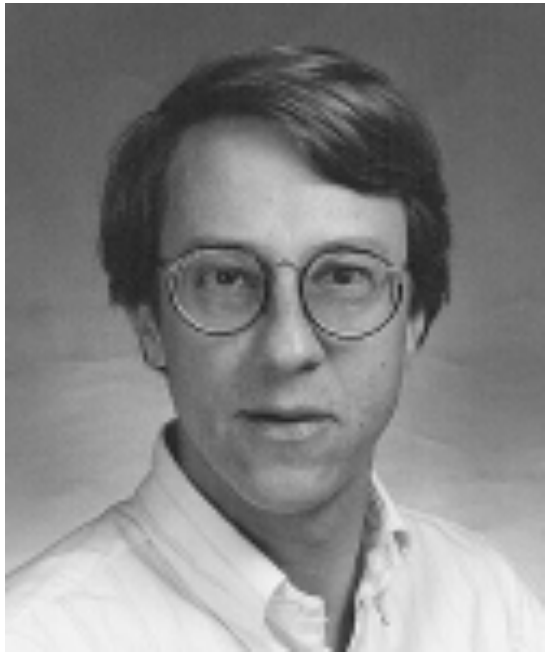
- *Create table tweet(attr1 int, attr2 int);*
- *Create view v_tweet_a1 AS*
  select attr1 from tweet;
- *Insert into tweet VALUES (1, 2);*
- *Select attr1, attr2 from tweet;*
- *Update tweet set attr2 = 3 where attr1 = 1;*
- *Delete from tweet where attr1 = 1;*
- *Drop table tweet;*

Usage: Python

```
import sqlite3
con = sqlite3.connect('tempdb')
cur = con.cursor()
cur.execute('create table tweet(attr1 int, attr2 int)')
cur.execute('insert into tweet VALUES (1, 2)')
con.commit()
cur.execute('select * from tweet')
for row in cur.fetchall():
        print row
```

Note: tempdb is a file created in the current directory

Distributed Databases



Bernstein

Mohan

DeWitt

Gray

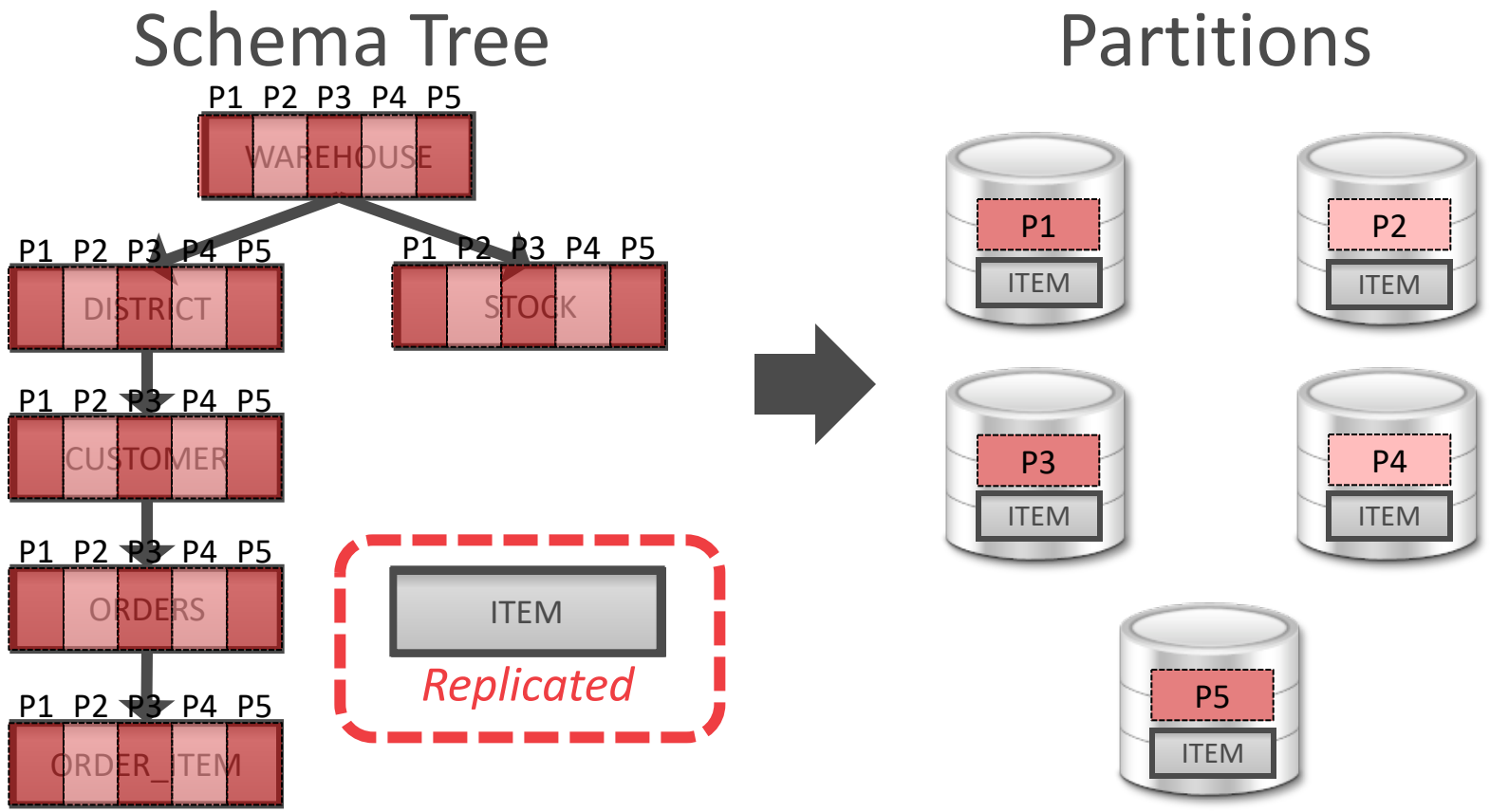
1980s – Distributed DBs

- SDD-1 – CCA
- System R* – IBM Research
- Gamma – Univ. of Wisconsin
- NonStop SQL – Tandem

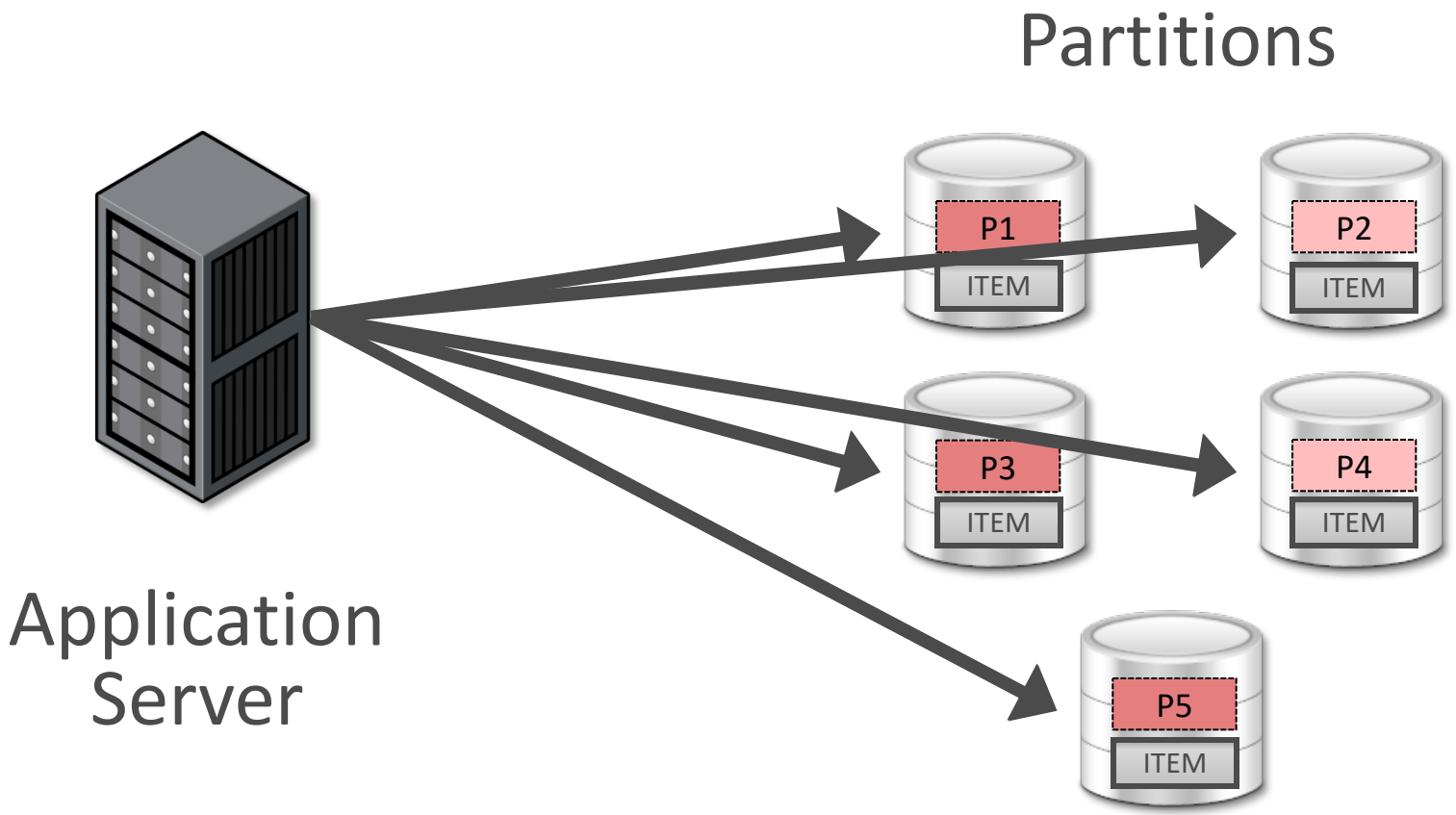
Distributed Databases

- Data split across many machines
- Execute queries that pull data from many machines
- Ensure results are correct
- Execute millions of operations / sec

Database Partitioning



Distributed OLTP



1980s – Object Oriented Databases

- Avoid "relational-object impedance mismatch."
- Tight coupling between objects and database.

Zdonik

Object-Oriented Model

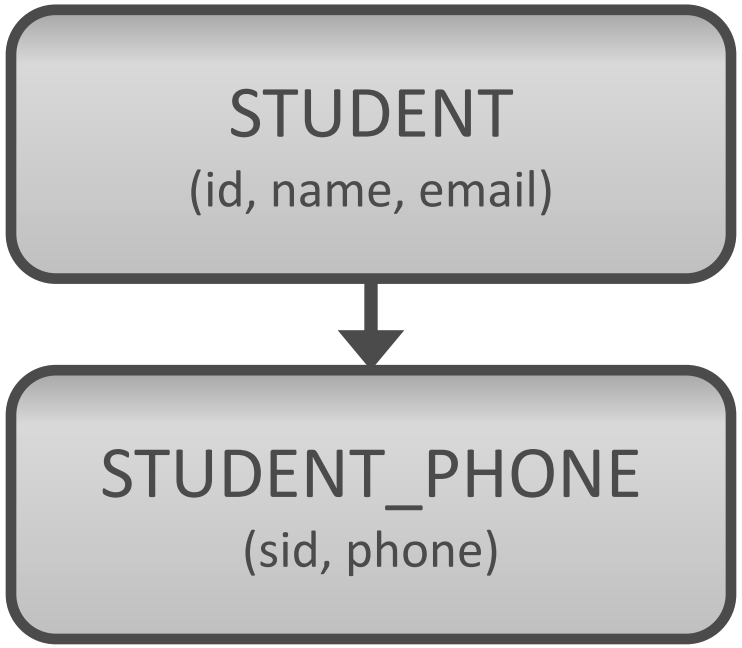
Application Code

```
class Student {
   int id;
   String name;
   String email;
   String phone[];
}
```

id	name	email
1001	Tone Loc	funky@medina.com

sid	phone
1001	444-444-4444
1001	555-555-5555

Schema



https://sequelize.readthedocs.io/en/v3/: ORM

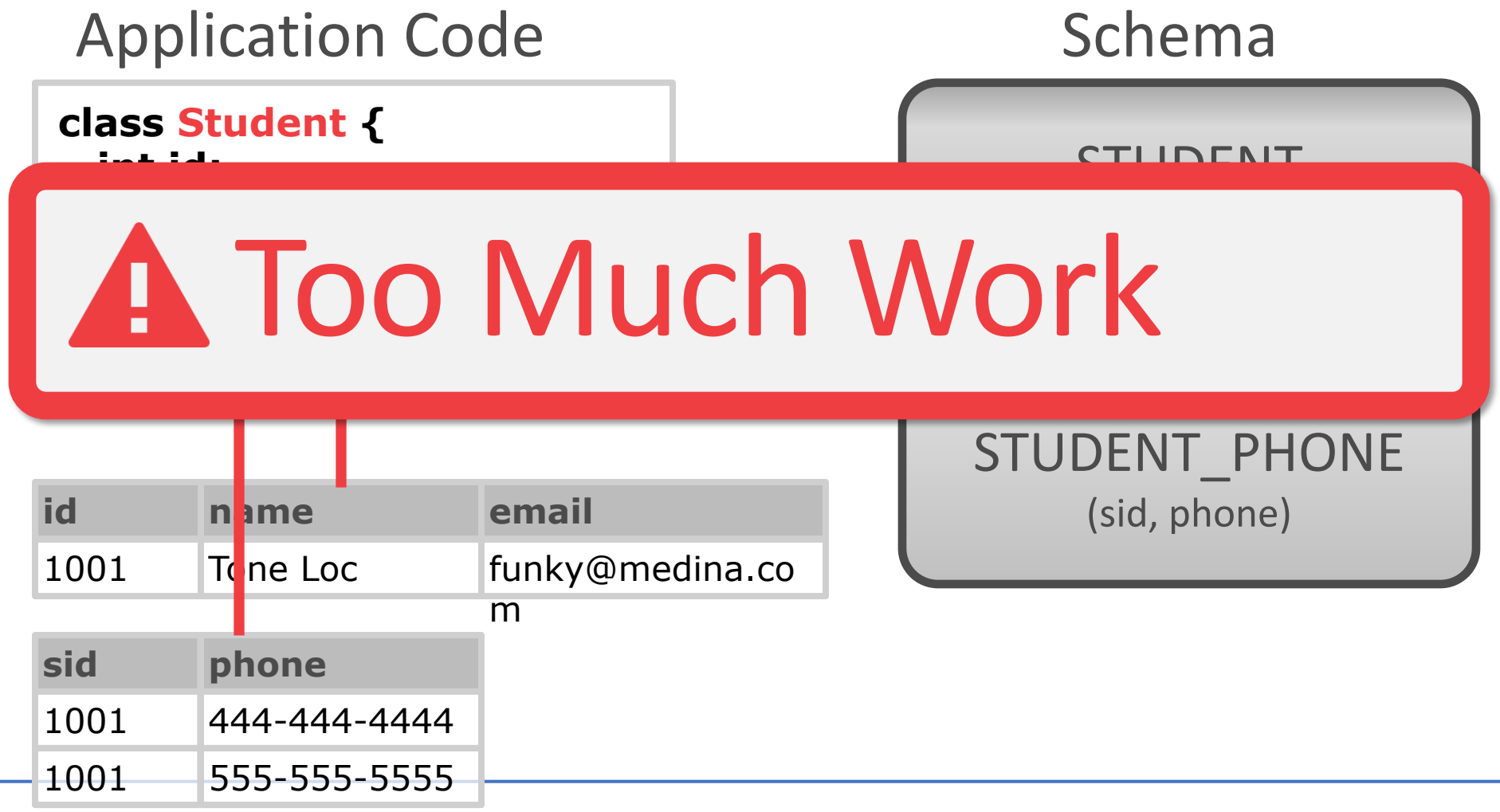
```
SELECT * FROM USERS WHERE zip_code=94107;
```

The equivalent Django ORM query would instead look like the following Python code:

```
# obtain everyone in the 94107 zip code and assign to users variable


users = Users.objects.filter(zip_code=94107)
```

Object-Oriented Model



1990s – Boring Days

- Microsoft forks Sybase and creates SQL Server.
- MySQL is written as a replacement for mSQL.
- Postgres gets SQL support.

2000s – Internet Boom

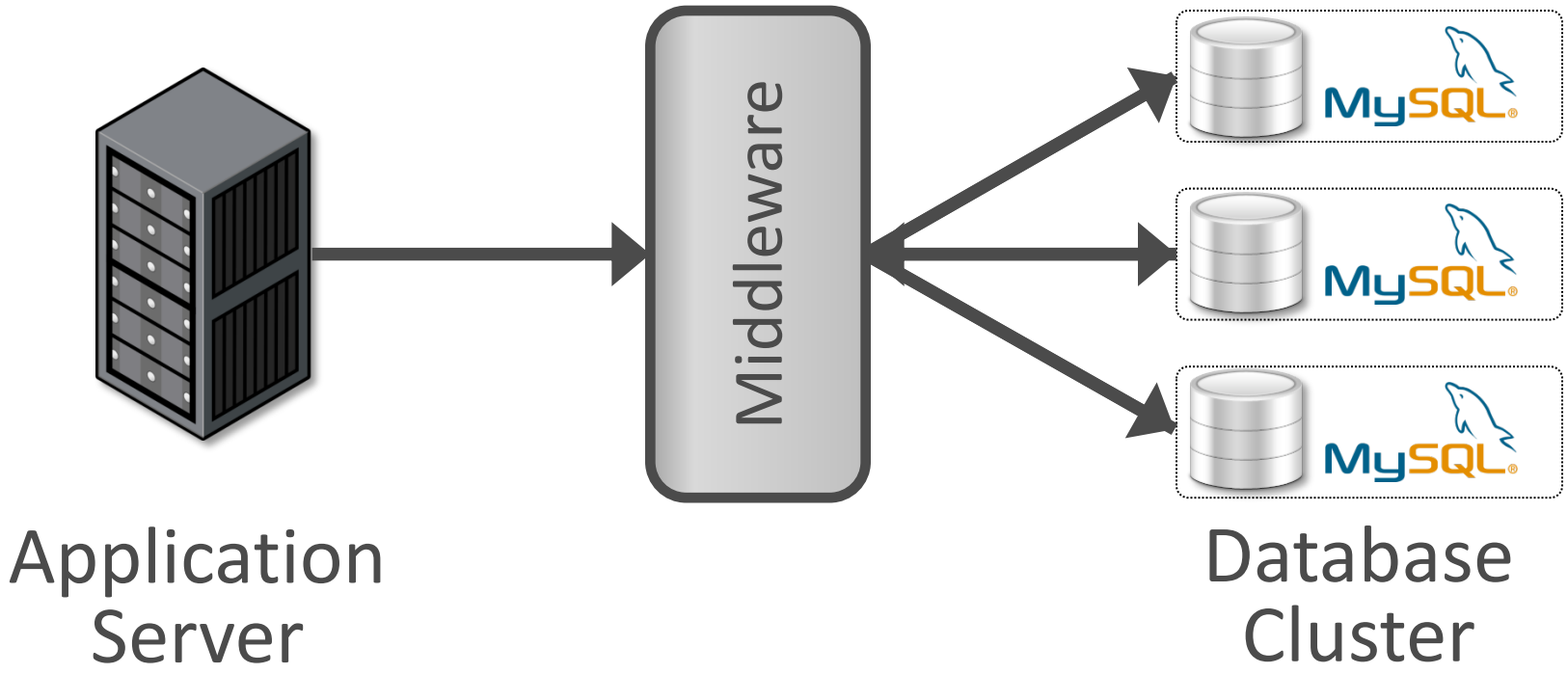
All the big players were heavyweight and expensive.

Open-source databases were missing important features.

Custom scale-out middleware.

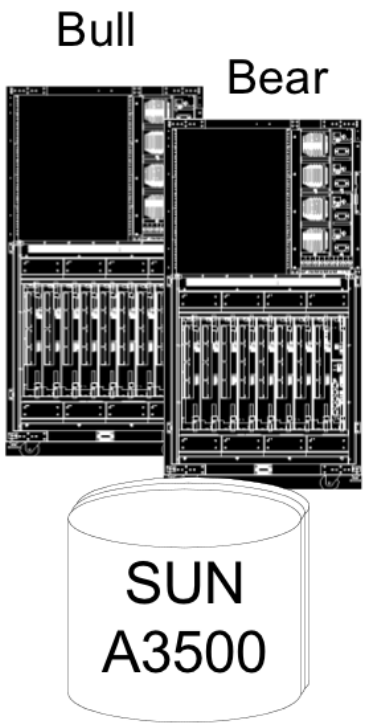
Examples: eBay, Facebook

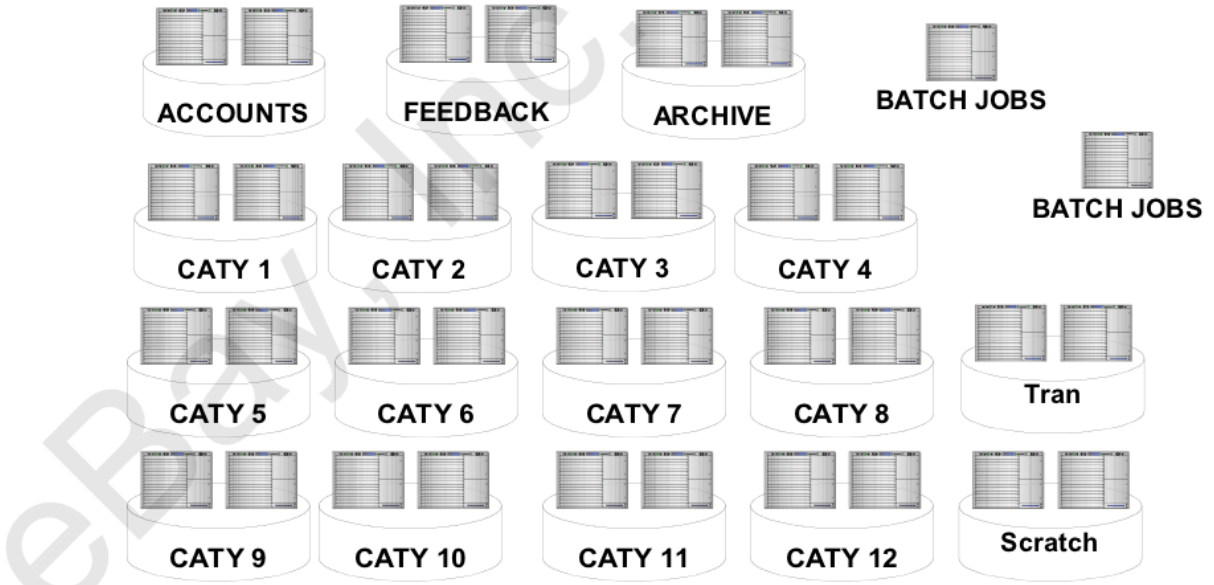
Middleware Approach



V2.5 *April 2001 – December 2002*

- Horizontal scalability through database splits
- Items split by category
- SPOF elimination





December, 2002

Randy Shoup - *"The eBay Architecture"*
http://highscalability.com/ebay-architecture

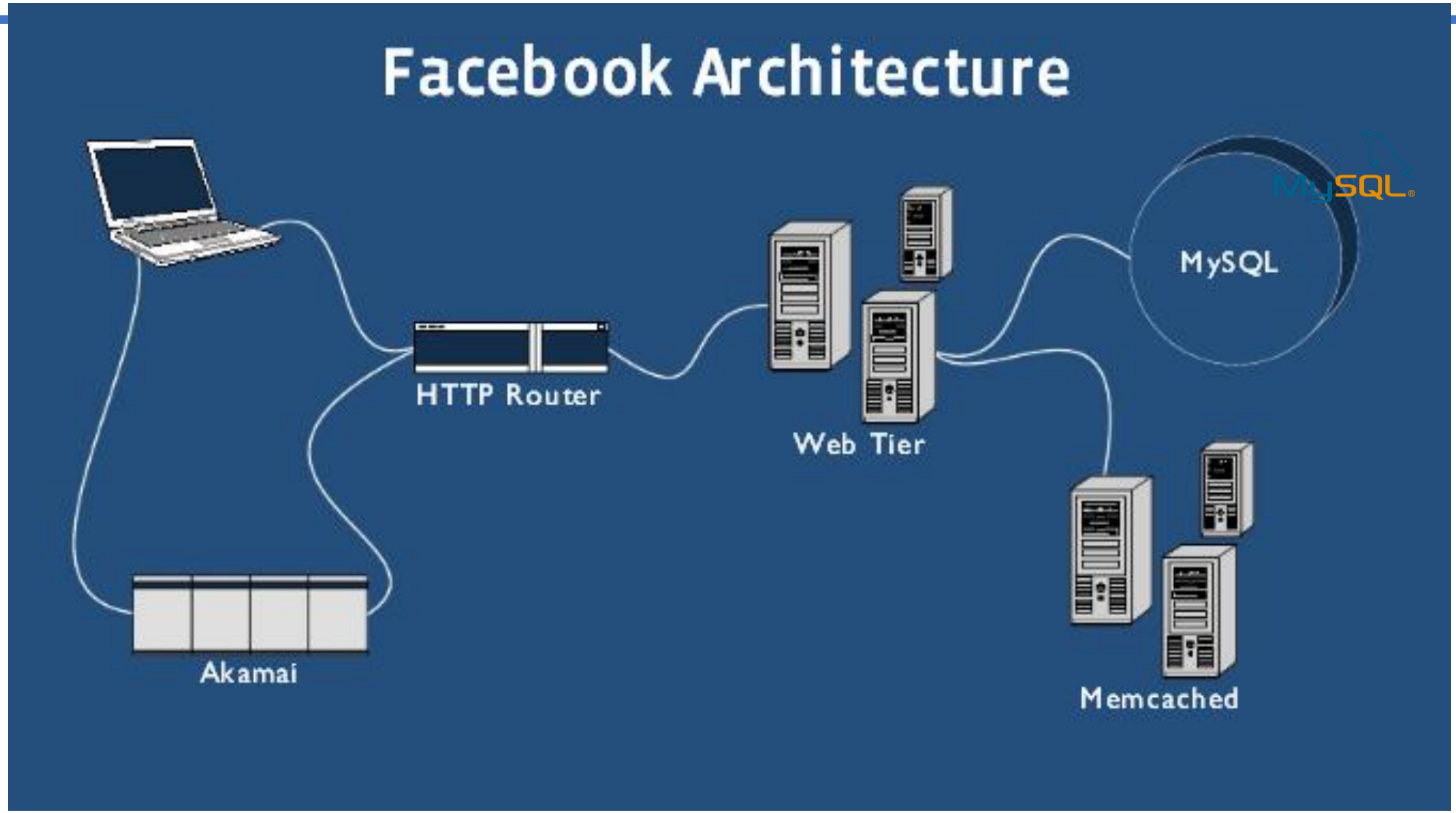
V2.5 *April 2001 – December 2002*

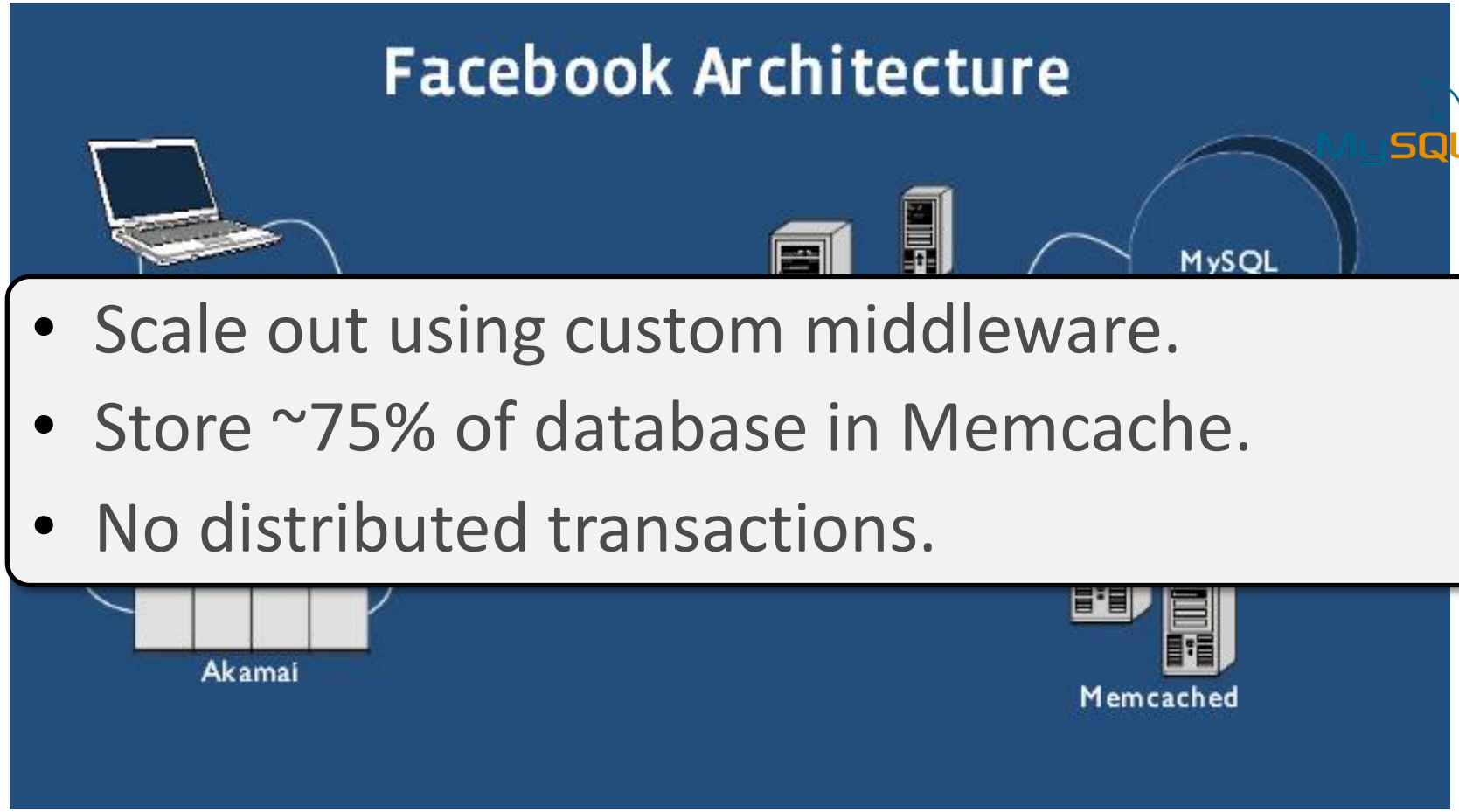
- Horizontal scalability through database splits
- Items split by category
- SPOF elimination

- Push functionality to application:
  - *Joins*
  - *Referential integrity*
  - *Sorting*
- No distributed transactions.

December, 2002

Randy Shoup - *"The eBay Architecture"*
http://highscalability.com/ebay-architecture


Facebook Architecture
MySQL
MySQL
HTTP Router
Web Tier
Akamai
Memcached



- Scale out using custom middleware.
- Store ~75% of database in Memcache.
- No distributed transactions.

2000s – NoSQL

- Focus on high-availability & high-scalability:
  - *Schema-less ("Schema Last")*
  - *Not ACID*
  - *Custom APIs instead of SQL.*

2000s – NoSQL

- Alternative data models:
  - *Column-family (Cassandra, HBase)*
  - *Document (MongoDB, CouchDB)*
  - *Key-value (Riak, Dynamo)*
  - *Graph (Neo4j, FlockDB)*
- Usually open-source.
- “A” + “P” in CAP Theorem

DETOUR: CAP THEOREM

CAP Theorem

- **C**onsistency:
  - All nodes should see the same data at the same time
- **A**vailability:
  - Node failures do not prevent survivors from continuing to operate
- **P**artition-tolerance:
  - The system continues to operate despite network partitions
- A distributed system can satisfy any two of these guarantees at the same time **but not all three**

CAP Theorem: Proof

- 2002: Proven by research conducted by Nancy Lynch and Seth Gilbert at MIT

Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT News 33.2 (2002): 51-59.

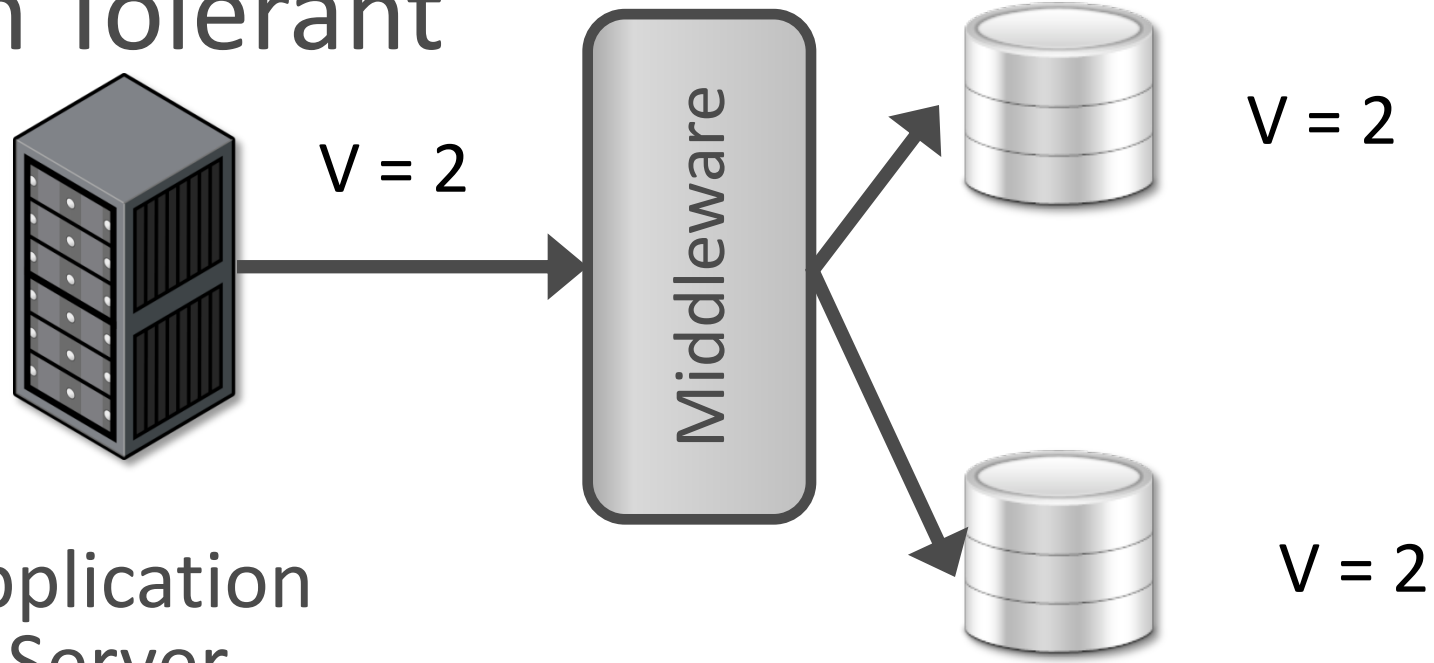
Consistency
Availability
Partition Tolerant

Application Server

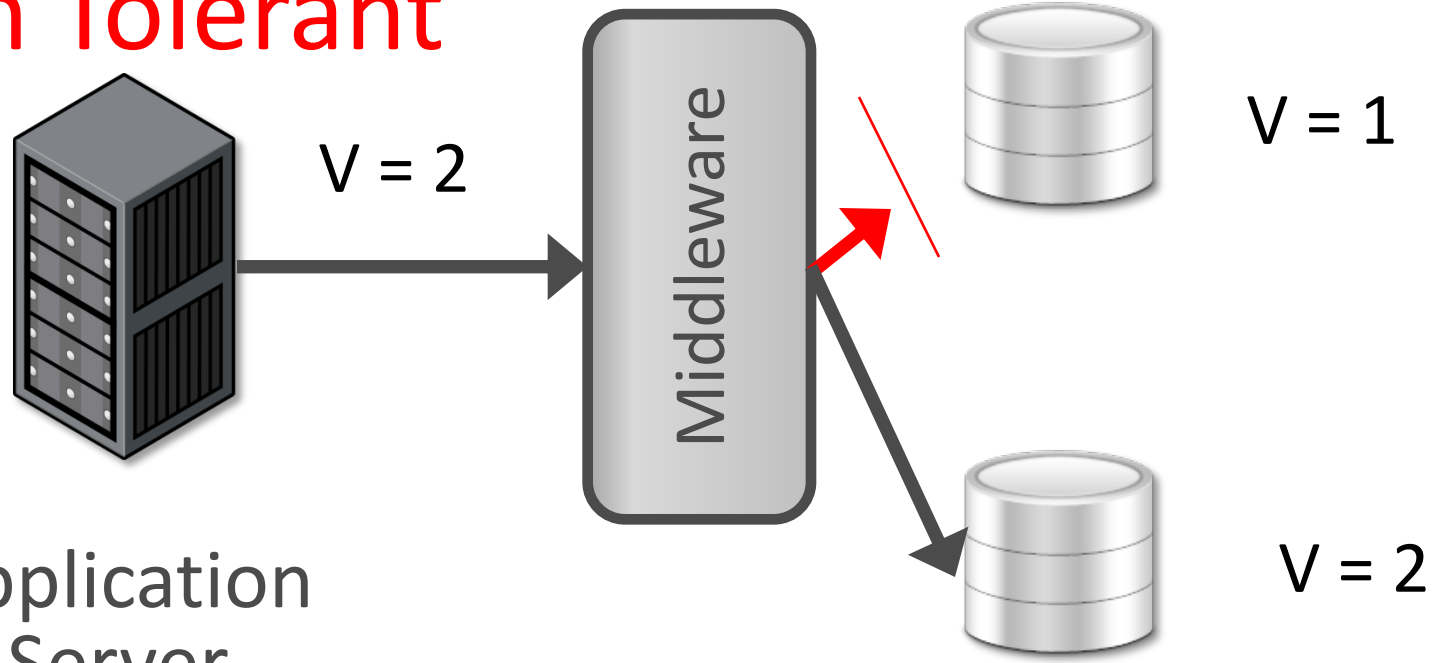
V = 1

V = 1

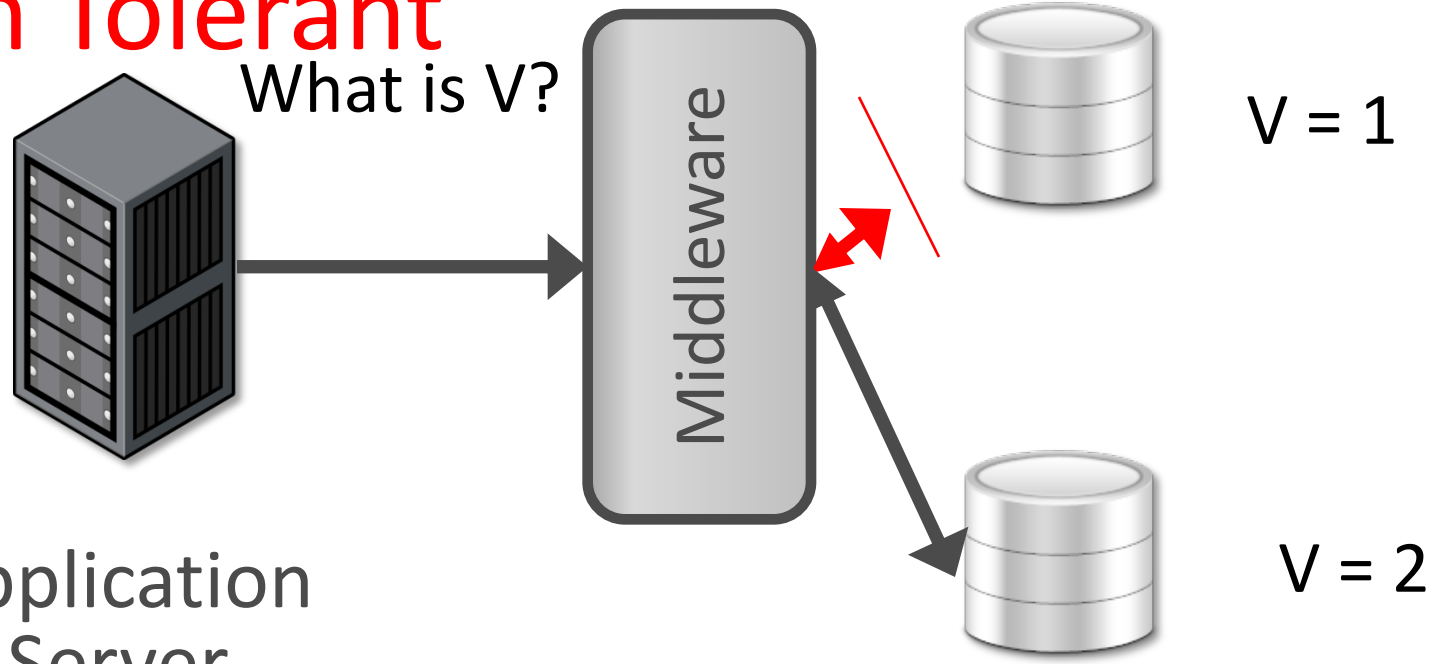
Consistency
Availability
Partition Tolerant



Consistency
Availability
Partition Tolerant



Consistency
Availability
Partition Tolerant



Do you respond with V = 1, V = 2 or **not** respond?

Why this is important?

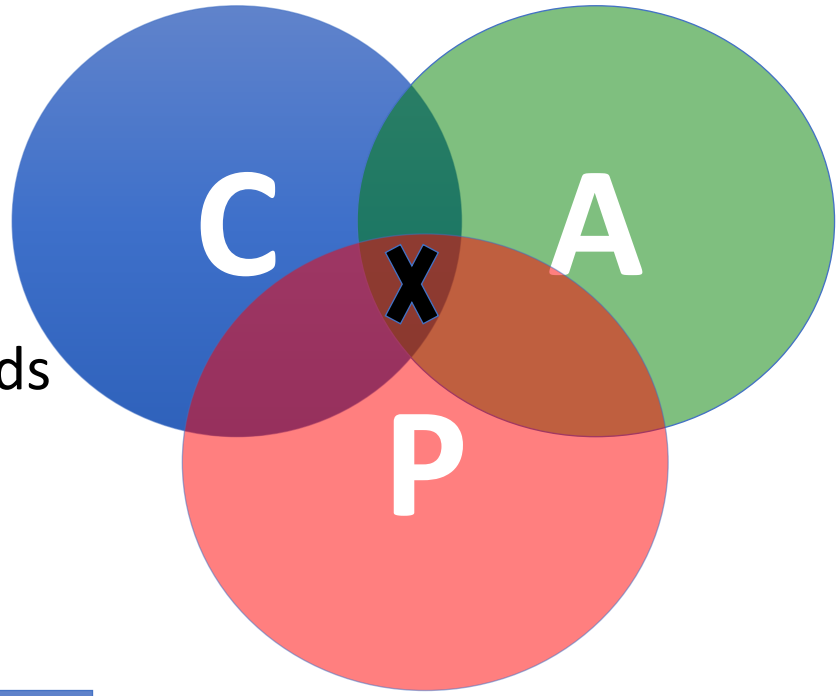
- All modern databases are **distributed** (Big Data Trend, etc.)
- CAP theorem describes the **trade-offs** involved in distributed systems
- A proper understanding of CAP theorem is essential to **making decisions** about the distributed database **design**
- Misunderstanding can lead to **erroneous or inappropriate** design choices

Problem for Relational Database to Scale

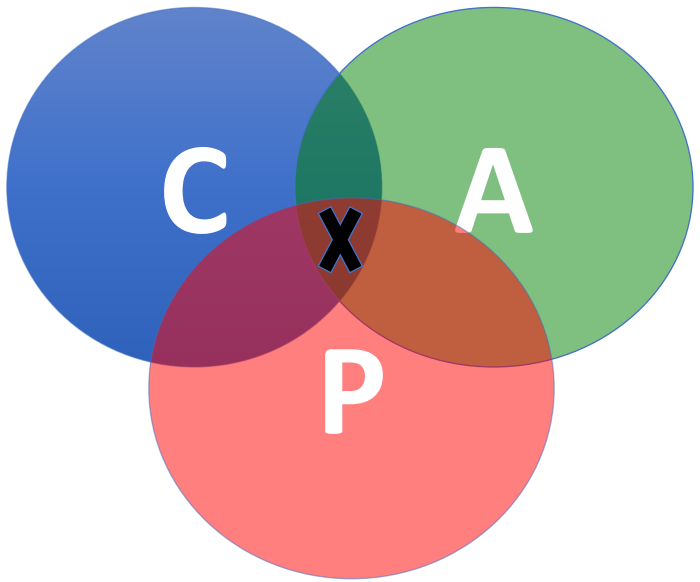
- The Relational Database is built on the principle of **ACID** (Atomicity, Consistency, Isolation, Durability)
- It implies that a truly distributed relational database should have **availability, consistency and partition tolerance**.
- Which unfortunately is **impossible** …

Revisit CAP Theorem

- Of the following three guarantees potentially offered a by distributed systems:
  - Consistency
  - Availability
  - Partition tolerance
- Pick two
- This suggests there are three kinds of distributed systems:
  - CP
  - AP
  - CA



Any problems?



Consistency or Availability

- Consistency and Availability is not “binary” decision
- AP systems relax consistency in favor of availability – but are not inconsistent
- CP systems sacrifice availability for consistency- but are not unavailable
- This suggests both AP and CP systems can offer a degree of consistency, and availability, as well as partition tolerance

AP: Best Effort Consistency

Example:

- Web Caching
- DNS

Trait:

- Optimistic
- Expiration/Time-to-live
- Conflict resolution

CP: Best Effort Availability

Example:

- Majority protocols
- Distributed Locking (Google Chubby Lock service)

Trait:

- Pessimistic locking
- Make minority partition unavailable

Types of Consistency

Strong Consistency

- After the update completes, **any subsequent access** will return the **same** updated value.

Weak Consistency

- It is **not guaranteed** that subsequent accesses will return the updated value.

Eventual Consistency

- Specific form of weak consistency
- It is guaranteed that if **no new updates** are made to object, **eventually** all accesses will return the last updated value (e.g., *propagate updates to replicas in a lazy fashion*)

Eventual Consistency Variations

Causal consistency

- Processes that have causal relationship will see consistent data

Read-your-write consistency

- A process always accesses the data item after it's update operation and never sees an older value

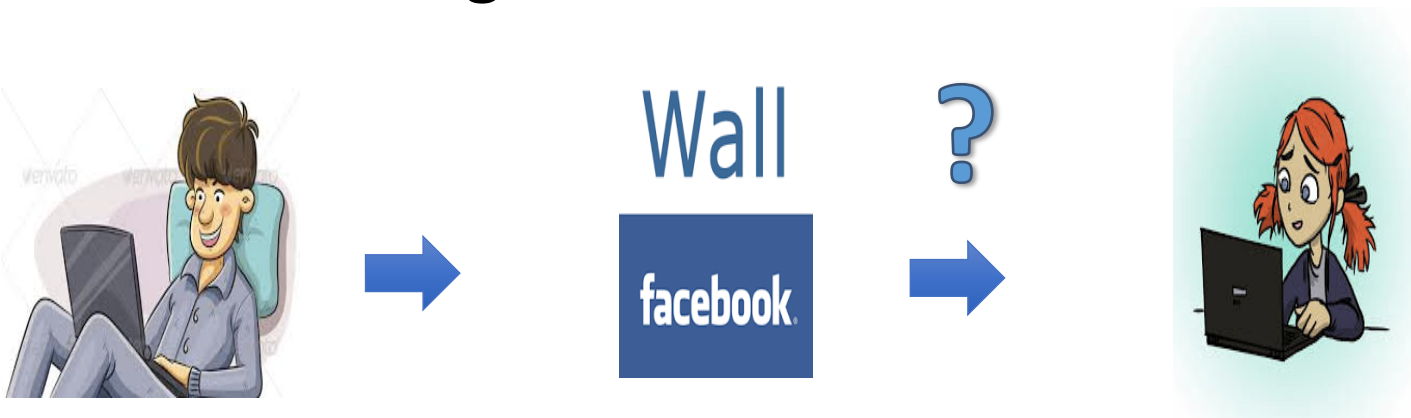
Session consistency

- As long as session exists, system guarantees read-your-write consistency
- Guarantees do not overlap sessions

Eventual Consistency : A facebook Example

- Bob finds an interesting story and shares with Alice by posting on her Facebook wall
- Bob asks Alice to check it out
- Alice logs in her account, checks her Facebook wall but finds:

  - **Nothing is there!**



Eventual Consistency : A facebook Example

- Bob tells Alice to wait a bit and check out later
- Alice waits for a minute or so and checks back:

- She finds the story Bob shared with her!

Eventual Consistency - A Facebook Example

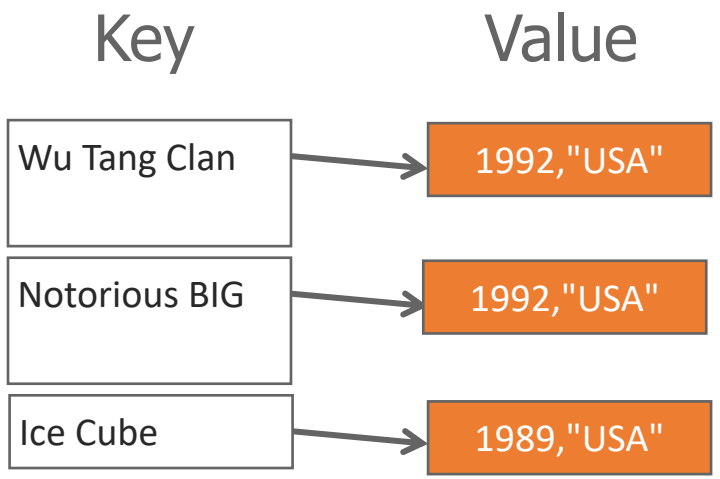
- Reason: it is possible because Facebook uses an **eventual consistent model**
- Why Facebook chooses eventual consistent model over the strong consistent one?
  - Facebook has more than 1 billion active users
  - It is non-trivial to efficiently and reliably store the huge amount of data generated at any given time
  - Eventual consistent model offers the option to **reduce the load and improve availability**

NON-RELATIONAL DATA MODELS

NON-RELATIONAL DATA MODELS: KEY-VALUE

- Store records in an **associative array** that maps a key to a value.
  - Sometimes called a **dictionary** or **hash table**.
- The contents of a record's value is often opaque to DBMS.
  - It is up to the application to interpret its contents.
  - Some systems allow for compound values.

Artist: name→(year,country)

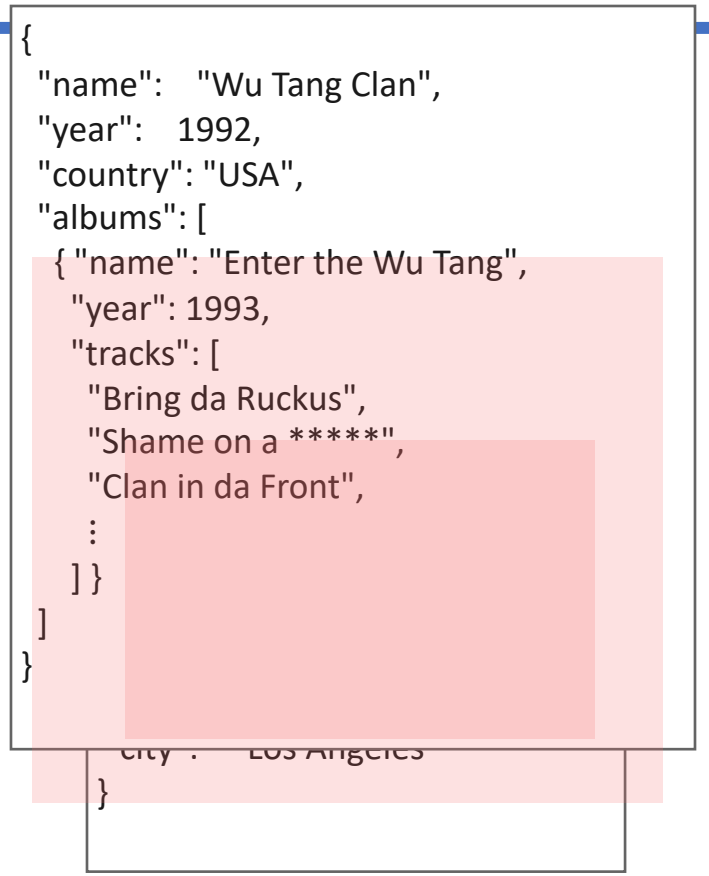


RocksDB amazon DynamoDB redis LMDB AEROSPIKE ORACLE BERKELEY DB

NON-RELATIONAL DATA MODELS: DOCUMENT

- A document is a self-contained record that contains the description of its data attributes and their corresponding values.

- Support nesting together all of the documents for a single entity.
  - This is called **denormalization** in the relational model.



MarkLogic CrateDB mongoDB CouchDB RAVENDB Couchbase

2000s – NoSQL

- Alternative data models:
  - *Column-family (Cassandra, HBase)*
  - *Document (MongoDB, CouchDB)*
  - *Key-value (Riak, Dynamo)*
  - *Graph (Neo4j, FlockDB)*
- Usually open-source.
- “A” + “P” in CAP Theorem

2010s – NewSQL

- Provide same performance of NoSQL without giving up ACID
  - *Relational / SQL*
  - *Distributed (Mostly)*
- Usually closed-source.

2010s – NewSQL

- Different solutions:
  - *Specialized OLTP (H-Store, VoltDB)*
  - *Distributed MVCC (NuoDB)*
  - *Custom Hardware (Clustrix, Spanner)*
  - *Relaxed Consistency (MemSQL, SQLFire)*
  - *Middleware (ScaleBase, dbShards)*

Observations

- Innovations come from both industry and academia.
- IBM was the vanguard during 1970-1980s.
- Google is current trendsetter.

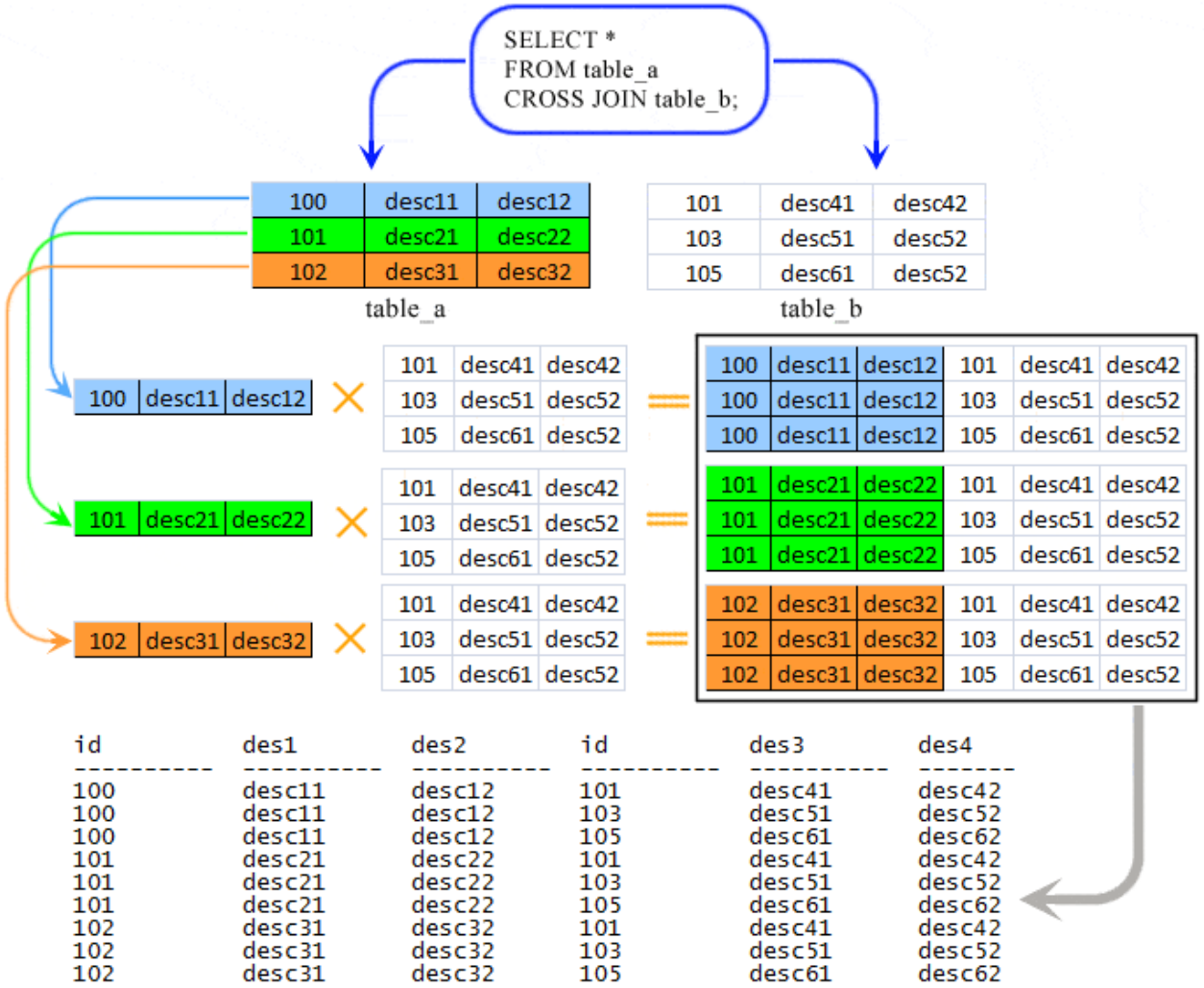
Summary

- Data independence
- Keep it simple
- Cycle between:
  - *Structured and non-structured DB*
  - *Functionality in-DB vs out-of-DB*
  - *Programming language tight vs loose integration*

ADVANCED SQL

Cross Join

- Generate a relation that contains all possible combinations of tuples from the input relations.
- **Syntax:** (R × S)
- **SELECT * FROM** R **CROSS JOIN** S;



https://www.w3resource.com/mysql/advance-query-in-mysql/mysql-cross-join.php

RELATIONAL ALGEBRA: JOIN

- Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.
- **Syntax:** (R ⋈ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R ⋈ S)

a_id	b_id
a3	103

```
SELECT * FROM R NATURAL JOIN S;
```

RELATIONAL ALGEBRA: JOIN TYPES

- **Cross Join**
  - Same thing as Cartesian product
- **Inner Join**
  - Each tuple in the first relation must have a corresponding match in the second relation.
- **Outer Join (left, right and full joins)**
  - Each tuple in one relation does not need to have a corresponding match in the other relation.

  - Let us see https://www.w3schools.com/sql/sql_quiz.asp