CSE 412 Database Management

Dr. Bharatesh Chakravarthi (BC)

Assistant Teaching Professor

School of Computing and Augmented Intelligence

Arizona State University

Instruction Team

- Dr. Bharatesh Chakravarthi
  - Assistant Teaching Professor, joined ASU in Fall 2022
  - 13+ years of academic and research experience
  - Office hour: Tuesday 4:30 – 5:30 PM at BYENG M1-40 or Zoom room https://asu.zoom.us/j/8095464858

Graduate TA

- **Kaustav Chanda (TA)**
  - Email: kchanda3@asu.edu
  - Office hour Time: Will be updated
  - Office hour Location: BYENG M1-45

Course Instruction

- **Class Schedule**
  - **Days:** Mondays (In-person)
  - **Time:** 3:00 PM – 4:15 PM
  - **Location:** Tempe – CDN60
- **Dates:** August 21, 2025 – December 5, 2025 (C)
- **Additional Session:**
  - Recorded lectures, materials, and resources will be shared every Thursday.
  - Students are expected to review these recordings promptly and stay up to date.
  - It is the student's responsibility to follow the sequence to attend the in-person session, then review the recorded lecture / shared resources, to ensure continuity and be prepared for the following week's in-person class.

Agenda

- Why do you need to learn about database?
- Course logistics
- A brief history of database systems

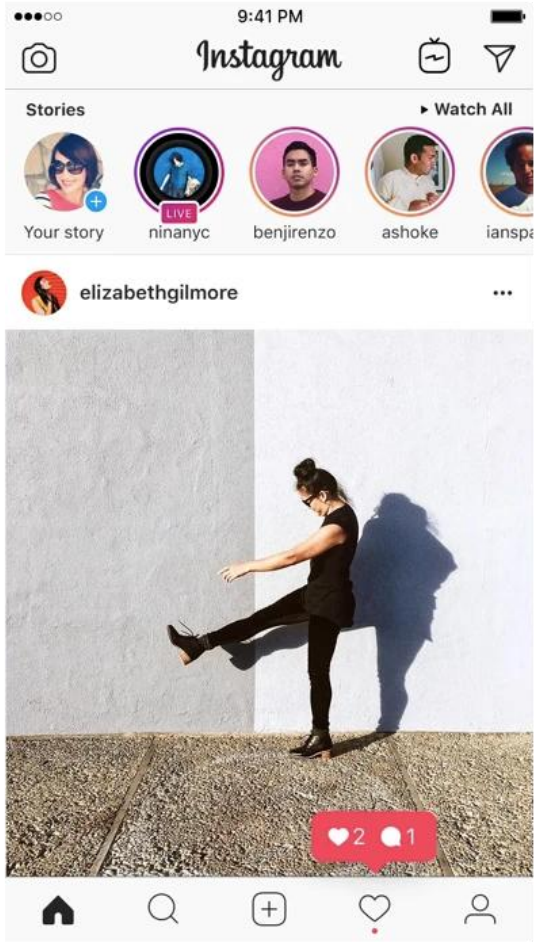
Discussion

- Does Anyone want to share your learning goal of CSE 412?
  - **Why** do you want to learn about database systems?
  - **What** do you expect to learn at the end of the semester?

Databases are everywhere

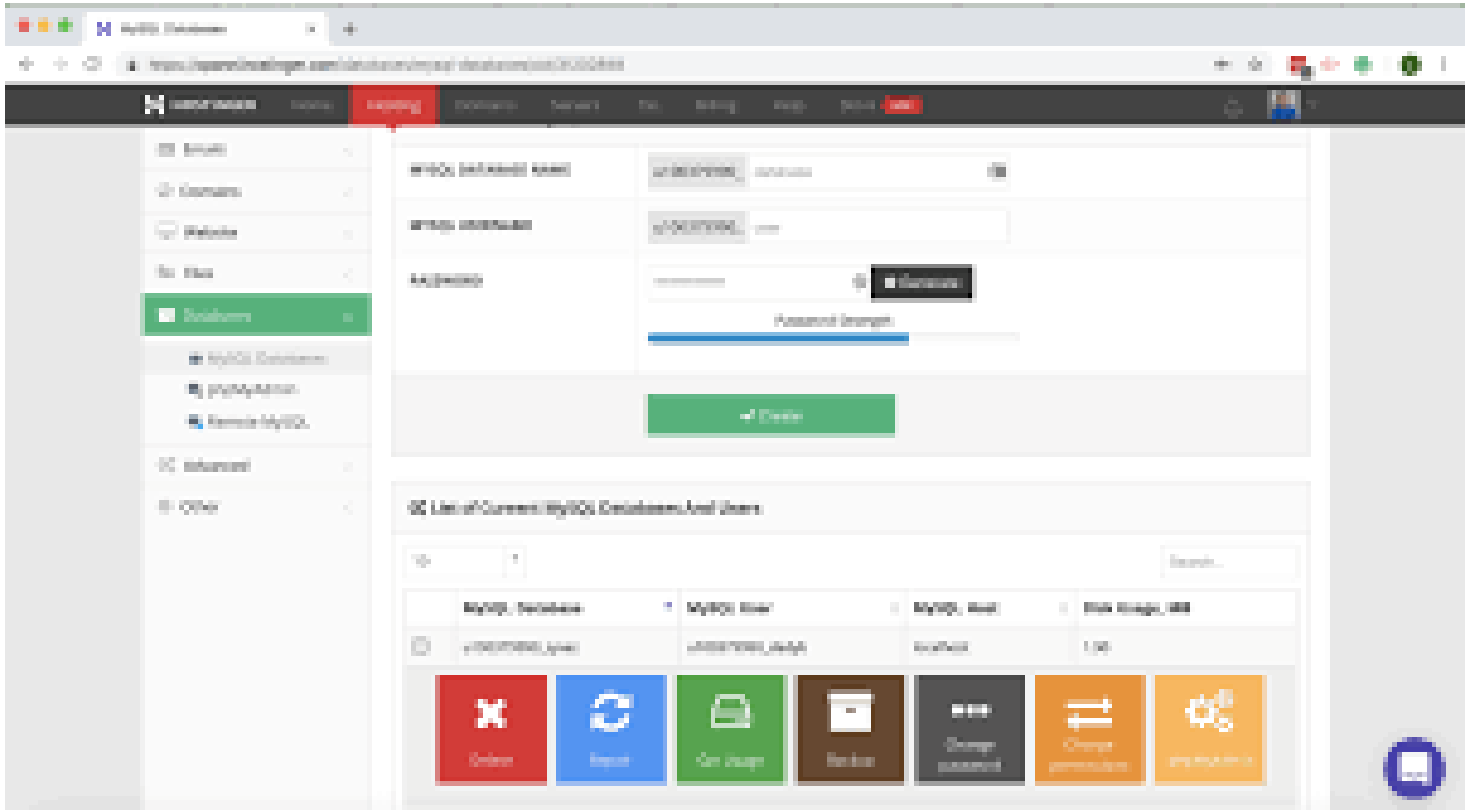
Facebook uses **MYSQL** as the primary database management system for all the structured data storage such as the posts, information of the various users, their timelines and so on.

Databases are everywhere



Instagram used **PostgreSQL** for handling user, media, friendship etc.

Databases are everywhere



WordPress uses **MySQL** for its database management system. **MySQL** is responsible for managing user data, user posts, comments, and so on.

Databases are everywhere

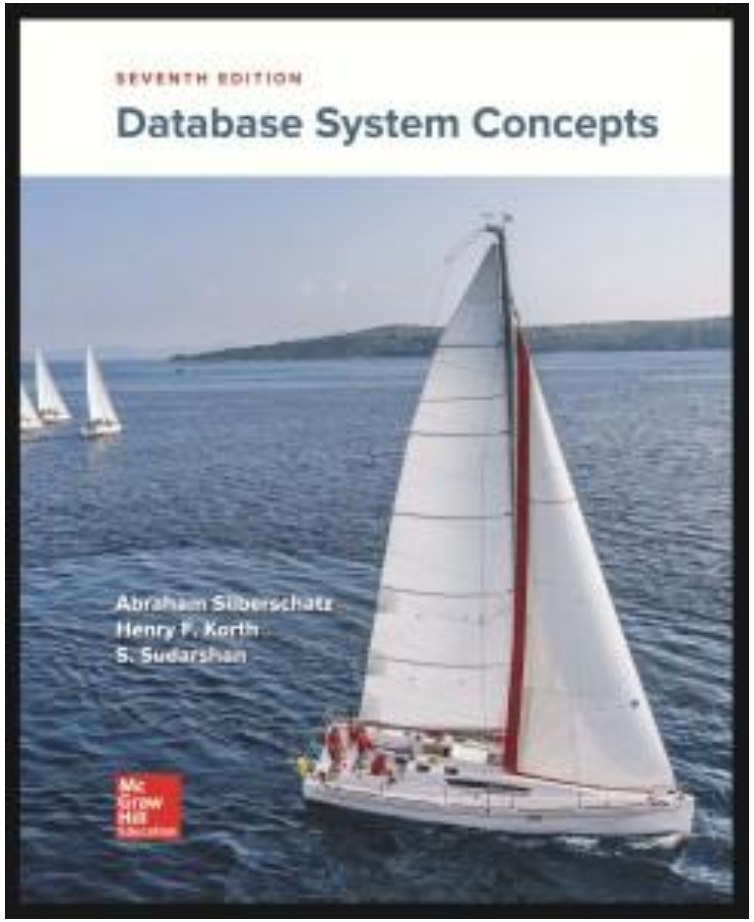
Expected Learning Outcome of this Course

- Understand **“What”** and **“Why”**
  - Relational algebra
  - SQL
  - database design
  - functional dependency
  - data storage
  - query execution/optimization/compilation
  - transaction
- Know **“How”**
  - Setup a database on a computer server
  - Design and create efficient relational data models that fit the application
  - Loading data into the database
  - Write SQL programs to issue queries to the database to read and analyze data
  - Build applications that access the database to retrieve relevant data and present it to end users

Course Logistics

Textbooks

- **[SKS]** Database System Concepts 7th edition https://www.db-book.com/



Grading Policy

Evaluation Component	Weightage
Quizzes	15%
Assignments (4)	30%
Group project (1)	25%
Midterm Exam (2)	20%
Final Exam	10%

Grade	Upper limit percentage	Lower limit percentage
A+	100 %	to 97 %
A	< 97.0 %	to 94.0 %
A-	< 94.0 %	to 90.0 %
B	< 90.0 %	to 87.0 %
B+	< 87.0 %	to 84.0 %
B-	< 84.0 %	to 80.0 %
C+	< 80.0 %	to 76.0 %
C	< 76.0 %	to 70.0 %
D	< 70.0 %	to 60.0 %
E	< 60.0 %	

Course Schedule

Course Plan CSE 412 – Database Management Fall 2025						
Week (W)	**Date**	**Lecture Topic**	**Quiz (Q)**	**Assignment (A)**	**Project (P)**	**Exam (E)**
01	25-Aug	Introduction				
02	08-Sep	ER Model				
03	15-Sep	ER Translation	**Q1**	A1 Release		
04	22-Sep	Relational Algebra		A01 Submission		
05	06-Oct	SQL 1	**Q2**	A02 Release	Phase 01 Report	
06	13-Oct	FALL BREAK				
07	20-Oct	SQL 2				**E01**
08	27-Oct	Schema Normalization		A02 Submission		
09	03-Nov	Relational Operators & Indexing	**Q3**	A03 Release		
10	10-Nov	Query Optimization		A03 Submission	Phase 02 Report	
11	17-Nov	Transaction Management	**Q4**			**E02**
12	24-Nov	Crash Recovery		A04 Release		
13	01-Dec	Concurrency Control, Review	**Q5**	A04 Submission	Phase 03 Report	
08-Dec		**FINAL EXAM**				

Grading Appeal

- Any questions, corrections, or appeals on grades of programs or tests must be done **in writing within one week** after it has returned to the class. State the problem and the rationale for any change in your grade in your appeal.

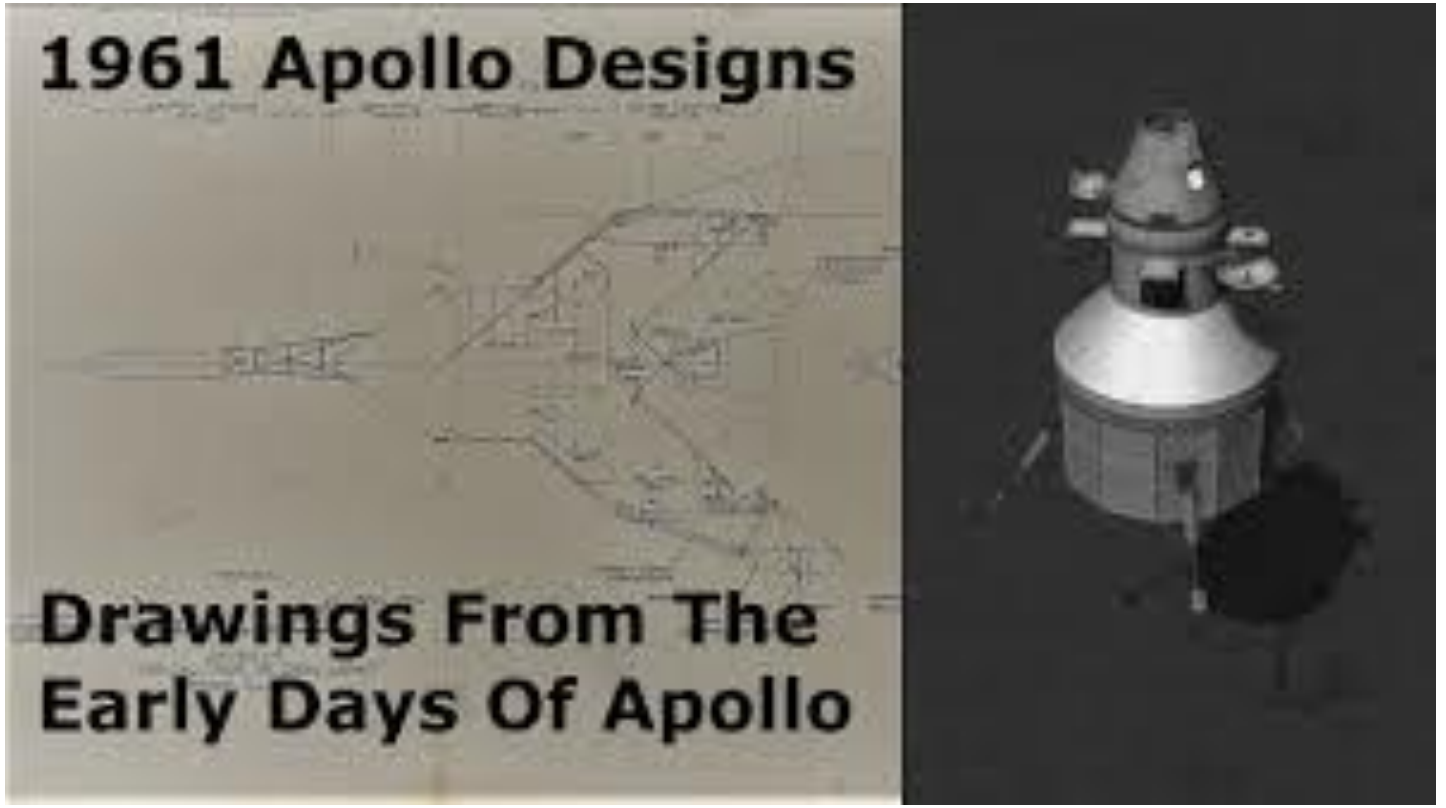
Honor Codes

- For the 4 individual assignments, you should not copy others' solutions, or use solutions that you find on the internet
- For the group project, you should not use another group's project, past year's projects for this or other courses, or projects you find on the internet
- Once we identify such plagiarism behavior, we have no problem failing you in this course

Please refer to the Syllabus for more details. The due dates may be subject to adjustments.

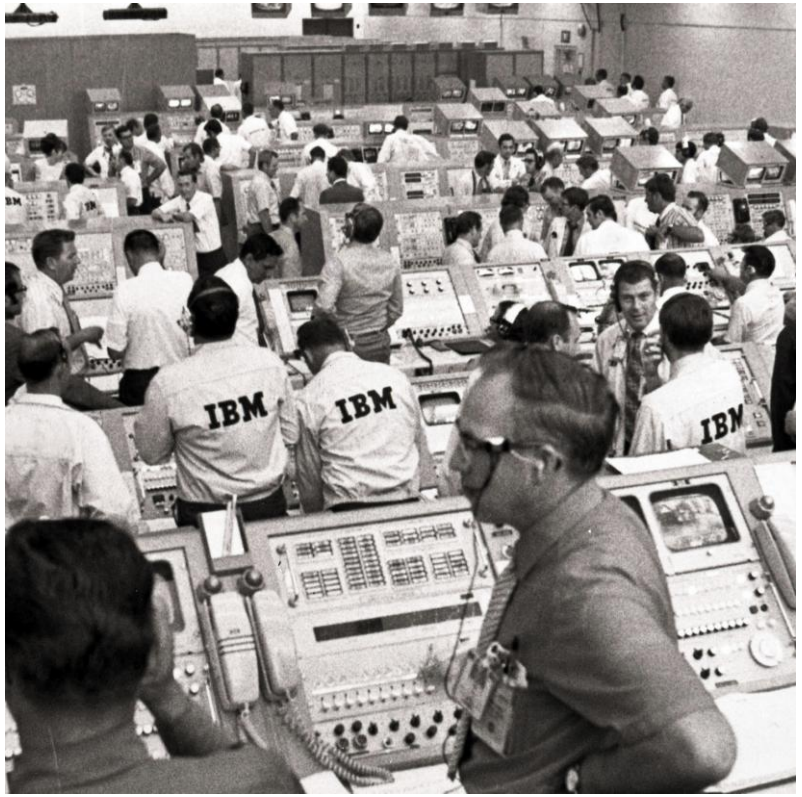
A History of Database Systems

1961: "We choose to go to the moon"



1961: "We choose to go to the moon"

- Apollo needed an automated system to manage the purchase information of a large number of rocket parts, and ***IBM*** took the job.



Why does a flat file strawman not work?

- **Why not simply write all of the information into a flat file that sits on the local disk?**
- Does anyone volunteer to share an opinion?

Why does a flat file strawman not work?

- **Why not simply write all of the information into a flat file that sits on the local disk?**
  - *No Efficient Querying*
  - *No Concurrency Control*
  - *Data Integrity & Consistency Issues*
  - *Poor Security & Access Control*
  - *Scalability Problems*
  - *Lack of Reliability*

Why a flat-file strawman doesn't work?

- **Integrity:** When multiple parties modify the file, how to ensure integrity and correctness?
- **Implementation:** each time we need to read, search, or update some information in the file, we need to implement an application from scratch.
- **Durability:** how to handle crashes and failures?
- **Security:** how to ensure security access to different information in the file for different users, that is, how to enforce access control policies for different users?

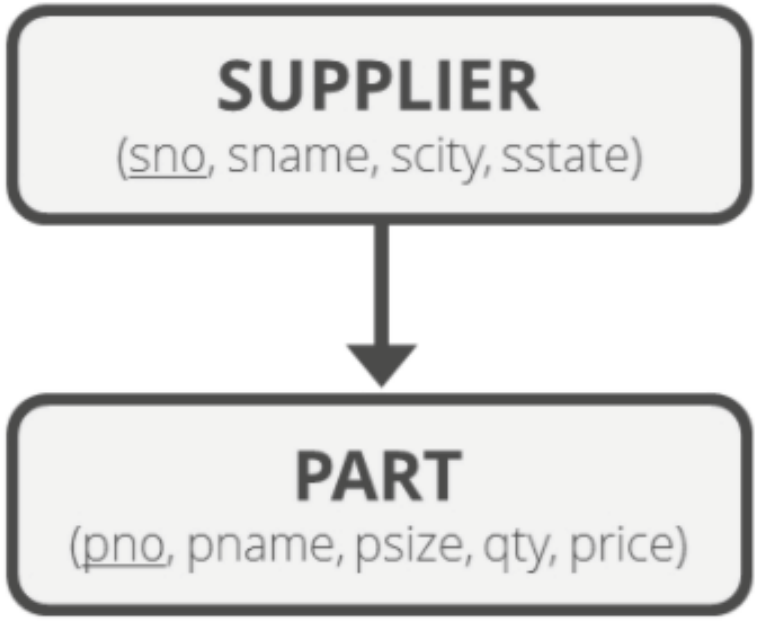
1966: IBM Information Management System (IMS)

- IBM designed IMS with Rockwell and Caterpillar
- To inventory the very large bill of materials and keep track of purchase orders for the Saturn V moon rocket and Apollo space vehicle
  - *Hierarchical data model*

Hierarchical Data Model

- It stores data as a tree, so parent nodes contain pointers to children nodes

Schema



Instances

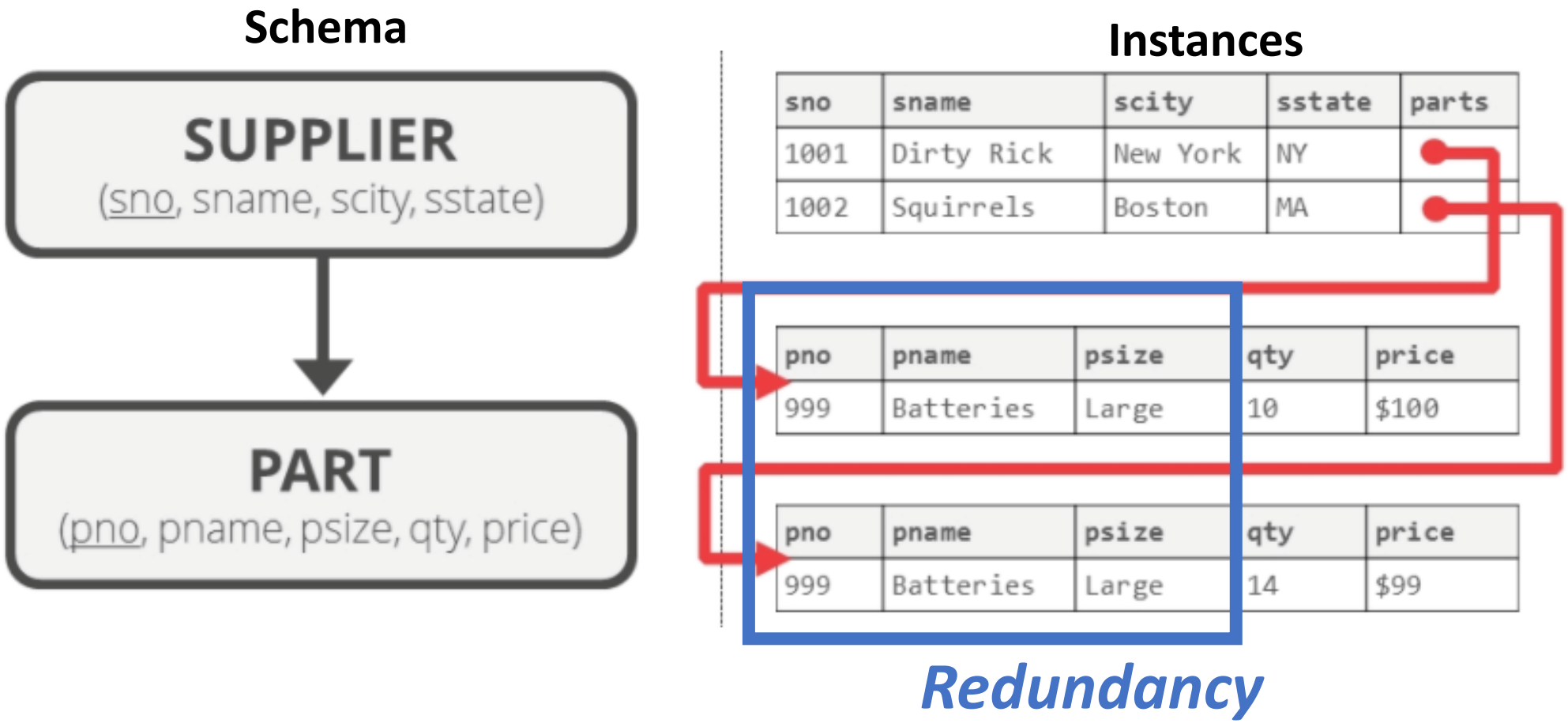
sno	sname	scity	sstate	parts
1001	Dirty Rick	New York	NY	
1002	Squirrels	Boston	MA	

pno	pname	psize	qty	price
999	Batteries	Large	10	$100

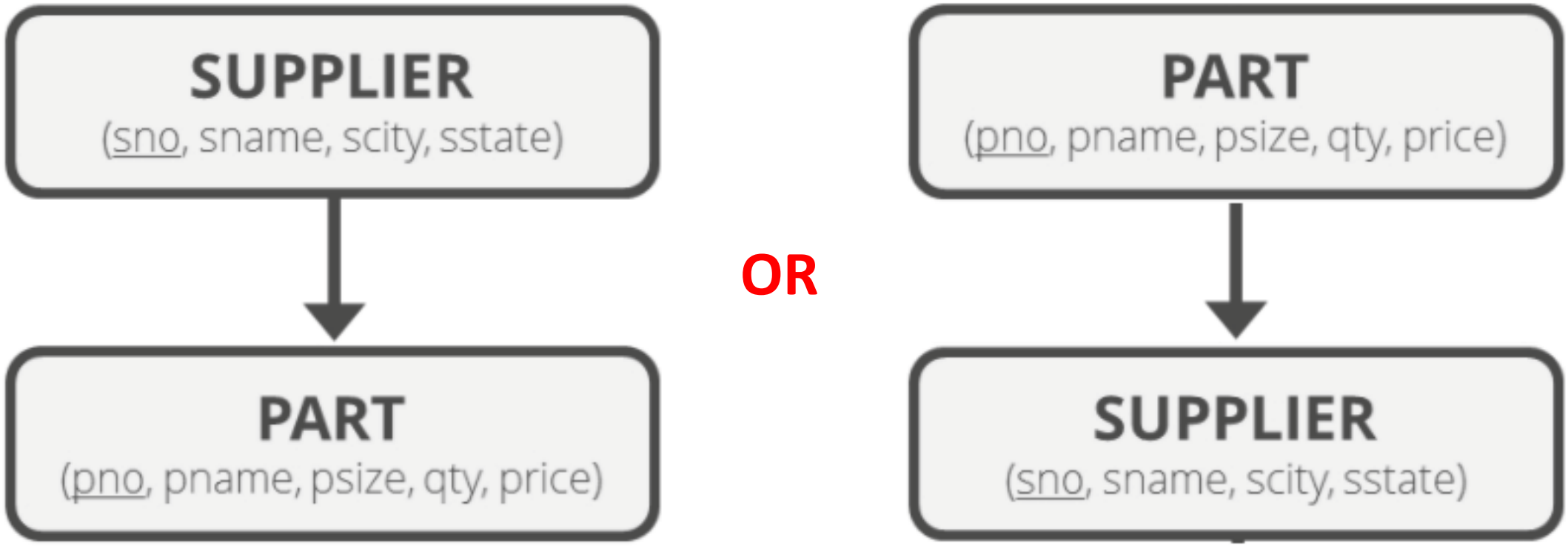
pno	pname	psize	qty	price
999	Batteries	Large	14	$99

Hierarchical Data Model

- However, this approach doesn't consider how to avoid redundant information shared by instances.



Hierarchical Data Model

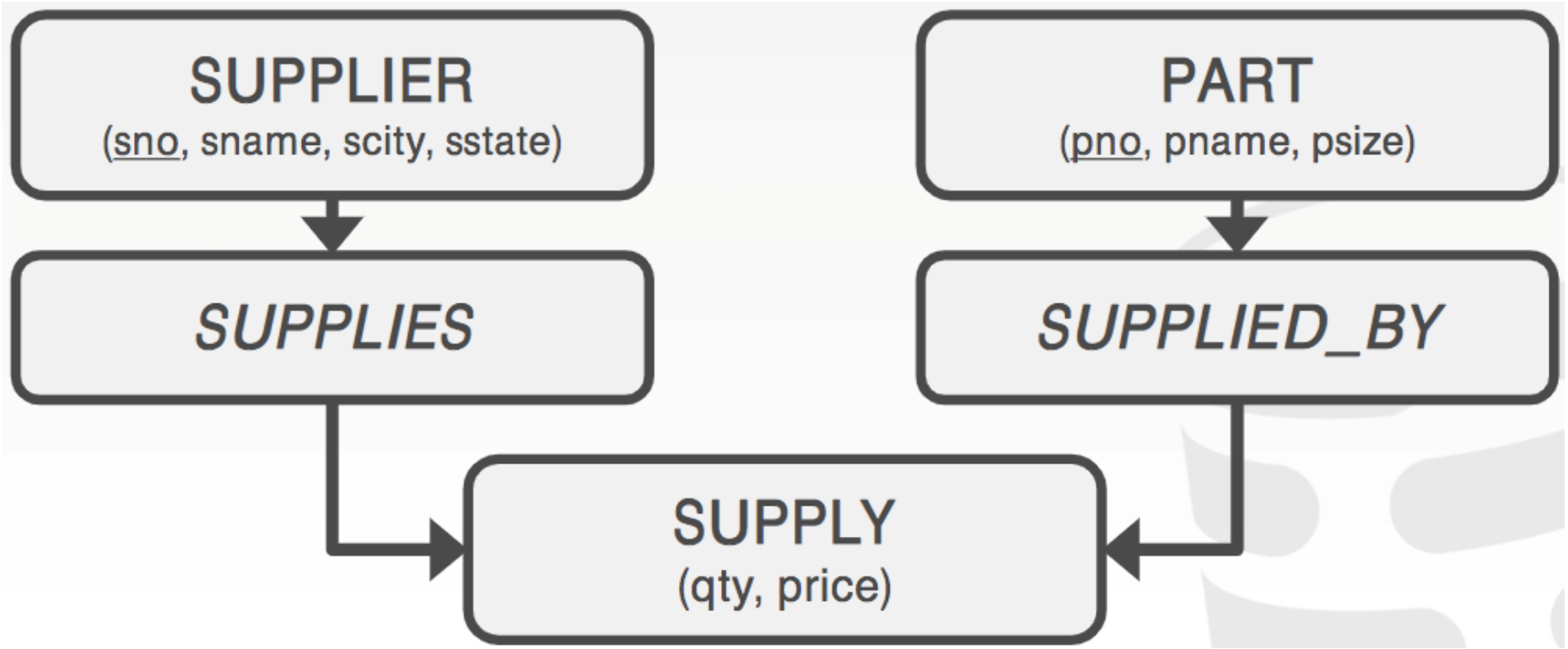


Lack Data Independence: Performance depends on choice of hierarchy

1969: CODASYL Data Model

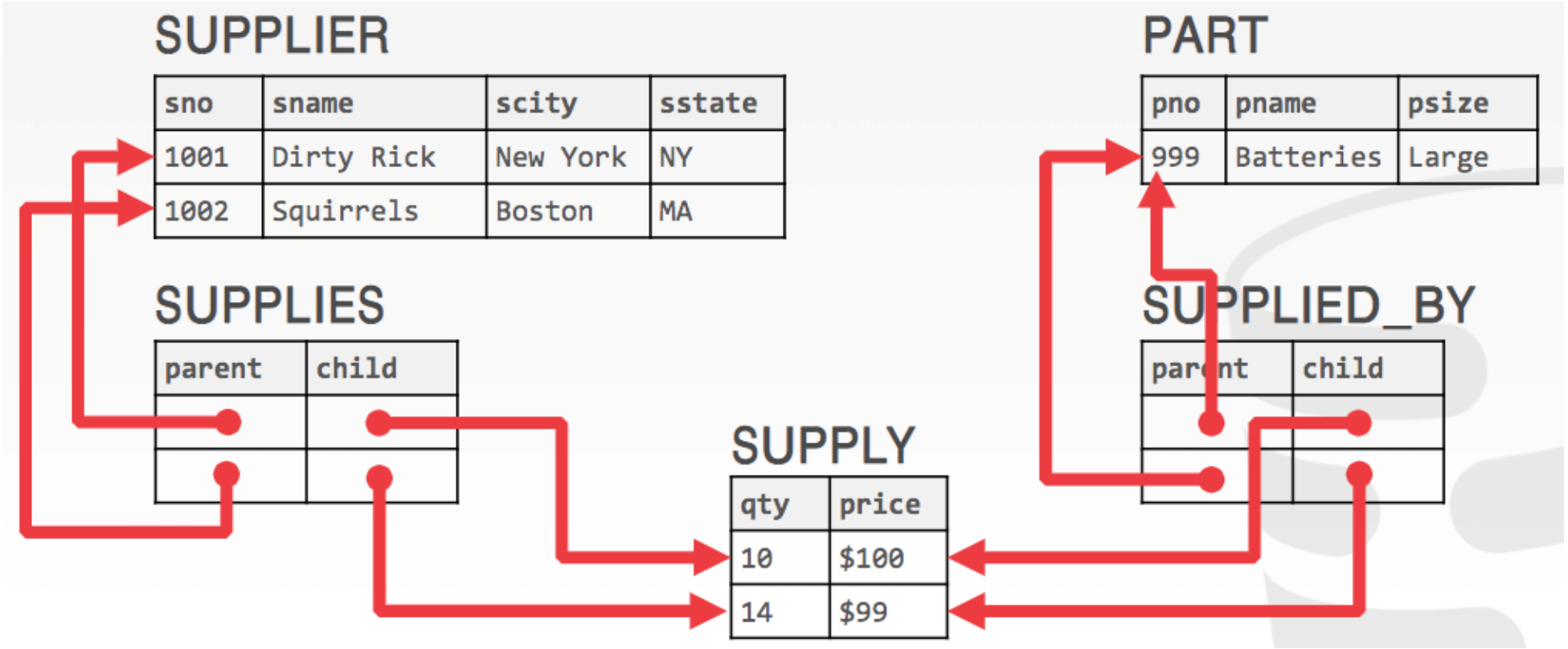
- Based on GE's Integrated Data Store (IDS) released in 1964
- Advocates for a ***network data model***
- IDS' developer ***Charles Bachman*** won the ***Turing Award*** in 1973

Network Data Model



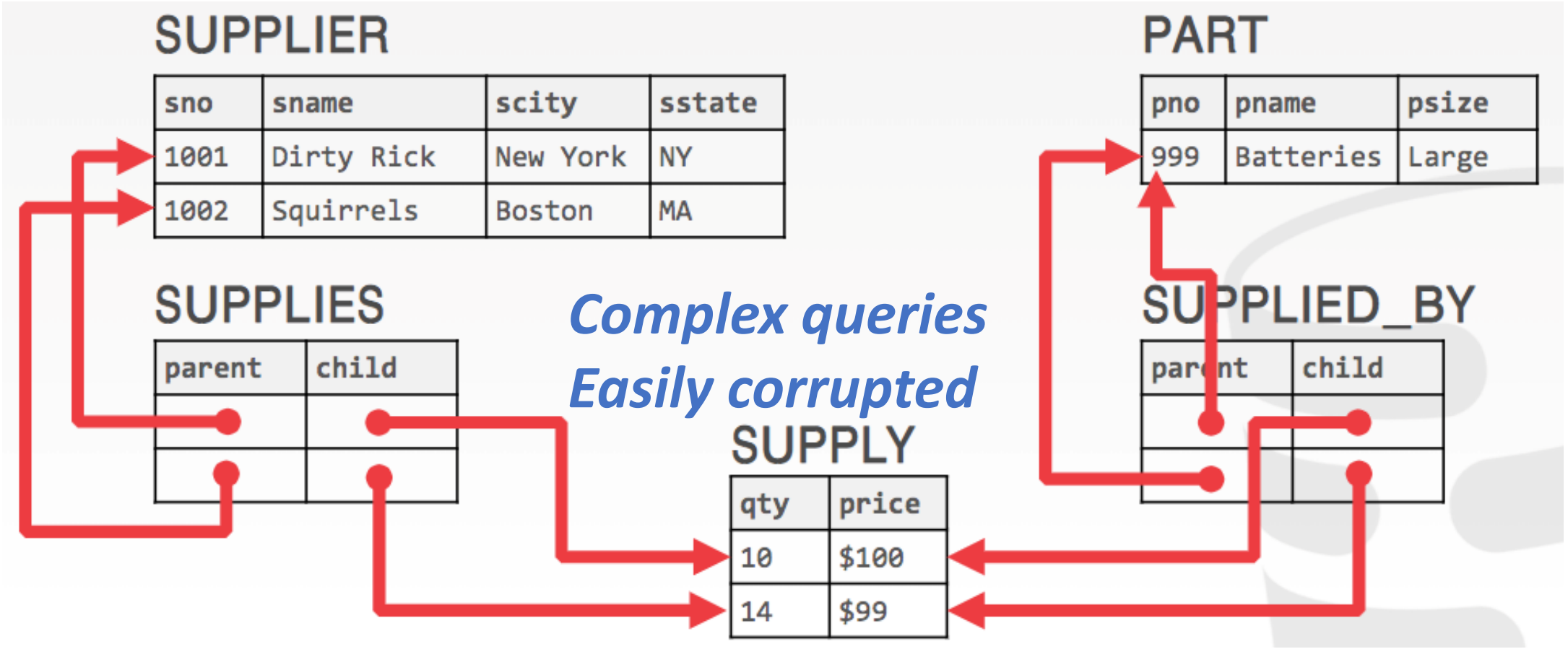
- Data is represented as ***records (nodes) and relationships (edges/links).***
- It's ***like a graph structure*** where a record can have multiple parent and child relationships (unlike the strict tree structure in the hierarchical model).

Network Data Model



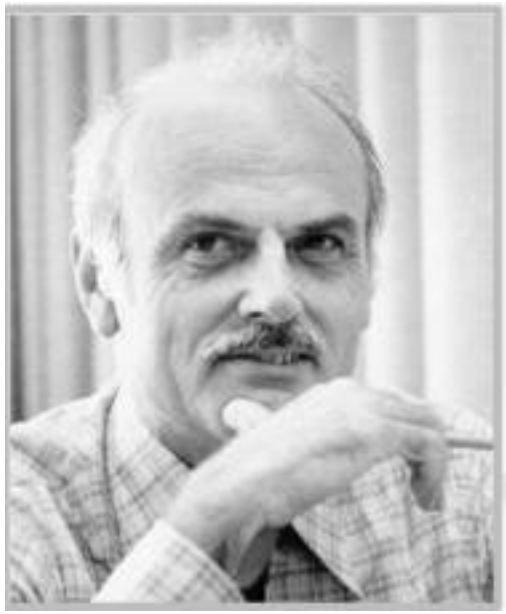
- *The structure is powerful,* ***but too complex to design and maintain****.*
- *You need to* ***manually navigate pointers/links to traverse data****.*

Network Data Model



1970s: Relational Model

- ***Edgar Frank Codd*** was a mathematician working at IBM Research. He saw developers represent a query like *"Find the employees who earn more than their managers"* in CODASYL *using complex code that was five pages long,* which would navigate through this labyrinth of pointers and stuff.
- He ***proposed the relational model*** as a new database abstraction, which is still in wide use today, and won him a Turing award in 1981



Information Retrieval

A Relational Model of Data for Large Shared Data Banks

E. F. Codd

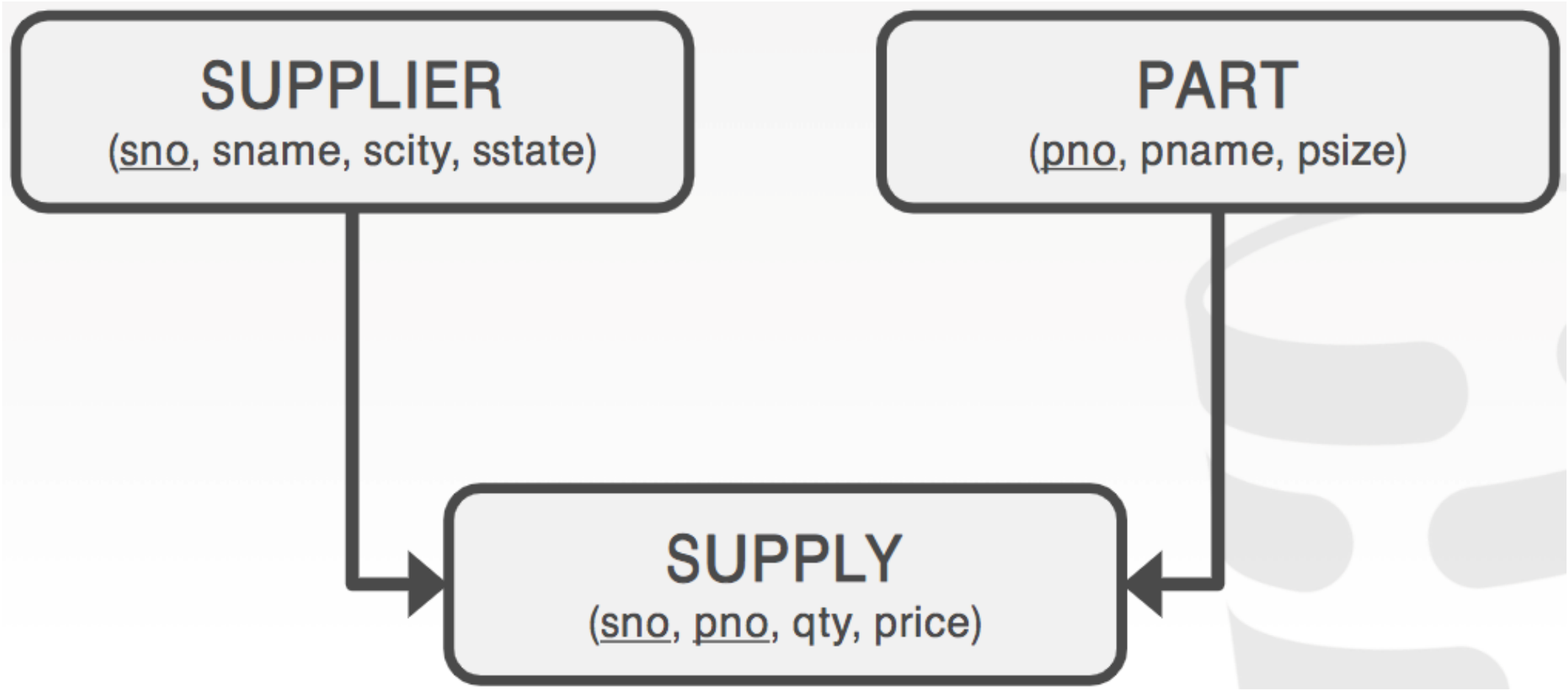
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be

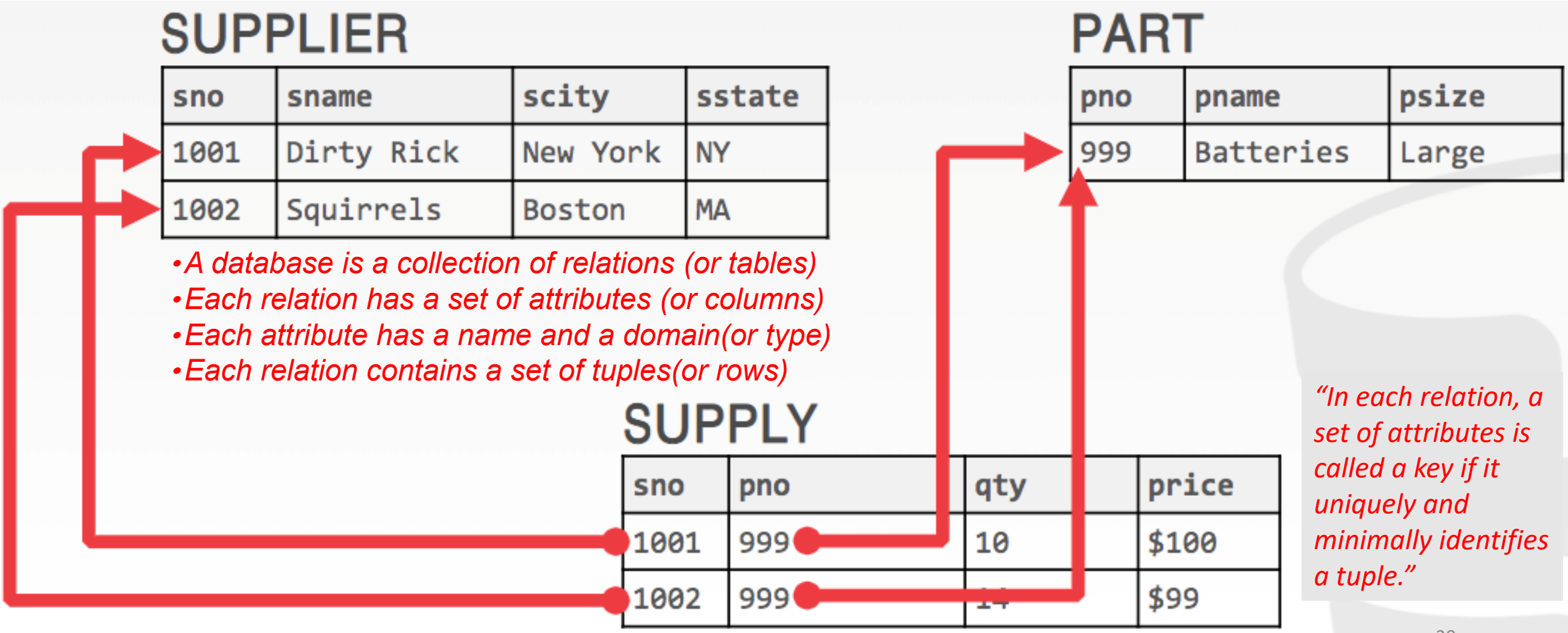
The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [2, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 3. The network model, on the other hand, has spawned a number of confusions, not the least of which is equating the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Relational Data Model: Conceptual Schema



Relational Data Model: Relations



- *A database is a collection of relations (or tables)*
- *Each relation has a set of attributes (or columns)*
- *Each attribute has a name and a domain(or type)*
- *Each relation contains a set of tuples(or rows)*

"In each relation, a set of attributes is called a key if it uniquely and minimally identifies a tuple."

Queries in the Relational Model

- **Examples:**
  - *Give me the names of all Suppliers, whose quantity for at least one part is larger than 10*
  - *Give me the names and states of all Suppliers, whose price for part with pno=999 is lower than 100*
  - *Give me the names of all parts which has more than one supplier*
  - *Insert one Supplier, whose name is XXXX, city is XXXX, state is XXXX*
  - *Remove a Supplier whose serial number is 1001, and remove all part information related to this Supplier*
  - *Update the price with serial number =1001, and serial number =999 to 200*

Queries in the Relational Model

- *Query Language*
  - Relational Calculus (a formal language based on mathematical logic)
  - Relational Algebra (based on a collection of operators for manipulating relations)
  - SQL (Structured Query Language for Relational Algebra)

- *Data Manipulation Language (DML)*
  - Insert, delete, update, query relations
  - Query Language is part of DML

- *Data Definition Language (DDL)*
  - Define relations and schemas
  - A schema is the blueprint or structure of a database.
  - It defines how data is organized and what relationships exist between different parts of the data.

Example

- Give me the names of all Suppliers, whose qty for at least one part is larger than 10

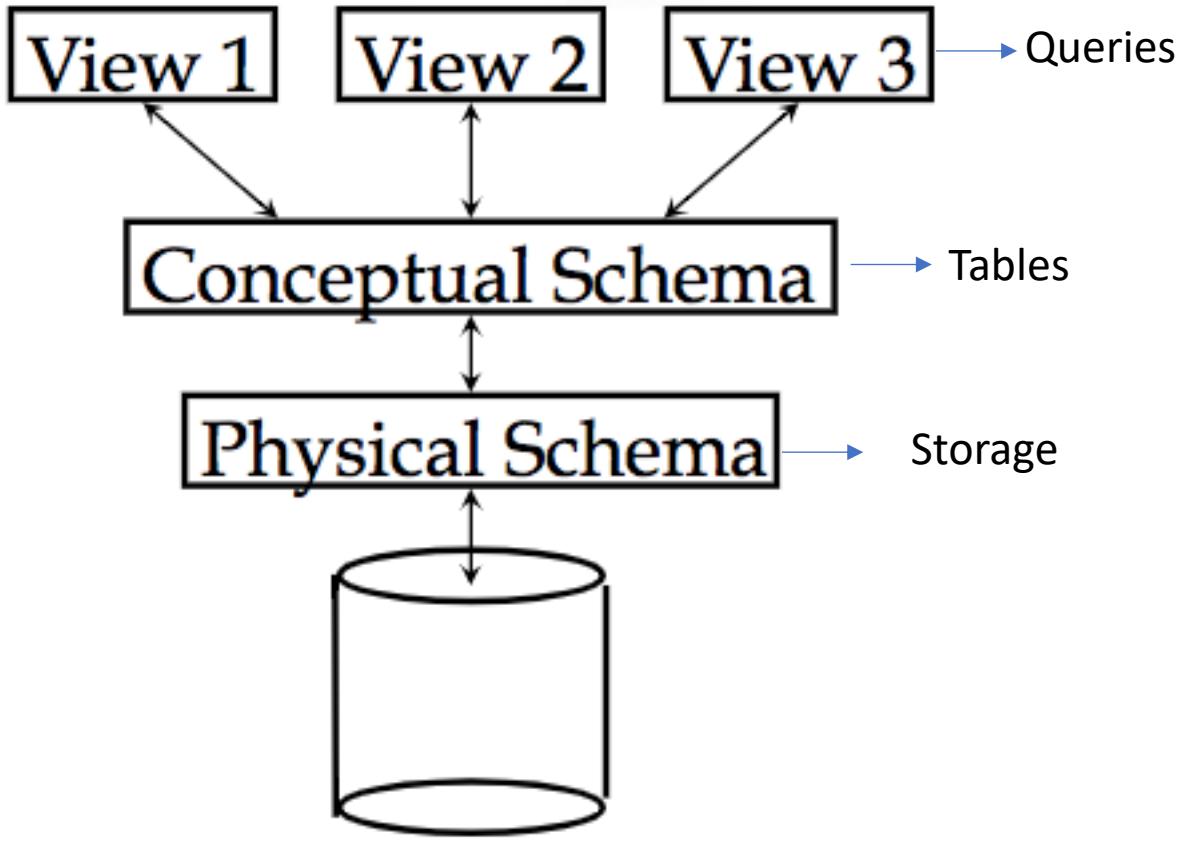
Select Suppliers.Name

from Suppliers, Supplies

Where Suppliers.sno == Supplies.sno ***AND*** Supplies.qty > 10

Relational Data Model

- ***Main concept:***
  - **Relation**: basically a table with rows and columns
  - Every relation has a **conceptual schema,** which describes the logical structure (columns, primary keys, foreign keys etc.)
  - Every relation also has a **physical schema**, which describes how the relation is stored (sorting, indexing, partitioning, column/row storage)
  - Every relation can have many **external schemas** (views)



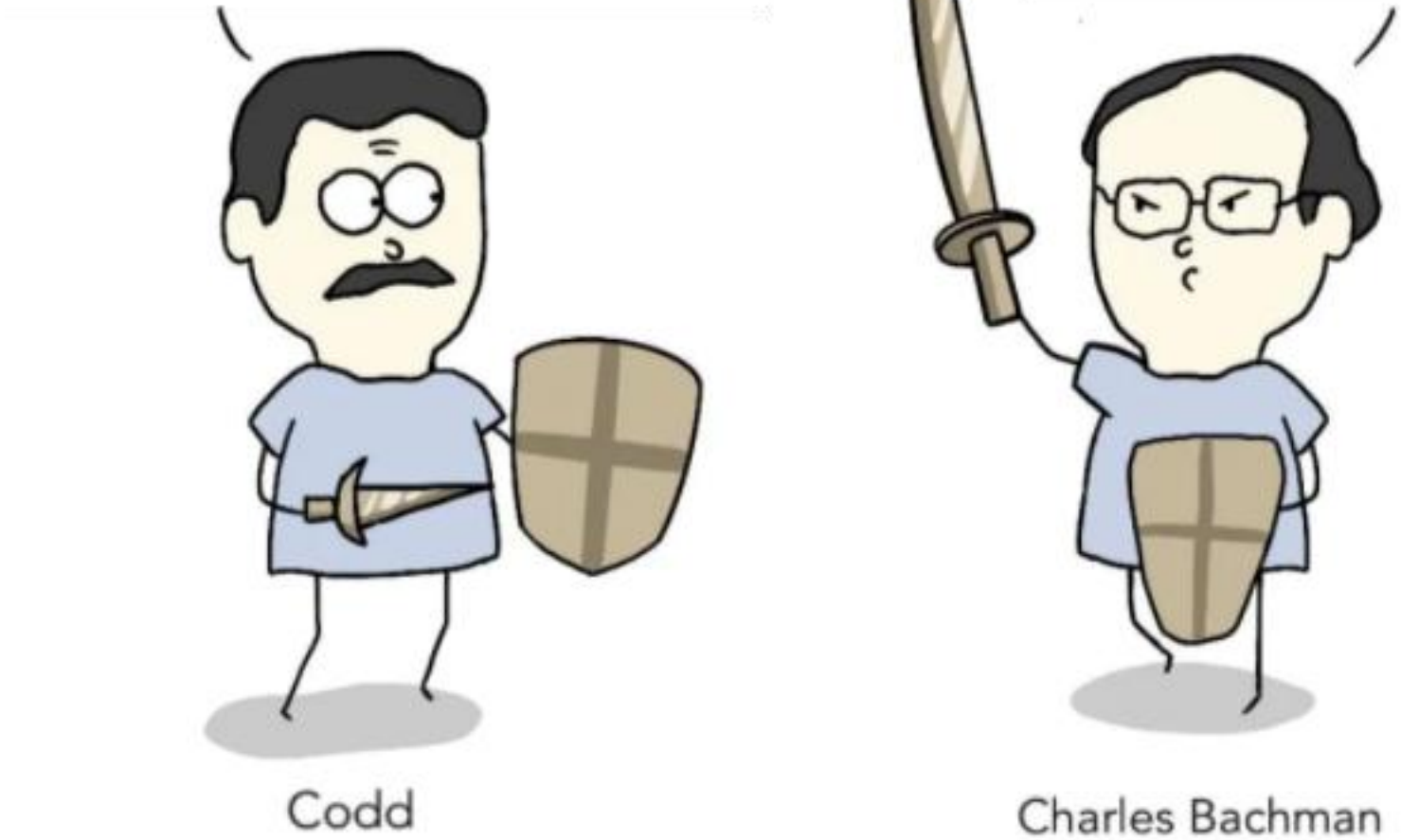
Benefits of the Relational Model: Data Independence

- A query only specifies what to compute, and not how to compute.
  - The database management system (DBMS) will be responsible for parsing the query, evaluating the query, optimizing the query execution plan, and executing the query.
- Logical schema is decoupled from physical schema
  - Changing the physical schema does not require any changes in the logical schema, and the applications (queries)

Then there were years' debate about which data model is better, CODASYL or Relational…

CODASYL is too complex!
Set-oriented queries in CODASYL is too difficult to develop!
CODASYL lacks a theoretical foundation!

Relational Model is hard to implement!
Most applications can work with a-tuple-at-a-time queries.
Set-oriented queries are not popular.



Early implementations of relational DBMS

- Systems R – IBM Research (Jim Gray, 1998 Turing Award Winner)
- INGRES – U.C. Berkeley (Michael Stonebraker, 2014 Turing Award Winner)
- Oracle – Larry Ellison

Jim Gray

Michael Stonebraker

Larry Ellison

1980s: Relational Models wins!

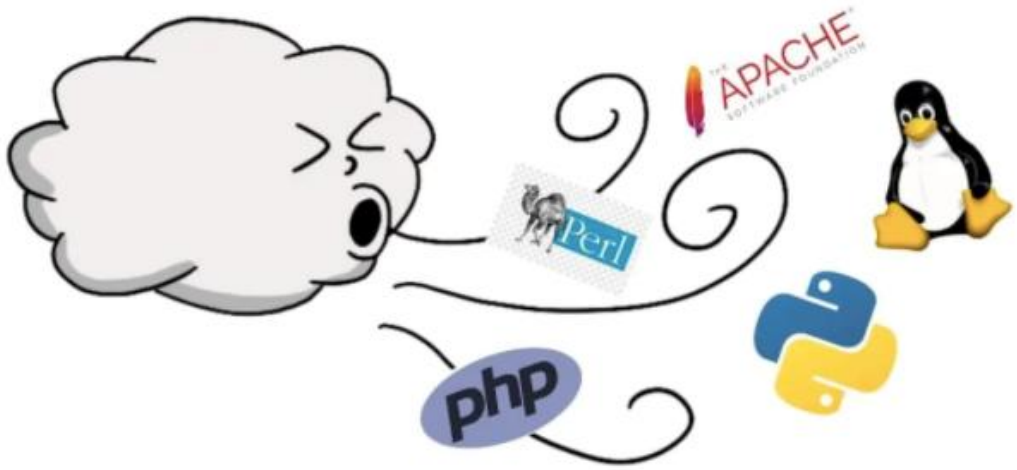
- "SEQUEL" becomes the standard (SQL)
- Many new "enterprise" DBMSs, but Oracle wins the market place

1990s: Boring Days

- No major advancements in database systems or application workloads
  - Microsoft creates SQL server based on Sybase
  - An open source database MySQL is written
  - Postgres gets SQL support
  - SQLite started in early 2000

2000s: Internet Boom

- All the big players were heavyweight and expensive
- Open-source databases were missing important features
- Companies wrote their own middleware to scale out database across single-node DBMS instances



2000s: NoSQL Systems

- Focus on high-availability & high-scalability:
  - Schema less
  - Non-relational data model: document, key/value, etc.
  - No ACID transactions
  - Custom APIs instead of SQL
  - Usually open-source

2000s: Data Warehouses

- Rise of special purpose OLAP DBMSs
  - Distributed/Shared-Nothing
  - Relational/SQL
  - Closed-source
  - Column-storage

2010s: Big Data/IoT Era

- Numerous specialized database systems
  - NewSQL
  - Hybrid Transactional-Analytical Processing
  - Cloud database
  - Shared-disk database (on distributed storage)
  - Graph database
  - Timeseries database
  - Embedded DBMSs
  - Multi-model DBMSs
  - Blockchain DBMSs
  - Hardware Acceleration

Today: AI + Database

- Database with a Natural Language interface?
- Database for retrieving contexts for Large Language Model (LLM) tasks?

https://dbdb.io

Database of Databases

Discover and learn about **710** database management systems

Search

Browse | Leaderboards

Most Recent

- GhostDB
- Grakn
- HAWQ
- Voldemort
- OrientDB

Most Viewed

- NeDB
- LevelDB
- BoltDB
- OceanBase
- LMDB

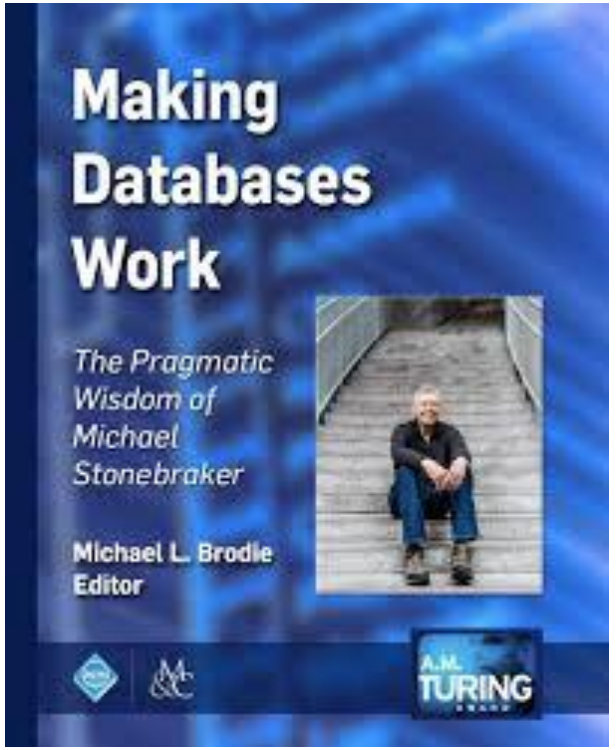
Most Edited

- Grakn
- HAWQ
- OrientDB
- GhostDB
- Voldemort


A lot of systems are still extending *SQL* interfaces and *relational models*
Even the NoSQL systems use a lot of RDBMS ideas

Reading

- Required
  - SKS 1.1~1.10
- Optional
  - Stonebraker, Michael, and Joey Hellerstein. "What goes around comes around." *Readings in database systems* 4 (2005): 1724-1735.
  - Making Databases Work



Examples

- https://github.com/topics/dbms-project
- https://github.com/yogeshk4/sales-and-inventory-management-system
- https://github.com/mayankpadhi/Hospital-Management
- https://github.com/chawat/BankingApp
- https://github.com/chiragchevli/Animal_Species_Repository_System
- https://github.com/dancancro/great-big-example-application
- https://github.com/dbs/postgresql-full-text-search-engine
- https://github.com/dushan14/books-store
- https://github.com/jai-singhal/croma
- https://github.com/carlalmadureira/Django-PostgreSQL-Application

More examples will be posted on Canvas

TO DO List

1. Reading
2. Take a look at example projects
3. Finish grouping in one week
4. Finish Assignment 0 in 1-2 weeks
5. Please think about what project you want to do, write it down (no need to submit it to me), and send it to your team members once the group formation are announced.

Next Class

- We will learn basics of relational algebra, which is the foundation of SQL, and ER diagram that you need for your group project phase 1