NATIONAL RESEARCH UNIVERSITY

Базы данных

Майнор “Интеллектуальный анализ данных”

Александр Брейман, к.т.н., доцент

Департамент программной инженерии

Факультет компьютерных наук

Организация курса

Материалы (будут ☺): https://github.com/adbadb/db-minor-22

Накопленная оценка (70% от финальной)

• Работа на семинарах	20%
• Групповой проект	40%
• Эссе	20%
• Квизы	10%
• Контрольная работа	10%

Экзамен (30% от финальной)

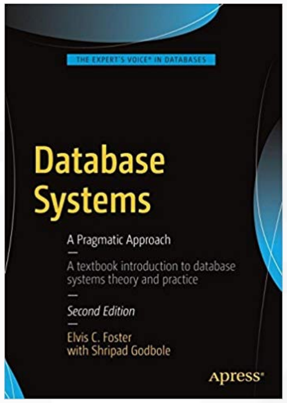
Textbooks

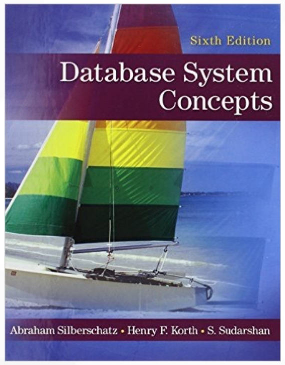
Foster, E. C., Godbole S.
Database Systems: A Pragmatic Approach,
2nd ed., 2016

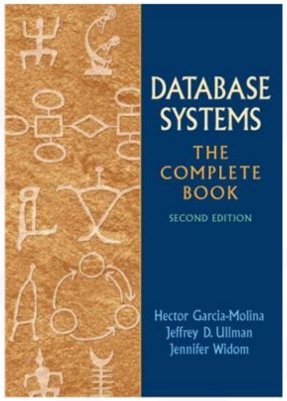
Silberschatz A., Korth H.F., Sudarshan S.
Database System Concepts
6th ed, McGraw-Hill, 2010.

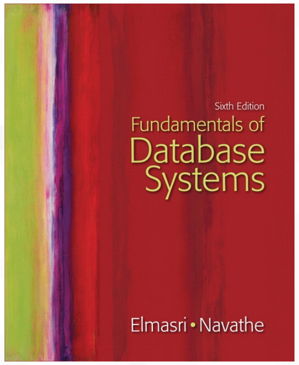
Garcia-Molina H., Ullman J., Widom J.
Database Systems: The Complete Book
2nd ed, Prentice Hall, 2009.

Elmasri R., Navathe S.B.
Fundamentals of Database Systems
6th ed., Addison Wesley, 2010.


THE EXPERT'S VOICE® IN DATABASES
Database Systems
A Pragmatic Approach
A textbook introduction to database systems theory and practice
Second Edition
Elvis C. Foster with Shripad Godbole
Apress®







Textbooks

Б.А.Новиков, Е.А.Горшкова
Основы технологий баз данных
https://postgrespro.ru/education/books/dbtech

Е.П.Моргунов
PostgreSQL. Основы языка SQL
https://postgrespro.ru/education/books/sqlprimer

Course structure

User perspective

- how to use a database system?
- conceptual data modeling, the relational and other data models, database schema design, relational algebra, SQL query language, object-relational mappings, application design and implementation

System perspective

- how to design and implement a database system?
- data representation, indexing

Project: Database-driven information system

1. Form a team of 1 to 5 students
2. Identify an application domain that requires a DB (for desktop/web/mobile access)
3. Define requirements for application
4. Design the relational DB (E/R, UML, SQL DDL)
5. * Design the application
6. * Implement database and application
7. Prepare a report and presentation

- use **relational** database management system: Oracle Database, MS SQL Server, IBM DB2, MySQL, PostgreSQL, etc.
- use **any programming language you** prefer: Java, Python, Ruby, C, C++, C#, Erlang, Go, PHP, etc.

Overview

1st Module

- Introduction
- Data modeling
- Database design: E/R and UML
- Relational model
- Relational database design
- Relational query languages
- SQL: Querying
- SQL: Updating
- SQL: Data Definition

2nd Module

- Application design and development
- Storage and file structure
- Indexing and Hashing

- Projects presentations

History of Data Management

Stone Age (- 1970)

-1900: Manual processing
1900-1955: Mechanical punched cards processing
1955-1970: Stored programs - sequential records processing

Age of Transactions (1970 -)

Goal: reliability - make sure no data is lost
1960s: IMS (hierarchical data model)
1980s: Oracle (relational data model)

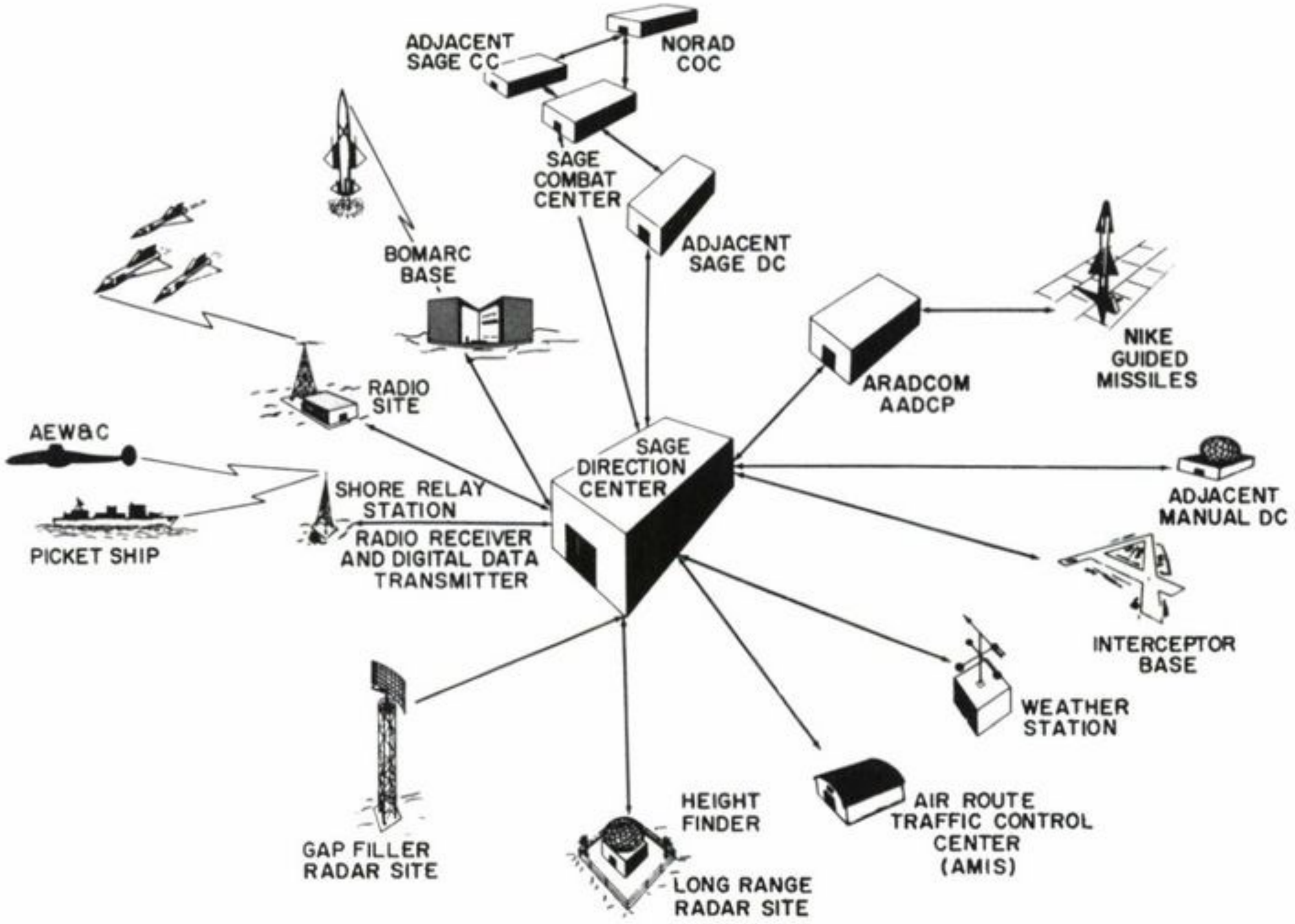
Age of Business Intelligence (1995 -)

Goal: analyze the data -> make business decisions
Aggregate data for boss. Tolerate imprecision!
SAP BW / Business Objects, Cognos, …, Essbase

Age of „Big Data " and „Data for the Masses" (2000-)

Goal: everybody has access to everything
Google (text), Cloud (XML, JSON: Services)

The very first database: SAGE (1957-1983)



History of Data Management Repeats Itself

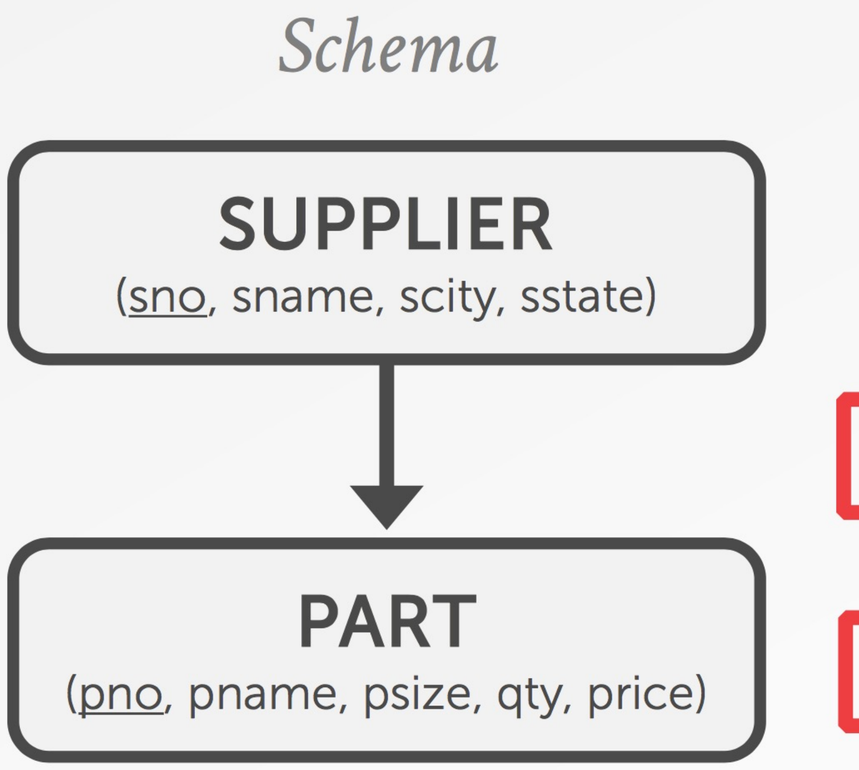
- Old database issues are still relevant today
- The “SQL vs NoSQL” debate is reminiscent of “Relational vs CODASYL” debate
- Many of the ideas in modern database systems are not new

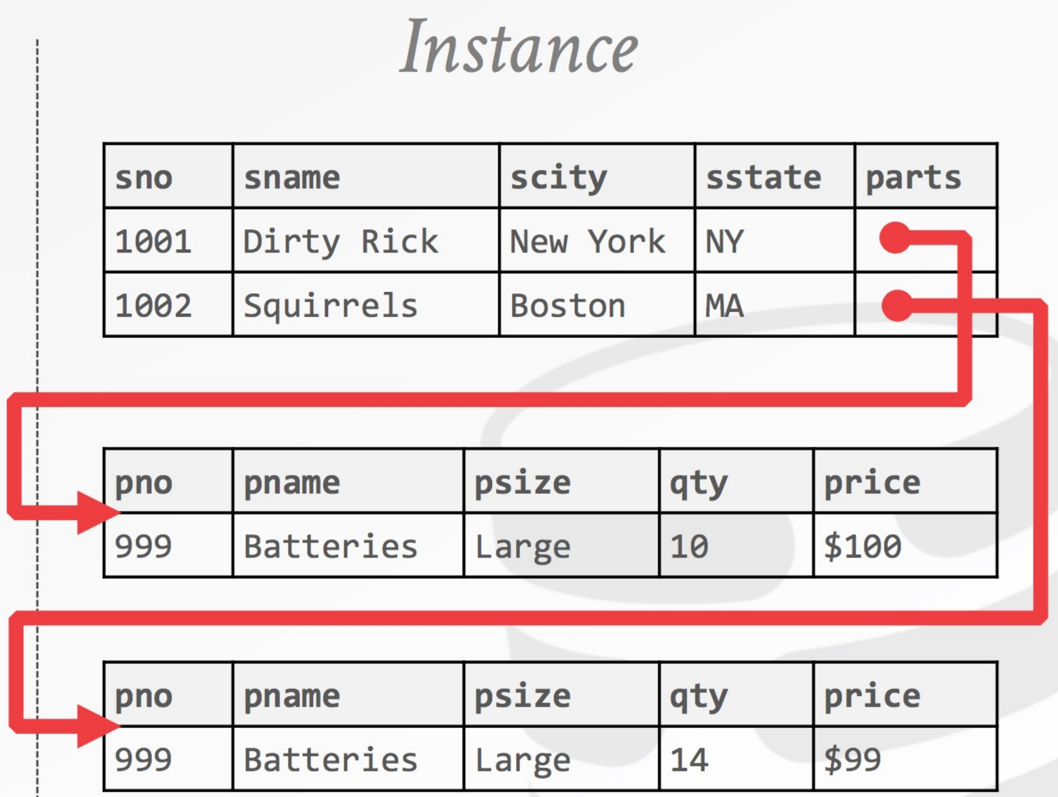
History of Data Management: IBM IMS (1960s)

IBM IMS – first database system developed to keep track of purchase orders for Apollo moon mission.

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

Hierarchical Data Model





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

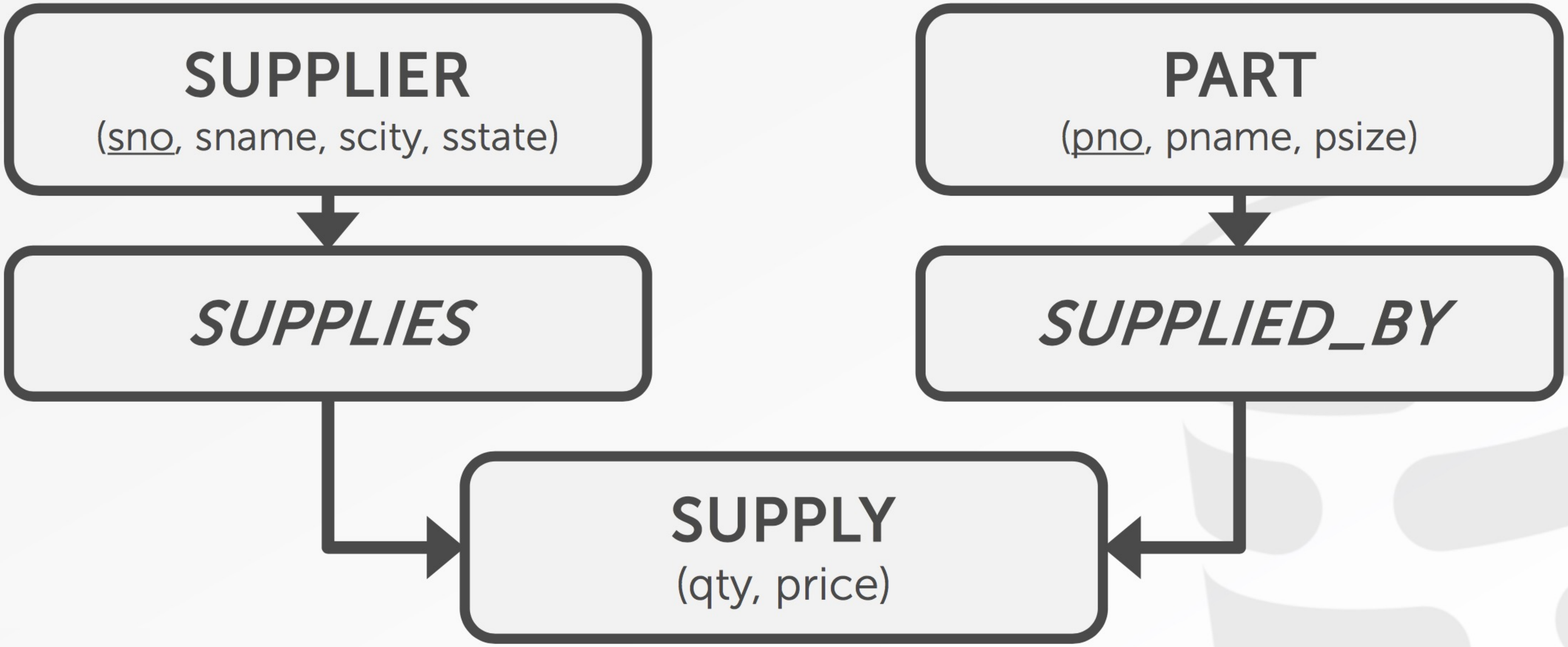
History of Data Management: CODASYL (1960s-1970s)

COBOL people got together and proposed a standard for how programs will access a database. Lead by Charles Bachman.

- Network data model.
- Tuple-at-a-time queries.

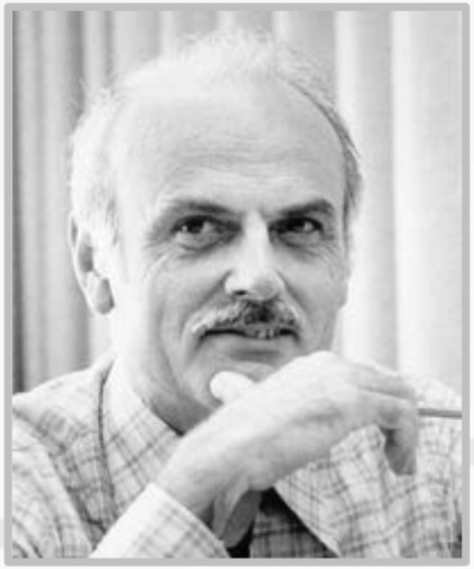
Network Data Model

Schema



History of Data Management: Relational (1960s-1970s)

Ted Codd was a mathematician working at IBM Research. He saw developers spending their time rewriting IMS and CODASYL programs every time the database’s schema or layout changed.

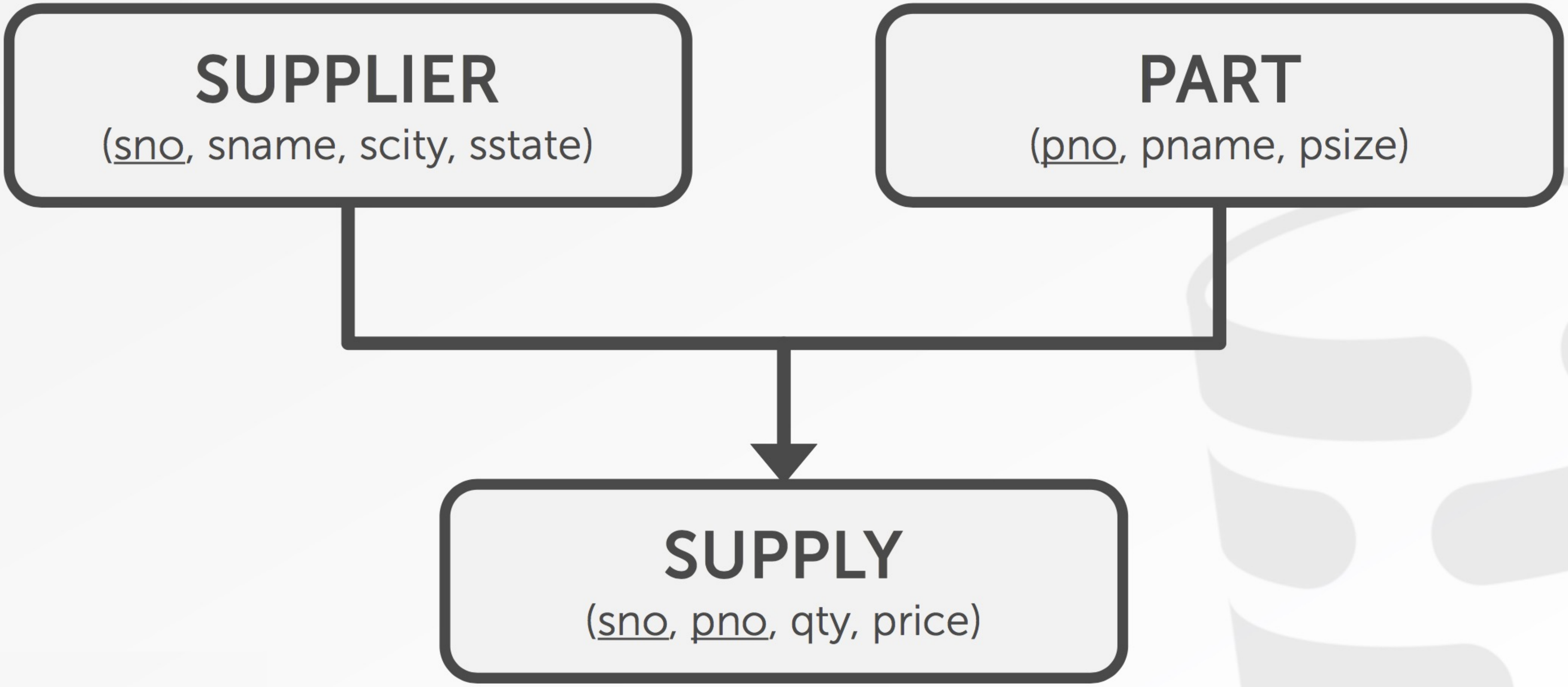


Database abstraction to avoid this maintenance:

- Store database in simple data structures.
- Access data through high-level language.
- Physical storage left up to implementation.

Relational Data Model

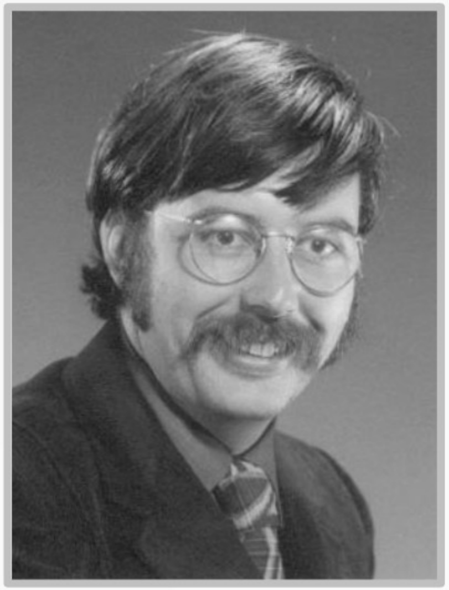
Schema



Relational Data Model

Early implementations of relational DBMS:

- **System R** – IBM Research
- **INGRES** – U.C. Berkeley
- **Oracle** – Larry Ellison



History of Data Management: Relational (1970s-1980s)

The relational model wins.

- IBM comes out with DB2 in 1983.
- “SEQUEL” becomes the standard (SQL).

Many new “enterprise” DBMSs but Oracle wins marketplace.

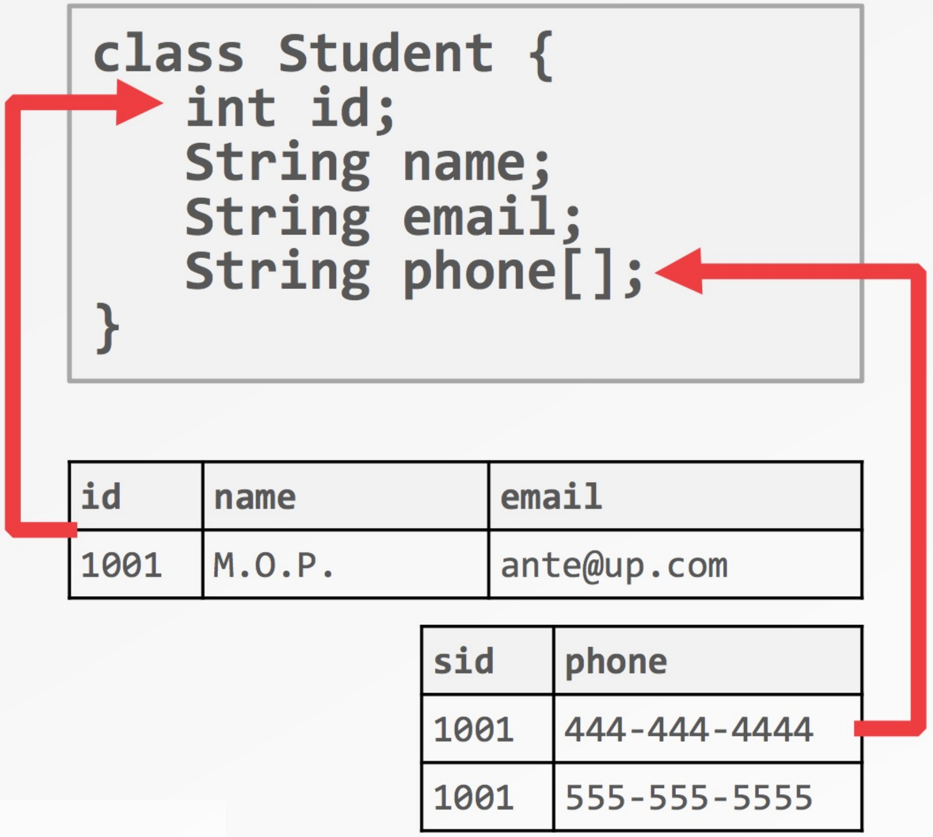
Stonebraker creates Postgres.

History of Data Management: Object-Oriented (1980s)

- Avoid “relational-object impedance mismatch” by tightly coupling objects and database.
- Few of these original DBMSs from the 1980s still exist today but many of the technologies exist in other forms (JSON, XML)

History of Data Management: Object-Oriented (1980s)

Application Code

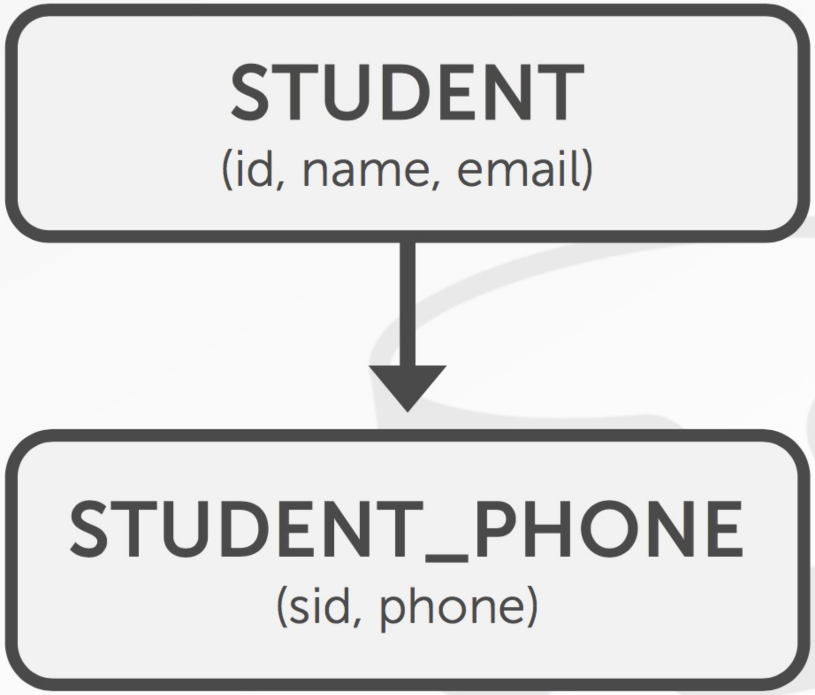


```
class Student {
    int id;
    String name;
    String email;
    String phone[];
}
```

id	name	email
1001	M.O.P.	ante@up.com

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

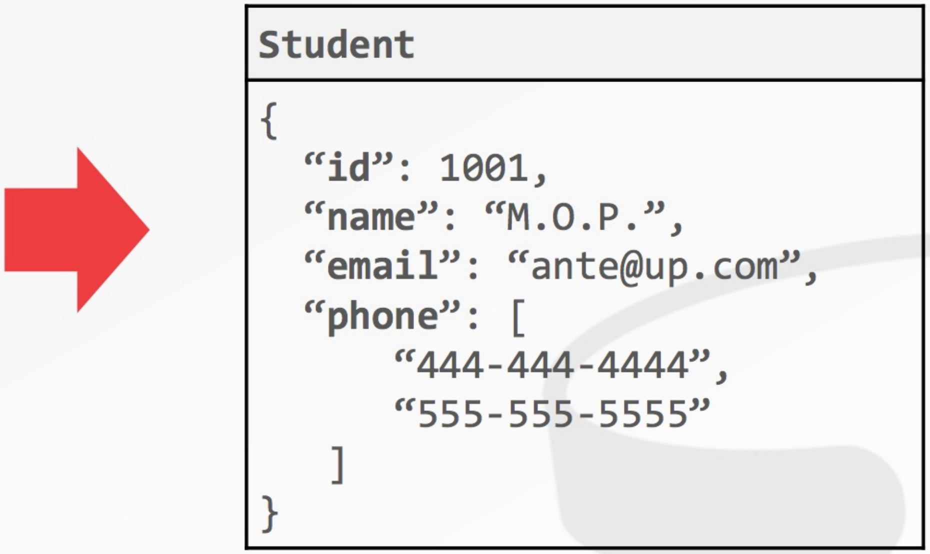
Relational Schema



History of Data Management: Object-Oriented (1980s)

Application Code

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



```
Student

{
  "id": 1001,
  "name": "M.O.P.",
  "email": "ante@up.com",
  "phone": [
      "444-444-4444",
      "555-555-5555"
    ]
}
```

History of Data Management: Boring Years (1990s)

No major advancements in database systems or application workloads.

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

History of Data Management: Internet Boom(2000s)

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

History of Data Management: Data Warehouses (2000s)

Rise of the special purpose OLAP DBMSs.

- Distributed / Shared-Nothing
- Relational / SQL
- Usually closed-source.

Significant performance benefits from using Decomposition Storage Model (i.e., columnar)

Vertica, Netezza, Greenplum, ParAccel, DATAllegro

History of Data Management: NoSQL Systems (2000s)

Focus on high-availability & high-scalability:

- Schemaless (i.e., “Schema Last”)
- Non-relational data models (document, key/value, etc)
- No ACID transactions
- Custom APIs instead of SQL
- Usually open-source

MongoDB, Cassandra, Redis, Riak, Aerospike, Neo4J, RethinkDB, DynamoDB, HBase, CouchDB, CouchBase

History of Data Management: NewSQL Systems (2000s)

Provide same performance for OLTP workloads as NoSQL DBMSs without giving up data consistency

- Relational / SQL
- Distributed
- Usually closed-source

SAP HANA, VoltDB, NuoDB, MemSQL, H-Store, dbShards, Clustrix, ScaleArc, HyPer, JustOne DB

History of Data Management: Hybrid Systems (2010s)

Hybrid **T**ransactional-**A**nalytical **P**rocessing.

Execute fast OLTP like a NewSQL system while also executing complex OLAP queries like a data warehouse system.

- Distributed / Shared-Nothing
- Relational / SQL
- Mixed open/closed-source.

SAP HANA, MemSQL, HyPer, JustOne DB, Snappy, Splice Machin

History of Data Management

There are many innovations that come from both industry and academia:

- Lots of ideas start in academia but few build complete DBMSs to verify them.
- IBM was the vanguard during 1970-1980s but now Google is current trendsetter.
- Oracle borrows ideas from anybody.

The relational model has won for operational databases.

Database Example

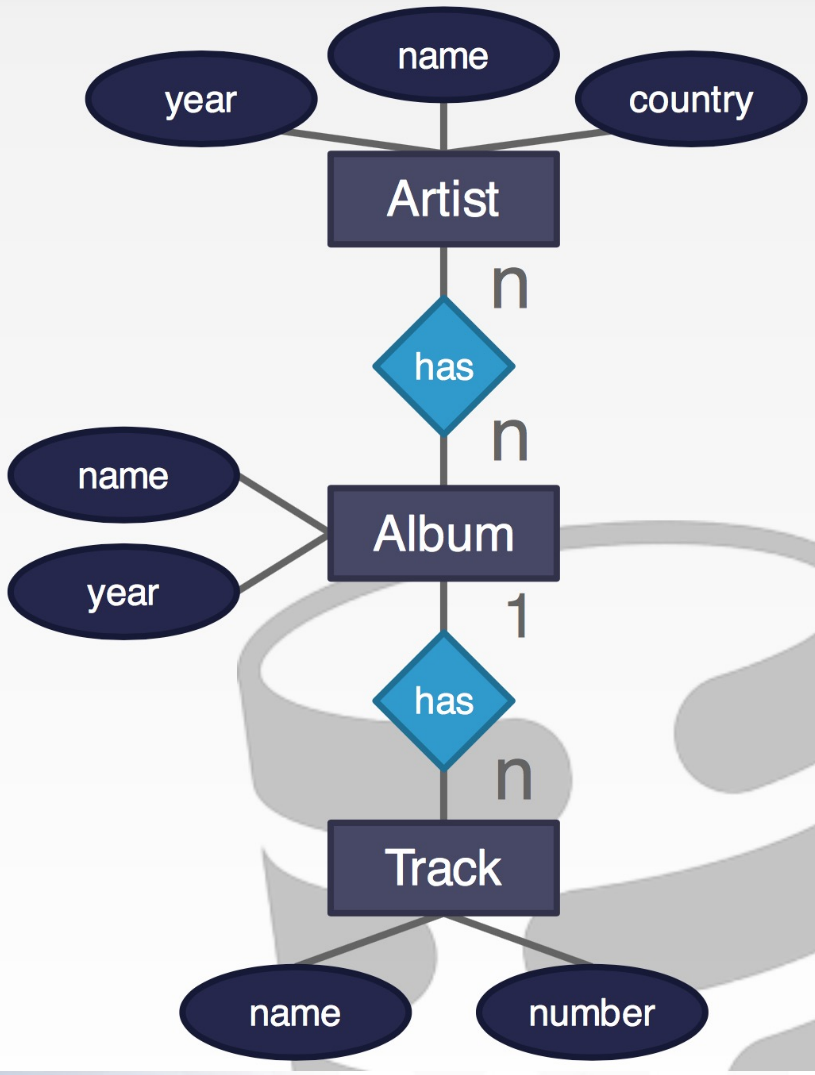
Create a database that models a digital music store.

Things we need to store:

- Information about Artists
- Albums released by Artists
- The Tracks on those Albums

Entity-Relationship Diagram

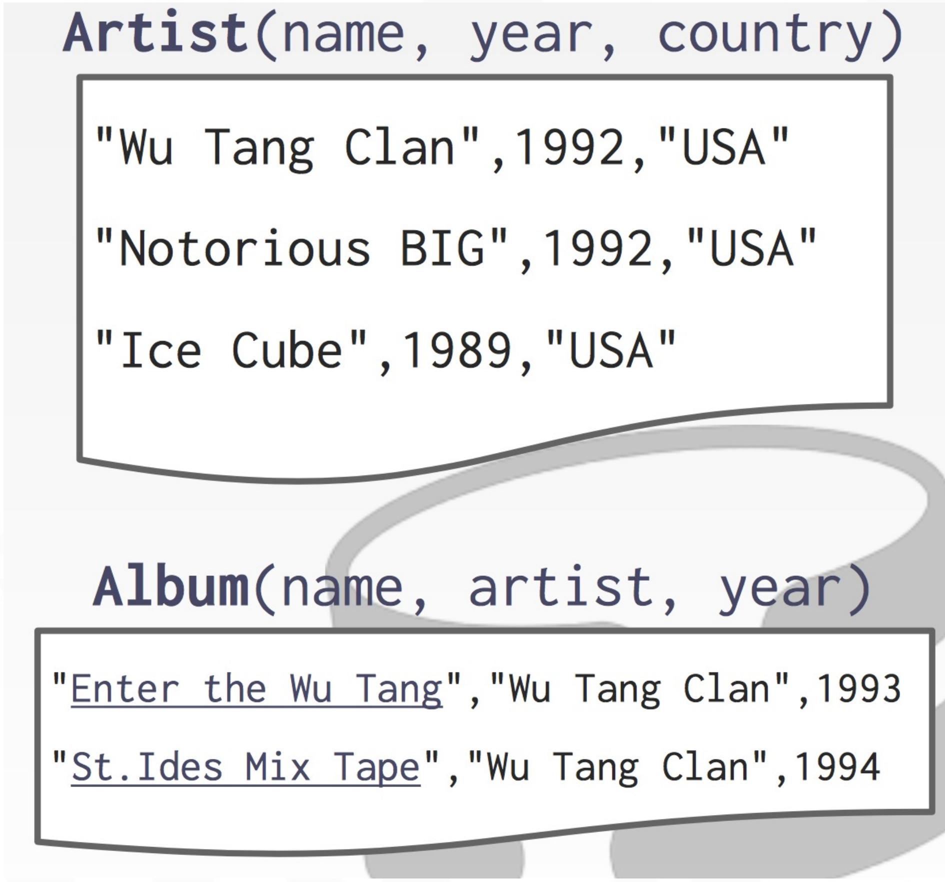
- Artists have names, year that they started, and country of origin.
- Albums have names, release year.
- Tracks have a name and number.
- An Album has one or more Artists.
- An Album has multiple Tracks.
- A Track can appear only on one Album.



Flat File Data Storage

Store the data in comma-separated value (CSV) files.

- Use a separate file per entity.
- The application has to parse the files each time they want to read/update records.



```
Artist(name, year, country)
"Wu Tang Clan",1992,"USA"
"Notorious BIG",1992,"USA"
"Ice Cube",1989,"USA"
```

```
Album(name, artist, year)
"Enter the Wu Tang","Wu Tang Clan",1993
"St.Ides Mix Tape","Wu Tang Clan",1994
```

Flat Files: Data Integrity, API, Durability

Data Integrity

- How do we ensure that the artist is the same for each album entry?
- What if somebody overwrites the album year with an invalid string?
- How do we store that there are multiple artists on an album?

API

- How do you find a particular record?
- What if we now want to create a new application that uses the same database?
- What if two threads try to write to the same file at the same time?

Durability

- What if the machine crashes while we're updating record?
- What if we want to replicate the database on multiple machines for high availability?

Database Management System (DBMS)

System for providing
EFFICIENT,
CONVENIENT and
SAFE
MULTI-USER storage of and access to
MASSIVE amounts of
PERSISTENT data

Why not direct implementation?

Storing data: file system is limited

- size limit by disk or address space
- when system crashes we may loose data
- password/file-based authorization insufficient

Query/update:

- need to write a new C++/Java program for every new query
- need to worry about performance

Concurrency: limited protection

- need to worry about interfering with other users
- need to offer different views to different groups of users

Schema change:

- entails changing file formats
- need to rewrite virtually all applications

More requirements for DBMS

SAFE:

from system failures

from malicious users

CONVENIENT:

simple commands to - debit account, get balance, etc.

unpredicted queries should also be easy

EFFICIENT:

don't search all files in order to get balance of one account

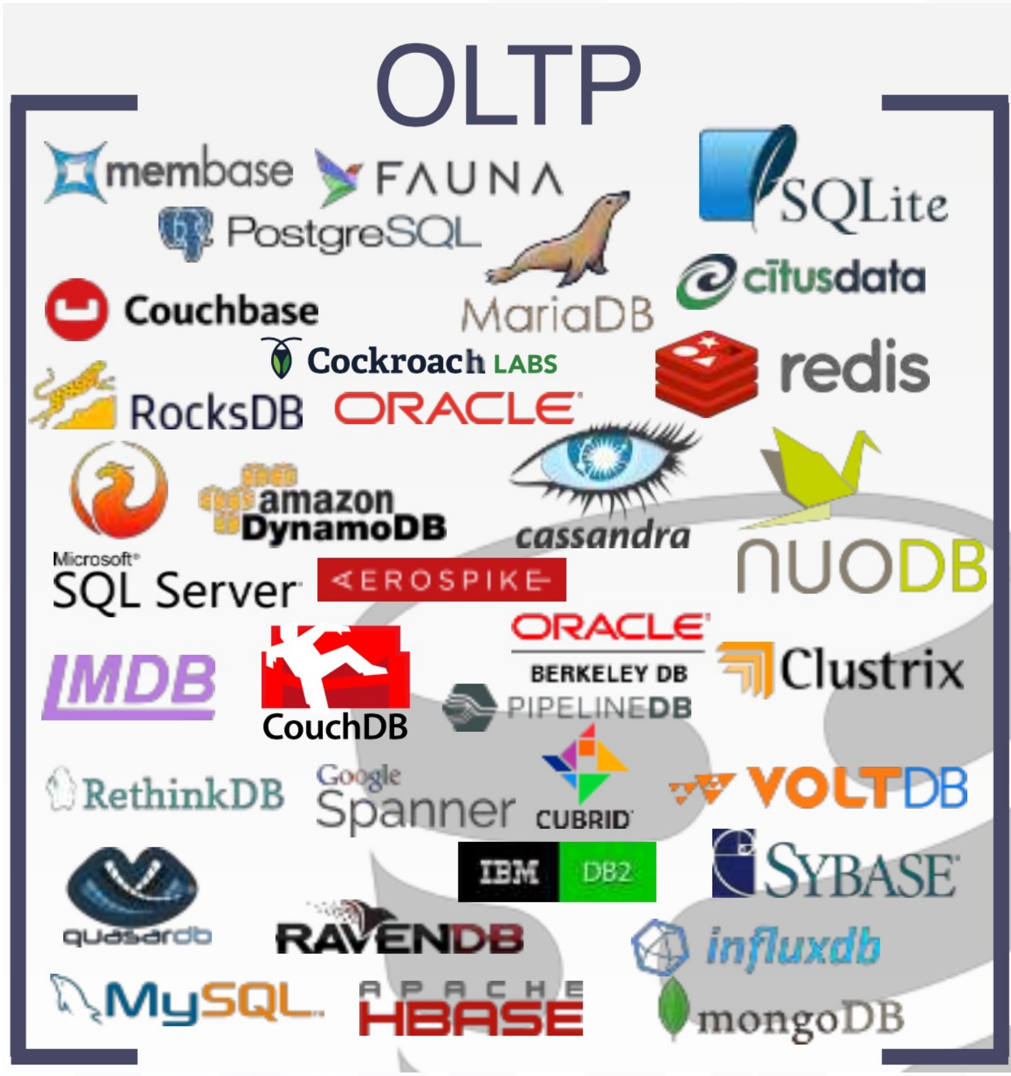
also: get all accounts with low balances, get large transactions, etc.

massive data! -> carefully tune DBMS for performance

DBMS Workload Types: OLTP

On-line Transaction Processing

- Fast operations that only read/update a small amount of data each time.



DBMS Workload Types: OLTP, OLAP, HTAP

On-line Transaction Processing

- Fast operations that only read/update a small amount of data each time.

On-line Analytical Processing

- Complex queries that read a lot of data to compute aggregates.

OLAP
presto
Hortonworks STINGER
VERTICA
TERADATA
amazon REDSHIFT
HyPer
SAP HANA
Microsoft SQL Server
MEMSQL
cloudera IMPALA
HIVE
NETEZZA
PARACCEL
IBM DB2
ORACLE
snowflake
HYRISE
vectorwise

NATIONAL RESEARCH UNIVERSITY

DBMS Workload Types: OLTP, OLAP, HTAP

On-line Transaction Processing

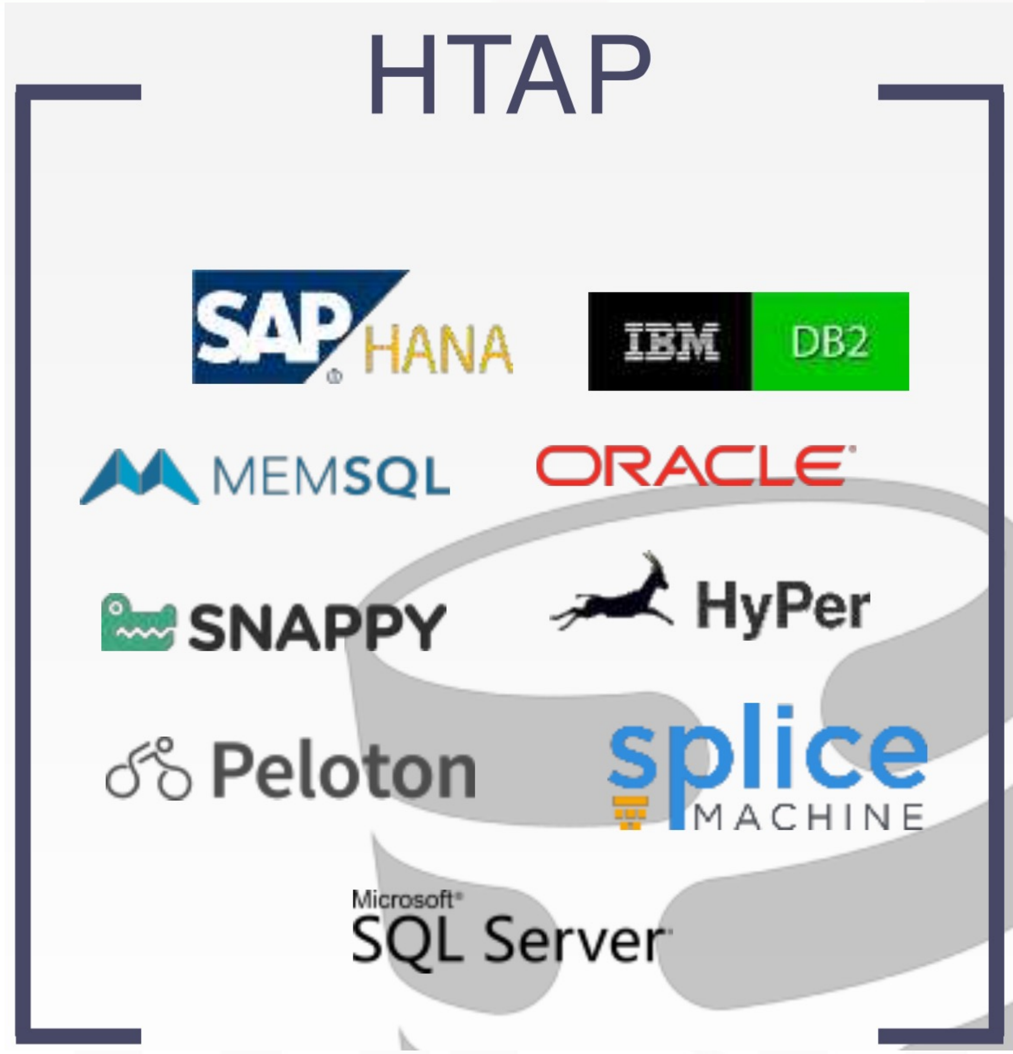
- Fast operations that only read/update a small amount of data each time.

On-line Analytical Processing

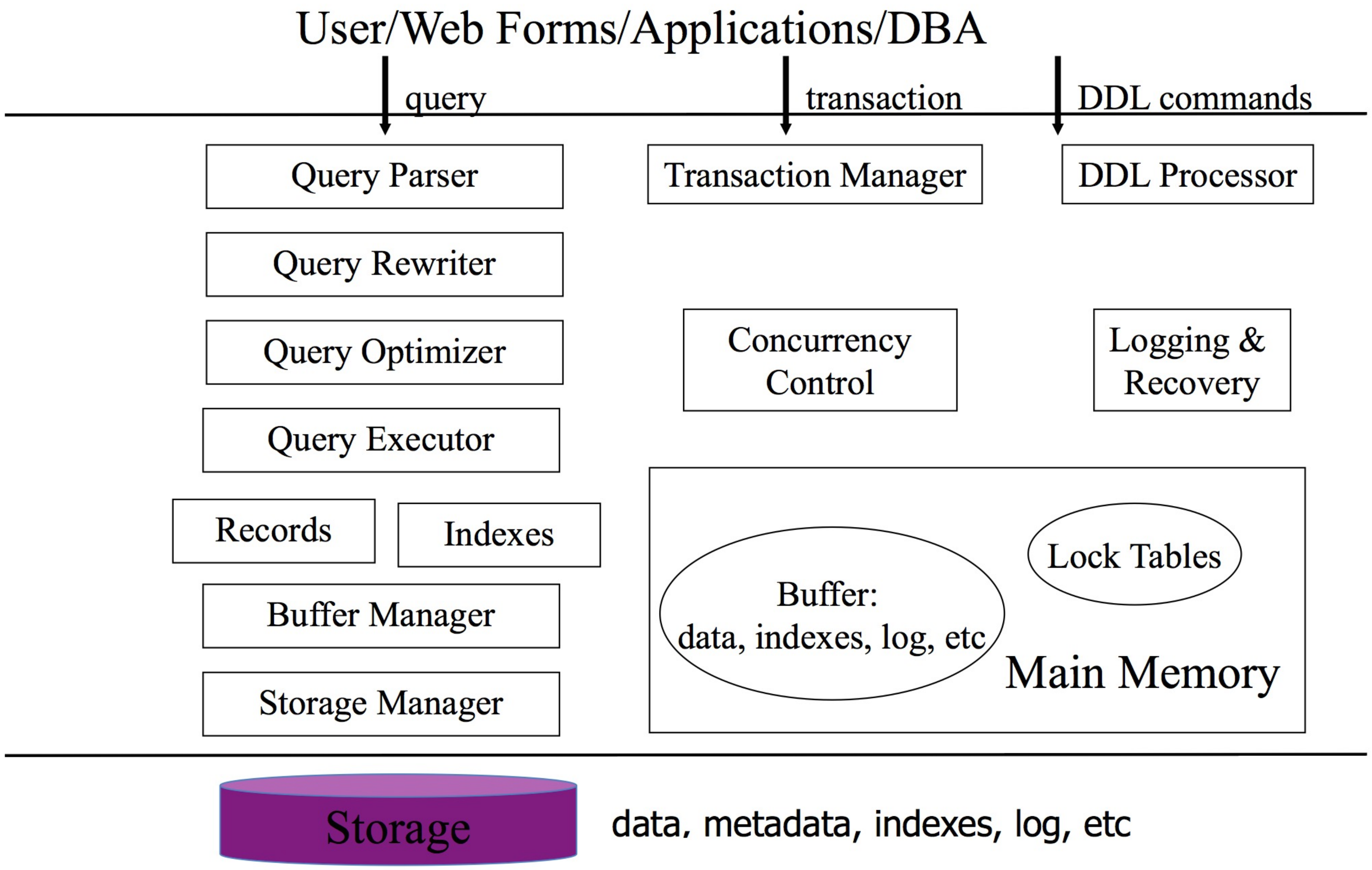
- Complex queries that read a lot of data to compute aggregates.

Hybrid Transaction + Analytical Processing

- OLTP+OLAP together on the same database instance



DBMS Internals



Data structuring: model, schema, data

Data model:

- conceptual structuring of data stored in database
- ex: data is set of records, each with student-ID, name, address, courses, photo
- ex: data is graph where nodes represent cities, edges represent airline routes

Schema versus data

- schema: describes how data is to be structured
  - defined at setup time, rarely changes (also called "metadata”)
- data is actual "instance" of database, changes rapidly
- vs. types and variables in programming languages

Data definition language (DDL)

- commands for setting up schema of database

Data manipulation language (DML)

- commands to retrieve and manipulate data in database
  - get, insert, delete, modify
- “query language”

HIGHER · SCHOOL · OF · ECONOMICS ·

NATIONAL RESEARCH
UNIVERSITY

Thank you for your attention!

20, Myasnitskaya str., Moscow, Russia, 101000
Tel.: +7 (495) 628-8829, Fax: +7 (495) 628-7931
www.hse.ru