01: Introduction

Andrew Crotty // CS497 // Fall 2023

Today's Agenda

Course Logistics
History of Database Systems
Modern OLAP

Course Objectives

Comprehensive overview of the internals of modern OLAP DBMSs.

Students will learn to:

- → Read + evaluate systems research papers
- → Identify system tradeoffs and justify design decisions
- → Craft + deliver presentations to convey research ideas
- → Plan + execute a final project that answers an interesting systems research question

This is **not** a course about classical DBMSs.

Background

This course is designed for students interested in systems research (grads + advanced undergrads).

I assume you have already taken an intro DB systems course (e.g., CS339) or equivalent.

Things that we will <u>not</u> cover:

Intro DBMS concepts like SQL, Serializability Theory, Relational Algebra, Basic Algorithms + Data Structures, etc.

Topics

Data Storage

Compression

Vectorized Execution

Query Compilation

Parallel Join Algorithms

Index Structures

Course Logistics

Course Policies + Schedule:

→ Refer to course web page

Academic Integrity:

- → Refer to Northwestern policy page
- → If you're not sure, ask me.
- → Seriously, don't plagiarize or you will get wrecked.

Grading Rubric

Compression Project – 40% **Encyclopedia Article** – 10% **Final Project** – 50%

Compression Project

Groups of 1-2 students will be given 3 real-world datasets to compress as much as possible.

Deliverables include:

- → Project presentation (5-10 minutes, modeled after a short conference talk) – 5%
- → Written report (at least 4 pages excluding references, modeled after a workshop paper) – 10%
- → Programming component (code, benchmarks, demo, etc.) 25%

Encyclopedia Article

The <u>Database of Databases</u> is an online encyclopedia of DBMSs maintained by the CMU DB Group.

Groups of 1-2 students will write an article.

- → Must provide citations and attributions
- → Avoid unscientific (i.e., marketing) language

You may <u>not</u> copy text / images directly from papers or other sources.

Final Project

Groups of 1-2 will complete a final project on an approved topic related to the course content.

Deliverables include:

- → Project proposal (5-10 minutes, modeled after a short conference talk) – 5%
- → Project presentation (20-30 minutes, modeled after a full conference talk) – 10%
- → Written report (at least 6 pages excluding references, modeled after a conference paper) – 10%
- → Programming component (code, benchmarks, demo, etc.) - 25%

HISTORY OF DATABASE SYSTEMS

WHAT GOES AROUND COMES AROUND READINGS IN DB SYSTEMS, 4TH EDITION, 2006.



WHAT GOES AROUND COMES AROUND... AND AROUND UNDER SUBMISSION 2023

History Repeats Itself

Old issues are still relevant today. Many of the ideas in today's systems are not new.

Every decade, someone invents a SQL replacement and some combination of the following happens.

- \rightarrow It fails.
- → The project slowly adds SQL features.
- → The SQL standard absorbs the best parts.

1960s - IDS

Integrated **D**ata **S**tore

Developed internally at GE in early 1960s.

GE sold their computing division to Honeywell in 1969.

One of the first DBMSs.





1960s - CODASYL

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

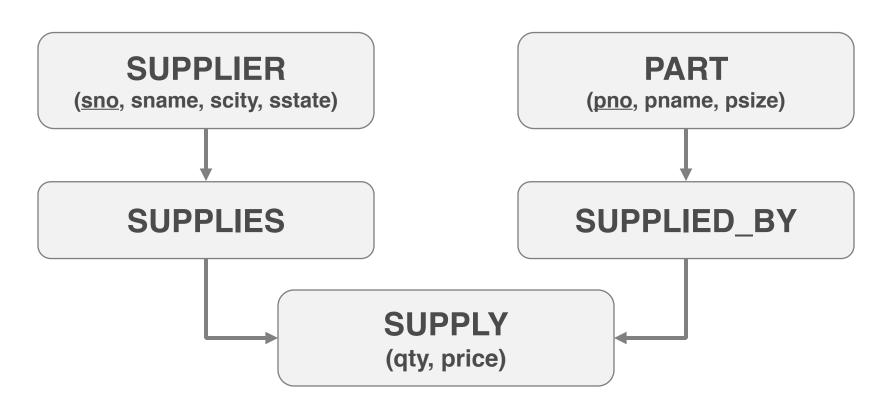
- → Network data model
- → Tuple-at-a-time queries



Bachman

Bachman also worked at <u>Culliane Database</u> <u>Systems</u> in the 1970s to help build **IDMS**.

Schema



Instance

SUPPLIER

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

PART

pno	pname	psize
999	Batteries	Large

SUPPLY

qty	price
10	\$100
14	\$99

Instance

SUPPLIER

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

SUPPLIES

parent	child

SUPPLY

qty	price
10	\$100
14	\$99

PART

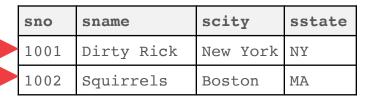
pno	pname	psize
999	Batteries	Large

SUPPLIED_BY

parent	child

Instance

SUPPLIER



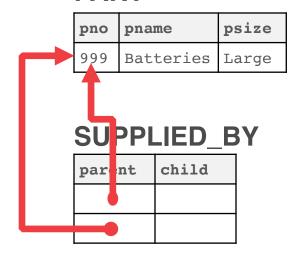
SUPPLIES

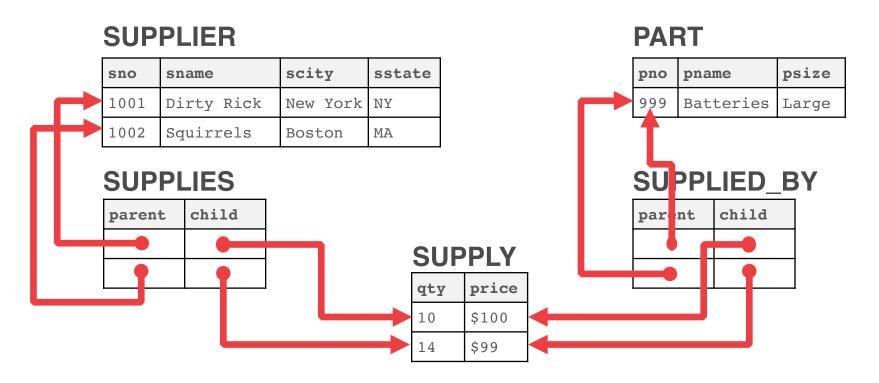
parent	child

SUPPLY

qty	price
10	\$100
14	\$99

PART





1960S - IBM IMS

Information Management System
Early DBMS developed to keep track of purchase orders for Apollo moon mission.

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



Schema

Instance

SUPPLIER

(sno, sname, scity, sstate)

PART

(pno, pname, psize, qty, price)

Schema

SUPPLIER

(sno, sname, scity, sstate)

PART

(pno, pname, psize, qty, price)

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

Schema

SUPPLIER

(sno, sname, scity, sstate)

PART

(pno, pname, psize, qty, price)

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

pno	pname	psize	qty	price
999	Batteries	Large	10	\$100

Schema

SUPPLIER

(sno, sname, scity, sstate)

PART

(pno, pname, psize, qty, price)

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

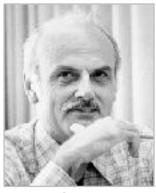
	pno	pname	psize	qty	price
1	999	Batteries	Large	10	\$100

pno	pname	psize	qty	price
999	Batteries	Large	14	\$99

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.



Codd

DESIGNABILITY, RECOMMENCY AND CONSISTENCY OF VELOCIONS SHOWER OF LAWER DATA BANKS.

Tenerch Division San Rosty California

ARCHARI: The large integrated data banks of the future will, excession wave solutions of marinan degrees is control form. It will not be install for this and of deterior being control to be reducible. To types of recombary are designed and discussed. See type may be each response to be provided part of control to discussed, and the type may be early report to be provided part of the discussed and the control to the control of the days have been about it must be recommended to the control of the days have them to be about it must have seen each of detecting any "lagging" increasing the control of the days have them to be control of the days have them to be control of the days have the days of the days of the days of the days of the control.

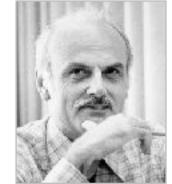
23 29804 (GAS) August 30, 1989

INVITED BENEFIT, THE MATTER. I this papers has been admitted for distribution elements and not seen denot as a beneath beport for early distribution of the underdy. As a contemp to the faringed publisher, to check and he whichy distributed word, after the care of particle

Ouries on to Depetited type 100 Years In Notes Researt Gentra, Fast 1975cc for 165, Yackloon Respect, the York 2018

Plational Model

hematician working saw developers writing IMS and every time the layout changed. to avoid this



Codd

e data structures. h-level language. to implementation.

E. T. Cald Despute the later See Root, California

APPRACE: The large, integrated data banks of the fature will contain many arbitrary of martine degrees in stored form. It will not be unusual for this set of stored relations to be nederlest. Two types of secondary are detired and discussed. One type mor be employed to improve accommutativity of certain ideas of information which rapper to be in great desord. When either type of peakedaway exists, these responsible for control of the dris basic should once about it and have some means of detecting any "inguital" intensistancies in the total put of chorac selections. Completency checking night be helpful in tracking down meaning tion (and possibly traudalent charges in the data bank porterio.

83 99904 (Glas) August 30, 1989

LEVIND DESCRIPTION WITH. - This papers has been submitted for AMELICATION eligances and not mean deleted as a best-such import for early three hallow of its noteds. As a source to to the latested publisher, it should not be whichly distributed world after the care of pathods

Cooles are be invested than 100 Money In Notice Research Contain Fact 1975 or tall, includes beinger, the York 2018

elatio

Information Retrieval

P. MARNEAUE, Febru

nematic saw de writing e data h-leve

A Relational Model of Data for Large Shared Data Banks

R. P. Corn. 1814 Emerch Leisenstone, San Jose, California

Puters over of large data basis need be profited from baring to linery have the date is proprieted in the modifier line. interest representation). A prompting remice which supplies such information is act a satisfactory solution, extinities of users at terminate and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the ecounty representation are dienard. Charges in data representation will after bespecied on a result of changes in query, update, and report halfic and actual growth in the types of stored information.

Existing resistances of decreased mate contains precise upon with translatered film or slightly more general extends. march of the darw. in Section 1, mediagramies of Press madels. are disposed. A model bound on a proportion, as executface for their base relation, and the covered of a universal dere sublenguage are infraduced, in Legion 2, cortain apera-Harry on relations father than implest informace) any observation and explice to the problems of reductioner and considerary in the part's probable

are written and health, over seed, even been, over draction, date, emperature, intermediate of white, emberging of white, exhibiting starbellishing reducting continues companies the mineral language, profession solubs, results, date benefits

CREATIONES SITE SITE SITE 610, 610, 610

I. Relational Model and Normal Form

11. Demoncornor

This paper is necessarily with the application of eleinjuriesy relative theory to systems which provide chared among to large harlor of fore attend data. Energyl for a paper lay Chible (1.), the principal application of relations to date systems for been to deductive constance awaring systems. Levels and Maron Mi provide numerous references to work in Dispers.

In ecotrary, the problems triated here are those of data delegandence—the independence of ambiention programs and terroinsi activities from growth in data types and changes in data representation—and service kinds of data increasing which are expected to become toroblesome over in mondadostive systems.

The relational view (or model) of data described in Section I appears to be expensor in several respects to the each or necessic model i2, 4) presently in segue for nonaferential systems. It provides a means of describing that with its natural structure unity—that it, without experies. point any odd formal structure for machine representation. purposes. Amendingly, it provides a busin for a high less! date language which will yield maximal independence by twees programs on the one hand and marking recreasing

tion and organization of data on the other, A further selvantage of the relational view is that it. forms a second basis for treating derivability, redundancy. and construction of relations—those are discussed in Section 2. The network model, on the other hand, has spreamed a number of senfedom, not the least of which is micraking the desiration of connections for the deciration of relations see remarks in Section 2 on the "connection trap").

Enally, the relational view permits a characteristics. of the stope and legical limitations of present formation. data systems, and also the relative movits (from a logisal standpoint) of competing representations of data within a simple system. Examples of this closure prospection are cited in various parts of this paper. Improventations of systems to respect the relational model are not discussed.

1.2. Barra Dispussionarios Princer Ricciona

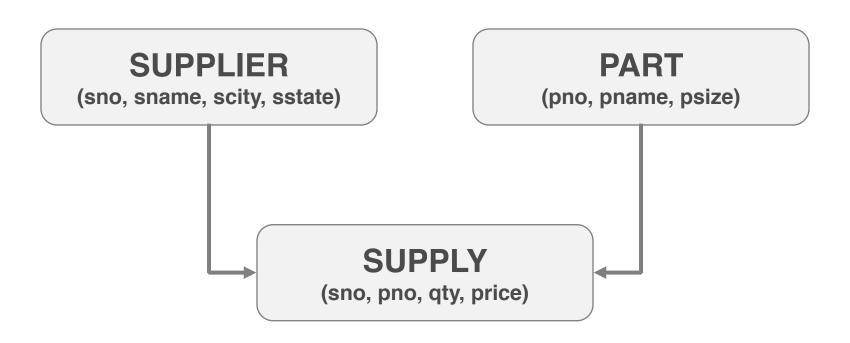
The receiving of data description taking is mornely do. velocal information materia represents a major editator. toward the grad of data independence [2, 4, 2]. Such taken buildate changing vertain characteristics of the data capra secution stored in a data bank. However, the variety of data representation characteristics which can be changed widout figurelly impairing some application programs is still gate limited. Further, the model of data with which mere intersect is still electrical with proposentational proportio, particularly in regard to the separatation of collections of data (as opposed to individual horse). Three of the principal kinds of data dependencies which still need to be removed are: artisting dependence, including dependence, and secons path dependence. In some systems these dependencies are not clearly separable from one another.

121. Ordering Dependence. Elimente of dela la a data bank may be stored in a variety of ways, some involving no economy for ordering some paraliting such element to participate in one ordering only, others possitting each element to participate in second inderings. Let us remailer these arising systems which either require or proudt date. adjunctive to be expeed in at least one total codesing which in rively associated with the hardware determined ordering al addresses. Far reample, the remade of a Six concerning racts might be stored in ascending order by part serial number. Such systems normally permit application prograve to severe that the order of prosestation of records from much a file is identical to (or is a sufferdering of) the

Vehicle II / Number 6 / June, 1904

Relational Data Model

Schema



Relational Data Model

Instance

SUPPLIER

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

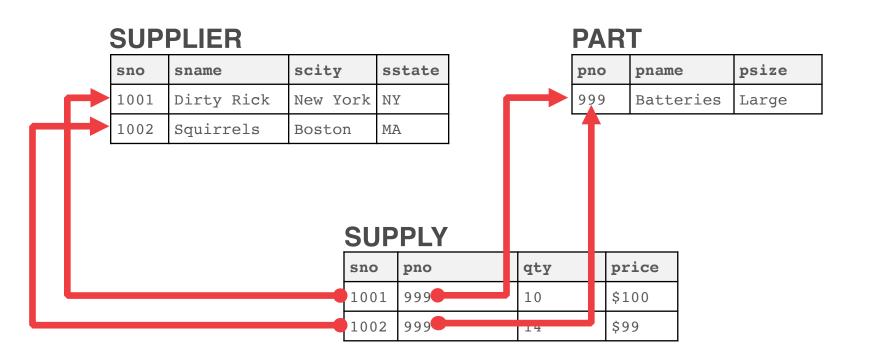
PART

pno	pname	psize
999	Batteries	Large

SUPPLY

sno	pno	qty	price
1001	999	10	\$100
1002	999	14	\$99

Relational Data Model



Early implementations of relational DBMS:

- → Peterlee Relational Test Vehicle IBM Research (UK)
- → **System R** IBM Research (San Jose)
- → INGRES U.C. Berkeley
- → Oracle Larry Ellison
- → Mimer Uppsala University



Gray



Stonebraker



Ellison

The relational model wins.

- → IBM first releases SQL/DS in 1981.
- → IBM then releases DB2 in 1983.
- → "SEQUEL" becomes the standard (SQL) after supposedly Stonebraker refused to talk to the ANSI standards committee

Many new "enterprise" DBMSs but Oracle wins marketplace.

Stonebraker creates Postgres as an "object-relational" DBMS.















InterBase[®]

The relational model wins.

- → IBM first releases SQL/DS in
- → IBM then releases DB2 in 198
- → "SEQUEL" becomes the stand after supposedly Stonebraker to the ANSI standards commit

Many new "enterprise" DBN but Oracle wins marketplace



But Ingres did not show up at the committee meetings because founder Mike Stonebraker detested the idea of having technology standards. Stonebraker was vocal about it. He thought they inhibited innovation and artificially restricted what got to the marketplace. Maybe so, but his hard-line position probably did not help his company. Don Deutsch, who served as chairman of the database committee, summed things up this way: "I tell you, QUEL was a much nicer language than SQL. No rational person would have chosen SQL instead of QUEL.... Ingres was stupid."

Stonebraker creates Postgres as an "object-relational" DBMS.

1980s – Object-Oriented Databases

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 (e.g., XML, JSON).



Object-Oriented Model

Application Code

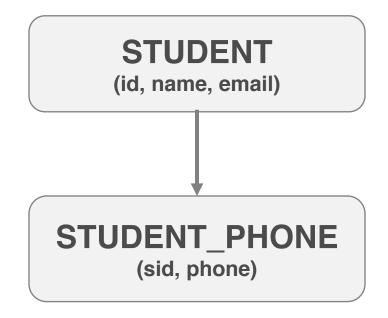
```
class Student {
  int id;
  String name;
  String email;
  String phone[];
}
```

Object-Oriented Model

Application Code

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

Relational Schema



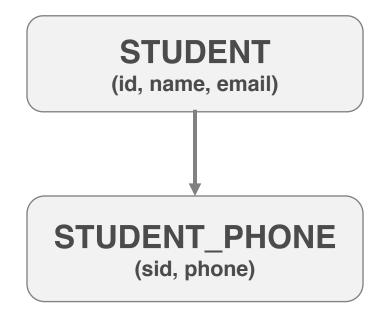
Application Code

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

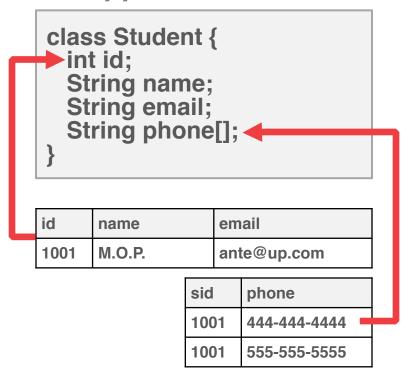
id	name	email
1001	M.O.P.	ante@up.com

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

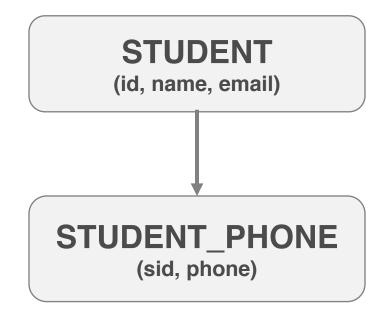
Relational Schema



Application Code



Relational Schema



Application Code

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

Application Code

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



1990s – Boring Years

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.
- → SQLite started in early 2000.

Some DBMSs introduced pre-computed <u>data</u> <u>cubes</u> for faster analytics.









2000s – Internet Boom

All the big players were heavyweight and expensive. Open-source DBMSs were missing important features.

Many companies wrote their own custom middleware to scale across many independent single-node DBMS instances.

2000s – Data Warehouses

Rise of the special purpose OLAP DBMSs.

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

Significant performance benefits from using columnar data storage model.













2000s - MapReduce

Distributed programming and execution model for analyzing large data sets.

- → First proposed by Google (MapReduce).
- → Yahoo! created an open-source version (**Hadoop**).
- → Data model decided by user-written functions.

People (eventually) realized this was a bad idea and grafted SQL on top of MapReduce. That was a bad idea too.



2000s - NoSQL

Focus on high-availability & high-scalability:

- → Schema-less (i.e., "Schema Last")
- → Non-relational data models (document, key/value, column-family)
- → No ACID transactions
- → Custom APIs instead of SQL
- → Usually open-source

























2010s - NewSQL

Provide same performance for OLTP as NoSQL without giving up ACID:

- → Relational / SQL
- → Distributed

Almost all of the first group of systems failed.

Second wave of "distributed SQL" systems are (potentially) doing better.















2010s - NewSQL

Provide same performance for OLTP as NoSQL without giving up ACID:

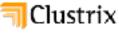
- → Relational / SQL
- → Distributed

Almost all of the first group of systems failed.

Second wave of "distributed SQL" systems are (potentially) doing better.







FOUNDATION DB

















2010s - HTAP

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.















2010s – Stream Processing

Execute continuous queries on streams of tuples, extending semantics to include notion of windows.

Often used in combination with batch-oriented systems in a **lambda architecture**.











2010s - The Cloud

First database-as-a-service (DBaaS) offerings were "containerized" versions of existing DBMSs.

There are newer DBMSs that are designed from scratch explicitly for running in a cloud environment.















2010s - Shared-Disk

Instead of writing a custom storage manager, the DBMS leverages distributed storage.

- → Scale execution layer independently of storage.
- → Favors log-structured approaches.

This is what most people think of when they talk about a **data lake**.















2010s – Graphs

Systems for storing and querying graph data.

→ Similar to the network data model (CODASYL)

Their (supposed) advantage over other data models is to provide a graph-centric query API → <u>SQL:2023</u> is adding graph query syntax (SQL/PCG)

Latest <u>research</u> (2023) shows that relational DBMSs outperform graph DBMSs.









🕦 TigerGraph

















2010s – Time Series

Specialized systems that are designed to store time series / event data.

The design of these systems make deep assumptions about the distribution of data and workload query patterns.

















2020s – Blockchains

Decentralized distributed log with incremental checksums (Merkle Trees).

→ Uses Byzantine Fault Tolerant (BFT) protocol to determine next entry to append to log.

Many blockchain use cases seem like they can be solved with a "traditional" OLTP DBMS and/or external policies (e.g., authentication).











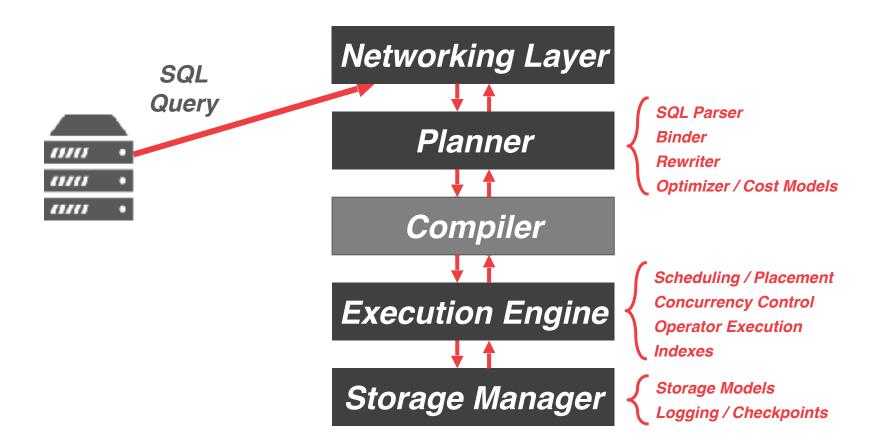
Current State of Affairs

The demarcation lines of DBMS categories will continue to blur over time as specialized systems expand the scope of their domains.

→ Every NoSQL DBMS (except for Redis) now supports SQL.

The relational model and declarative query languages promote better data engineering.

DBMS Overview



Executing OLAP queries in a distributed DBMS is roughly the same as on a single node.

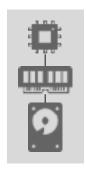
→ Query plan is a DAG of physical operators.

For each operator, the DBMS considers where input is coming from and where to send output.

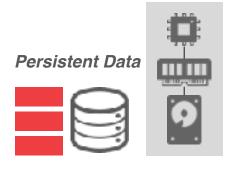
- → Table Scans
- → Joins
- \rightarrow Aggregations
- \rightarrow Sorting

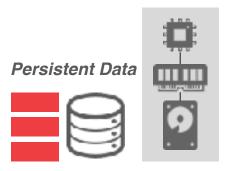


•

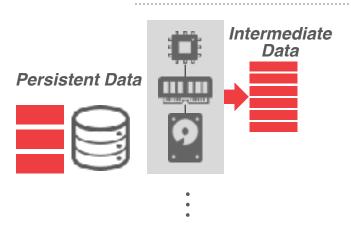


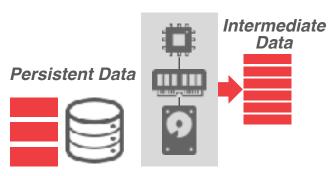
Worker Nodes



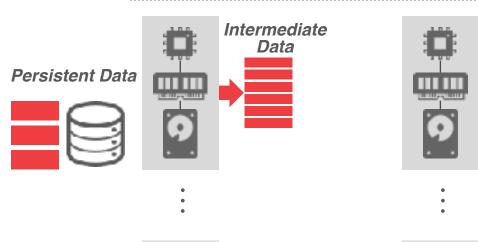


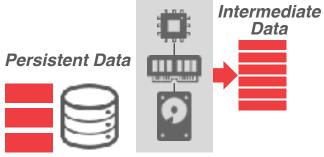
Worker Nodes





Worker Nodes

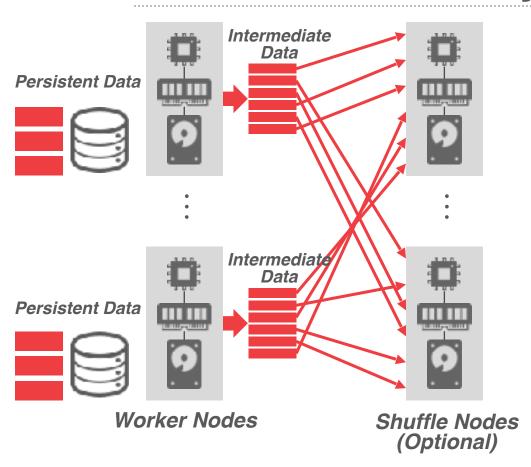


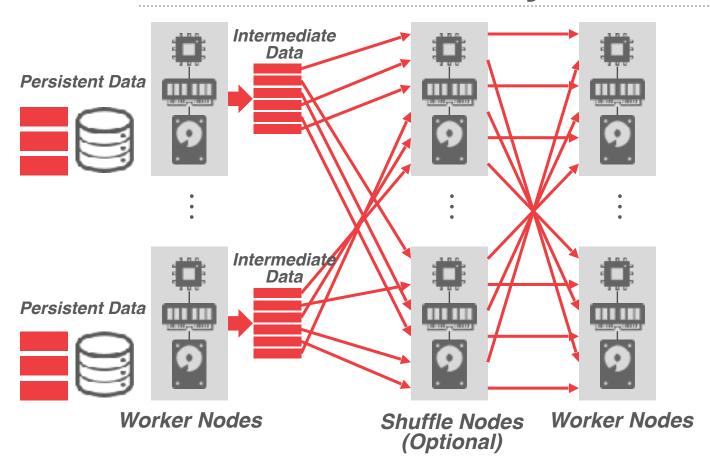


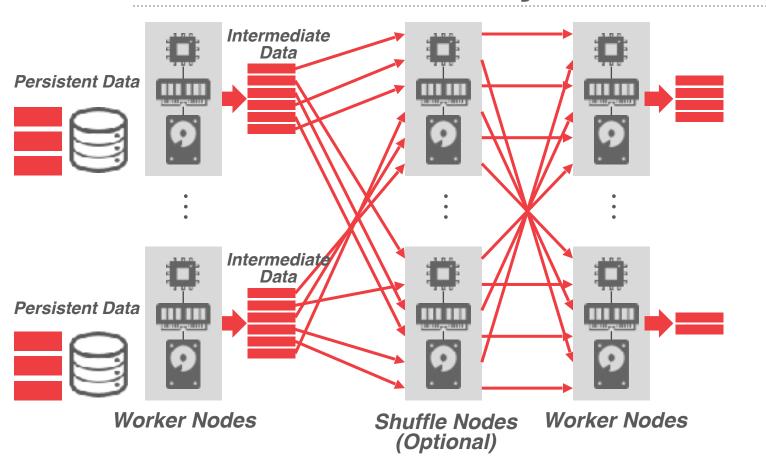
Worker Nodes

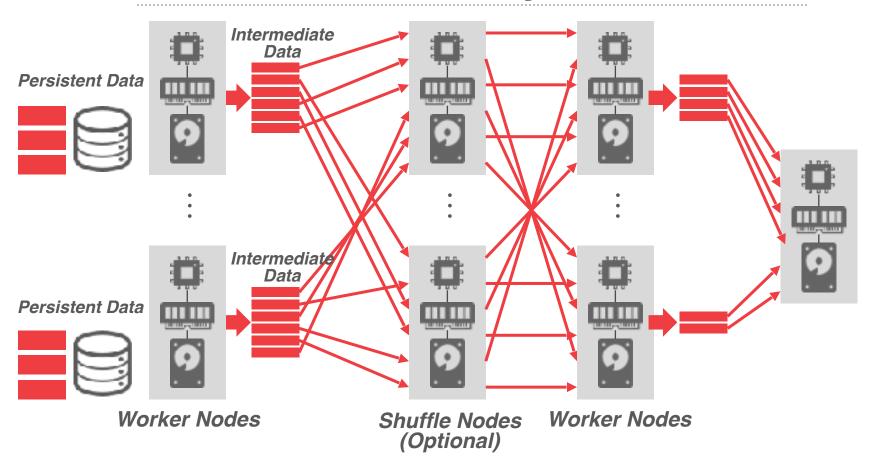


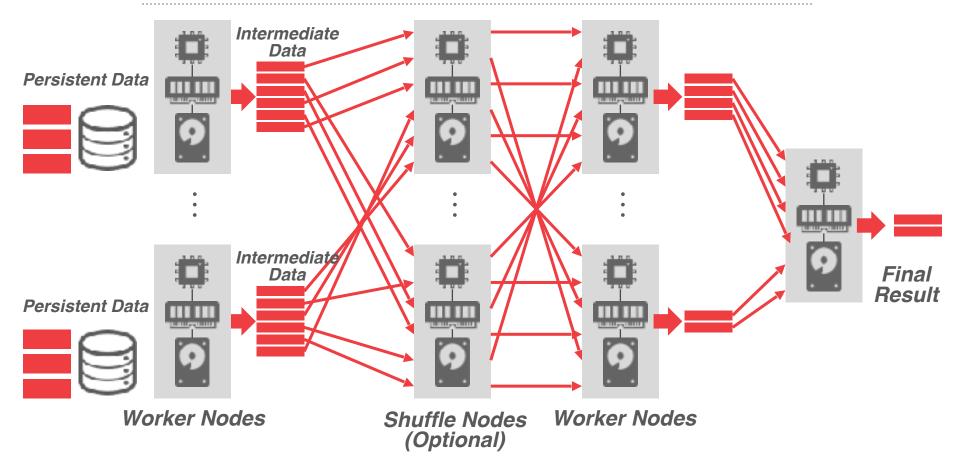
Shuffle Nodes (Optional)











Types of Data

Persistent Data:

- → The "source of record" for the database (e.g., tables).
- → Modern systems assume that these data files are immutable but can support updates by rewriting them.

Intermediate Data:

- → Short-lived artifacts produced by query operators during execution and then consumed by other operators.
- → The amount of intermediate data that a query generates has little correlation to amount of persistent data that it reads or the overall execution time.

Distributed Architecture

A distributed DBMS's system architecture specifies the location of the database's persistent data files. This affects how nodes coordinate with each other and where they retrieve / store objects in the database.

Two approaches (not mutually exclusive):

- → Push Query to Data
- → Pull Data to Query

Push vs. Pull

Approach #1: Push Query to Data

- → Send the query (or a portion of it) to the node that contains the data.
- → Perform as much filtering and processing as possible where data resides before transmitting over network.

Approach #2: Pull Data to Query

- → Bring the data to the node that is executing a query that needs it for processing.
- → This is necessary when there is no compute resources available where persistent data files are located.

Filtering and retrieving data using Amazon S3 Select



Approa

- \rightarrow Send t contail
- → Perfor where

With Amazon 53 Select, you can use simple structured query language (SQL) statements to filter the contents of an Amazon 53 object and retrieve just the subset of data that you need. By using Amazon 53 Select to filter this data, you can reduce the amount of data that Amazon S3 transfers, which reduces the cost and latency to retrieve this data.

Amazon S3 Select works on objects stored in CSV, JSON, or Apache Parquet format, it also works with objects that are compressed with GZIP or BZIP2 (for CSV and JSON objects only), and server-side encrypted objects. You can specify the format of the results as either CSV or JSON, and you can determine how the records in the result are delimited.

You pass SQL expressions to Amazon SS in the request. Amazon SS Select supports a subset of SQL. For more information about the SQL elements that are supported by Amazon S3 Select, see SQL reference for Amazon S3 Select.

You can perform SQL queries using AWS SDKs, the SELECT Object Content REST API, the AWS Command Line Interface (AWS CLI), or the Amazon S3 console. The Amazon S3 console limits the amount of data returned to 40 MB. To retrieve

Approa

- \rightarrow Bring that needs it for processing.
- → This is necessary when there is no compute resources available where persistent data files are located.

Filtering and retrieving data using Amazon S3 Select



Approa

With Amazon \$3 Select ..

Microsoft

Feedback

Query Blob Contents Article • 07/20/2021 • 10 minutes to read • 3 contributors

The query Blob contents API applies a simple Structured Query Language (SQL) statement on a blob's contents and returns only the queried subset of the data. You can also call query 8105 contents, to query the contents of a version or snapshot.

Request

The Query Blob Contents request may be constructed as follows. HTTPS is recommended. Replace myaccount with the name of your storage account:

hydecount with the ham a significant	HTTP Version
PGST Method Request URI	HTTP/1.0
https://myaccount.blob.come.windows.met/mycontainer/myblob?compropery	
https://myaccount.blot.comp.windows.net/mycontainer/myblot/comp-queryscrapshot= <pre>https://myaccount.blot.comp.windows.net/mycontainer/myblot/comp-queryscrapshot=</pre>	HTTP/IL1
constitution of the second state of the second state of the second secon	
mttps://www.mttps://www.mtps.compagegryaversionid=kpaceTime>	
https://myaccount.bloc.comp.windows.met/mycontainer/myblob/comprquerysversionid=clatering>	

uery language (SQL) statements to filter the contents of an at you need. By using Amazon 53 Select to filter this data, you can ch reduces the cost and latency to retrieve this data.

or Apache Parquet format. It also works with objects that are only), and server-side encrypted objects. You can specify the etermine how the records in the result are delimited.

azon S3 Select supports a subset of SQL. For more information Select, see SQL reference for Amazon S3 Select.

bbject Content REST API, the AWS Command Line Interface le limits the amount of data returned to 40 MB. To retrieve

compute ent data files are

Shared-Nothing

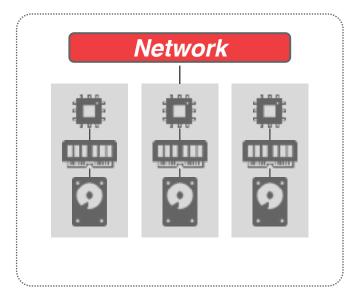
Each DBMS instance has its own CPU, memory, locally-attached disk.

→ Nodes only communicate with each other via network.

Database is partitioned into disjoint subsets across nodes.

→ Adding a new node requires physically moving data between nodes.

Since data is local, the DBMS can access it via POSIX API.



Shared-Nothing

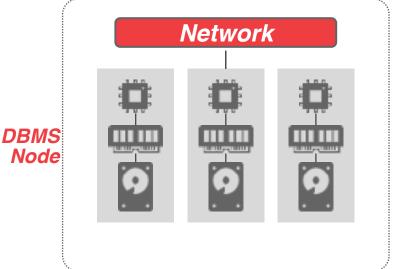
Each DBMS instance has its own CPU, memory, locally-attached disk.

→ Nodes only communicate with each other via network.

Database is partitioned into disjoint subsets across nodes.

→ Adding a new node requires physically moving data between nodes.

Since data is local, the DBMS can access it via POSIX API.

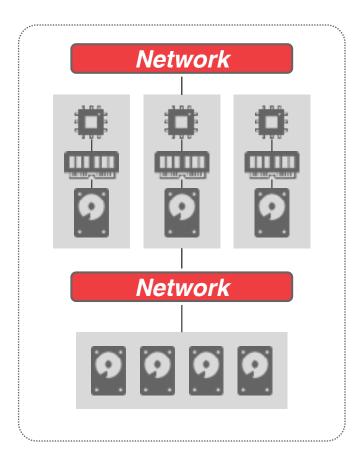


Shared-Disk

Each node accesses a single logical disk via an interconnect, but also has its own private memory and ephemeral storage.

→ Must send messages between nodes to learn about their current state.

Instead of a POSIX API, the DBMS accesses disk via a userspace API.



Shared-Disk

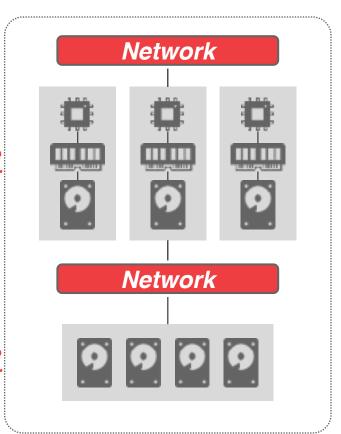
Each node accesses a single logical disk via an interconnect, but also has its own private memory and ephemeral storage.

Compute

→ Must send messages between nodes Layer to learn about their current state.

Instead of a POSIX API, the DBMS accesses disk via a userspace API.

Storage Layer



System Architecture

Choice #1: Shared-Nothing:

- → Harder to scale capacity (data movement).
- → Potentially better performance & efficiency.
- → Apply filters where the data resides before transferring.

Choice #2: Shared-Disk:

- → Scale compute / storage layers independently.
- → Easy to shut down idle compute layer resources.
- → May need to pull uncached persistent data from storage layer to compute layer before filtering.

System Architecture

Choice #1: Shared-Nothing:

- → Harder to scale capacity (data movement).
- → Potentially better performance & efficiency.
- → Apply filters where the data resides before transferring.

Choice #2: Shared-Disk:

- → Scale compute / storage layers independently.
- → Easy to shut down idle compute layer resources.
- → May need to pull uncached persistent data from storage layer to compute layer before filtering.

Shared-Disk

Traditionally the storage layer in shared-disk DBMSs were dedicated on-prem NAS.

→ Example: Oracle Exadata

Cloud **object stores** are now the prevailing storage target for modern OLAP DBMSs because they are "infinitely" scalable.

→ Examples: Amazon S3, Azure Blob, Google Cloud Storage

Object Stores

Partition the database's tables (persistent data) into large, immutable files stored in an object store.

- → All attributes for a tuple are stored in the same file in a columnar layout (PAX).
- → Header (or footer) contains meta-data about columnar offsets, compression schemes, indexes, and zone maps.

The DBMS retrieves a block's header to determine what byte ranges it needs to retrieve (if any).

Each cloud vendor provides their own proprietary API to access data (PUT, GET, DELETE).

→ Some vendors support predicate pushdown (S3).

Object Stores

Partitil into la

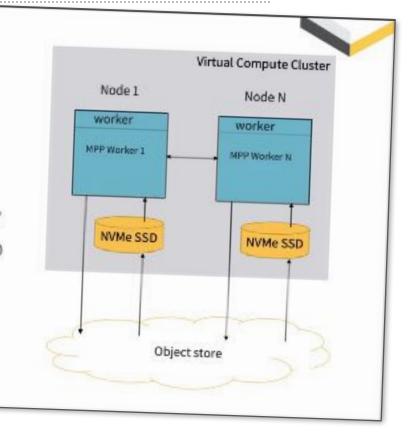
- \rightarrow All a colu
- → Hea offs

The [what

Each API t

Workers

- Separated compute / storage
- One Worker pod per compute node
 - Executes portions of the query plan
- Custom network protocol over UDP
 - Data distribution between workers
 - Uses Intel DPDK
 - 50% higher throughput on AWS over TCP/IP
- Shard files cached in local NVMe SSD
- Shards persisted in object store
 - Custom AWS S3 access library
 - 3X better throughput than stock S3 lib





Yellowbrick

Observation

Snowflake is a monolithic system comprised of components built entirely in-house.

Most of the non-academic DBMSs we will cover this semester will have a similar overall architecture.

But this means that multiple organizations are writing the same DBMS software...

OLAP Commoditization

One recent trend in the last decade is the refactoring of OLAP engine sub-systems into standalone open-source components.

→ This is typically done by organizations <u>not</u> in the business of selling DBMS software.

Examples:

- → System Catalogs
- → Query Optimizers
- → File Format / Access Libraries
- → Execution Engines

OLAP Commoditization

One recent trend in the last de refactoring of OLAP engine su standalone open-source comp

→ This is typically done by organiza business of selling DBMS softwa

Examples:

- → System Catalogs
- → Query Optimizers
- → File Format / Access Libraries
- → Execution Engines

Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System

Sunjt Chadhari Marosef Awards Rodrose, WA 9800, USA Brajta-Brake redecom

Abstract

Dealess meanthquisess of the companys for he new mikesion's I projuge Browns. decision according to a real of court provinging and deployment on at a reservab communical system have formulating homeon the a long store and save new second combinate that makes them . difficult design on tomo of tom "printprin arte', or a second platform the networked BORDERS STYLING SET IN EXP OF COMPANY. It is written that deathers continue he come to menge, probabile in the performer employees, are unracy we sent for the chake ped \$157-tyle simplification of server test implify and may not to similarly traced Wy supply a recical authorized departure in which district exhausings is purkaged into much make BEC sale des mangen and less provides, of its and windows is self-assessment ord management repolations.

1. The Need for a New Departure

Designe schwiegt has at correctly secretal trainfront, in a Surface of information technology (IFthoughout for hard free design, Highest of Suffersion Surface), and the second of Suffersion Surface, a more to the old-distance of conformation are less a more to the old-distance of conformation of information sections. Needlessness, dealers software, software appears a subject of an absolute ordering a conformation section. Needlessness, dealers or observed software appears as a subject of the section of the second applications ferrors and figured formers as into assessment as a proper conformation for the last, deliction furnitive, for an approximate process of hearths and sever to remain conclude vectorian file neacedlessness, etc.

Permission on explainable for effect part of elicitorisms risk is matted perceivable that the order consequently or distributed for effects consequently, advanced to 19 Me outputted notice and these consequently, advanced on the data respects and notice; in the consequently of the permission of the first large "Matthews (See Matthews of the permission of the first large "Matthews Consequently of the first large "Matthews and Consequently of the first large "Matthews and Consequently for the Section pages."

Presentings of the 25th International Conference on Very Larry Decisions, Calva Serial, 5000 Griked Written Batterety visits Sortend D 64-123 Sortewarker, Germany well-and countries, k

Suppose a leasy teacher to handleng fell fined and flowber we should not reached that the findance cases, in the cult of regionsing, slighting, and restaure partiest addenic community, is in long affaire. It could lied at notic important application areas and rajes track in the release industry stought HISCHO HIS DESCRIPTION THE RE COME HIS "gampin role" Ples, with the francis drap of hardware and neffronts private the expenses can be human symmetric on times out period the con of country by a bidger trains. The complexity and one of these field and care male in Blody to probble Anabuse specie-free larie pletin our rainingly pressure. rile at the fetter I streetunes. Seel Abdress technology is story likely soler adopted in advantable and congressed have winter addressived application province-

Both of the three problems then from pulsaging of deathers and acting into a maple and of descriptions. IMMERICAL Appropriate, for operation, Not signed that the architecture is a design appropriate for the new age of splittings an applications. For this mater, approach due were covered and advance in this paper is no specula of Carach. Ventratelly material provided dues contained to the control operation of the action for the control operation of the control operation.

The one of the paper is required or follows. Seather 1 per largetime come conjunted the evaluation industing that distinct primary is their industrial force or makes. Seather 1 profily uniform sealor assume to a notice making the paper and paper are seather assume the paper are distincted from the force of the paper and th

2 Crisis Indicators

To begin our radius, for an entireption after ensuring relevantions are low determine systems are recorded by excession tradius, and the except determines.

(Perturbe I Economic chief purches become exceptable, Dadres system offer new and man feature deaths to extend would and the proping

System Catalogs

A DBMS tracks a database's schema (e.g., table, columns) and data files in its catalog.

- → If the DBMS is on the data ingestion path, then it can maintain the catalog incrementally.
- → If an external process adds data files, then it also needs to update the catalog so that the DBMS is aware of them.

Notable implementations:

- → <u>HCatalog</u>
- → Google Data Catalog
- → Amazon Glue Data Catalog

Query Optimizers

Extendible search engine framework for heuristic- and cost-based query optimization.

- → DBMS provides transformation rules and cost estimates.
- → Framework returns either a logical or physical query plan.

This is the hardest part to build in any DBMS.

Notable implementations:

- → Greenplum Orca
- → Apache Calcite



File Formats

Most DBMSs use a proprietary on-disk binary file format for their databases. The only way to share data between systems is to convert data into a common text-based format

→ Examples: CSV, JSON, XML

There are open-source binary file formats that make it easier to access data across systems and libraries for extracting data from files.

→ Libraries provide an iterator interface to retrieve (batched) columns from files.

Storage Formats

Apache Parquet (2013)

→ Compressed columnar storage from Cloudera/Twitter

Apache ORC (2013)

→ Compressed columnar storage from Apache Hive.

Apache CarbonData (2013)

→ Compressed columnar storage with indexes from Huawei.

Apache Iceberg (2017)

→ Flexible data format that supports schema evolution from Netflix.

HDF5 (1998)

→ Multi-dimensional arrays for scientific workloads.

Apache Arrow (2016)

→ In-memory compressed columnar storage from Pandas / Dremio.

Execution Engines

Standalone libraries for executing vectorized query operators on columnar data.

- → Input is a DAG of physical operators.
- → Require external scheduling and orchestration.

Notable implementations:

- → <u>Velox</u>
- → DataFusion
- → Intel OAP

Conclusion

Today was about understanding the high-level history and concept of modern OLAP DBMSs.

→ Fundamentally, they are not very different from previous distributed / parallel DBMSs <u>except</u> for their use of cloud-based object stores.

Our focus for the rest of the quarter will be on state-of-the-art implementations of the various system components.

Next Class

Data Storage