CS 4440 A

Emerging Database Technologies

Lecture 18

03/18/26

Agenda

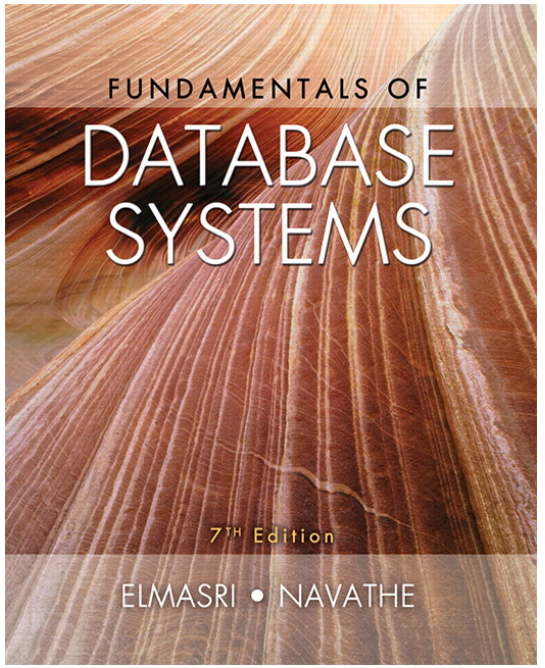
1. OLTP vs OLAP

2. A brief intro to HTAP Databases

Reading Materials

Fundamental of Database Systems (7th Edition)

- Chapter 29 - Overview of Data Warehousing and OLAP



1. OLTP vs OLAP

So far we've been dealing with OLTP

OLTP: OnLine Transactional Processing

- Often used to store and manage relevant data to the day-to-day operations of a system or company
  - e.g., ATM transactions, online hotel bookings
- INSERT, UPDATE, DELETE commands
- Handles real-time transactions (response times often in milliseconds)
- ACID properties are often important

This is where relational databases shine!

Ok, so what's OLAP?

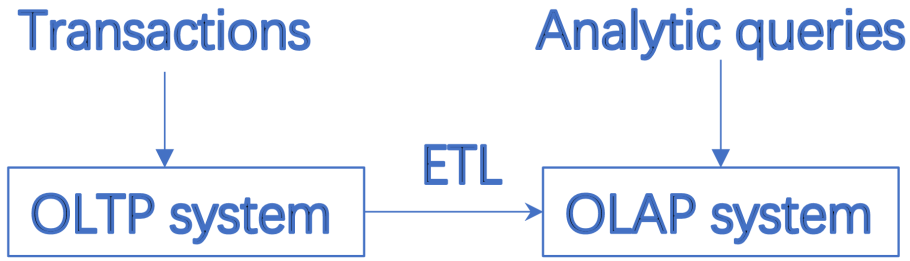
OLAP: OnLine Analytical Processing

- Also known as decision support or business intelligence (BI), but now BI has grown to include more (e.g., AI)
- A specialization of relational databases that prioritizes the reading and summarizing large volumes (TB, PB) of relational data to understand high-level trends and patterns
  - E.g., the total sales figures of each type of Honda car over time for each county
- "Read-only" queries

Contrast this to OLTP

- "Read-write" queries
- Usually touch a small amount of data
  - e.g., append a new car sale into the sales table

How is OLAP Performed?

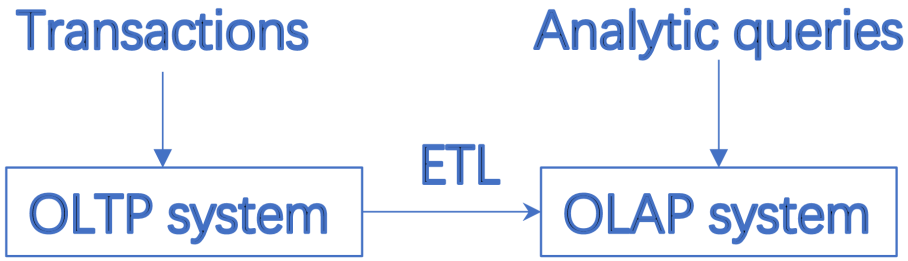


Usually, OLAP is performed on a separate data warehouse away from the critical path of OLTP transactions (a live/transactional database).

This data warehouse is periodically updated with data from various sources (e.g., once an hour or once a day)

- This is through a process of ETL (Extract, Transform, Load)
  - Extract useful business that needs to be summarized, transform it (e.g., canonicalize values, clean it up), load it in the data warehouse
- By doing it periodically, this data warehouse can become stale

How is OLAP Performed?



Usually, OLAP is performed on a separate data warehouse away from the critical path of OLTP transactions (a live/transactional database).

Why?

- Because OLAP queries end up reading most of the data, and will prevent OLTP queries from taking precedence
  - it is more important to ensure that sales are not prevented than to make sure that a report for a manager is generated promptly
  - the latter will anyway take a long time, so might as well have them wait a bit longer
- It's OK if the warehouse data is a bit stale.

OLAP in Data Warehouses

Here are some data warehouses that you might have heard of

Data Warehouse vs Data Lake

Data warehouse:

- structured data (schema-on-write)
- expensive for large data volumes
- managers and business analysts

Image source: https://www.sap.com/insights/what-is-a-data-lake.html

Data Lake vs Data Warehouse

Data lake:

raw data, can be unstructured (schema-on-read)

low-cost storage, but no transactions, data quality checks

data scientists and engineers

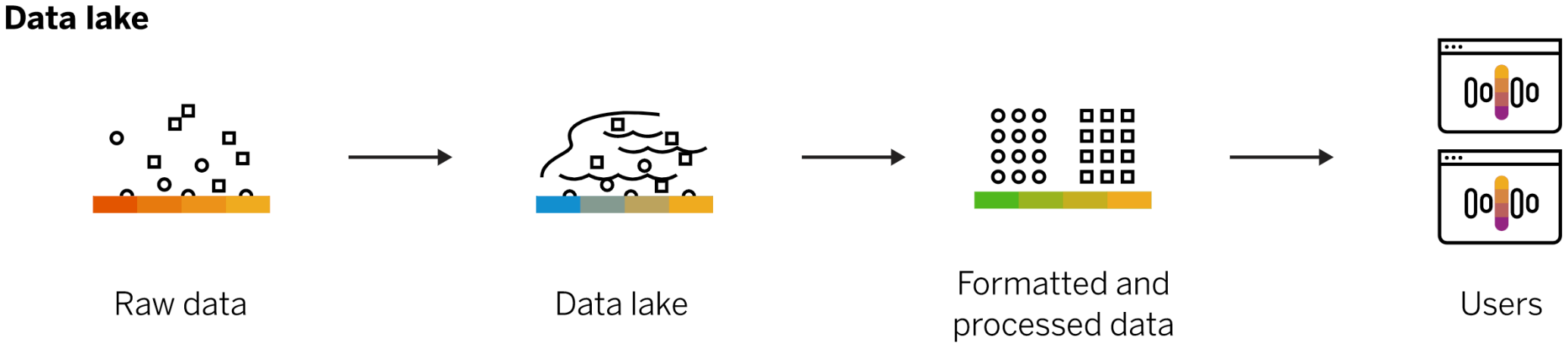


Image source: https://www.sap.com/insights/what-is-a-data-lake.html

We will focus on the following three aspects of OLAP systems

#1 Data Model

- Relational vs multi-dimensional schema

#2 Storage Model

- NSM vs DSM

#3 Query Processing Model

- Iterator vs Materialization vs Vectorized Model

#1 Data Model: Multi-dimensional Model

The multi-dimensional data model includes two types of tables:

- Fact table
  - Contain the actual metrics or measurements of business processes. Store quantitative data (numbers that can be aggregated).
    - E.g., Sales amount, Quantity sold
  - Typically has many rows but fewer columns
  - Usually contains foreign keys that link to dimension tables
- Dimension table
  - Contain descriptive attributes that provide context to facts
  - Store qualitative data
    - E.g., Product dimension (name, category, brand), Time dimension (date, month, year)

Multi-dimensional Schemas

Star schema:

- Consists of a fact table with a single table for each dimension.

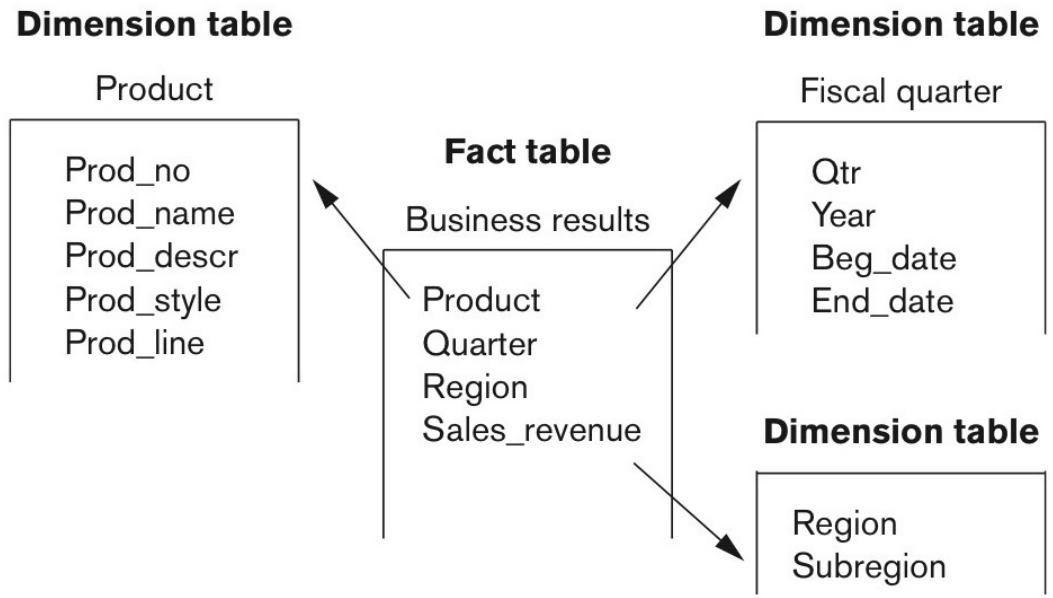


Figure 29.7 A star schema with fact and dimensional tables.

Multi-dimensional Schemas

Snowflake Schema:

- It is a variation of star schema, in which the dimensional tables from a star schema are normalized to eliminate redundancy

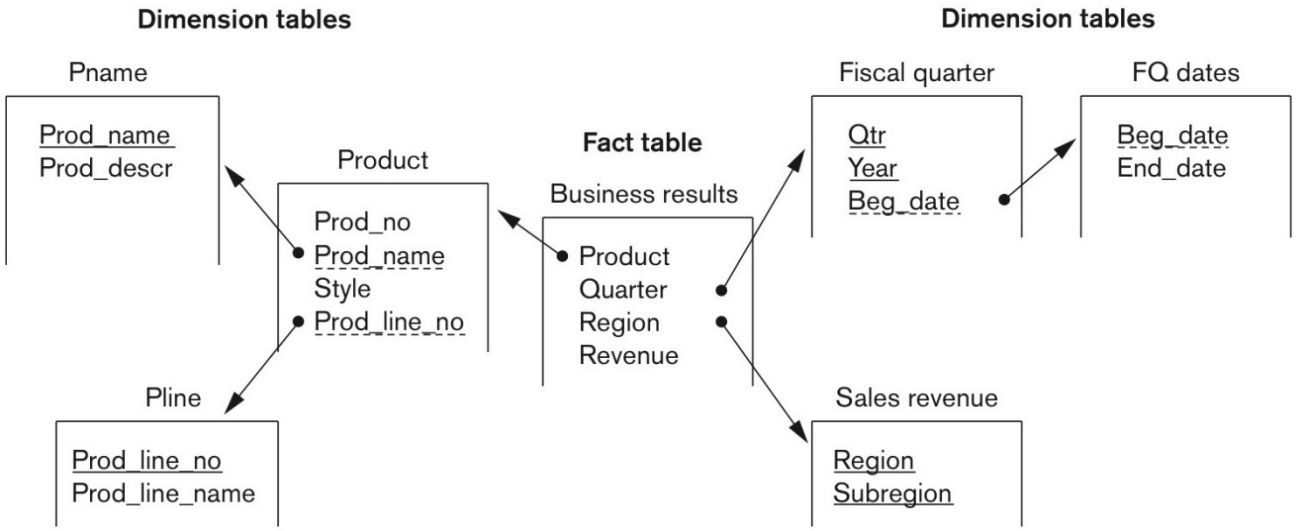


Figure 29.8 A snowflake schema.

Comparison with the Relational Model

The relational model (used by OLTP) keeps tables normalized

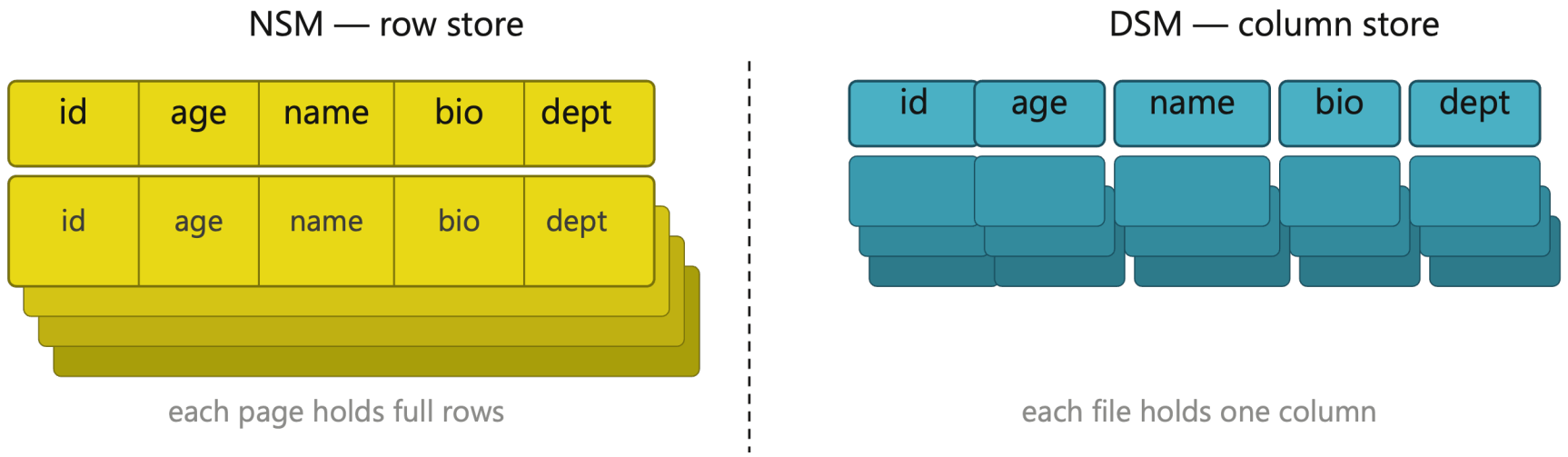
- Minimal updates required for data inserts and deletes => help improve transaction performance

In the multi-dimensional model, the fact table is often in 3NF, but the dimensional tables are denormalized.

- This means columns in a table contains data which is repeated throughout the table => help improve read performance.
- Data is stored in fewer tables, which removes the overhead of having to perform complex joins => helps improve read performance.

#2 Storage Model: NSM vs DSM

A DBMS's **storage model** specifies how it physically organizes tuples on disk and in memory.

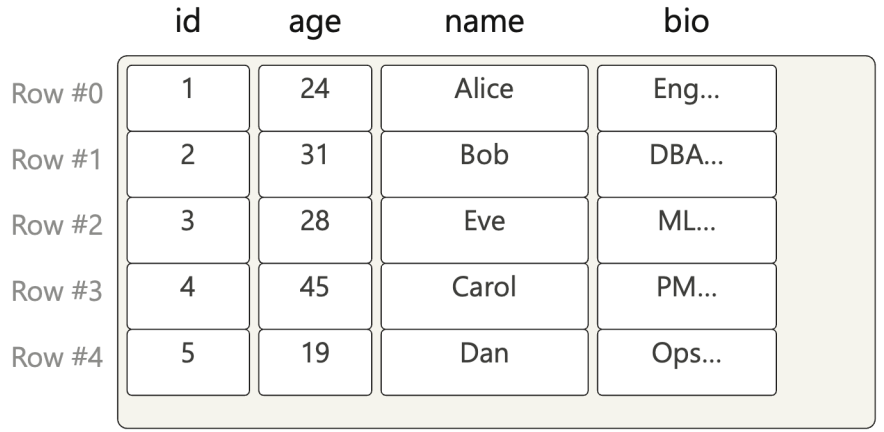


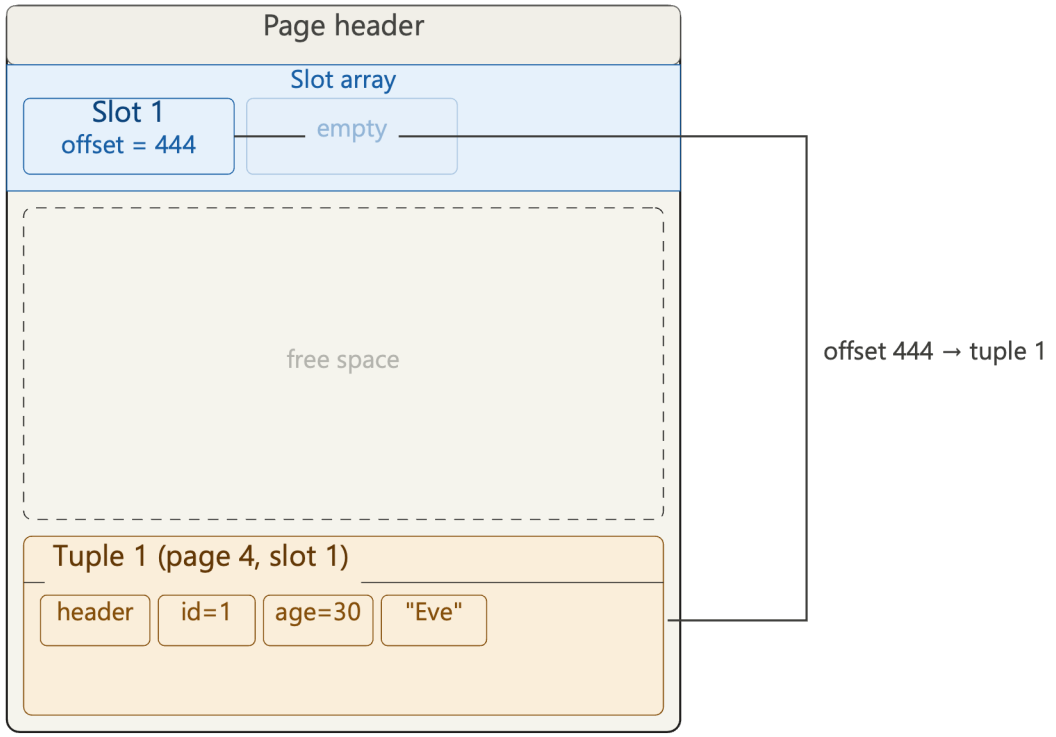
Q: Which format do you think is better suited for OLTP vs OLAP?

N-ary Storage Model (NSM)

- The DBMS stores (almost) all the attributes for a single tuple contiguously in a single page.
  - Record organization: fixed length before variable length attributes

- The tuple's record id (page#, slot#) is how the DBMS uniquely identifies a physical tuple.



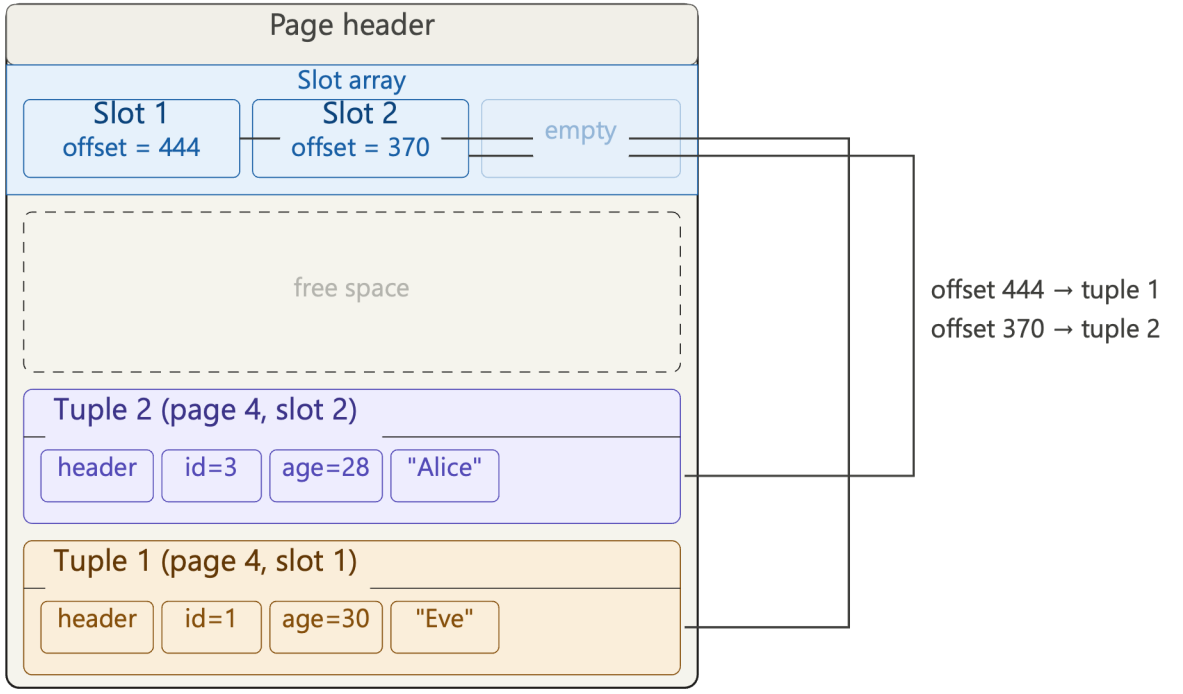


Slides adapted from CMU CS 15-721 by Andy Pavlo

N-ary Storage Model (NSM)

- The DBMS stores (almost) all the attributes for a single tuple contiguously in a single page.
  - Record organization: fixed length before variable length attributes

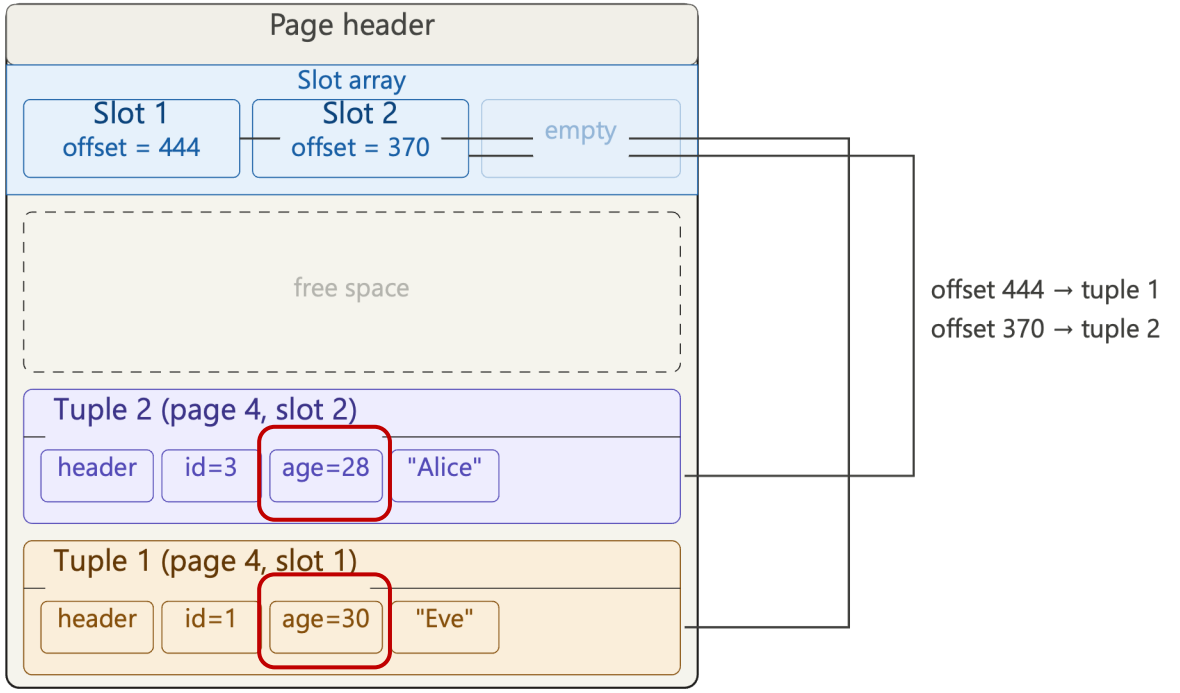
- The tuple's record id (page#, slot#) is how the DBMS uniquely identifies a physical tuple.



N-ary Storage Model (NSM)

	id	age	name	bio
Row #0	1	24	Alice	Eng…
Row #1	2	31	Bob	DBA…
Row #2	3	28	Eve	ML…
Row #3	4	45	Carol	PM…
Row #4	5	19	Dan	Ops…

```
SELECT …
FROM …
WHERE age < 30;
```



Slides adapted from CMU CS 15-721 by Andy Pavlo

N-ary Storage Model (NSM)

Advantages

- Fast inserts, updates, and deletes.
- Good for queries that need the entire tuple (OLTP).
- Can use index-oriented physical storage for clustering.

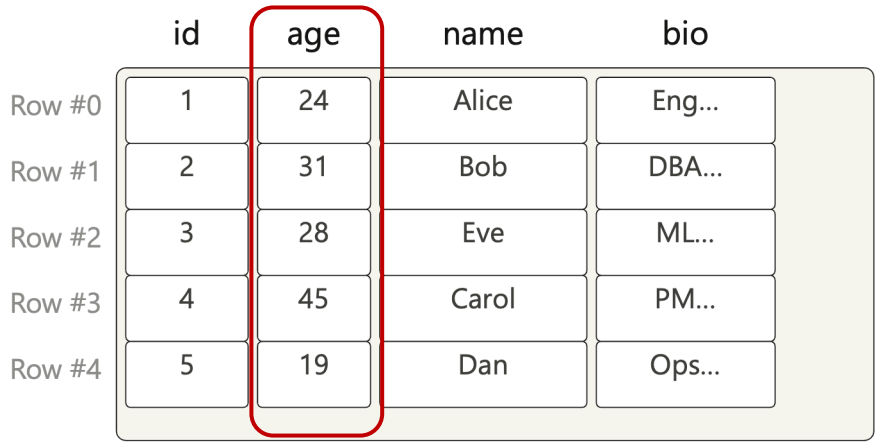
Disadvantages

- Not good for scanning large portions of the table and/or a subset of the attributes.
- Terrible memory locality in access patterns.
- Not ideal for compression because of multiple value domains within a single page.

Decomposition Storage Model (DSM)

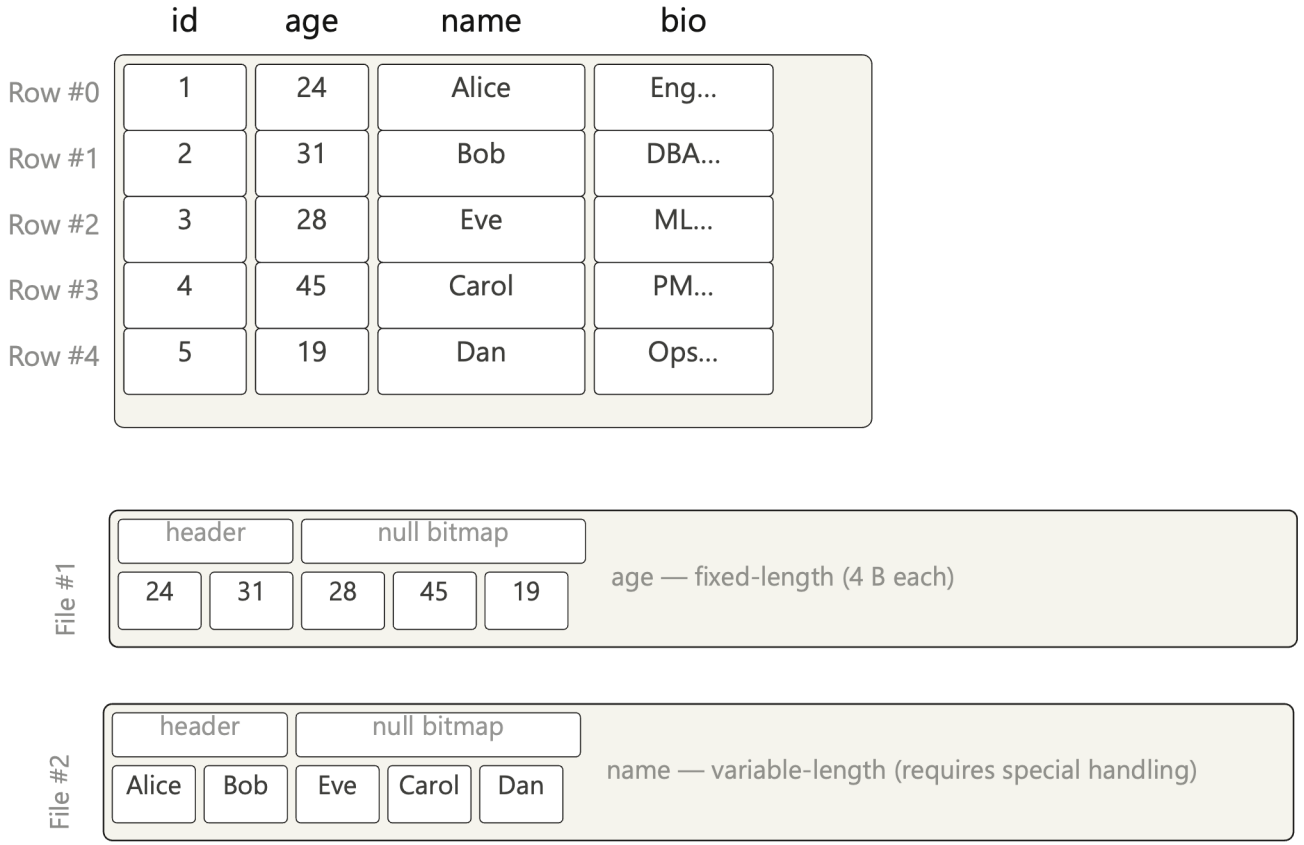
- The DBMS stores a single attribute for all tuples contiguously in a block of data

- Maintain a separate file per attribute with a dedicated header area for metadata about entire column.



Decomposition Storage Model (DSM)

- Store attributes and meta-data (e.g., nulls) in separate arrays of **fixed length** values.
  - Identify tuples using offsets into these arrays
  - Need to handle variable length attributes (e.g., padding, dictionary compression)



Slides adapted from CMU CS 15-721 by Andy Pavlo

Decomposition Storage Model (DSM)

Advantages

- Reduces the amount wasted I/O per query because the DBMS only reads the data that it needs.
- Faster query processing because of increased locality and cached data reuse.
- Better data compression

Disadvantages

- Slow for point queries, inserts, updates, and deletes because of tuple splitting/stitching/reorganization.

Observations

OLAP queries almost never access a single column in a table by itself

- At some point during query execution, the DBMS must get other columns and stitch the original tuple back together.

But we still need to store data in a columnar format to get the storage + execution benefits.

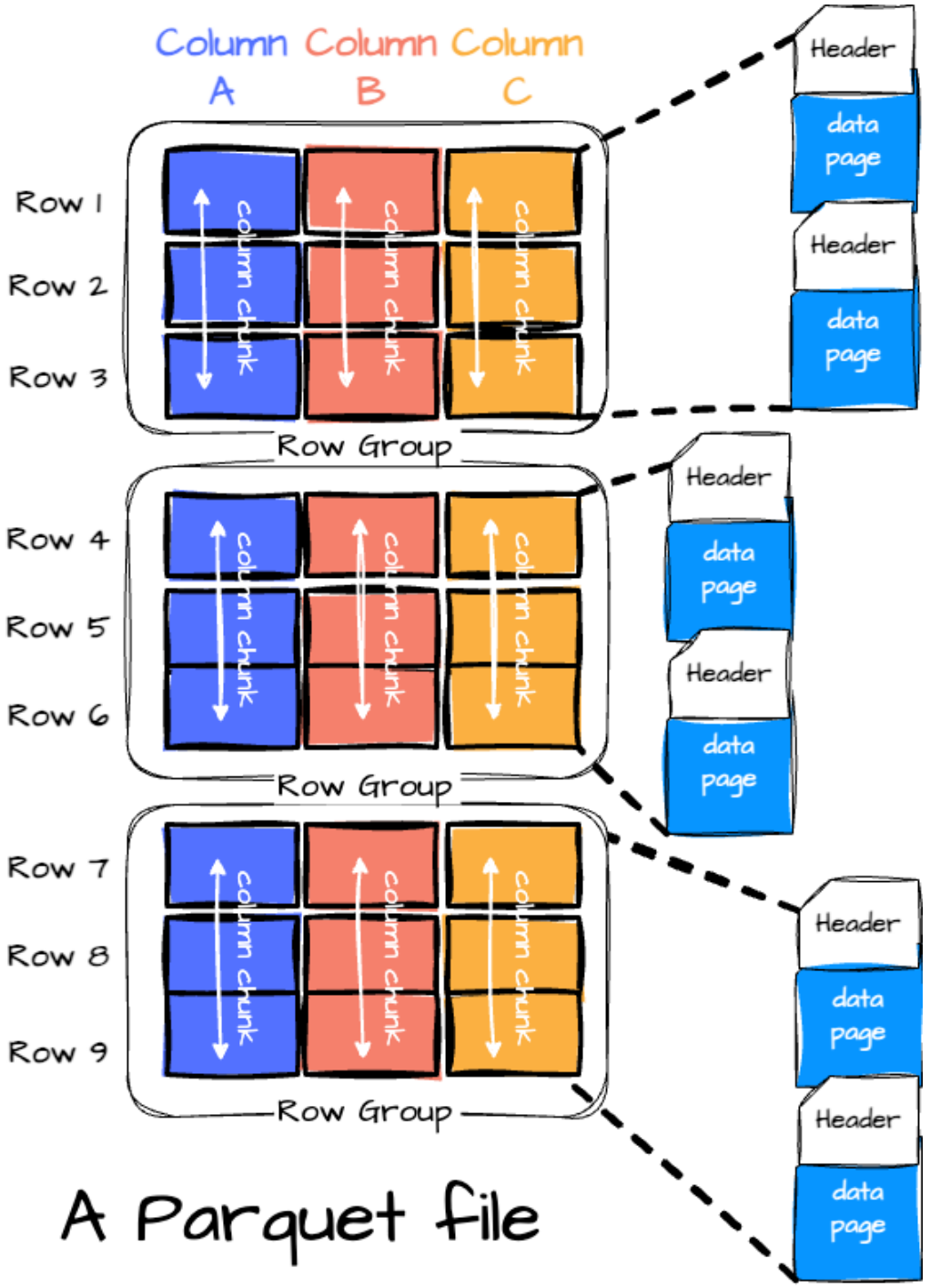
We need columnar scheme that still stores attributes separately but keeps the data for each tuple physically close to each other…

PAX Storage Model

Goal: Get the benefit of faster processing on columnar storage while retaining the spatial locality benefits of row storage

Example: Parquet format

- **Row Groups**: Horizontally partition rows into groups
- **Column chunk**: Data for a single column within a row group, stored in a compressed and encoded format.



#3 Query Processing

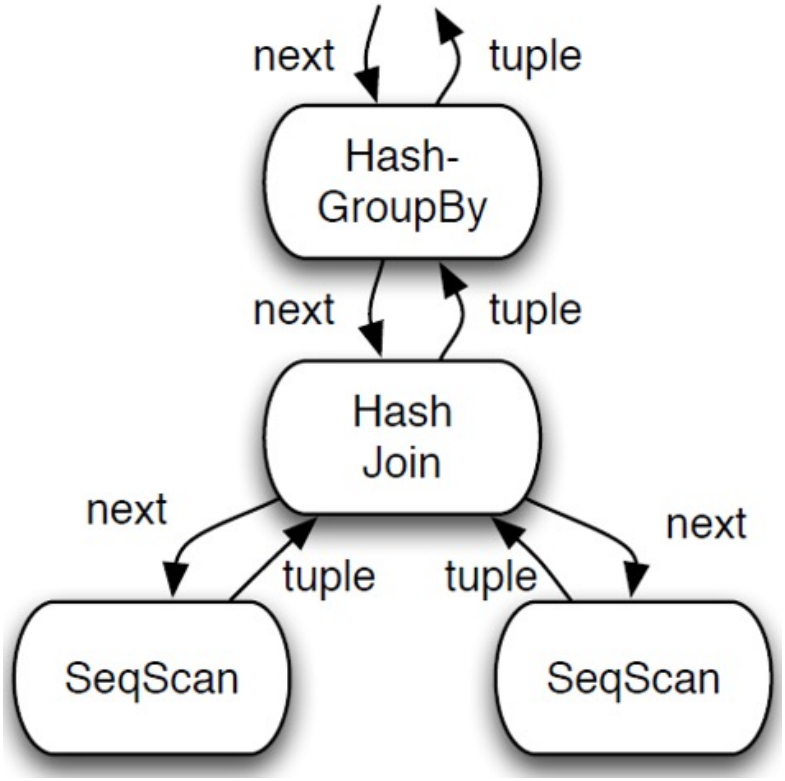
A DBMS's processing model defines how the system executes a query plan.

- Different trade-offs for different workloads.

- Approach 1: Iterator Model
- Approach 2: Materialization Model
- Approach 2: Vectorized / Batch Model

Approach 1: Iterator Model

- Each query plan operator implements a `next()` function.
  - On each invocation, the operator returns either a single tuple or a null marker if there are no more tuples.
  - The operator implements a loop that calls next on its children to retrieve their tuples and then process them.
  - Works well with NSM



- Also called **volcano** or **pipeline** model.

Image source: https://chatwithengineers.wordpress.com/2016/08/29/a-survey-of-query-execution-engines-from-volcano-to-vectorized-processing/

Approach 1: Iterator Model

- This is used in almost every DBMS. Allows for tuple pipelining.
- Output control (e.g., LIMIT) works easily with this approach.

- Some operators have to block until their children emit all of their tuples. These operators are known as pipeline breakers.
  - Joins, Subqueries, Order By

- Examples: SQLite, MySQL, PostgreSQL

Approach 2: Materialization Model

- Each operator processes its input all at once and then emits its output all at once.
  - The operator "materializes" its output as a single result.
  - The DBMS can push down hints into to avoid scanning too many tuples (e.g., LIMIT).
  - Can send either a materialized row or a single column.

- The output can be either whole tuples (NSM) or subsets of columns (DSM)

Approach 2: Materialization Model

- Better for OLTP workloads because queries only access a small number of tuples at a time.
  - Lower execution / coordination overhead.
  - Fewer function calls.

- Not good for OLAP queries with large intermediate results.

- Examples: MonetDB, VoltDB

Approach 3: Vectorization Model

- Like the Iterator Model where each operator implements a `next()` function in this model.

- Each operator emits **a batch of tuples** instead of a single tuple.
  - The operator's internal loop processes multiple tuples at a time.
  - The size of the batch can vary based on hardware or query properties.
  - Useful in in-memory DBMSs (due to fewer function calls)
  - Useful in disk-centric DBMSs (due to fewer IO operations)
  - Works well with DSM

Approach 3: Vectorization Model

- Ideal for OLAP queries because it greatly reduces the number of invocations per operator.
- Allows for operators to use vectorized (SIMD) instructions to process batches of tuples.

- Examples: Vectorwise, Snowflake, SQL Server, Oracle, Amazon RedShift

Assignment 4:
Postgres OLAP Extension

Assignment Overview

Goal: Implement PostgreSQL extensions to support vectorized query execution over Parquet files.

```
-- Install the wrapper
CREATE EXTENSION pgvparquet;

-- Define the external server
CREATE SERVER parquet_server FOREIGN DATA WRAPPER pgvparquet;

-- Define the foreign table
CREATE FOREIGN TABLE employees (
    id    int,
    age   int,
    name  text
)
SERVER parquet_server
OPTIONS (filename '/data/employees.parquet');
```

Two extensions:

- **pgvparquet:** reads Parquet files and exposes them to PostgreSQL
- **pgvrewrite:** transforms standard row-based execution plans into vectorized execution plans whenever possible.

Gap1: Storage Model

PostgreSQL (Row Store)

Stores all columns of a row together — great for INSERT/UPDATE (OLTP), wasteful for reading just a few columns across millions of rows.

Parquet (Columnar Format)

Stores each column as a contiguous, compressed buffer. Reading only the columns you need is fast.

Goal: read Parquet without immediately converting to PostgreSQL rows.

Gap1: Storage Model

- A `TupleTableSlot` is PostgreSQL's unit of data between plan nodes. Each slot holds one row.

- Our custom `ArrowTupleTableSlot` replaces that with an Arrow `RecordBatch` — thousands of rows stored column-by-column. Vectorized operators process the whole batch in one shot.

- Only at the devectorization boundary does the batch get unpacked into individual PostgreSQL tuples.

Gap 2: Execution Model

Volcano / Iterator Model

PostgreSQL's executor calls "give me the next row" one at a time. Every operator — filter, join, aggregate — processes a single tuple per call.

Vectorized Model

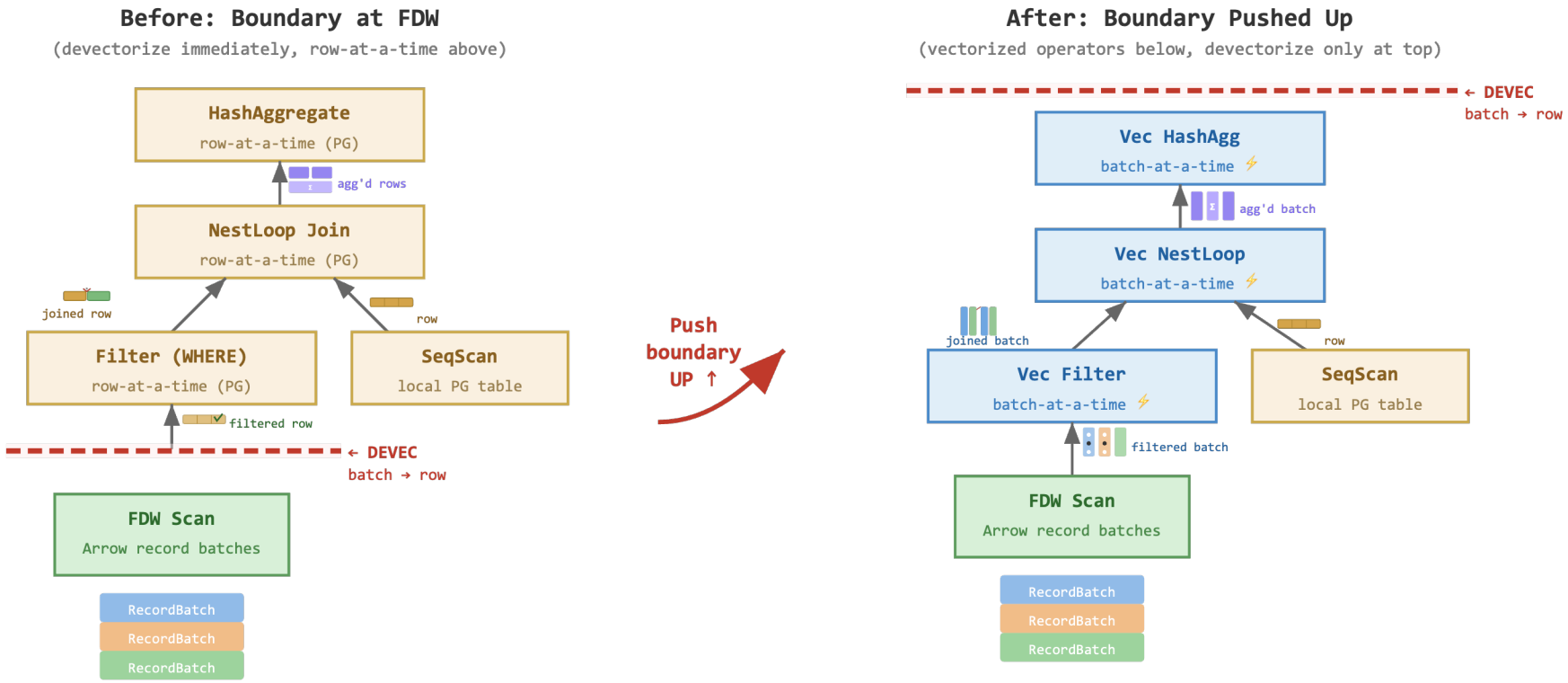
Process a batch of thousands of values at once using SIMD and columnar Arrow buffers.

Goal: rewrite query plans to push as much work as possible into batch-at-a-time operators.

Gap 2: Execution Model

Devectorization boundary: the point in the query plan where columnar batches are converted into PostgreSQL tuples

- the higher the boundary, the more work runs in batch mode



Misc

Grading:

- Correctness tests: Checks whether operators are implemented correctly
- Performance tests: Checks whether vectorized implementation outperforms Postgres baseline
- Extra credit for top 3 entries on the performance leaderboard

AI policy: Can use AI coding agents, but you are responsible for understanding the code you submit

Start early!

2. A brief intro to HTAP Databases

Slides adapted from *SIGMOD'22 Tutorial HTAP Databases: A Tutorial* by Guoliang Li and Chao Zhang

Observations

Data is "hot" when it enters the database

- A newly inserted tuple is more likely to be updated again the near future.

As a tuple ages, it is updated less frequently.

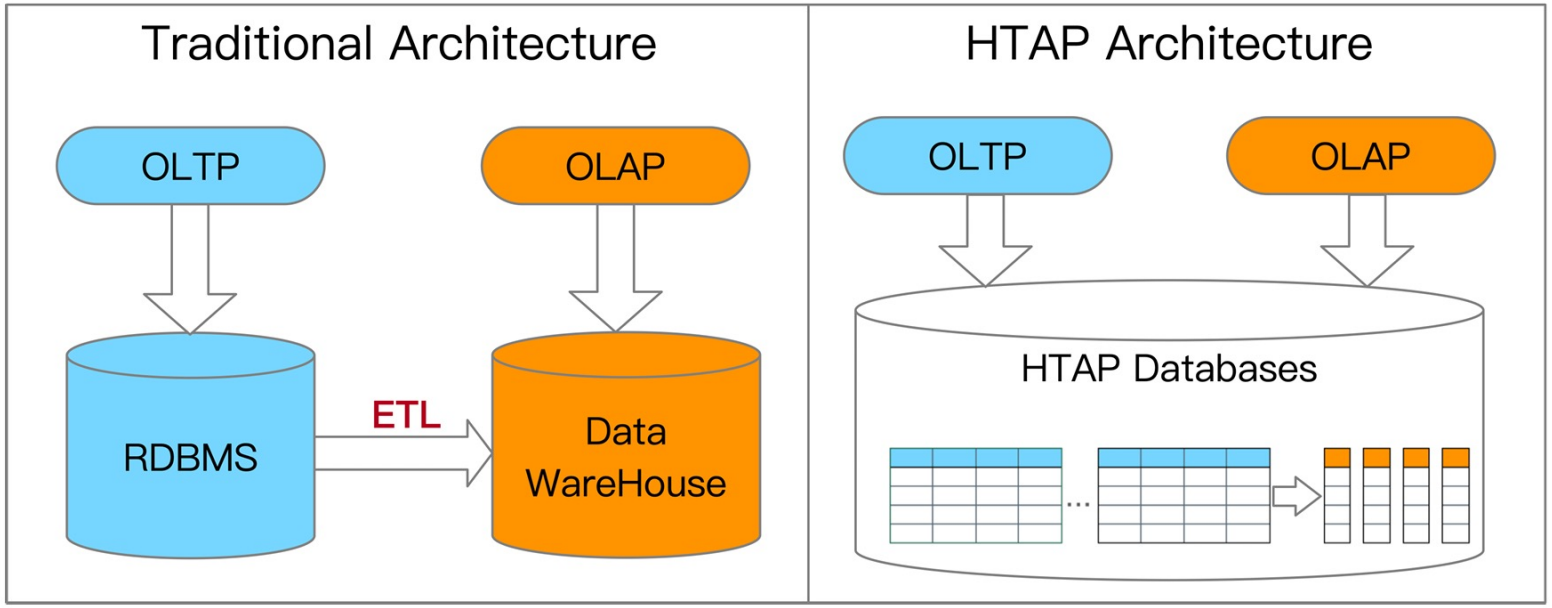
- At some point, a tuple is only accessed in read-only queries along with other tuples.

Many applications require a mix of transactional and analytical workloads

Motivation

HTAP: Hybrid Transaction Analytical Processing

- Gartner's definition in 2018: supports weaving analytical and transaction processing techniques together as needed to accomplish the business task.

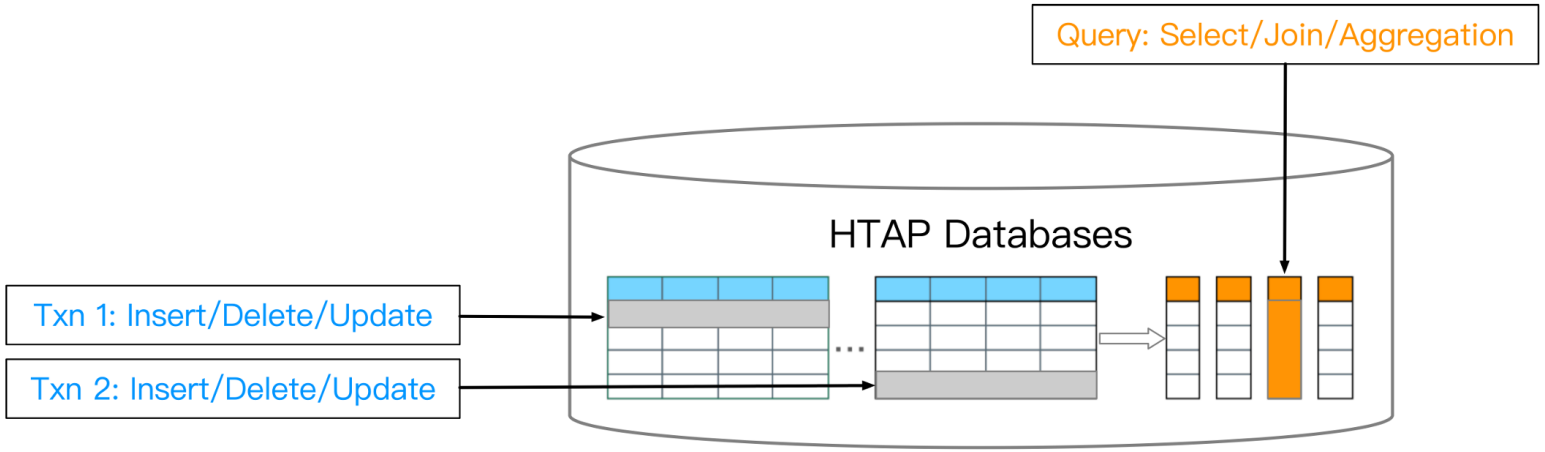


Slides Adapted from SIGMOD'22 Tutorial

Motivation

- Rule of thumb 1: row store is ideal for OLTP workloads
  - Row-wise, update-heavy, short-lived transactions

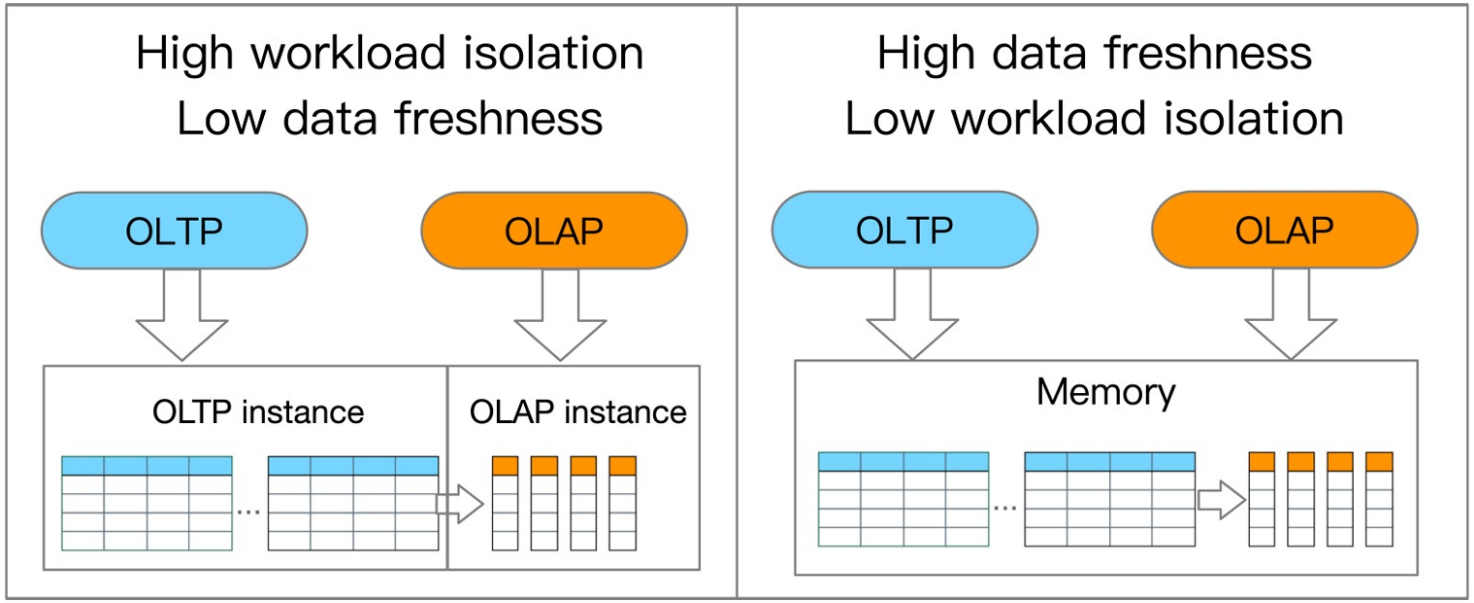
- Rule of thumb 2: column store is best suited for OLAP workload
  - Column-wise, read-heavy, bandwidth-intensive queries



Slides Adapted from SIGMOD'22 Tutorial

A trade-off for HTAP databases

- Workload isolation: the isolation level of handling the mixed workloads
- Data freshness: the portion of latest transaction data that is read by OLAP
- Trade-off for workload isolation and data freshness
  - High workload isolation leads to low data freshness
  - Low workload isolation results in high data freshness



Slides Adapted from SIGMOD'22 Tutorial

Example Approach: Delta Store

Incoming writes land in a small, **row-oriented delta store** — optimized for fast inserts and updates

The bulk of historical data sits in a **main column store** — optimized for analytical reads

- A background process periodically **migrates rows** from the delta into the main column store

Queries execute against both stores simultaneously and merge results at runtime

Examples: SAP HANA, Vertica, SingleStore, Databricks

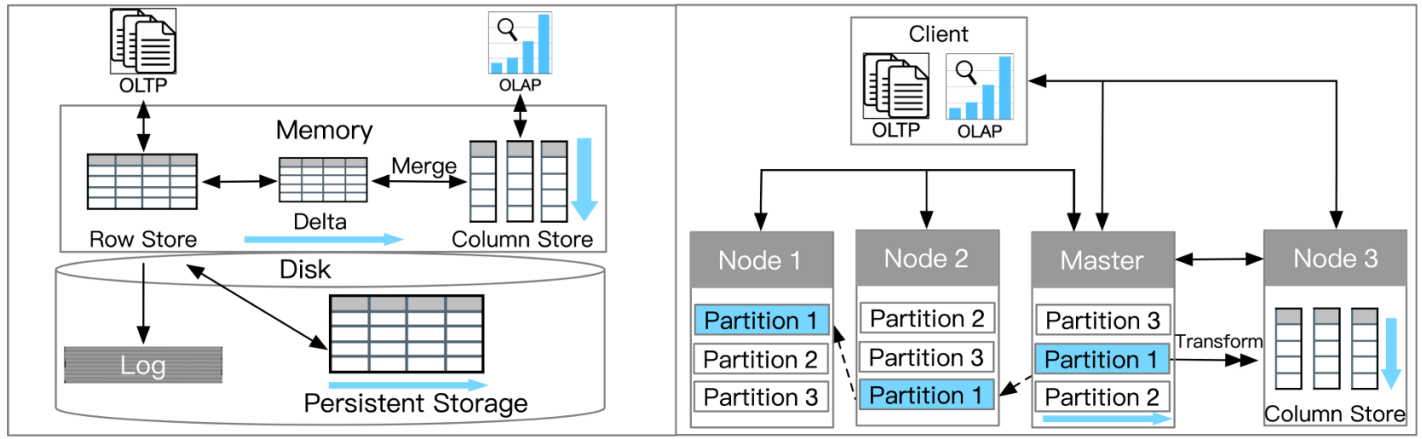
An Overview of HTAP Architectures

(a) **Primary Row Store + In-Memory Column Store**

(b) **Distributed Row Store + Column Store Replica**

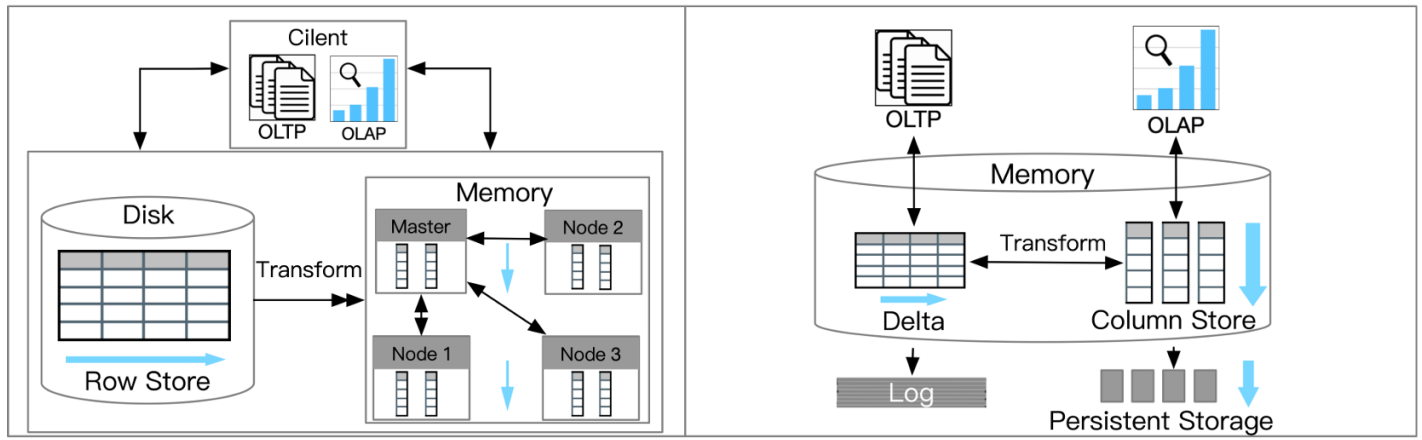
(c) **Disk Row Store + Distributed Column Store**

(d) **Primary Column Store + Delta Row Store**



(a) Primary Row Store+In–Memory Column Store

(b) Distributed Row Store+Column Store Replica



(c) Disk Row Store+Distributed Column Store

(d) Primary Column Store+Delta Row Store