CS 6400 A

Database Systems Concepts and Design

Lecture 21

11/11/26

Agenda

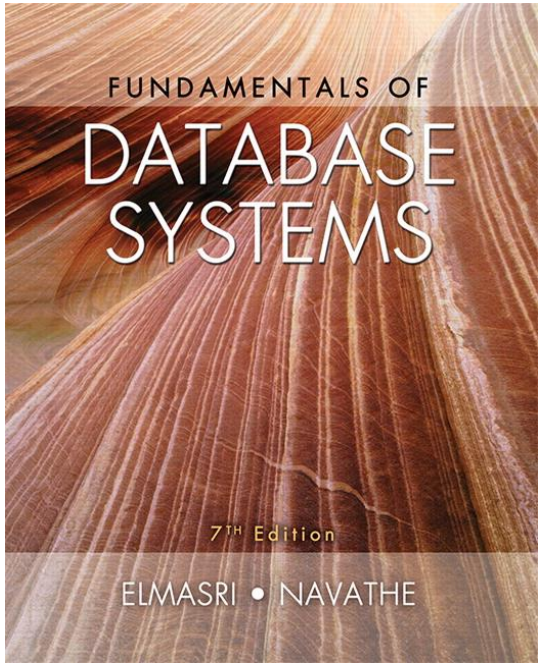
Comparison OLTP vs OLAP:

1. Data Model
2. Storage Format
3. Query processing

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?

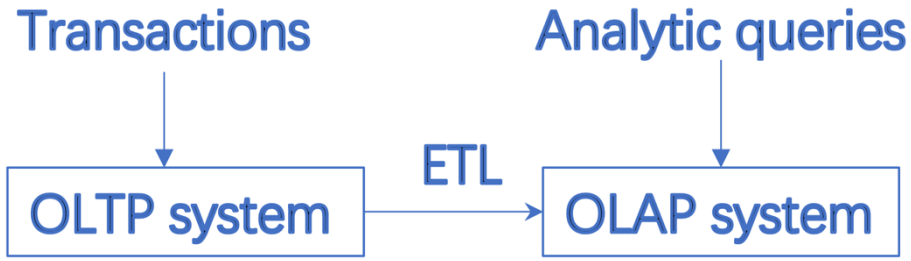
OLAP: OnLine Analytical Processing

- Also known as decision support or business intelligence (BI), but now BI has grown to include more (e.g., AI)
- OLAP focuses on analyzing and summarizing large volumes of data to identify 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?

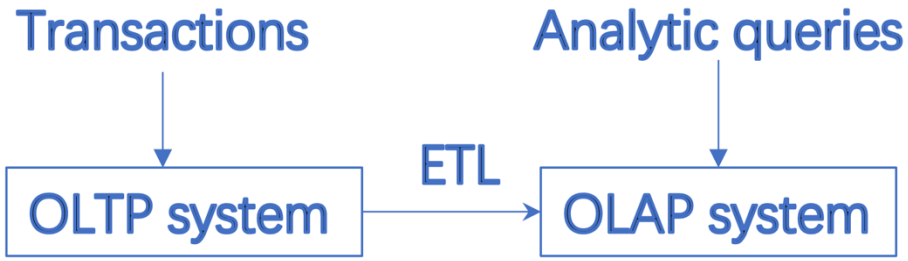


Usually, OLAP is performed on a separate data warehouse away from the critical path of OLTP transactions (a live/transactional database).

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

fewer built-in transaction and data-quality guarantees.

data scientists and engineers

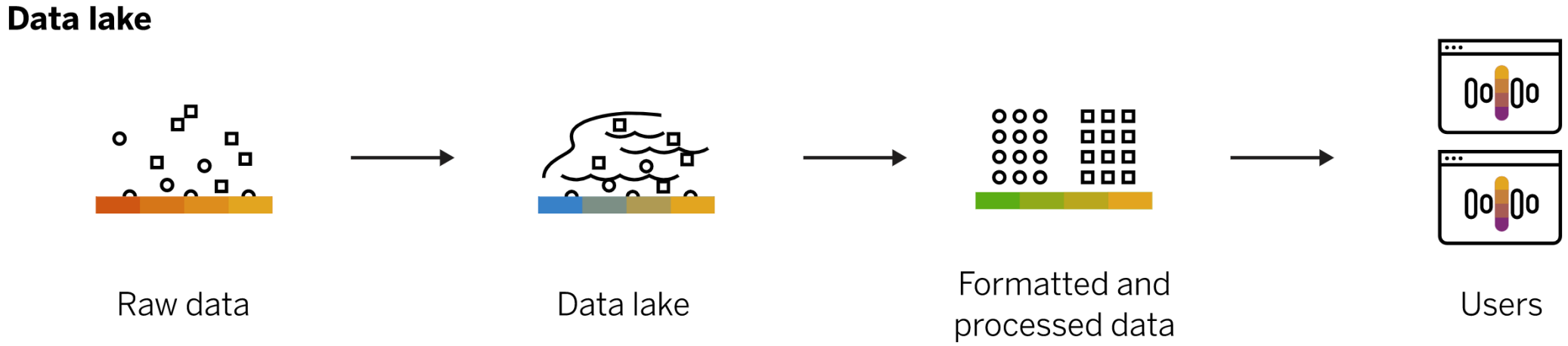


Image source: https://www.sap.com/insights/what-is-a-data-lake.html

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.

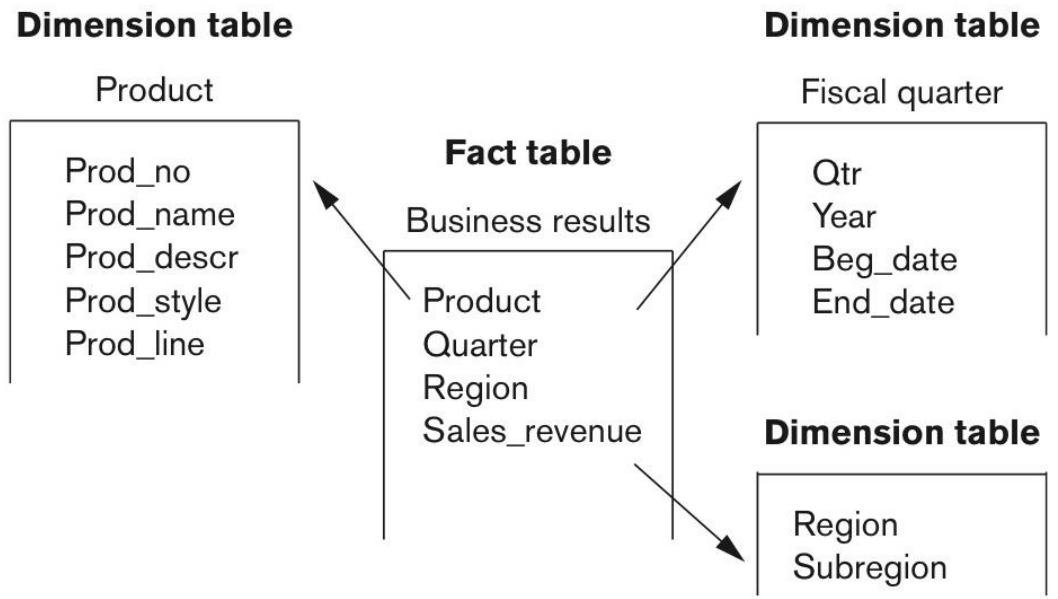


Figure 29.7 A star schema with fact and dimensional tables.

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

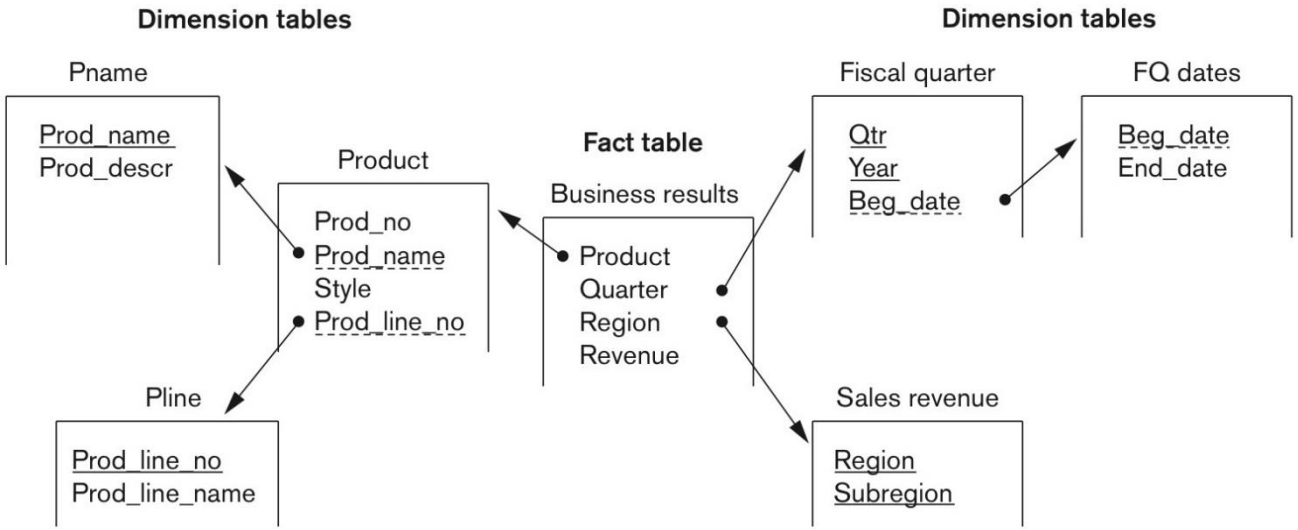


Figure 29.8 A snowflake schema.

Transactional vs. Dimensional Schema Design

The relational model (used by OLTP) keeps tables normalized

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

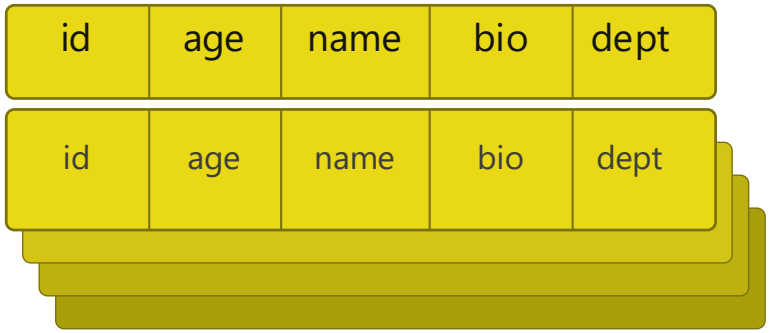
In the multi-dimensional model, the fact table is often in 3NF, but the dimensional tables are denormalized.

- 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

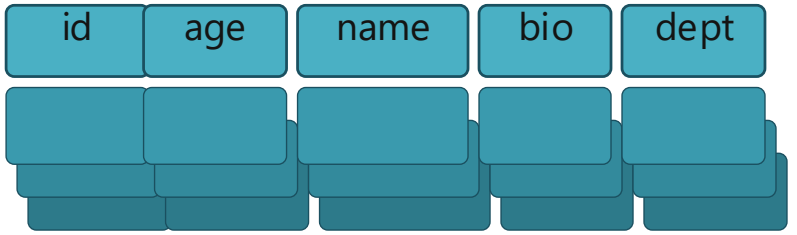
A DBMS's **storage model** specifies how it physically organizes tuples on disk and in memory.

NSM — row store



each page holds full rows

DSM — column store

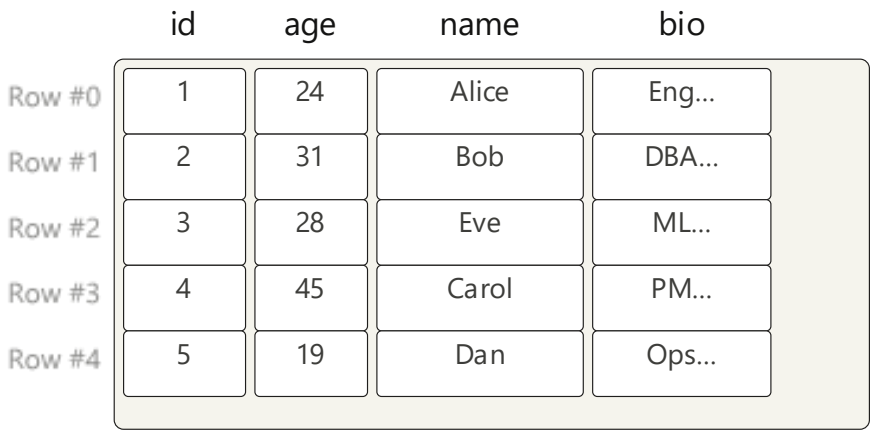


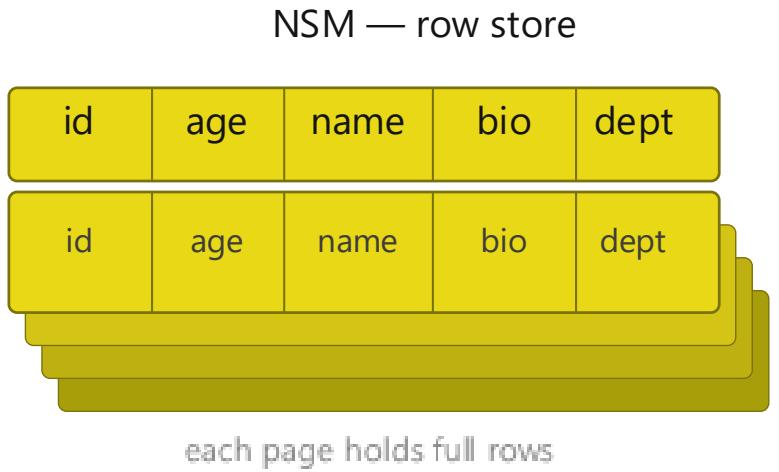
each file holds one column

N-ary Storage Model (NSM)

- Store all attributes of each tuple together within a database page.

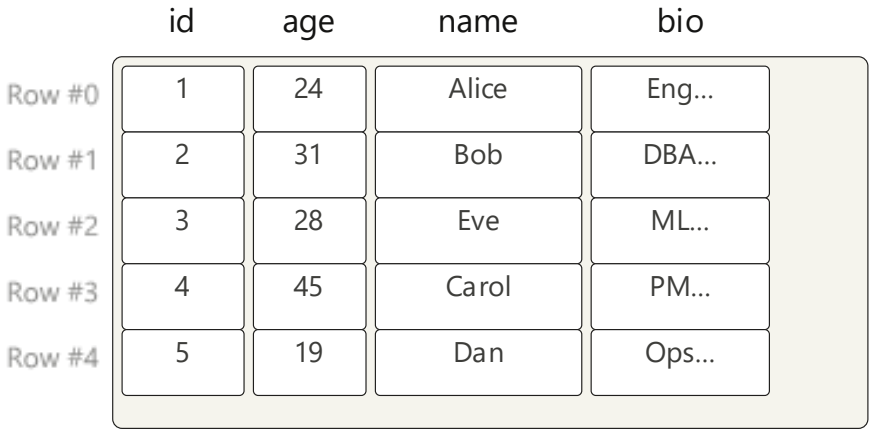
- Record organization (lec 9): fixed length before variable length attributes

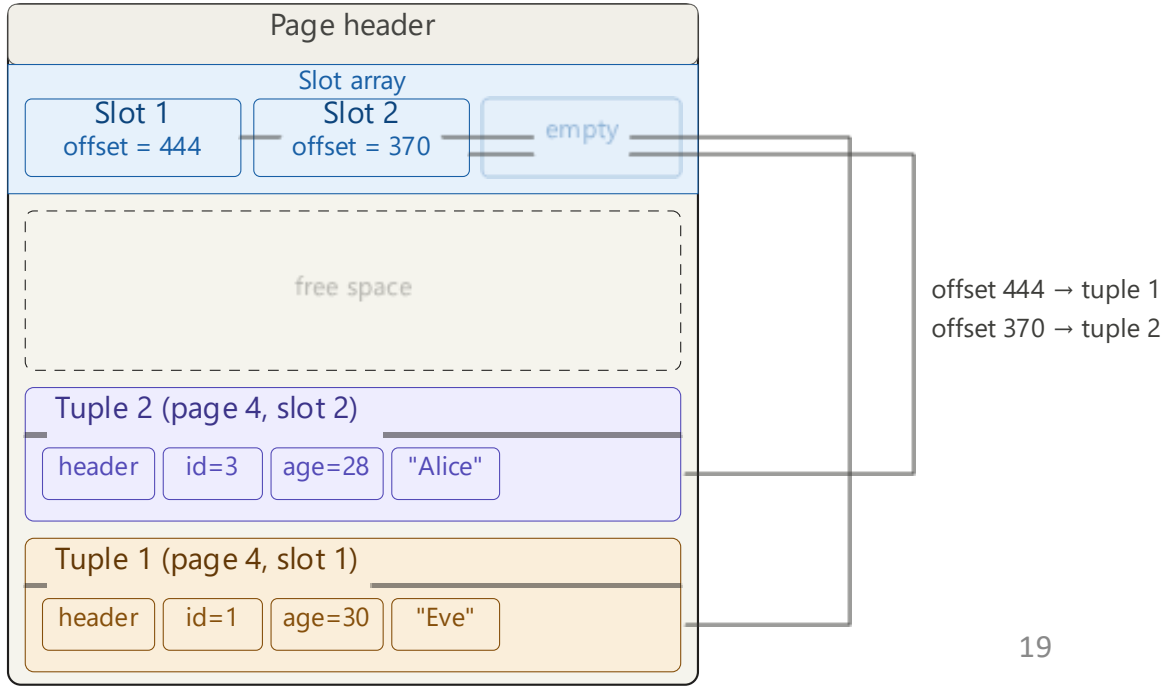




NSM: Page Organization

- Record ID (RID) = (page number, slot number) identifies a tuple's physical location
- The slot directory stores offsets to tuples within the page.

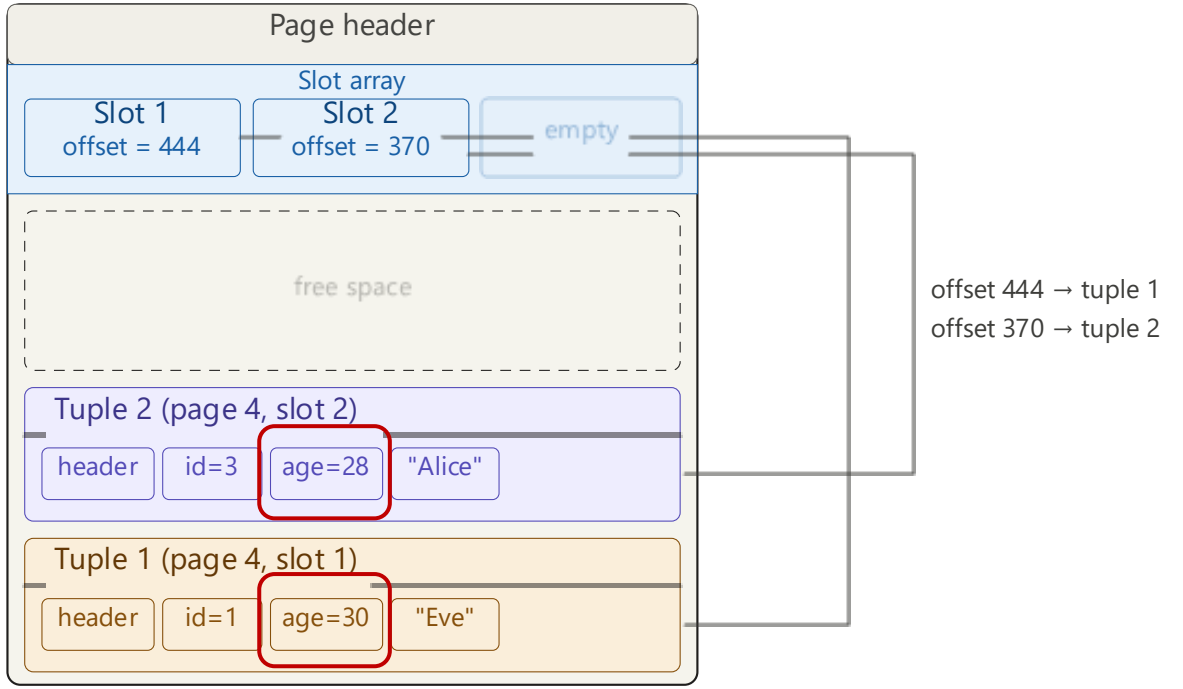




NSM: Query Execution

- To evaluate age < 30, locate each tuple using the slot directory and read its age attribute.

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



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

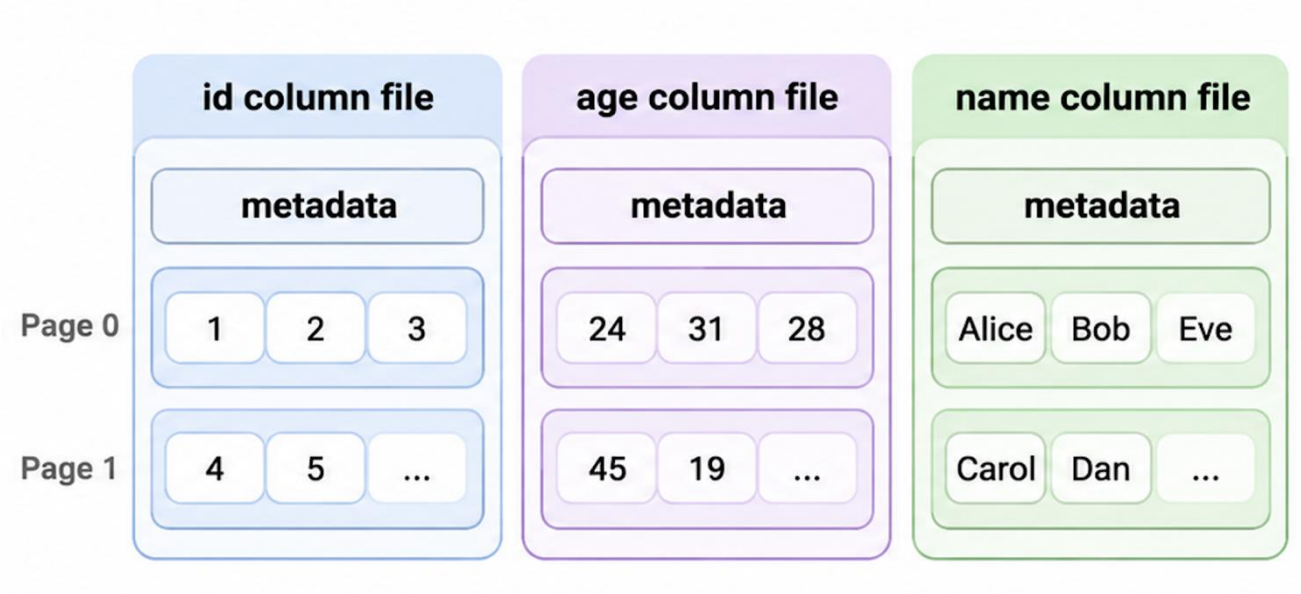
- Inefficient when queries scan many tuples but access only a few attributes (poor cache locality)
- Not ideal for compression because of multiple value domains within a single page.

Decomposition Storage Model (DSM)

- DSM stores values of each attribute together, rather than storing complete tuples together.

	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…

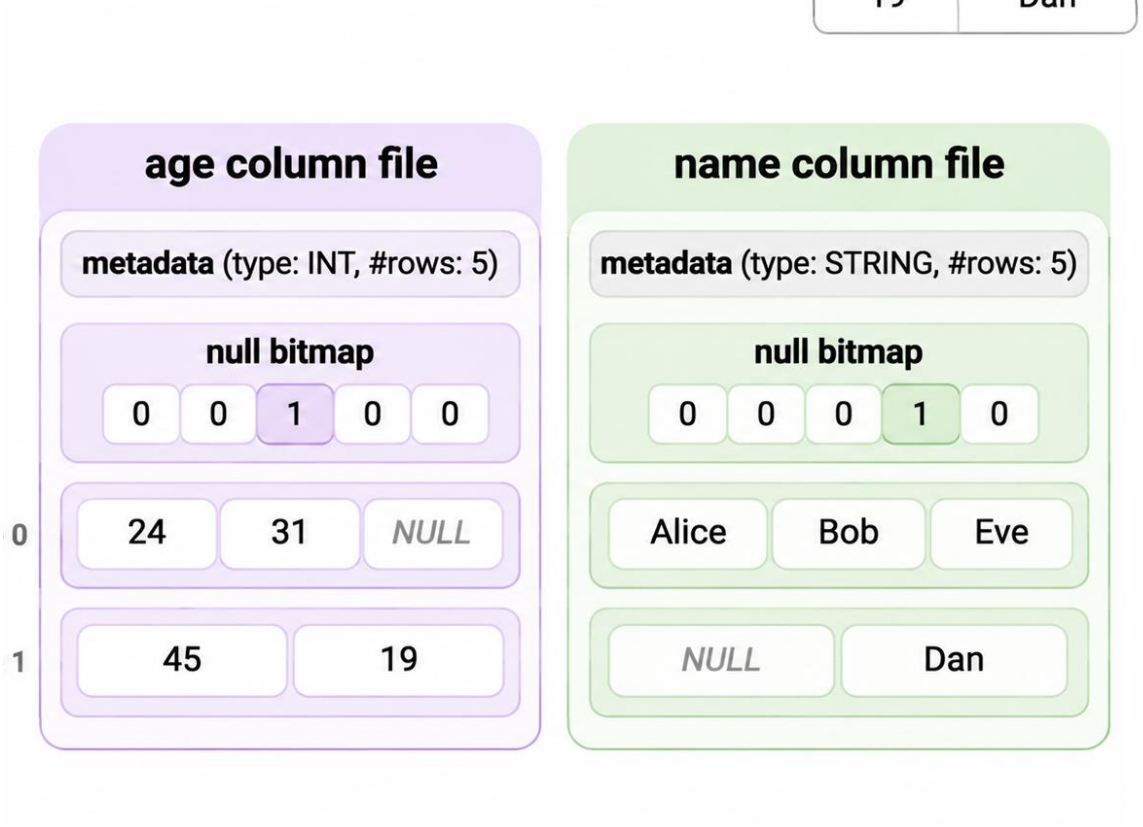
- Store each attribute separately, typically across multiple pages, with metadata describing the column.



Decomposition Storage Model (DSM)

age	name
24	Alice
31	Bob
NULL	Eve
45	*NULL*
19	Dan

- Store each attribute in a separate column-oriented array.
- Identify corresponding tuples using positions or row IDs.
- Handle different data types:
  - Fixed-length values: Direct access by offset (e.g., integers).
  - Variable-length values: Use offsets or dictionary encoding (e.g., strings).
  - NULLs: Use a bitmap or other validity representation.



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.

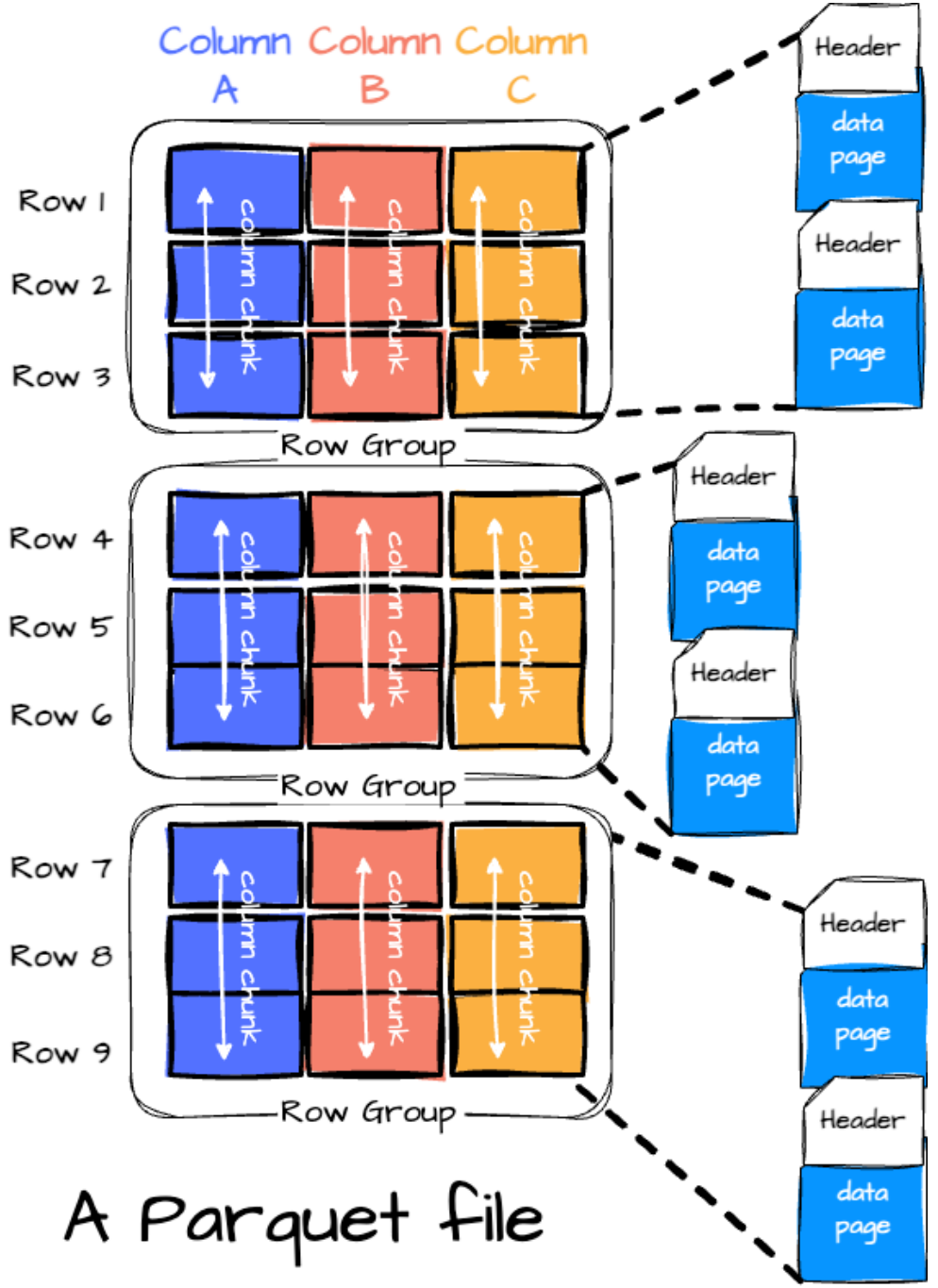
Beyond Pure Columnar Storage

- DSM stores each attribute separately, enabling efficient column scans.
- Queries often access multiple columns from the same rows.
  - At some point during query execution, the DBMS must get other columns and stitch the original tuple back together.

- Idea: Partition rows into groups, then store columns separately within each group.

Example: Parquet Format

- Partition rows into **row groups**.
- Within each row group, store each column separately in a **column chunk**.
- Encode and compress column values to reduce storage and I/O
- Use metadata to skip irrelevant row groups or pages.



#3 Query Processing

A query processing model determines how operators exchange intermediate results during execution.

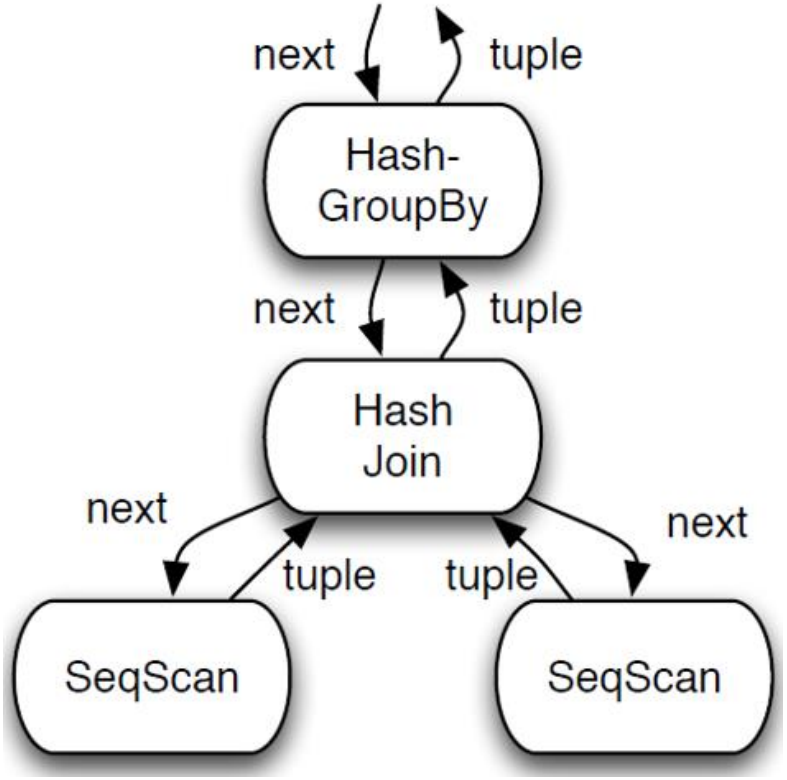
Different trade-offs for different workloads.

Three common models:

- Iterator: one tuple at a time.
- Materialization: entire intermediate result at a time.
- Vectorized: one batch at a time.

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

- Operators can produce and consume tuples incrementally, without materializing entire intermediate results.
- 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.
  - Sorting, hash aggregation etc.
- Limitation: Calling next() for every tuple introduces substantial per-tuple overhead.

- Examples: SQLite, MySQL, PostgreSQL

Approach 2: Materialization Model

- Each operator consumes its input and produces a complete intermediate result.
- The next operator processes that result after it is materialized.
- Intermediate results can contain full tuples or selected columns.

Approach 2: Materialization Model

Advantages

- Fewer per-tuple function calls.
- Efficient when intermediate results are small.
- Can work well for OLTP queries that access a small number of tuples.

Disadvantages

- Large intermediate results consume substantial memory.
- Operators cannot generally pipeline results tuple by tuple.
- Less suitable for OLAP queries that produce 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.
  - Batch size can be tuned for cache efficiency.
  - 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

Advantages

- Fewer per-tuple function calls.
- Better CPU cache locality.
- Opportunities for SIMD execution.
- Efficient for analytical queries that process many rows.

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

Summary: OLAP vs OLTP

Design dimension	Typical OLTP preference	Typical OLAP preference
Data model	Normalized schemas	Dimensional schemas
Storage	Row-oriented (NSM)	Column-oriented (DSM/PAX-like)
Query processing	Tuple-at-a-time often works well	Batch/vectorized execution
Workload	Small reads and updates	Large scans and aggregations

Database systems optimize data organization and query execution for different workload characteristics.