**Database Storage (Part II)**

1. **Data Representation**

* The data in a tuple is essentially just byte arrays.
* A data representation scheme is how a DBMS stores the bytes for a value.
* There are five high level datatypes that can be stored in tuples:
* Integers => INTEGER, BIGINT, SMALLINT, TINYINT.
* Variable Precision Numbers => FLOAT, REAL.
* Fixed-Point Precision Numbers => NUMERIC, DECIMAL.
* Variable-Length Data => VARCHAR, VARBINARY, TEXT, BLOB.
* Dates and Times => TIME, DATE, TIMESTAMP.

* System Catalogs : maintains an internal catalog to tell it meta-data about the databases. The meta-data will contain information about what tables and columns the databases have along with their types and the orderings of the values.

1. **Workloads**

* OLTP: Online Transaction Processing :
* is characterized by fast, short running operations, simple queries that operate on single entity at a time, and repetitive operations.
* typically handle more writes than reads.
* An example of an OLTP workload is the Amazon storefront. Users can add things to their cart, they can make purchases, but the actions only affect their account.
* OLAP: Online Analytical Processing :
* is characterized by long running, complex queries, reads on large portions of the database.
* s, the database system is analyzing and deriving new data from existing data collected on the OLTP side.
* An example of an OLAP workload would be Amazon computing the five most bought items over a one month period for these geographical locations.
* HTAP: Hybrid Transaction + Analytical Processing : tries to do OLTP and OLAP together on the same database.

1. **Storage Models**

**N-Ary Storage Model (NSM)**

In the n-ary storage model, the DBMS stores all of the attributes for a single tuple contiguously in a single page, so NSM is also known as a “row store.” This approach is ideal for OLTP workloads where requests are insert-heavy and transactions tend to operate only an individual entity. It is ideal because it takes only

one fetch to be able to get all of the attributes for a single tuple.

Advantages:

• Fast inserts, updates, and deletes.

• Good for queries that need the entire tuple.

Disadvantages:

• Not good for scanning large portions of the table and/or a subset of the attributes. This is because it pollutes the buffer pool by fetching data that is not needed for processing the query.

**Decomposition Storage Model (DSM)**

In the decomposition storage model, the DBMS stores a single attribute (column) for all tuples contiguously in a block of data. Thus, it is also known as a “column store.” This model is ideal for OLAP workloads with many read-only queries that perform large scans over a subset of the table’s attributes.

Advantages:

• Reduces the amount of wasted work during query execution because the DBMS only reads the data that it needs for that query.

• Enables better compression because all of the values for the same attribute are stored contiguously.

Disadvantages:

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

To put the tuples back together when using a column store, there are two common approaches: The most commonly used approach is fixed-length offsets. Assuming the attributes are all fixed-length, the DBMS can compute the offset of the attribute for each tuple. Then when the system wants the attribute for a specific tuple, it knows how to jump to that spot in the file from the offest. To accommodate the variable-length fields, the system can either pad fields so that they are all the same length or use a dictionary that takes a fixed-size integer and maps the integer to the value. A less common approach is to use embedded tuple ids. Here, for every attribute in the columns, the DBMS stores a tuple id (ex: a primary key) with it. The system then would also store a mapping to tell it how to jump to every attribute that has that id. Note that this method has a large storage overhead because it needs

to store a tuple id for every attribute entry.