Lecture #04: Database Storage (Part II)

15-445/645 Database Systems (Fall 2019)
https://15445.courses.cs.cmu.edu/fall2019/
Carnegie Mellon University
Prof. Andy Pavlo

1 Data Representation

Tuples' data is essentially just byte arrays. It is up to the DBMS to know how to interpret those bytes to derive the values for attributes. A *data representation* scheme is how a DBMS stores the bytes for a value.

There are four main types that can be stored in tuples: integers, variable precision numbers, fixed point precision numbers, variable length values, and dates/times.

Integers

- Most DBMSs store integers using their "native" C/C++ types as specified by the IEEE-754 standard. These values are fixed length.
- Examples: `INTEGER`, `BIGINT`, `SMALLINT`, `TINYINT`.

Variable Precision Numbers

- Inexact, variable-precision numeric type that uses the "native" C/C++ types specified by IEEE-754 standard. These values are also fixed length.
- Variable-precision numbers are faster to compute than arbitrary precision numbers because the CPU can execute instructions on them directly.
- Examples: `FLOAT`, `REAL`.

Fixed Point Precision Numbers

- These are numeric data types with arbitrary precision and scale. They are typically stored in exact, variable-length binary representation with additional meta-data that will tell the system things like where the decimal should be.
- These data types are used when rounding errors are unacceptable, but the DBMS pays a performance penalty to get this accuracy.
- Example: `NUMERIC`, `DECIMAL`.

Variable Length Data

- An array of bytes of arbitrary length.
- Has a header that keeps track of the length of the string to make it easy to jump to the next value.
- Most DBMSs do not allow a tuple to exceed the size of a single page, so they solve this issue by writing the value on an overflow page and have the tuple contain a reference to that page.
- Some systems will let you store these large values in an external file, and then the tuple will contain a pointer to that file. For example, if our database is storing photo information, we can store the photos in the external files rather than having them take up large amounts of space in the DBMS. One downside of this is that the DBMS cannot manipulate the contents of this file.
- Example: `VARCHAR`, `VARBINARY`, `TEXT`, `BLOB`.

Dates and Times

- Usually, these are represented as the number of (micro/milli)seconds since the unix epoch.
- Example: `TIME`, `DATE`, `TIMESTAMP`.

System Catalogs

In order for the DBMS to be able to read these values, it maintains an internal catalog to tell it meta-data about the databases. The meta-data will contain what tables and columns the databases have along with their types and the orderings of the values.

Most DBMSs store their catalog inside of themselves in the format that they use for their tables.

2 Workloads

OLTP: On-line Transaction Processing

- Fast, short running operations
- Queries operate on single entity at a time
- More writes than reads
- Repetitive operations
- Usually the kind of application that people build first
- Example: User invocations of Amazon. They can add things to their cart, they can make purchases, but the actions only affect their account.

OLAP: On-line Analyitical Processing

- Long running, more complex queries
- Reads large portions of the database
- Exploratory queries
- Deriving new data from data collected on the OLTP side
- Example: Compute the five most bought items over a one month period for these geographical locations.

3 Storage Models

There are different ways to store tuples in pages. We have assumed the **n-ary storage model** so far.

N-Ary Storage Model (NSM)

The DBMS stores all of the attributes for a single tuple contiguously, so NSM is also known as a "row store." This approach is ideal for OLTP workloads where transactions tend to operate only an individual entity and insert heavy workloads. 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.

There are two different ways to organize a NSM database:

- **Heap-Organized Tables:** Tuples are stored in blocks called a heap, and the heap does not necessarily define an order.
- **Index-Organized Tables:** Tuples are stored in the primary key index itself, but different from a clustered index.

Decomposition Storage Model (DSM)

The DBMS stores a single attribute (column) for all tuples contiguously in a block of data. Also known as a "column store." This model is ideal for OLAP workloads where read-only queries 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 we are using a column store, we can use:

- **Fixed-length offsets**: Start by assuming the attributes are all fixed-length. Then when the system wants the attribute for a specific tuple, it knows how to jump to that spot in the file. To accommodate the variable-length fields, the system can pad them so that they are all the same length, or you could use a dictionary that takes a fixed-size integer and maps the integer to the value.
- **Embedded Tuple Ids**: For every attribute in the columns, store the tuple id with it. The system would also need extra information to tell it how to jump to every attribute that has that id.

Most DBMSs use fixed-length offsets.

Row stores are usually better for OLTP, while column stores ar better for OLAP.