Lecture 5: Data Representation

Discussion

Deep learning job postings have collapsed in past 6 months

- Something I've learned: when non-engineers ask for an AI or ML implementation, they almost certainly don't understand the difference between that and an "algorithmic" solution.
- If you solve "trending products" by building a SQL statement that *e.g.*, selects items with the largest increase of purchases this month in comparison to the same month a year ago, that's still "AI" to them.
- Knowing this can save you a lot of wasted time.
- Any sufficiently misunderstood algorithm is indistinguishable from AI.

Discussion

Deep learning job postings have collapsed in past 6 months

- I've worked in lots of big corps as a consultant. Every one raced to harness the power of "big data" 7 years ago. They couldn't hire or spend money fast enough. And for their investment they (mostly) got nothing. The few that managed to bludgeon their map/reduce clusters in to submission and get actionable insights discovered... they paid more to get those insights than they were worth!
- I think this same thing is happening with ML. It was a hiring bonanza. Every big corp wanted to get an ML/AI strategy in place. They were forcing ML in to places it didn't (and may never) belong. This "recession" is mostly COVID related I think - but companies will discover that ML is (for the vast majority) a shiny object with no discernible ROI.
- Like Big Data, I think we'll see a few companies execute well and actually get some value, while most will just jump to the next shiny thing in a year or two.

Recap

Memory Mapping Files

`mmap()` is used to manage the virtual address space of a process.

- One use case for `mmap()` is to map the contents of a file into the virtual memory.
- `mmap()` can also be used to allocate memory by not associating it with a file.
- With `mmap()`, data migration is automatically done by the OS (**not** by the DBMS).
- The key limitation of `mmap()` is that it **does not** provide fine-grained control over when and which pages are moved from DRAM to SSD.
- We will learn about how to design a buffer manager that allows us to gain this control in a DBMS.

Disk Block Mapping

The units of database space allocation are **disk blocks, extents, and segments**.

- A disk block is the smallest unit of data used by a database.
- An extent is a logical unit of database storage space allocation made up of a number of **contiguous** disk blocks.
- A segment is made up of one or more extents (and is hence not always **contiguous** on disk).

System Catalog

- A DBMS stores **meta-data** about databases in its internal catalog.
- List of tables, columns, indexes, views
- Almost every DBMS stores their catalog as a **private database**.
- Specialized code for "bootstrapping" catalog tables.

Today's Agenda

- Data Representation
- Storage Models

Data Representation

Data Representation

- A catalog table contain the schema information about the user tables
- The DBMS uses this schema information to figure out the tuple's **data representation**.
- In this way, it interprets the tuple's bytes into a set of attributes (types and values).

Data Representation

- `INTEGER/BIGINT/SMALLINT/TINYINT`
  - C/C++ Representation
- `FLOAT/REAL` vs. `NUMERIC/DECIMAL`
  - IEEE-754 Standard / Fixed-point Decimals
- `VARCHAR/VARBINARY/TEXT/BLOB`
  - Header with length, followed by data bytes.
- `TIME/DATE/TIMESTAMP`
  - 32/64-bit integer of (micro)seconds since Unix epoch

Variable Precision Numbers

- Inexact, variable-precision numeric type that uses the "native" C/C++ types.
  - Examples: `FLOAT`, `REAL/DOUBLE`
- Store directly as specified by IEEE-754.
- Typically faster than arbitrary precision numbers but can have rounding errors. . .

Variable Precision Numbers

Rounding Example

```
include <stdio.h>

int main(int argc, char* argv[]) {
float x = 0.1;
float y = 0.2;
printf("x+y = %f\n", x+y);
printf("0.3 = %f\n", 0.3);
}
```

Output

```
x+y = 0.300000
0.3 = 0.300000
```

Variable Precision Numbers

Rounding Example

```
include <stdio.h>

int main(int argc, char* argv[]) {
float x = 0.1;
float y = 0.2;
printf("x+y = %.20f\n", x+y);
printf("0.3 = %.20f\n", 0.3);
}
```

Output

```
x+y = 0.30000001192092895508
0.3 = 0.29999999999999998890
```

Fixed Precision Numbers

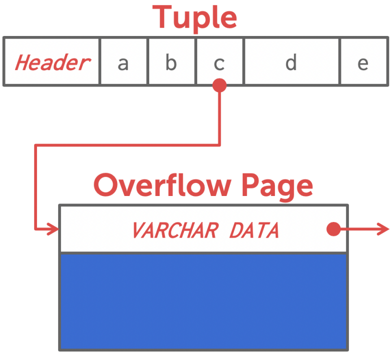
- Numeric data types with **arbitrary precision and scale**.
- Used when rounding errors are unacceptable.
  - Example: `NUMERIC`, `DECIMAL`
- Typically stored in a exact, variable-length binary representation with additional meta-data.
  - Like a `VARCHAR` but not stored as a string

Postgres: Numeric

```
typedef unsigned char NumericDigit;
typedef struct {
int ndigits;               // number of digits
int weight;                // weight of 1st Digit
int scale;                 // scale factor
int sign;                  // positive/negative/NaN
NumericDigit *digits;      // digit storage
} numeric;
```

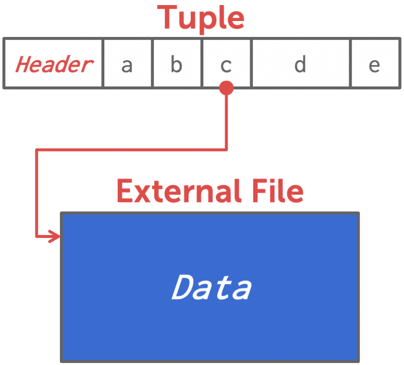
Large Values

- Most DBMSs don't allow a tuple to exceed the size of a single page.
- To store values that are larger than a page, the DBMS uses separate **overflow** storage pages.
  - Postgres: `TOAST` (>2KB)
  - MySQL: Overflow (>½ size of page)
  - SQL Server: Overflow (>size of page)



External Value Storage

- Some systems allow you to store a really large value in an external file. Treated as a `BLOB` type.
  - Oracle: `BFILE` data type
  - Microsoft: `FILESTREAM` data type
- The DBMS **cannot** manipulate the contents of an external file.
  - No durability guarantees.
  - No transaction protections.
- Objects < 256 KB are best stored in a database
- Objects > 1 MB are best stored in the filesystem
- Reference: To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?



Schema Changes: Index

- CREATE INDEX:
  - Scan the entire table and populate the index (*e.g.*, hash table: tuple id $\longrightarrow$ tuple pointer).
  - Have to record changes made by txns that modified the table while another txn was building the index.
  - When the scan completes, lock the table and resolve changes that were missed after the scan started.
- DROP INDEX:
  - Just drop the index logically from the catalog.
  - It only becomes "invisible" when the txn that dropped it **commits**.
  - All ongoing txns will still have to update it.

Observation

- The relational model does **not** specify that we have to store all of a tuple's attributes together in a single page.
- This may **not** actually be the best **storage layout** for some workloads. . .

Storage Models

Wikipedia Example

```
CREATE TABLE pages (
userID INT PRIMARY KEY,
userName VARCHAR UNIQUE,
);

CREATE TABLE pages (
pageID INT PRIMARY KEY,
title VARCHAR UNIQUE,
latest INT  REFERENCES revisions (revID),
);

CREATE TABLE revisions (
revID INT PRIMARY KEY,
userID INT REFERENCES useracct (userID),
pageID INT REFERENCES pages (pageID),
content TEXT,
updated DATETIME
);
```

OLTP Workload

On-line Transaction Processing (OLTP)

- Simple queries that read/update a small amount of data that is related to a single entity in the database.
- This is usually the kind of application that people build first.

```
SELECT * FROM useracct
WHERE userName = ? AND userPass = ?

UPDATE useracct
SET lastLogin = NOW(), hostname = ?
WHERE userID = ?

INSERT INTO revisions VALUES (?,?...,?)
```

On-line Transaction Processing (OLTP)

- Simple queries that read/update a small amount of data that is related to a single entity in the database.
- This is usually the kind of application that people build first.

```sql
SELECT P.*, R.*
FROM pages AS P INNER JOIN revisions AS R  ON P.latest = R.revID
WHERE P.pageID = ?
```

OLAP Workload

On-line Analytical Processing (OLAP)

- Complex queries that read large portions of the database spanning multiple entities.
- You execute these workloads on the data you have collected from your OLTP application(s).

```
SELECT P.*, R.*
FROM pages AS P INNER JOIN revisions AS R  ON P.latest = R.revID
WHERE P.pageID = ?
```

```
SELECT COUNT(U.lastLogin), EXTRACT(month FROM U.lastLogin) AS month
FROM useracct AS U
WHERE U.hostname LIKE '%.gov'
GROUP BY EXTRACT(month FROM U.lastLogin)
```

Workload Characterization

Workload	Operation Complexity	Workload Focus
OLTP	Simple	Writes
OLAP	Complex	Reads
HTAP	Medium	Mixture

Source

Workload Types

- OLTP: On-line Transaction Processing; Simple + Write-intensive
- OLAP: On-line Analytical Processing; Complex + Read-intensive
- HTAP: Hybrid Transaction/Analytical Processing; Medium + Mixed

Data Storage Models

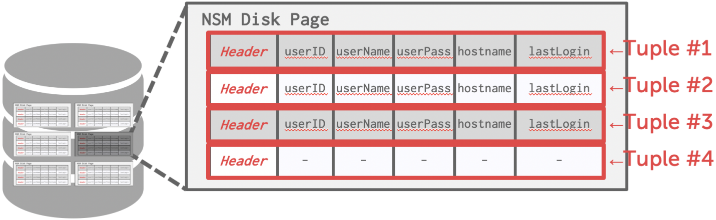
- The DBMS can store tuples in different ways that are better for either OLTP or OLAP workloads.
- We have been assuming the **n-ary storage model** (**NSM**) (*a.k.a.*, row storage) so far this semester.

N-ary Storage Model (NSM)

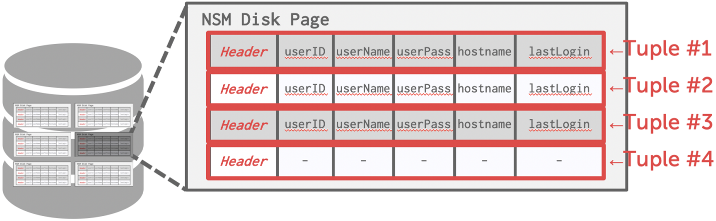
- The DBMS stores all attributes for a single tuple contiguously in a page.
- Ideal for OLTP workloads where queries tend to operate only on an individual entity and insert-heavy workloads.

N-ary Storage Model (NSM)

- The DBMS stores all attributes for a single tuple contiguously in a page.



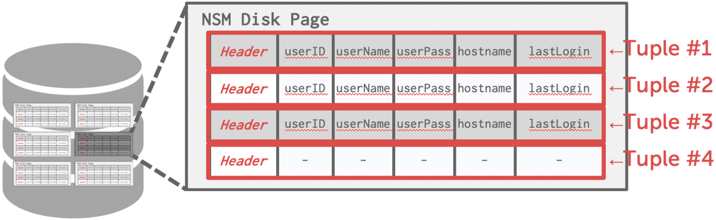
N-ary Storage Model (NSM)




```
SELECT * FROM useracct
WHERE userName = ? AND userPass = ?
```

Use index to access the particular user's tuple.

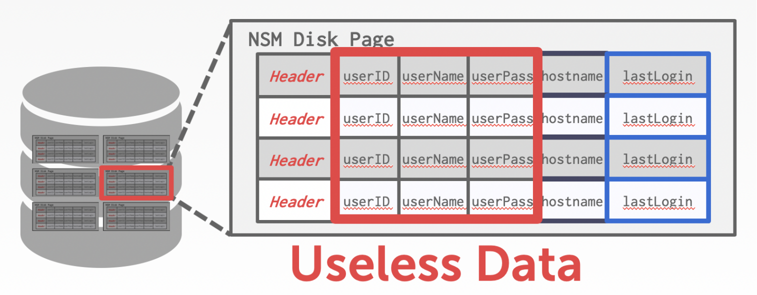
N-ary Storage Model (NSM)




```
INSERT INTO useracct VALUES (?,?,...?)
```

Add the user's tuple using `std::memcpy`.

N-ary Storage Model (NSM)




```sql
SELECT COUNT(U.lastLogin), EXTRACT(month FROM U.lastLogin) AS month
FROM useracct AS U
WHERE U.hostname LIKE '%.gov'
GROUP BY EXTRACT(month FROM U.lastLogin)
```

Useless data accessed for this query.

N-ary Storage Model (NSM)

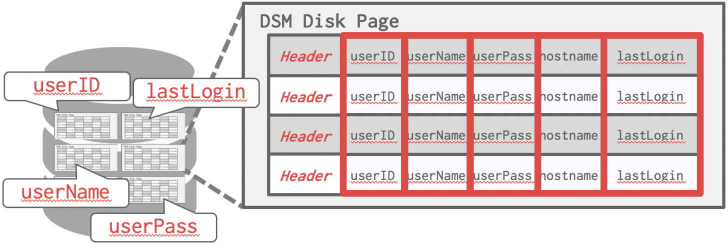
- Advantages
  - Fast inserts, updates, and deletes.
  - Good for queries that need the entire tuple.
- Disadvantages
  - Not good for scanning large portions of the table and/or a subset of the attributes.

Decomposition Storage Model (DSM)

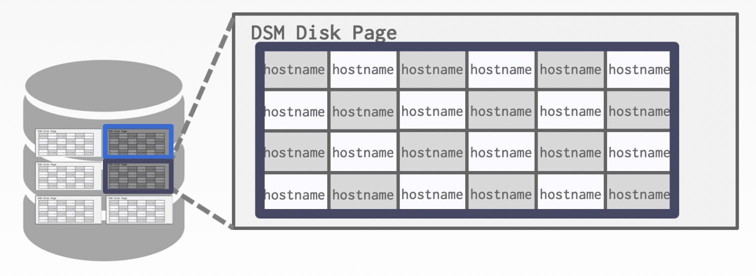
- The DBMS stores the values of a **single attribute** for all tuples contiguously in a page.
  - Also known as a "column store".
- Ideal for OLAP workloads where read-only queries perform large scans over a subset of the table's attributes.

Decomposition Storage Model (DSM)

- The DBMS stores the values of a single attribute for all tuples contiguously in a page.
  - Also known as a "column store".



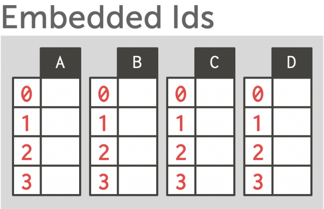
Decomposition Storage Model (DSM)




```
SELECT COUNT(U.lastLogin), EXTRACT(month FROM U.lastLogin) AS month
FROM useracct AS U
WHERE U.hostname LIKE '%.gov'
GROUP BY EXTRACT(month FROM U.lastLogin)
```

Tuple Identification

- Choice 1: Fixed-length Offsets
  - Each value is the same length for an attribute.
- Choice 2: Embedded Tuple Ids
  - Each value is stored with its tuple id in a column.



Decomposition Storage Model (DSM)

- Advantages
  - Reduces the amount wasted I/O because the DBMS only reads the data that it needs.
  - Better query processing and data compression (more on this later).
- Disadvantages
  - Slow for point queries, inserts, updates, and deletes because of **tuple splitting/stitching**.

DSM History

- 1970s: Cantor DBMS
- 1980s: DSM Proposal
- 1990s: SybaseIQ (in-memory only)
- 2000s: Vertica, VectorWise, MonetDB
- 2010s: Everyone

Schema Changes: Columns

- ADD COLUMN:
  - NSM: Copy tuples into new region in memory.
  - DSM: Just create the new column segment on disk.
- DROP COLUMN:
  - NSM-1: Copy tuples into new region of memory.
  - NSM-2: Mark column as "deprecated", clean up later.
  - DSM: Just drop the column and free memory.
- CHANGE COLUMN:
  - Check whether the conversion is allowed to happen. Depends on default values.

Conclusion

- A DBMS encodes and decodes the tuple's bytes into a set of attributes based on its schema.
- It is important to choose the right storage model for the target workload
  - OLTP $\longrightarrow$ Row-Store
  - OLAP $\longrightarrow$ Column-Store

References I