Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos – A. Pavlo

Lecture#25: Column Stores

Today's Class

- Storage Models
- System Architectures
- Vectorization
- Compression
- Data Modification

Wikipedia Example

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


OLTP

- On-line Transaction Processing:
  - Short-lived txns.
  - Small footprint.
  - Repetitive operations.

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

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

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

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

OLAP

- On-line Analytical Processing:
  - Long running queries.
  - Complex joins.
  - Exploratory queries.

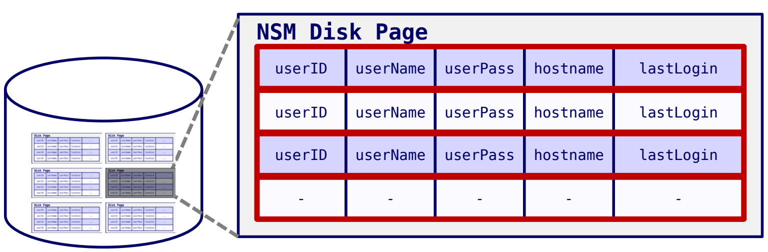
```
SELECT COUNT(U.lastLogin),
       EXTRACT(month FROM U.lastLogin) AS month
  FROM useracct AS U
 WHERE U.hostname LIKE '%.gov'
 GROUP BY EXTRACT(month FROM U.lastLogin)
```


Data Storage Models

- There are different ways to store tuples.
- We have been assuming the ***n*-ary storage model** this entire semester.

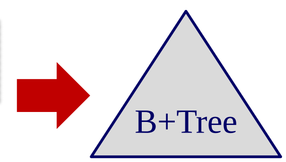
n-ary Storage Model

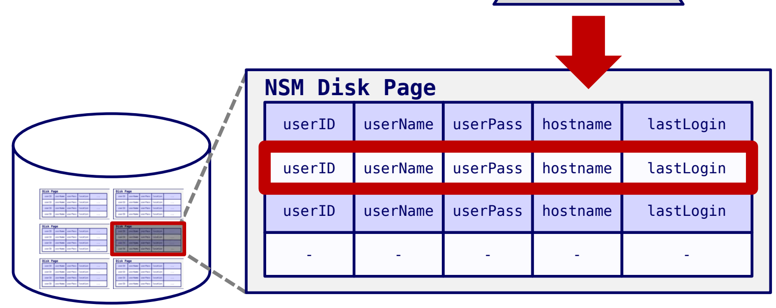
- The DBMS stores all attributes for a single tuple contiguously in a block.



n-ary Storage Model

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

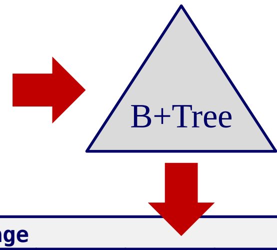


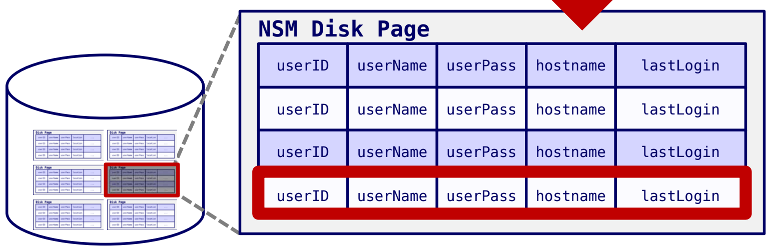


n-ary Storage Model

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

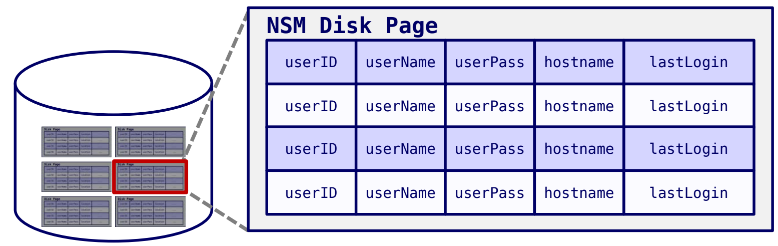
```
INSERT INTO useracct
VALUES (?,?,…?)
```





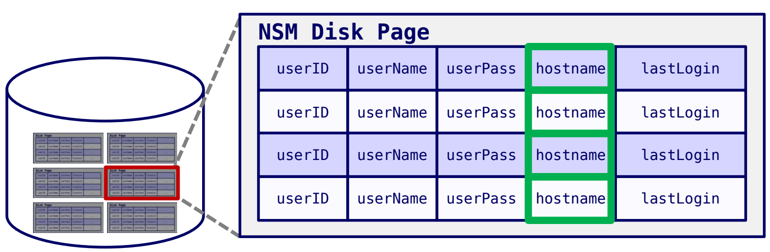
n-ary Storage Model

```
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)
```



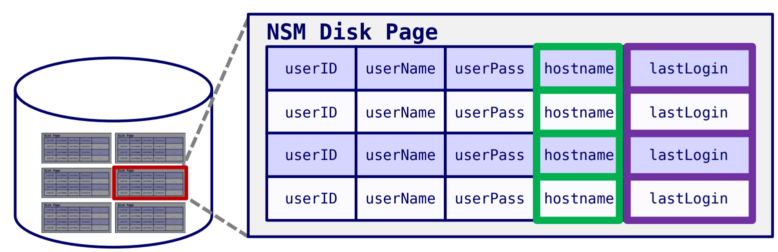
n-ary Storage Model

```
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)
```



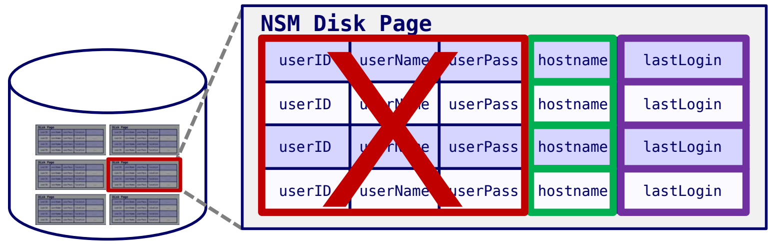
n-ary Storage Model

```
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)
```



n-ary Storage Model

```
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)
```



n-ary Storage Model

- 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.

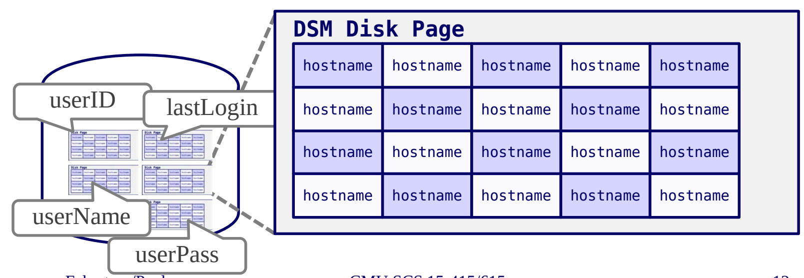
Decomposition Storage Model

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

userID	userName	userPass	hostname	lastLogin
userID	userName	userPass	hostname	lastLogin
userID	userName	userPass	hostname	lastLogin
-	-	-	-	-

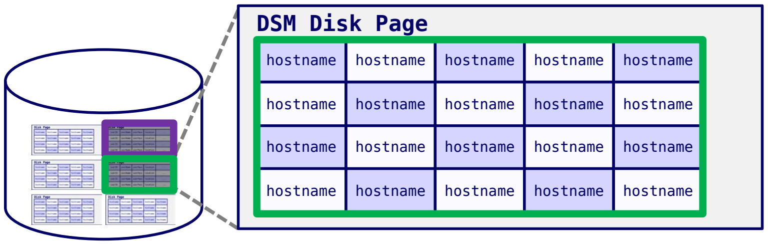
Decomposition Storage Model

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



Decomposition Storage Model

```
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)
```



Decomposition Storage Model

- 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.

History

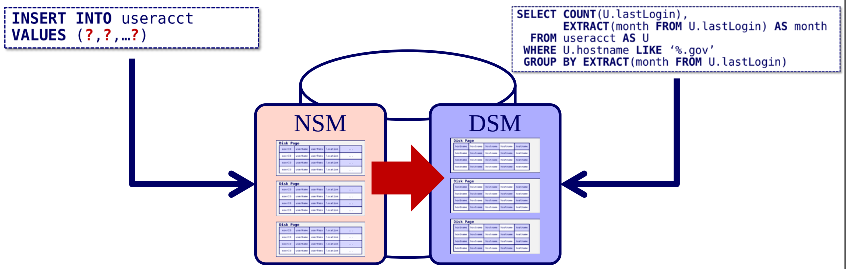
- **1970s:** Cantor DBMS
- **1980s:** DSM Proposal
- **1990s:** SybaseIQ (in-memory only)
- **2000s:** Vertica, VectorWise, MonetDB
- **2010s:** Cloudera Impala, Amazon Redshift, "The Big Three"

System Architectures

- Fractured Mirrors
- Partition Attributes Across (PAX)
- Pure Columnar Storage

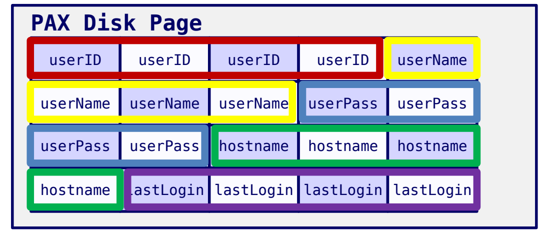
Fractured Mirrors

- Store a second copy of the database in a DSM layout that is automatically updated.
  - Examples: Oracle, IBM DB2 BLU



PAX

- Data is still stored in NSM blocks, but each block is organized as mini columns.



Column Stores

- Entire system is designed for columnar data.
  - Query Processing, Storage, Operator Algorithms, Indexing, etc.
  - Examples: Vertica, VectorWise, Paraccel, Cloudera Impala, Amazon Redshift

Today's Class

- Storage Models
- System Architectures
- Vectorization
- Compression
- Data Modification

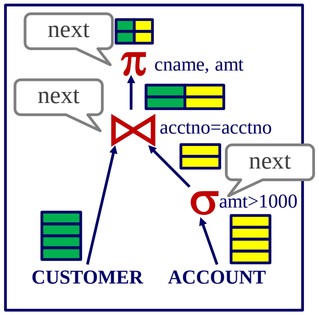
Query Processing Strategies

- The DBMS needs to process queries differently when using columnar data.
- We have already discussed the Iterator Model for processing tuples in the DBMS query operators.

Iterator Model

- Each operator calls **next()** on their child operator to process tuples one at a time.

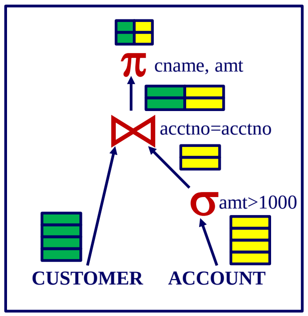
```
SELECT cname, amt
  FROM customer, account
 WHERE customer.acctno =
       account.acctno
   AND account.amt > 1000
```



Materialization Model

- Each operator consumes its entire input and generates the full output all at once.

```
SELECT cname, amt
  FROM customer, account
 WHERE customer.acctno =
       account.acctno
   AND account.amt > 1000
```



Observations

- The Iterator Model is bad with a DSM because it requires the DBMS to stitch tuples back together each time.
- The Materialization Model is a bad because the intermediate results may be larger than the amount of memory in the system.

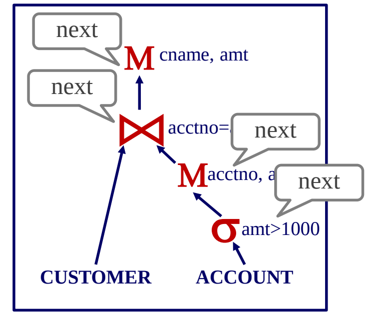
Vectorized Model

- Like the Iterator Model but each **`next()`** invocation returns a vector of tuples instead of a single tuple.
- This vector does not have to contain the entire tuple, just the attributes that are needed for query processing.

Vectorized Model

- Each operator calls **`next()`** on their child operator to process vectors.

```
SELECT cname, amt
  FROM customer, account
 WHERE customer.acctno =
       account.acctno
   AND account.amt > 1000
```

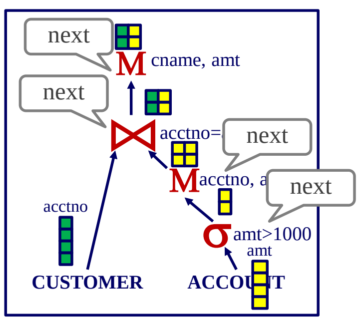


CMU SCS

Vectorized Model

- Each operator calls **`next()`** on their child operator to process vectors.

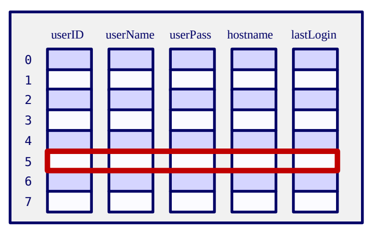
```
SELECT cname, amt
  FROM customer, account
 WHERE customer.acctno =
       account.acctno
   AND account.amt > 1000
```

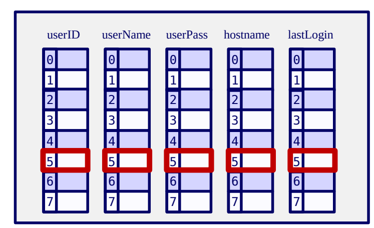
Faloutsos/Pavlo CMU SCS 15-415/615 26

Virtual IDs vs. Offsets

- Need a way to stitch tuples back together.
- Two approaches:
  - Fixed length offsets
  - Virtual ids embedded in columns



Offsets



Virtual Ids

Vectorized Model

- Reduced interpretation overhead.
- Better cache locality.
- Compiler optimization opportunities.

- AFAIK, VectorWise is still the only system that uses this model. Other systems use query compilation instead…

Today's Class

- Storage Models
- System Architectures
- Vectorization
- Compression
- Data Modification

Compression Overview

- Compress the database to reduce the amount of I/O needed to process queries.
- DSM databases compress much better than NSM databases.
  – Storing similar data together is ideal for compression algorithms.

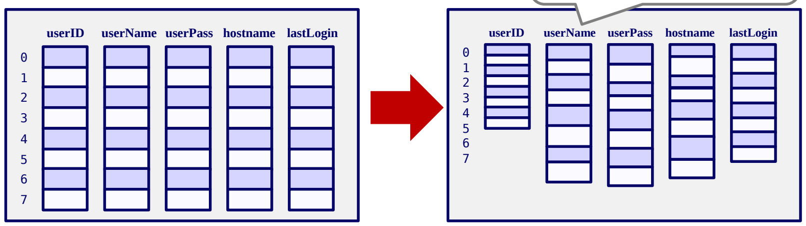
Naïve Compression

- Use a general purpose algorithm to compress pages when they are stored on disk.
  – Example: 10KB page in memory, 4KB compressed page on disk.
- Do we have to decompress the page when it is brought into memory? Why or why not?

Fixed-width Compression

- Sacrifice some compression in exchange for having uniform-length values per attribute.

Tuples are no longer aligned at offsets



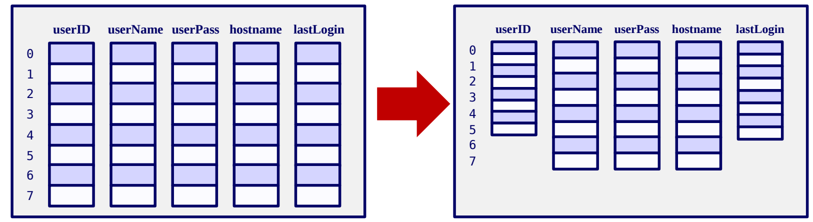
Original Data

Variable-Length Compression

CMU SCS

Fixed-width Compression

- Sacrifice some compression in exchange for having uniform-length values per attribute.

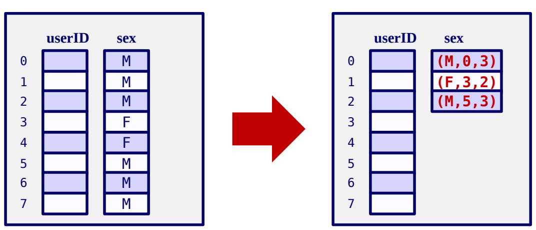


Original Data

Fixed-Length Compression

Run-length Encoding

- Compress runs of the same value into a compact triplet:
  - **`(value, startPosition, runLength)`**



Original Data

Unsorted RLE

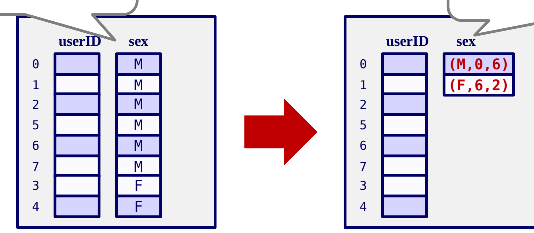
CMU SCS

Run-length Encoding

- Compress runs of the same value into a compact triplet:
  - **`(value, startPosition, runLength)`**

All tuples are sorted on this column.

Reduces the # of triplets

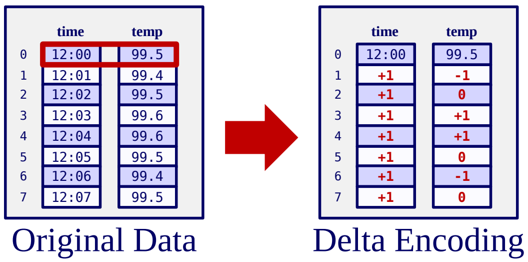


Sorted Data

Sorted RLE

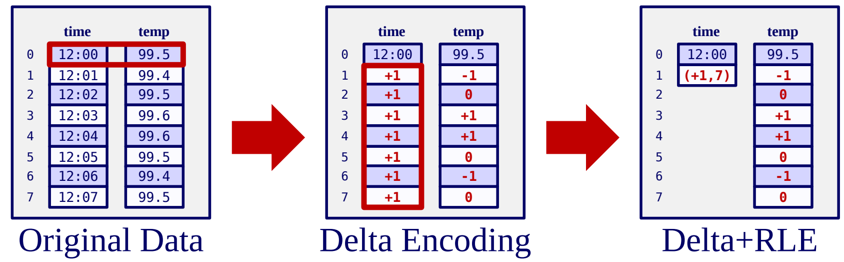
Delta Encoding

- Record the difference between successive values in the same column.



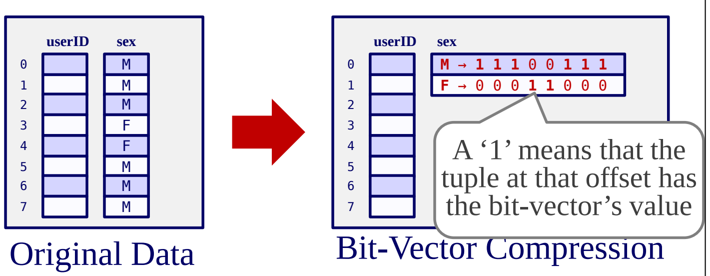
Delta Encoding

- Record the difference between successive values in the same column.



Bit-Vector Encoding

- Store a separate bit-vector for each unique value for a particular attribute where an offset in the vector corresponds to a tuple.



Dictionary Compression

- Replace frequent patterns with smaller integer codes.
  - Need to support fast encoding and decoding.
  - Need to also support range queries.

Dictionary Compression

- Construct a separate table of the unique values for an attribute sorted by value.

```
SELECT * FROM users
 WHERE name LIKE 'Tru%'
```

```
SELECT * FROM users
 WHERE name BETWEEN 70 AND 80
```

	userId	name
0	101	Truman
1	102	Obama
2	103	Bush
3	104	Reagan
4	105	Trump
5	106	Nixon
6	107	Carter
7	108	Ford

Original Data

	userId	name
0	101	70
1	102	50
2	103	10
3	104	60
4	105	80
5	106	40
6	107	20
7	108	30

value	code
Bush	10
Carter	20
Ford	30
Nixon	40
Obama	50
Reagan	60
Truman	70
Trump	80

Compressed Data

Dictionary Compression

- A dictionary needs to support two operations:
  - **Encode:** For a given uncompressed value, convert it into its compressed form.
  - **Decode:** For a given compressed value, convert it back into its original form.
- We need two data structures to support operations in both directions.

Summary

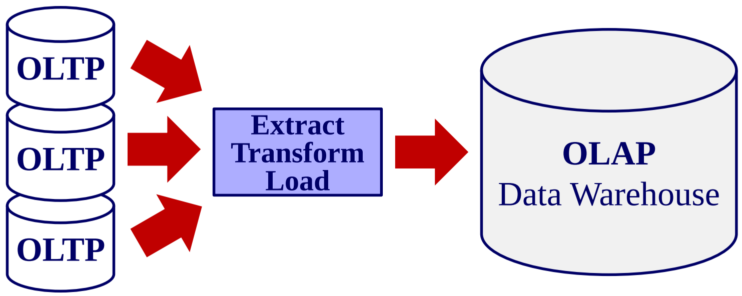
- Some operator algorithms can operate directly on compressed data
  - Saves I/O without having to decompress!
- Difficult to implement when the DBMS uses multiple compression schemes.
- It's generally good to wait as long as possible to materialize/decompress data when processing queries…

Today's Class

- Storage Models
- System Architectures
- Vectorization
- Compression
- Data Modification

Bifurcated Architecture

- All txns are executed on OLTP database.
- Periodically migrate changes to OLAP database.

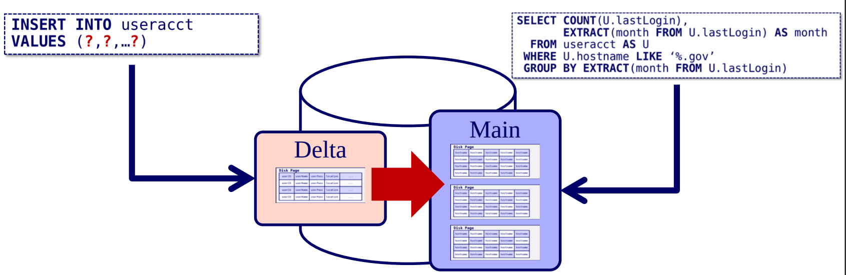


Modifying a Column Store

- Updating compressed data is expensive.
- Updating sorted data is expensive.
- The DBMS will store updates in an staging area and then apply them in batches.
  - Have to make sure that we execute queries on both the staging and main storage.

Delta Store

- Stage updates in delta store and periodically apply them in batches to the main storage.
  - Examples: Vertica, SAP HANA



HTAP

- **H**ybrid **T**ransaction-**A**nalytical **P**rocessing
- Single database instance that can handle both OLTP workloads and OLAP queries.
  - Row-store for OLTP
  - Column-store for OLAP
  - Examples: SAP HANA, MemSQL, HyPer, SpliceMachine, Peloton, Cloudera Kudu (???)