Georgia Tech

DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2019 // JOY ARULRAJ

LECTURE #07: STORAGE MODELS & COMPRESSION

CREATING THE NEXT®

ADMINISTRIVIA

- Reminder
  - Assignment 1 due on next Wednesday
  - Sign up for discussion slots on next Thursday

LAST CLASS

- Disk-centric & in-memory DBMSs
  - Buffer management (ACI**D**)
  - Query processing
  - Concurrency control (AC**I**D)
  - Logging and recovery (**A**CI**D**)

TODAY'S AGENDA

- Storage Models
- Compression
- Visual Storage Engine

STORAGE MODELS

ANATOMY OF A DATABASE SYSTEM

Query

Connection Manager + Admission Control	*Process Manager*
Query Parser	*Query Processor*
Query Optimizer	*Query Processor*
Query Executor	*Query Processor*
Lock Manager (Concurrency Control)	*Transactional Storage Manager*
Access Methods (or Indexes)	*Transactional Storage Manager*
Buffer Pool Manager	*Transactional Storage Manager*
Log Manager	*Transactional Storage Manager*
Memory Manager + Disk Manager	*Shared Utilities*
Networking Manager	*Shared Utilities*

Source: Anatomy of a Database System

DATA ORGANIZATION

- One can think of an in-memory database as just a large array of bytes.
  - The schema tells the DBMS how to convert the bytes into the appropriate type (e.g., `INTEGER`, `DATE`).
  - Each tuple is prefixed with a header that contains **meta-data** (e.g., last modified time-stamp).

TABLE STORAGE FORMAT

- Storage Models
  - *N*-ary Storage Model (NSM) / Row-Store
  - Decomposition Storage Model (DSM) / Column-Store
  - Flexible or Hybrid Storage Model

N-ARY STORAGE MODEL (NSM)

- The DBMS stores all of the attributes for a single tuple contiguously.
- Ideal for OLTP workloads where txns tend to operate only on an individual entity and insert-heavy workloads.
- Use the tuple-at-a-time iterator model.

N-ARY STORAGE MODEL (NSM)

ID	University	Enrollment	City
1	Georgia Tech	15000	Atlanta
2	Wisconsin	30000	Madison
3	Carnegie Mellon	6000	Pittsburgh
4	UC Berkeley	30000	Berkeley

NSM PHYSICAL STORAGE

- **Choice #1: Heap-Organized Tables**
  - Tuples are stored in blocks called a **heap**.
  - The heap does not necessarily define an order
- **Choice #2: Index-Organized Tables**
  - Tuples are stored in the primary key **index** itself.
  - Index does define an order based on the primary key

N-ARY STORAGE MODEL (NSM)

- **Advantages**
  - Fast inserts, updates, and deletes.
  - Good for queries that need the entire tuple.
  - Can use index-oriented physical storage.
- **Disadvantages**
  - Not good for scanning large portions of the table and/or a subset of the attributes.
  - OLAP workloads & wide tables with lots of attributes

DECOMPOSITION STORAGE MODEL (DSM)

- The DBMS stores a single attribute for all tuples contiguously in a block of data.
  - Sometimes also called **vertical partitioning**.
- Ideal for OLAP workloads where read-only queries perform large scans over a subset of the table's attributes.
- Use the vector-at-a-time iterator model.

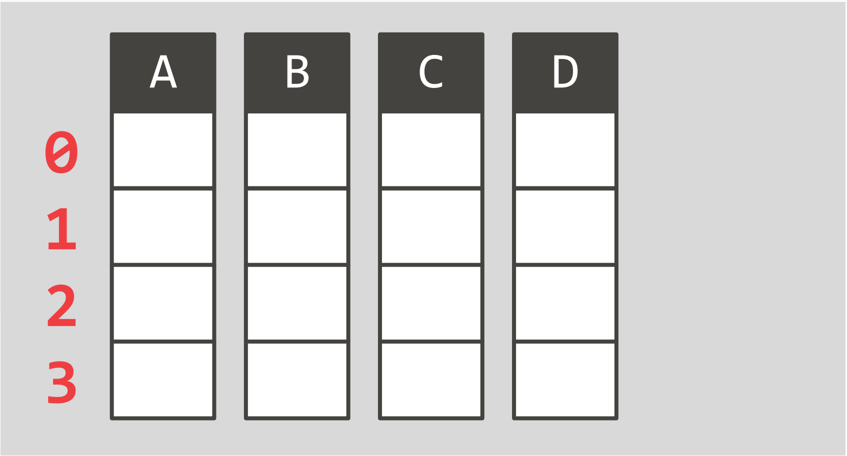
DECOMPOSITION STORAGE MODEL (DSM)

ID	University	Enrollment	City
1	Georgia Tech	15000	Atlanta
2	Wisconsin	30000	Madison
3	Carnegie Mellon	6000	Pittsburgh
4	UC Berkeley	30000	Berkeley

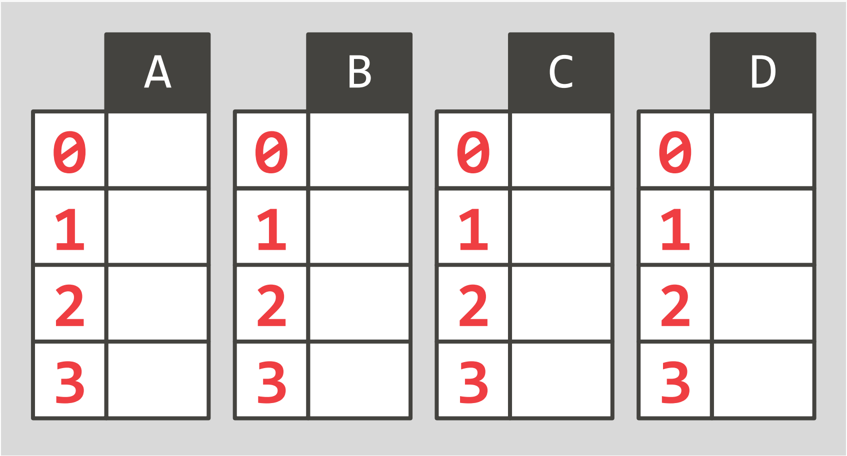
TUPLE IDENTIFICATION IN DSM

- **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.

Offsets



Embedded Ids



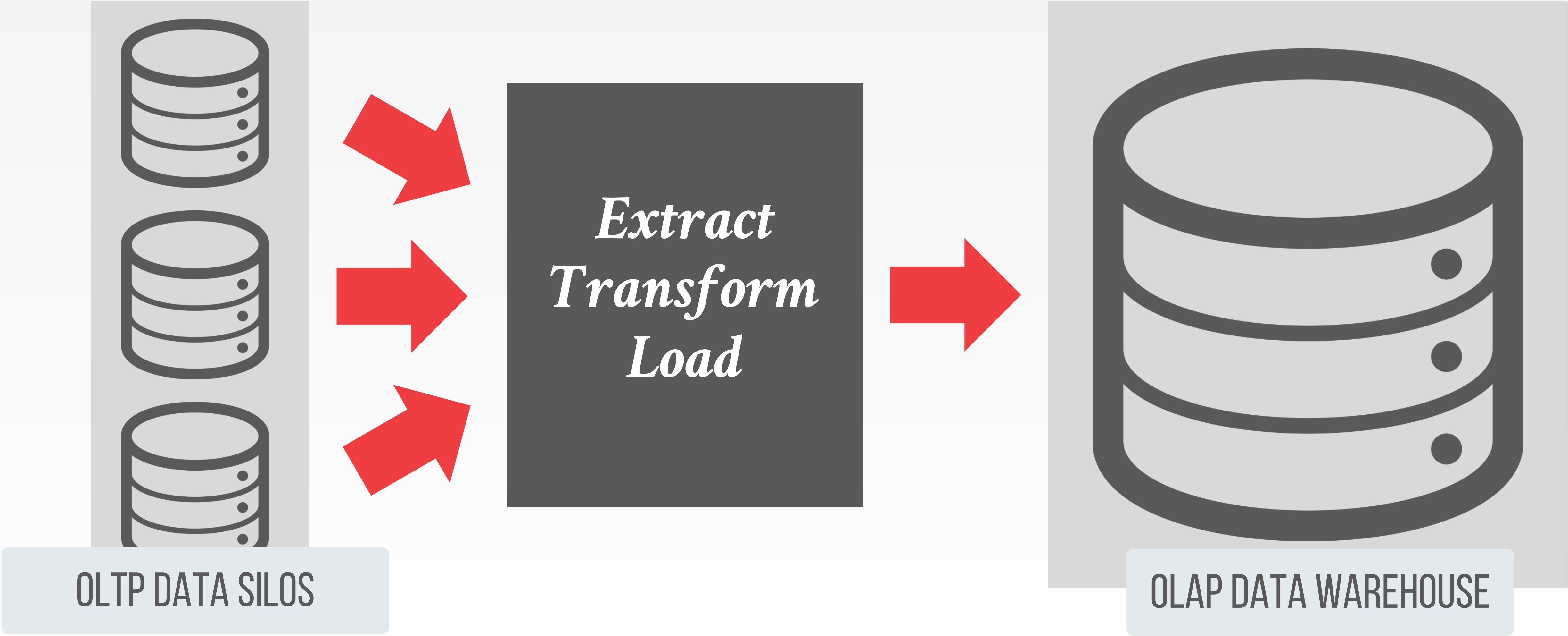
DECOMPOSITION STORAGE MODEL (DSM)

- **Advantages**
  – Reduces the amount wasted work because the DBMS only reads the data that it needs.
  – Better compression.
- **Disadvantages**
  – Slow for point queries, inserts, updates, and deletes because of tuple splitting/stitching (OLTP workloads).

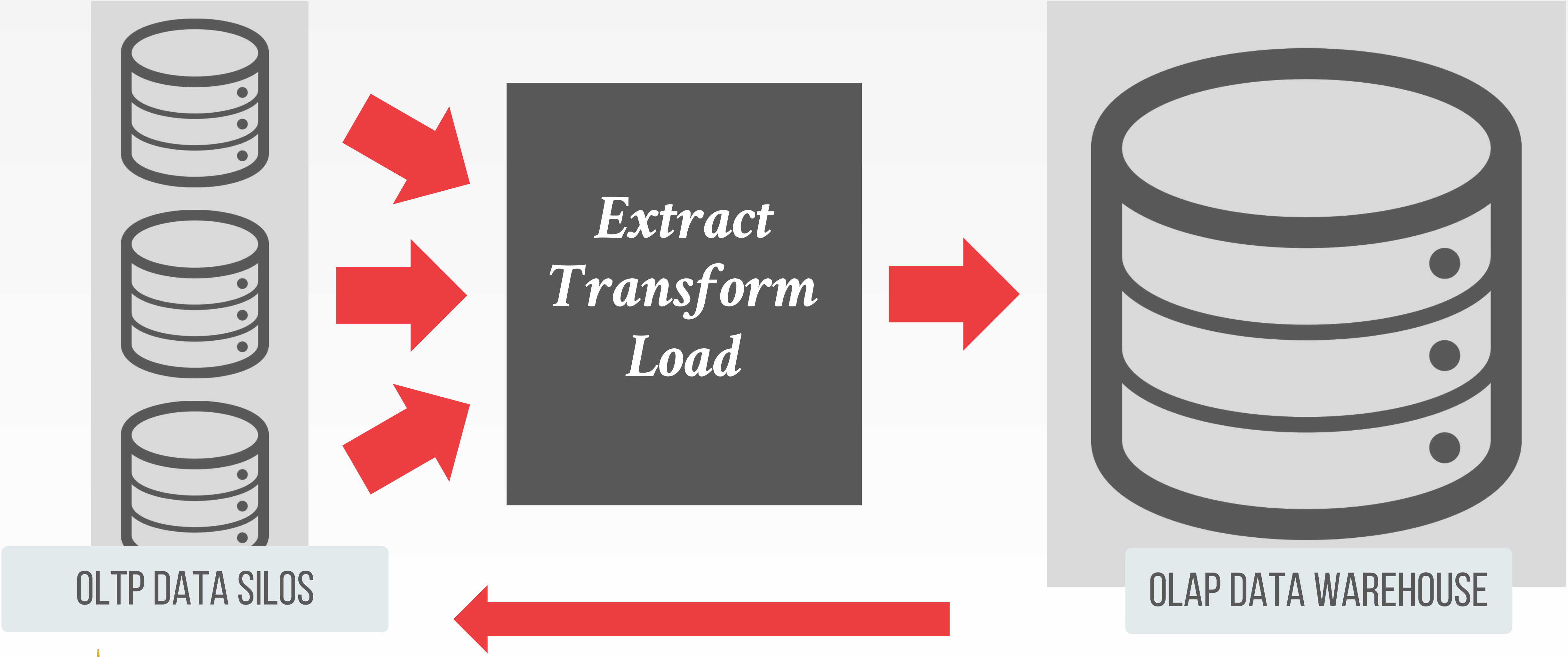
OBSERVATION

- Can we build a single system that supports both OLTP and OLAP workloads?
- Data is "hot" when first entered into database
  - A newly inserted tuple is more likely to be updated again the near future.
- As a tuple ages, it is updated less frequently.
  - At some point, a tuple is only accessed in read-only queries along with other tuples.

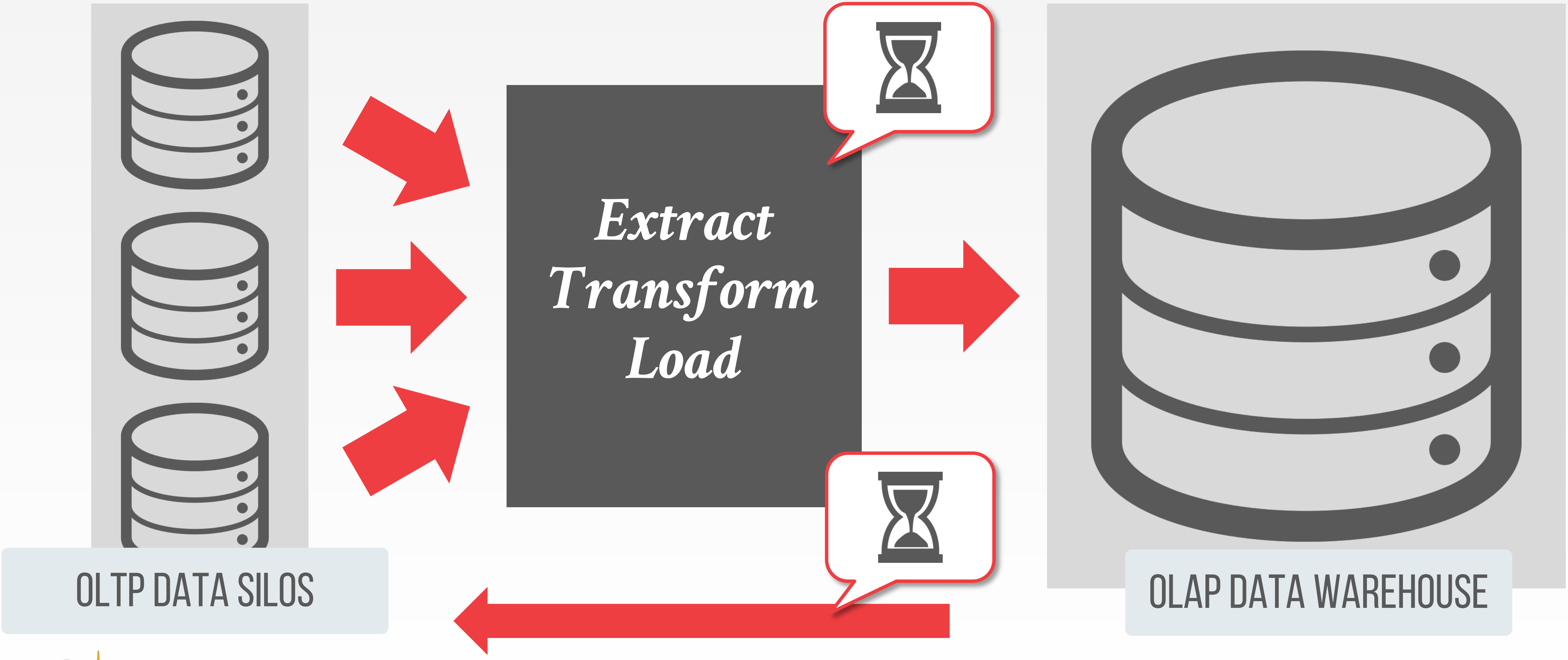
BIFURCATED ENVIRONMENT



BIFURCATED ENVIRONMENT



BIFURCATED ENVIRONMENT



HYBRID STORAGE MODEL

- Single database instance that uses different storage models for hot and cold data.
- Store new data in NSM for fast OLTP
  Migrate data to DSM for more efficient OLAP

HYBRID STORAGE MODEL

ID	University	Enrollment	City
1	Georgia Tech	15000	Atlanta
2	Wisconsin	30000	Madison
3	Carnegie Mellon	6000	Pittsburgh
4	UC Berkeley	30000	Berkeley

PELOTON ADAPTIVE STORAGE

- Employ a single execution engine architecture that is able to operate on both NSM and DSM data.
  - Don't need to store two copies of the database.
  - Don't need to sync multiple database segments.
- Note that a DBMS can still use the delta-store approach with this single-engine architecture.

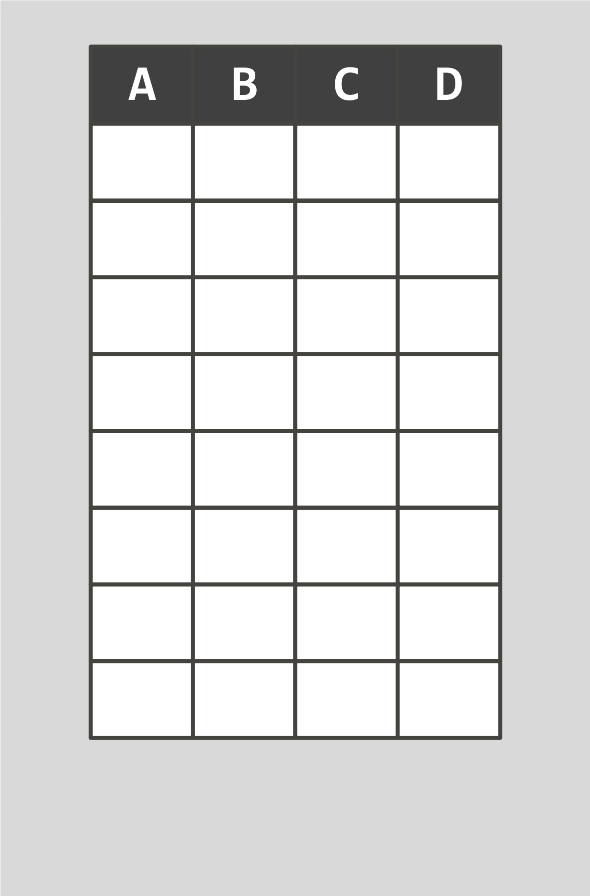
BRIDGING THE ARCHIPELAGO BETWEEN ROW-STORES AND COLUMN-STORES FOR HYBRID WORKLOADS
SIGMOD 2016

PELOTON ADAPTIVE STORAGE

```
UPDATE AndySux
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = “xxx”
```

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = “yyy”
```

Original Data



PELOTON ADAPTIVE STORAGE

```
UPDATE AndySux
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = "xxx"
```

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = "yyy"
```

Original Data

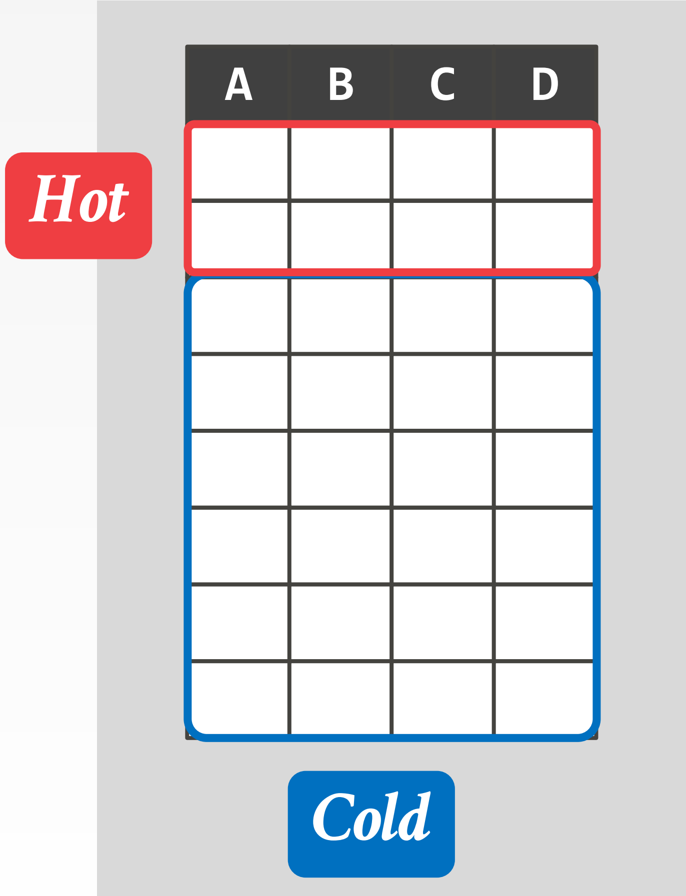
A	B	C	D

PELOTON ADAPTIVE STORAGE

```
UPDATE AndySux
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = "xxx"
```

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = "yyy"
```

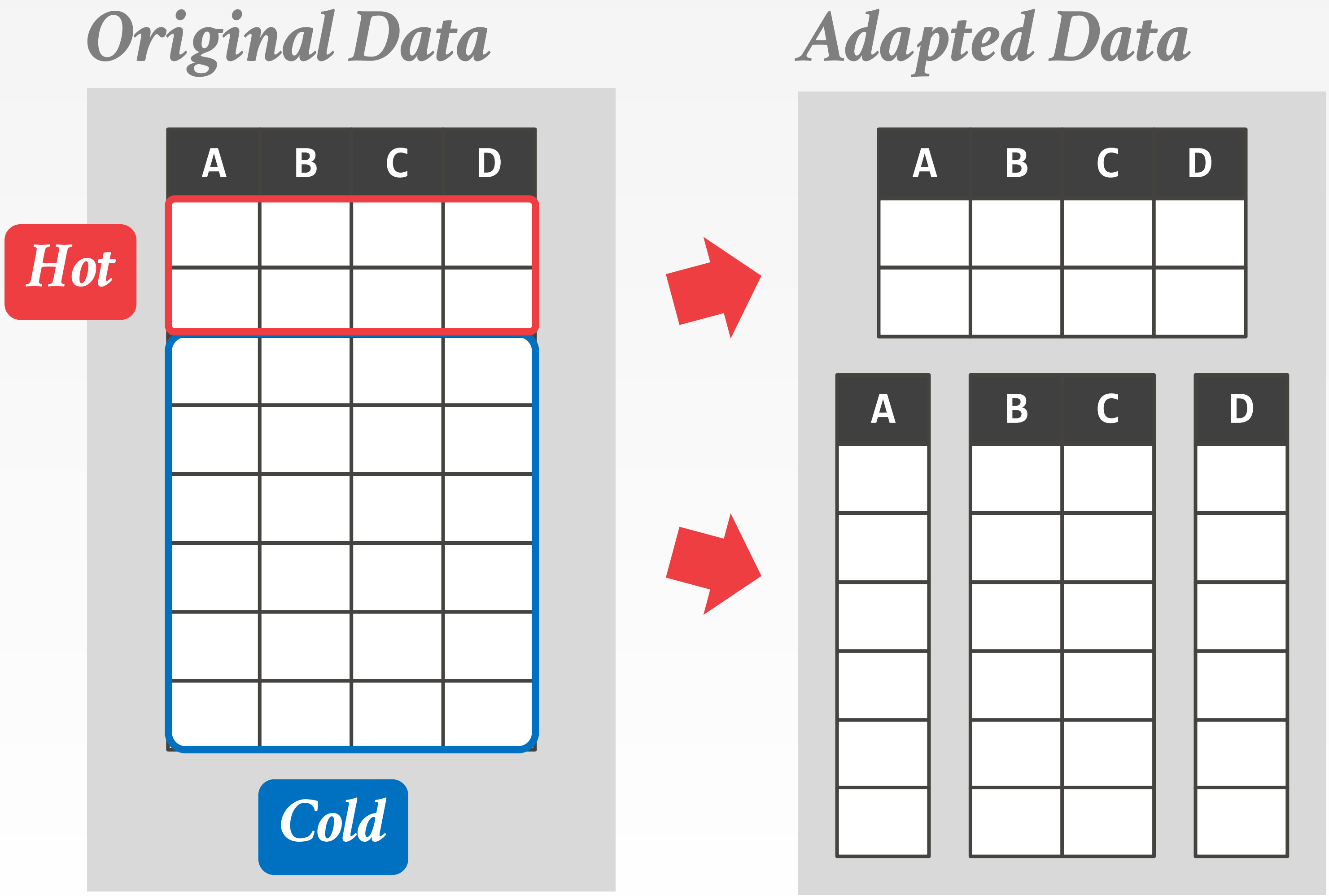
Original Data



PELOTON ADAPTIVE STORAGE

```
UPDATE AndySux
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = “xxx”
```

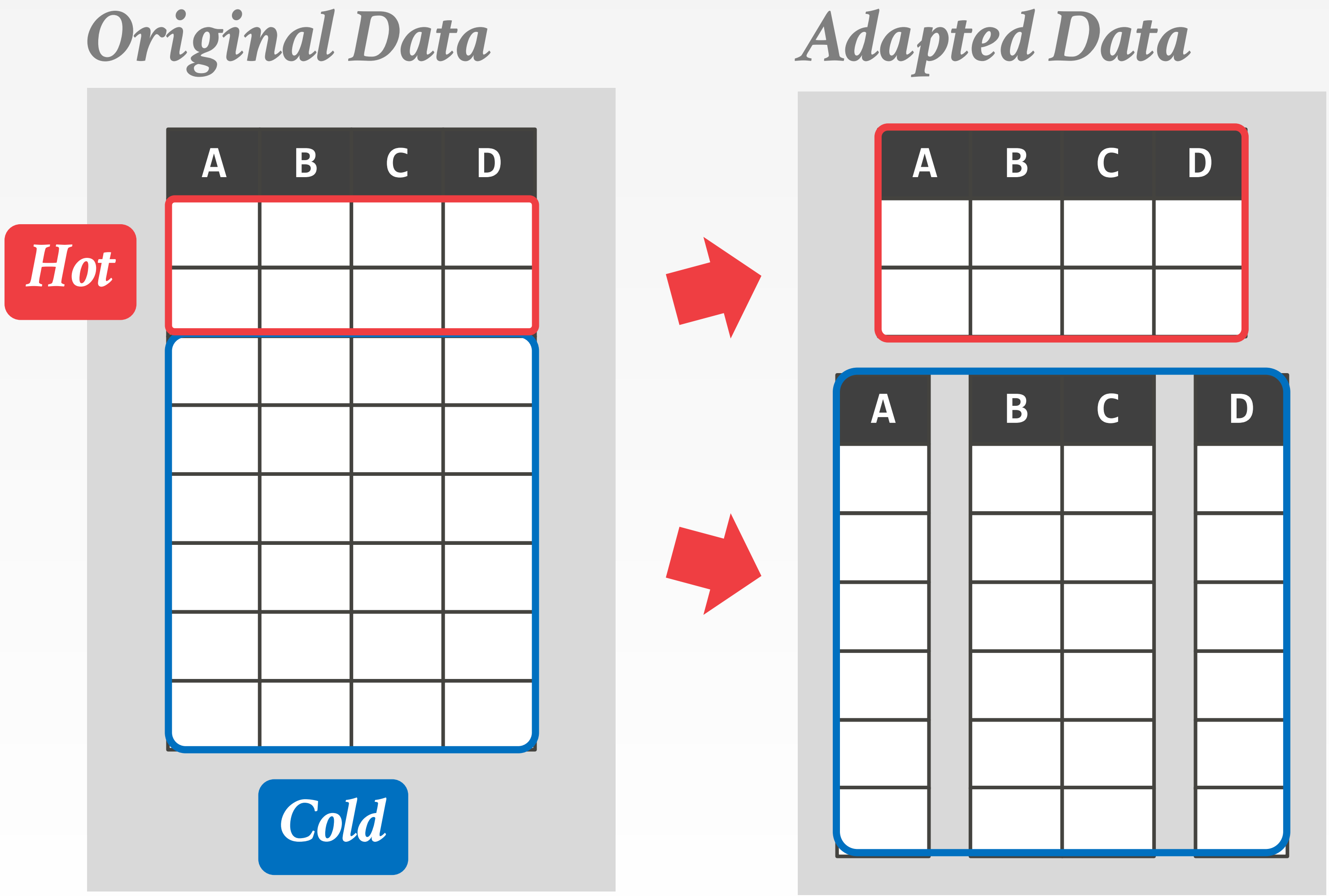
```
SELECT AVG(B)
  FROM AndySux
 WHERE C = “yyy”
```



PELOTON ADAPTIVE STORAGE

```
UPDATE AndySux
   SET A = 123,
       B = 456,
       C = 789
 WHERE D = “xxx”
```

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = “yyy”
```

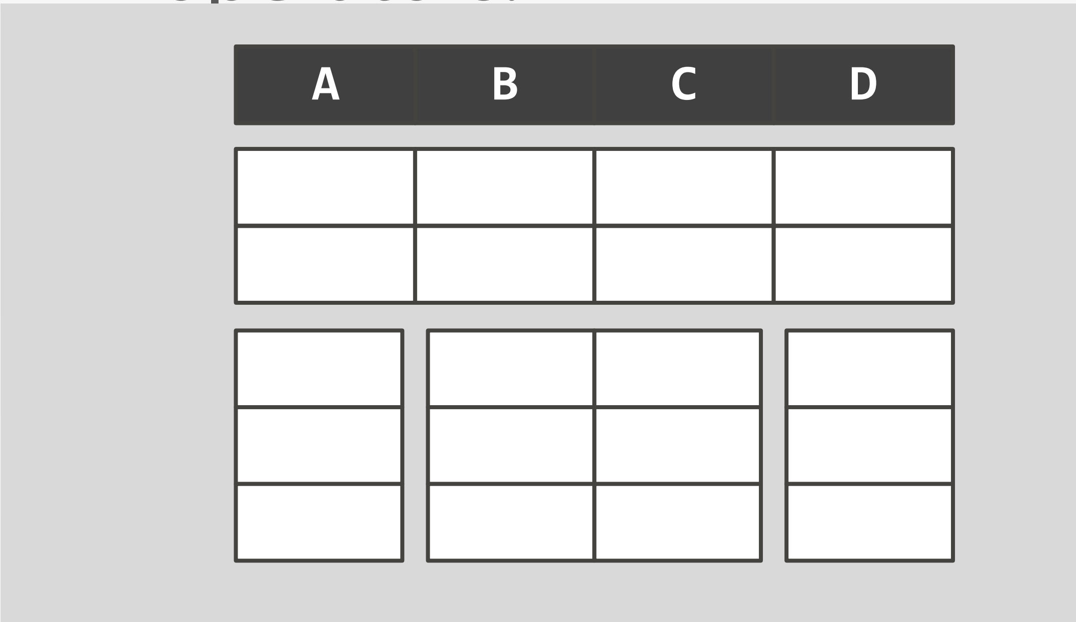


FLEXIBLE STORAGE MODEL

ID	University	Enrollment	City
1	Georgia Tech	15000	Atlanta
2	Wisconsin	30000	Madison
3	Carnegie Mellon	6000	Pittsburgh
4	UC Berkeley	30000	Berkeley

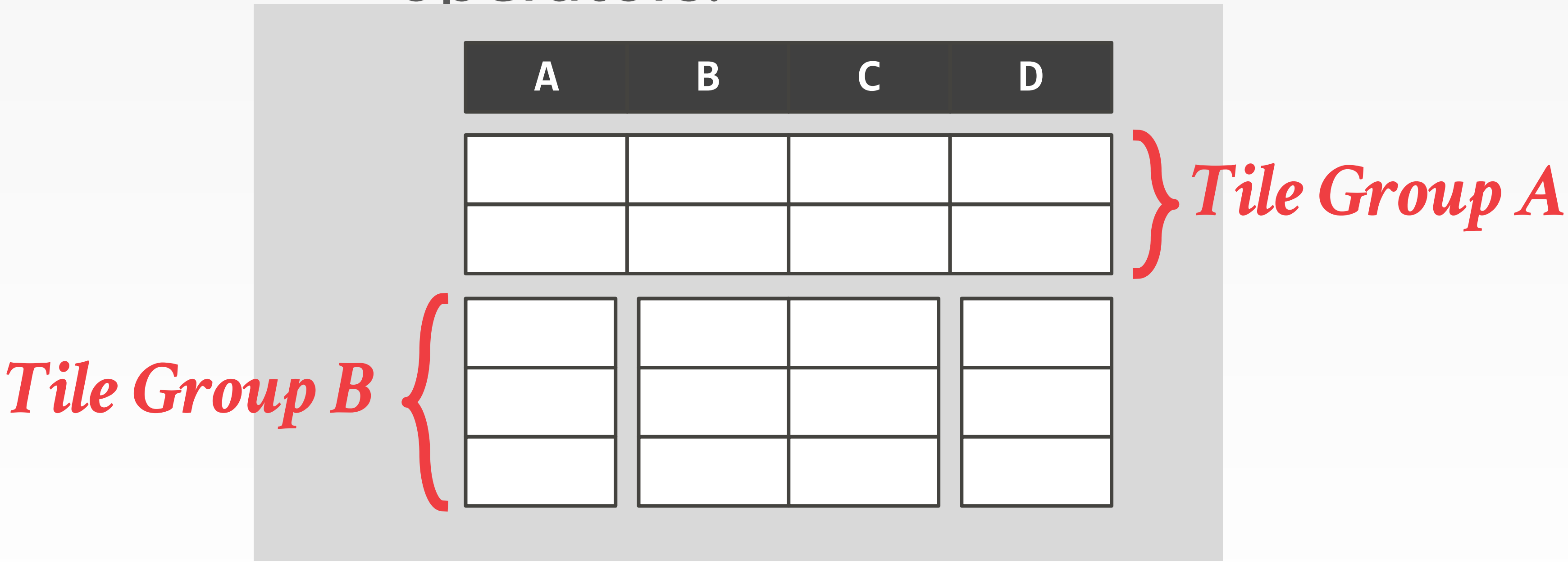
TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.



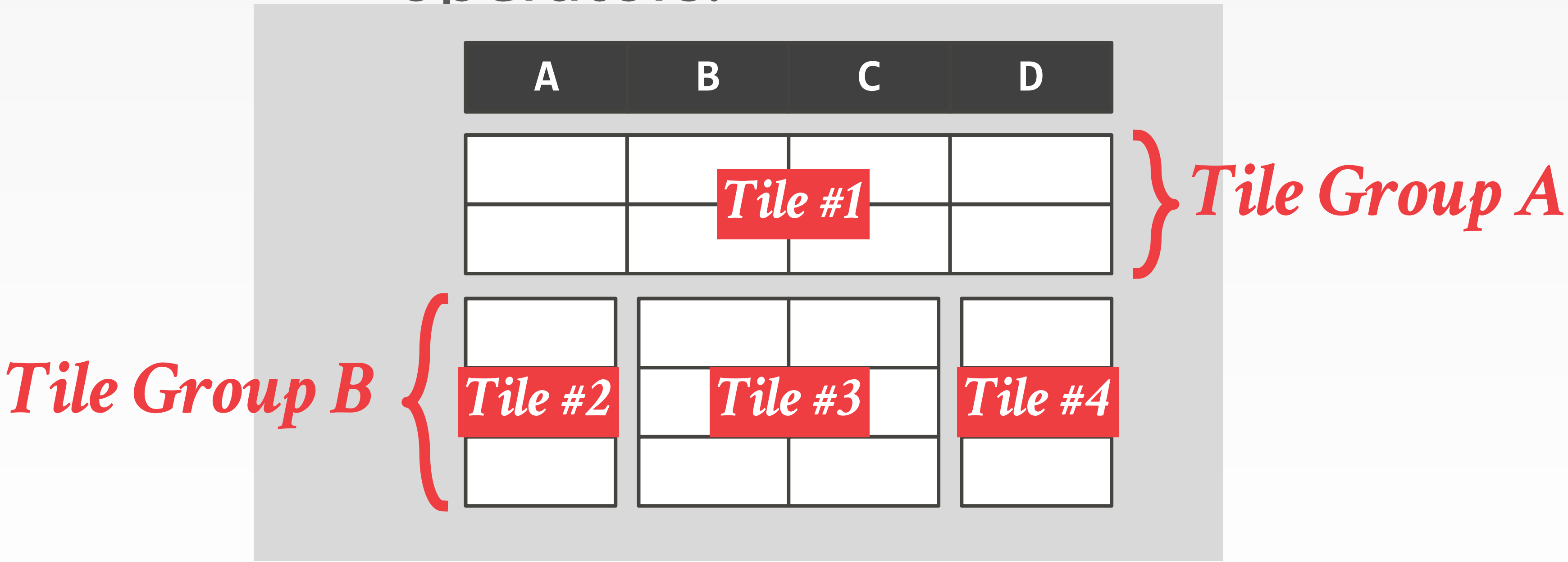
TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.



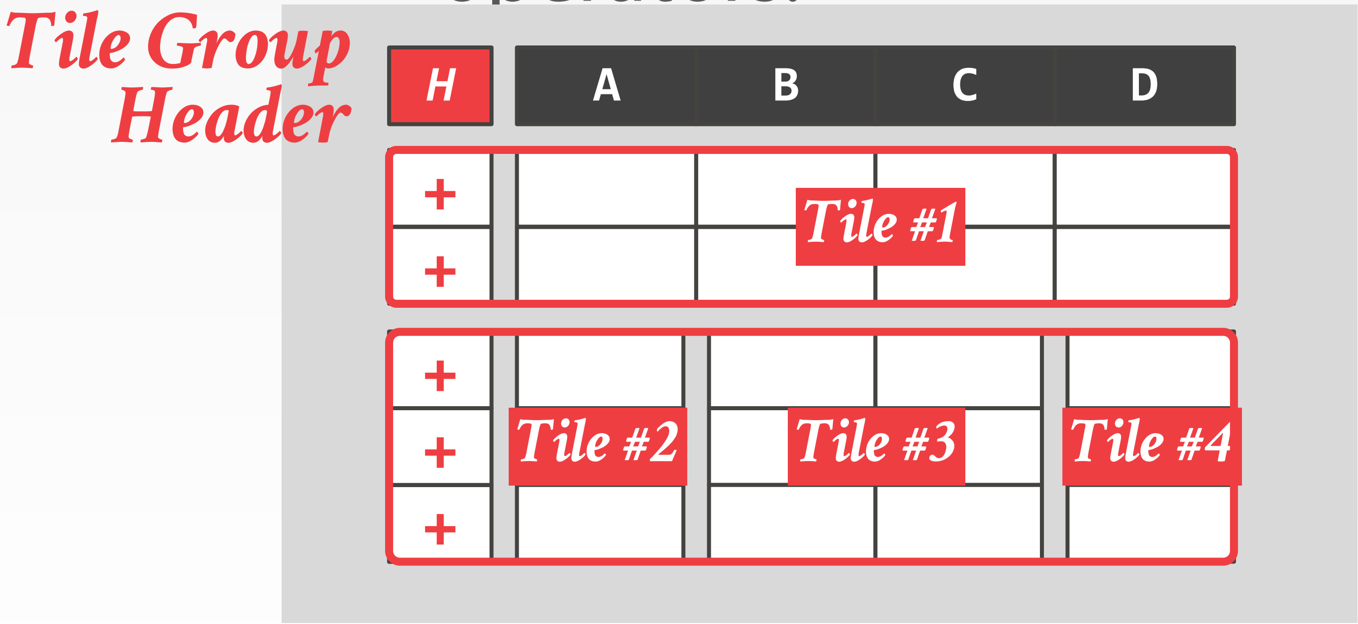
TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.



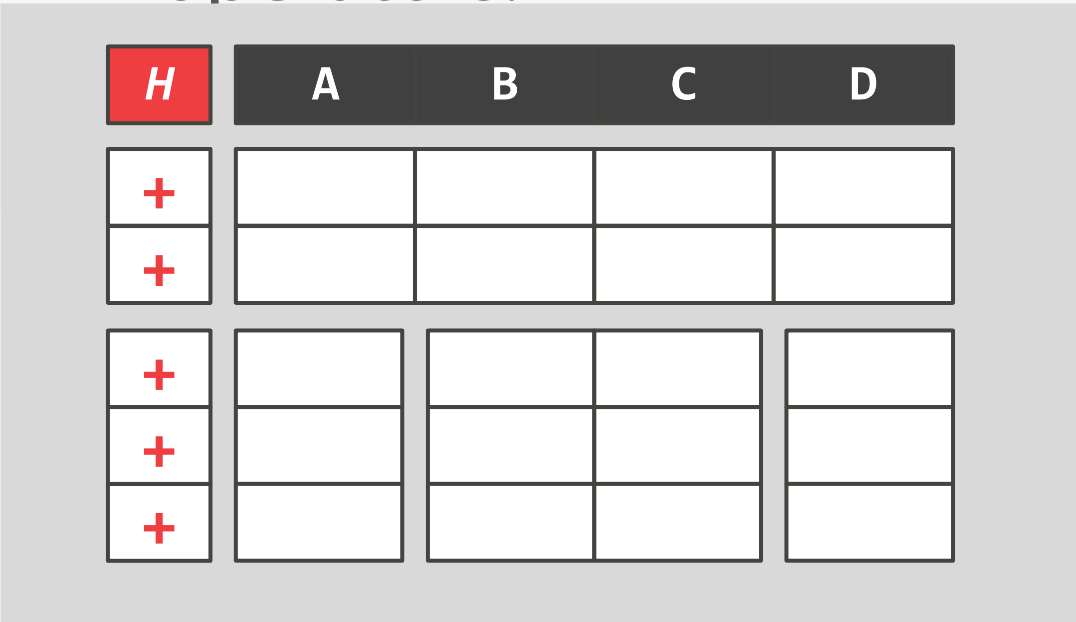
TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.



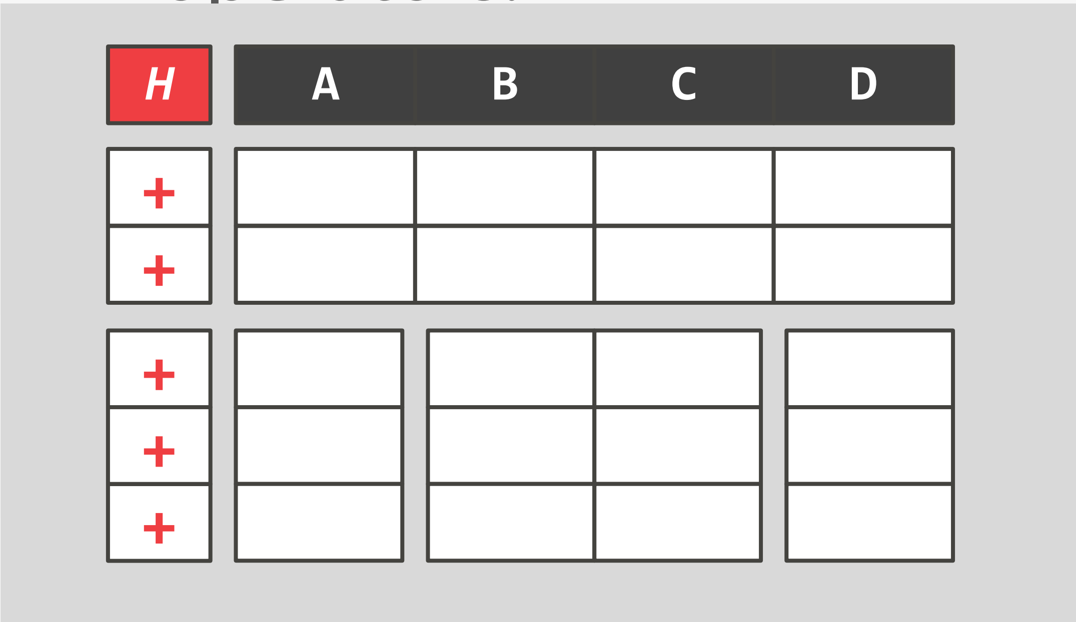
TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.

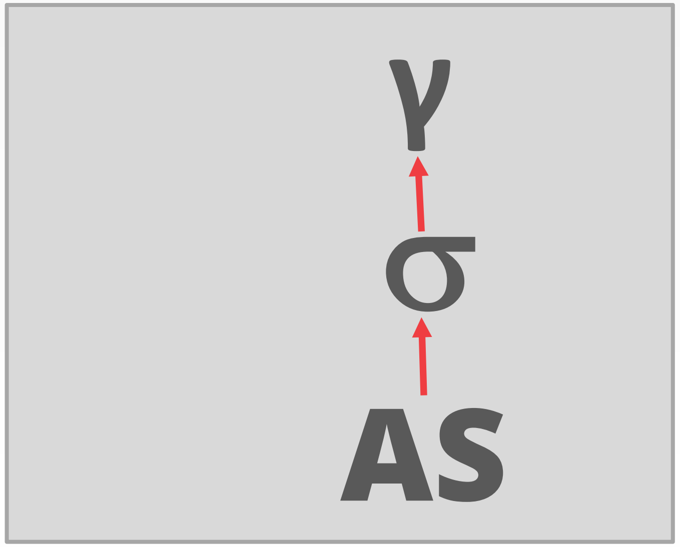


TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.

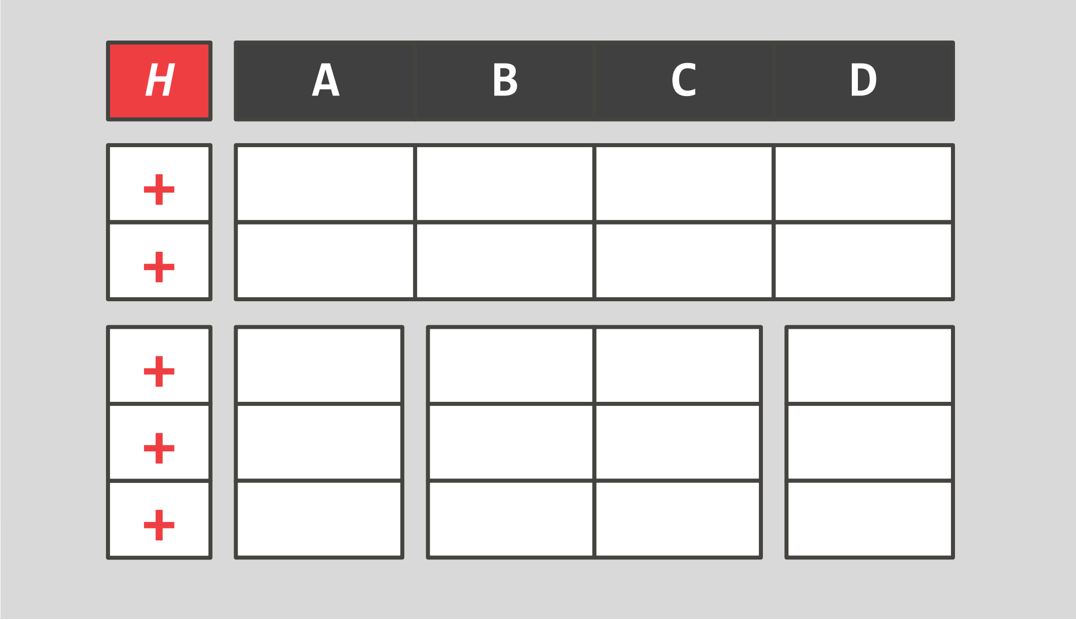



```
SELECT AVG(B)
  FROM AndySux
 WHERE C = "yyy"
```

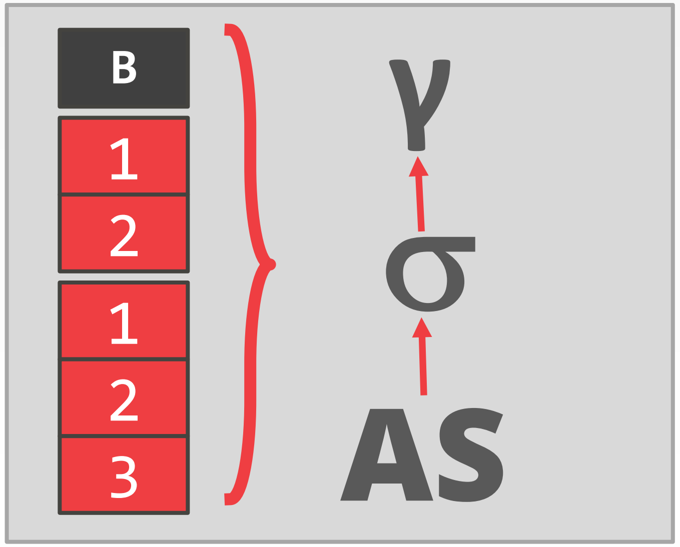


TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.



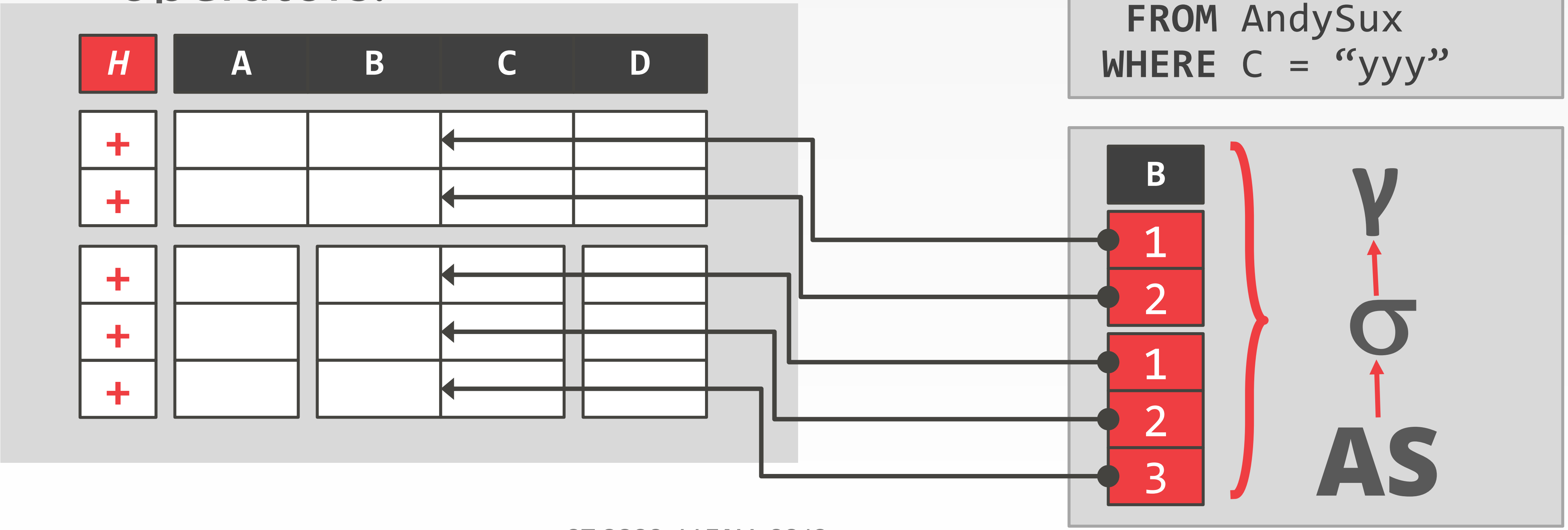

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = "yyy"
```



TILE ABSTRACTION

- Introduce an indirection layer that abstracts the true layout of tuples from query operators.

```
SELECT AVG(B)
  FROM AndySux
 WHERE C = "yyy"
```



PARTING THOUGHTS

- A flexible architecture that supports a hybrid storage model is the next major trend in DBMSs
  - This will enable relational DBMSs to support both OLTP and OLAP database workloads.

COMPRESSION

OBSERVATION

- I/O is the main bottleneck if the DBMS has to fetch data from disk.
  - CPU cost for decompressing data **<**
  - I/O cost for fetching un-compressed data.
- Compression always helps.

OBSERVATION

- In-memory DBMSs are more complicated
  - Compressing the database reduces DRAM requirements and processing.
- Key trade-off is **speed** vs. **compression ratio**
  - In-memory DBMSs (always?) choose speed.

REAL-WORLD DATA CHARACTERISTICS

- Data sets tend to have highly **skewed** distributions for attribute values.
  – Example: Zipfian distribution of the Brown Corpus
  – Words like "the", "a" occur very frequently in books

REAL-WORLD DATA CHARACTERISTICS

- Data sets tend to have high **correlation** between attributes of the same tuple.
  - Example: Order Date to Ship Date (few days)
  - (June 5, +5) instead of (June 5, June 10)

DATABASE COMPRESSION

- **Goal #1:** Must produce fixed-length values. Allows us to efficiently access tuples.
- **Goal #2:** Allow the DBMS to postpone decompression as long as possible during query execution. Operate directly on compressed data.
- **Goal #3:** Must be a **<u>lossless</u>** scheme. No data should be lost during this transformation.

LOSSLESS VS. LOSSY COMPRESSION

- When DBMS uses compression, it is always **lossless** since people don't like losing data.
- Any kind of **lossy** compression is has to be performed at the application level.
  - Example: Sensor data. Readings are taken every second, but we may only store average per minute.
- New DBMSs support approximate queries
  - Example: BlinkDB, SnappyData, XDB, Oracle (2017)

ZONE MAPS

- Pre-computed aggregates for blocks of data.
- DBMS can check the zone map first to decide whether it wants to access the block.

Original Data

val
100
200
300
400
400

ZONE MAPS

- Pre-computed aggregates for blocks of data.
- DBMS can check the zone map first to decide whether it wants to access the block.

Original Data

val
100
200
300
400
400

Zone Map

type	val
MIN	100
MAX	400
AVG	280
SUM	1400
COUNT	5

ZONE MAPS

- Pre-computed aggregates for blocks of data.
- DBMS can check the zone map first to decide whether it wants to access the block.

```
SELECT * FROM table
 WHERE val > 600
```

Original Data

val
100
200
300
400
400

Zone Map

type	val
MIN	100
MAX	400
AVG	280
SUM	1400
COUNT	5

COLUMNAR COMPRESSION SCHEMES

- Compression Schemes
  - Run-length Encoding
  - Bitmap Encoding
  - Delta Encoding
  - Incremental Encoding
  - Mostly Encoding
  - Dictionary Encoding

DICTIONARY COMPRESSION

- Most pervasive compression scheme in DBMSs.
- Replace frequent patterns with smaller codes.
- Need to support fast encoding and decoding.

DICTIONARY-BASED ORDER-PRESERVING STRING COMPRESSION FOR MAIN MEMORY COLUMN STORES
SIGMOD 2009

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 hash tables to support operations in both directions.

DICTIONARY COMPRESSION

- When to construct the dictionary?
- What is the scope of the dictionary?

DICTIONARY CONSTRUCTION

- **Choice #1: All At Once**
  - Compute the dictionary for all the tuples at a given point of time.
  - New tuples must use a separate dictionary or the all tuples must be recomputed.
- **Choice #2: Incremental**
  - Merge new tuples in with an existing dictionary.
  - Likely requires re-encoding of existing tuples.

DICTIONARY SCOPE

- **Choice #1: Block-level**
  - Only include a subset of tuples within a single table.
  - Lower compression ratio, but can add tuples easily
  - Impact of dictionary data corruption is localized
- **Choice #2: Table-level**
  - Construct a dictionary for the entire table.
  - Better compression ratio, but expensive to update.
- **Choice #3: Multi-Table**
  - Sometimes helps with joins and set operations.

PARTING THOUGHTS

- Dictionary encoding is probably the most useful compression scheme because it does not require pre-sorting.
- The DBMS can combine different approaches for even better compression.
- It is important to wait as long as possible during query execution to decompress data.

VISUAL STORAGE ENGINE

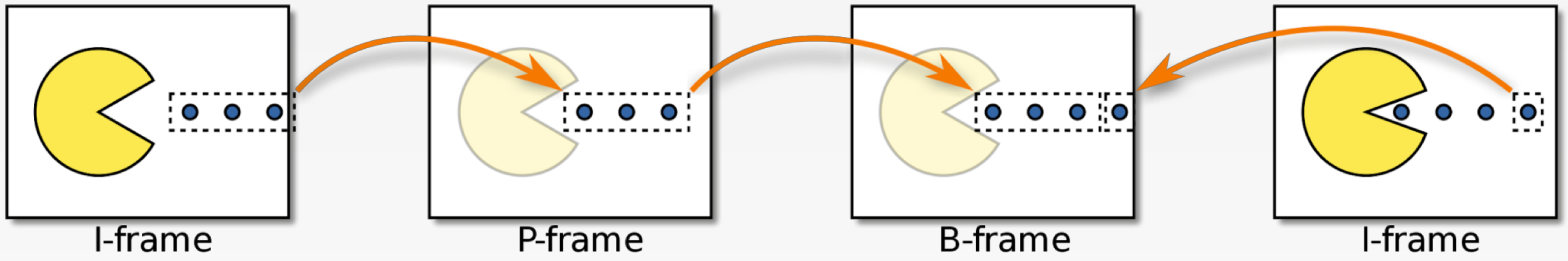
VIDEO ANALYTICS

- Components of a video analytics DBMS
  - Query parser
  - Query optimizer
  - Query execution engine
  - Storage engine

CHALLENGES: STORAGE ENGINE

- Loading frames from disk takes time
  - This slows down model training
  - Traditional video compression formats are optimized for human consumption
- Goals
  - Accelerate model training
  - Leverage the observation that the compression format need not be optimized for human consumption

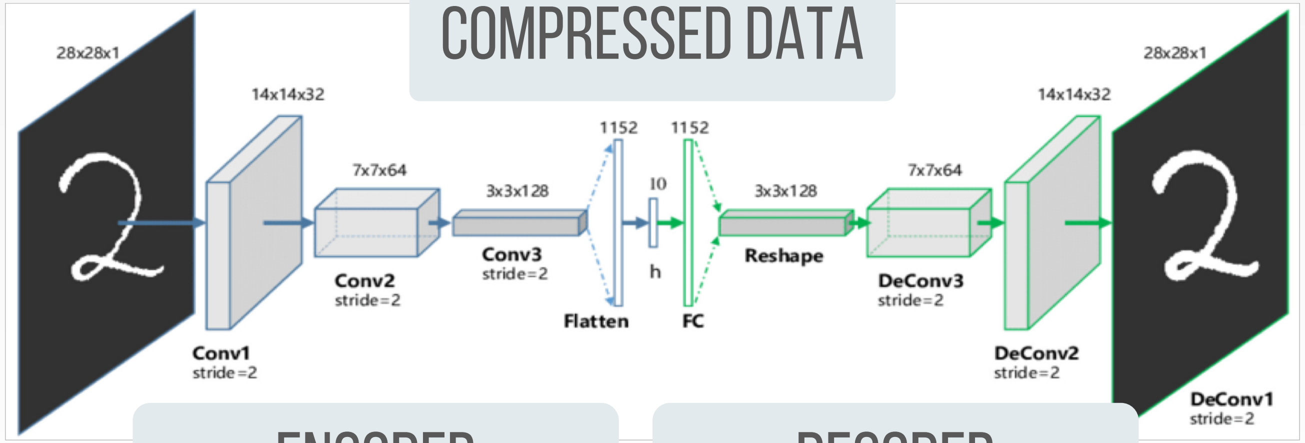
TRADITIONAL VIDEO COMPRESSION



- Three types of frame encoding
  - I-frame (intra-coded picture)
  - P-frame (predicted picture i.e. delta from I-frame)
  - B-frame (bi-directional predicted picture i.e. deltas from both the preceding and following frames)

CONVOLUTIONAL AUTO ENCODER

- An autoencoder is a neural network used to learn an efficient data coding.



COMPRESSED DATA

ENCODER (CONVOLUTIONS)

DECODER (DECONVOLUTIONS)

COMPRESSION USING AUTO ENCODER

- Train the auto encoder using videos
  - Compress frames using the auto encoder
  - Store compressed frames in the storage engine
- Execute queries on compressed data
  - Reduce storage footprint by orders of magnitude
  - Accelerate query processing

PARTING THOUGHTS

- Convolutional auto-encoders are capable of efficiently encoding visual data sets.
- What can we do with them?

NEXT LECTURE

- Query Execution