DataBase Storage

Type Representation

使用FLOAT,REAL/DOUBLE 与 使用 NUMERIC,DECIMAL,VARCHAR的区别

FLOAT是根据IEEE-754 Standard用二进制来对浮点数进行编码，上过ICS都知道其实这种方式表示浮点数是有误差的。因此在一些不能容忍误差的场合如金融统计等精度要求极高的场景，不能使用float/double，而应该使用numeric/decimal。当然使用float/double速度会快很多

Data Layout / Alignment

- 如何表示NULL DATA TYPES IN DB?

choice 1：Special Values

如用INT_MIN来表示NULL

choice 2：Null Column Bitmap Header

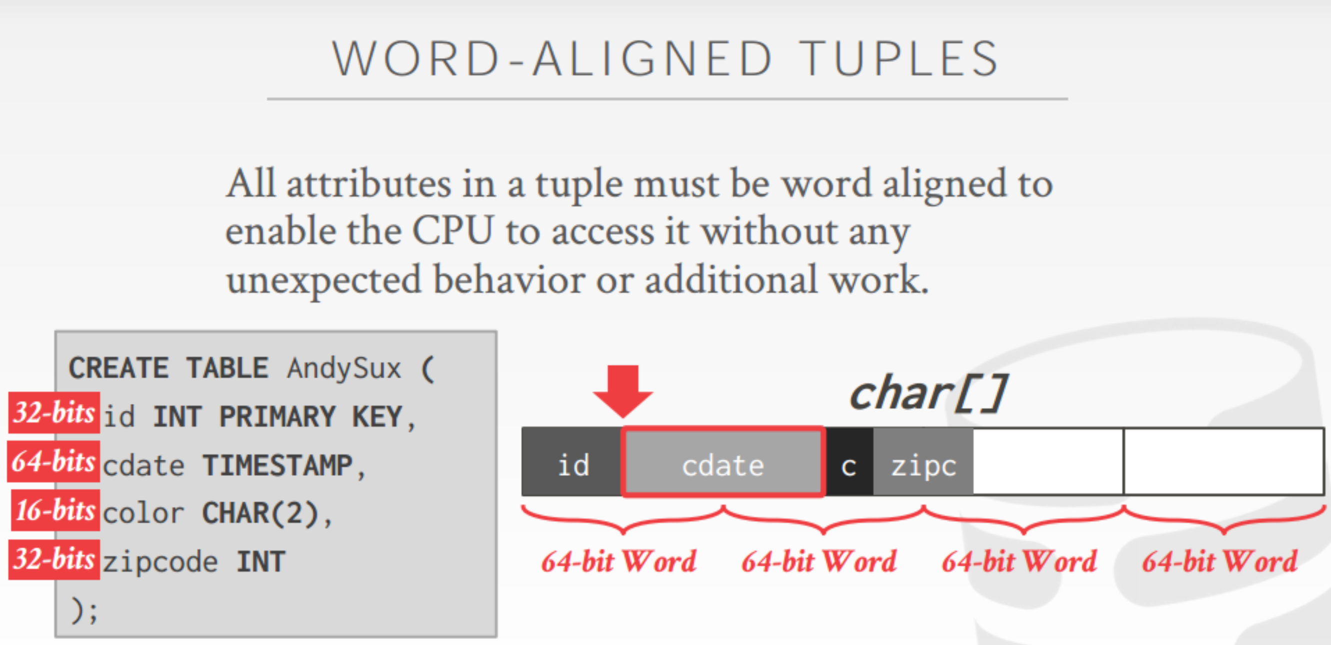
在tuple的header存一个bitmap,用单个bit来表示哪些属性是null

choice 3：Per Attribute Null Flag

用一个Flag来表示Null,但是会比较浪费存储空间，并且会有word alignment问题

- Unalignment data reading issue

下图所示：如果要读取cdate这个TS，由于其不是对齐存放的，跨越了第一个64bit word和第二个64bit word，导致读一个64bit的数据可能要读取两个64bit的word。这很影响性能



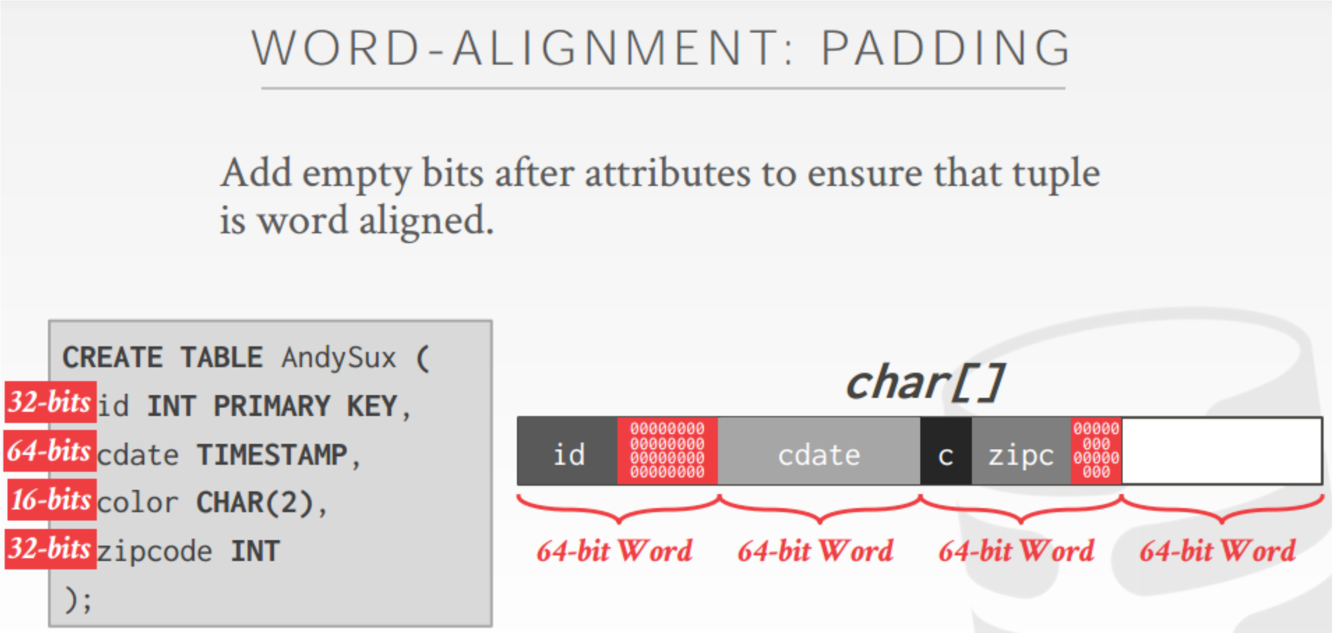
WORD-ALIGNED TUPLES

All attributes in a tuple must be word aligned to enable the CPU to access it without any unexpected behavior or additional work.

```
CREATE TABLE AndySux (
  id INT PRIMARY KEY,
  cdate TIMESTAMP,
  color CHAR(2),
  zipcode INT
);
```

Possible Solution:

填充一些0字符，使得强制对齐。



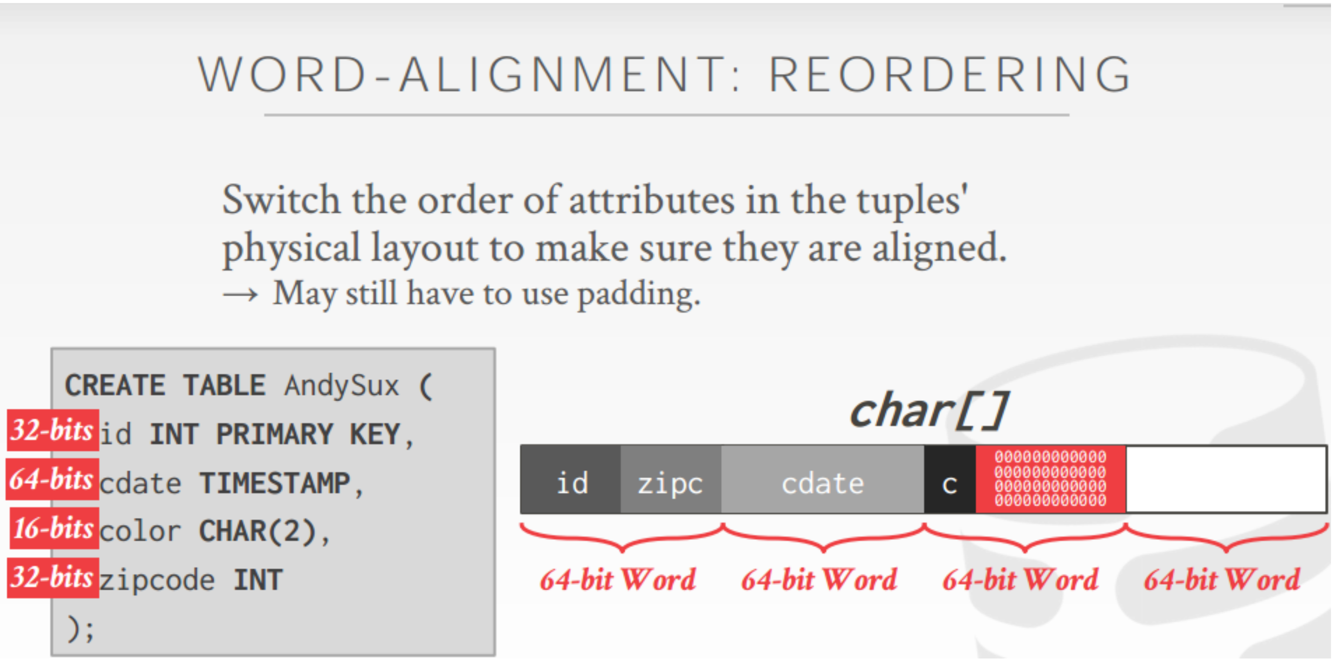
WORD-ALIGNMENT: PADDING

Add empty bits after attributes to ensure that tuple is word aligned.

```
CREATE TABLE AndySux (
32-bits id INT PRIMARY KEY,
64-bits cdate TIMESTAMP,
16-bits color CHAR(2),
32-bits zipcode INT
);
```

char[]

也可以尝试先排序，尽量把字段对齐地放在一个64bit word里面。但可能也会需要padding



WORD-ALIGNMENT: REORDERING

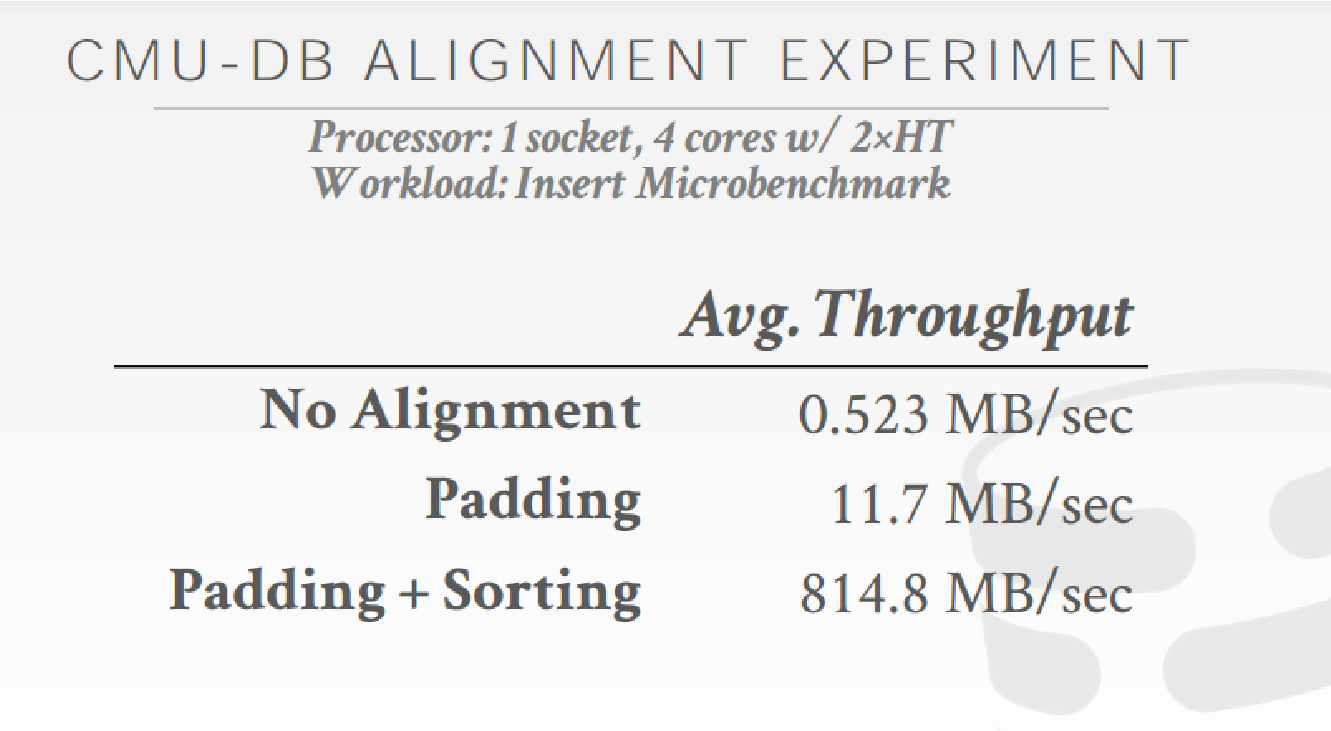
Switch the order of attributes in the tuples' physical layout to make sure they are aligned.

→ May still have to use padding.

```
CREATE TABLE AndySux (
32-bits id INT PRIMARY KEY,
64-bits cdate TIMESTAMP,
16-bits color CHAR(2),
32-bits zipcode INT
);
```

char[]

测试结果：可以看到如果解决好字段存储的对齐问题，性能提高非常明显



CMU-DB ALIGNMENT EXPERIMENT

Processor: 1 socket, 4 cores w/ 2×HT
Workload: Insert Microbenchmark

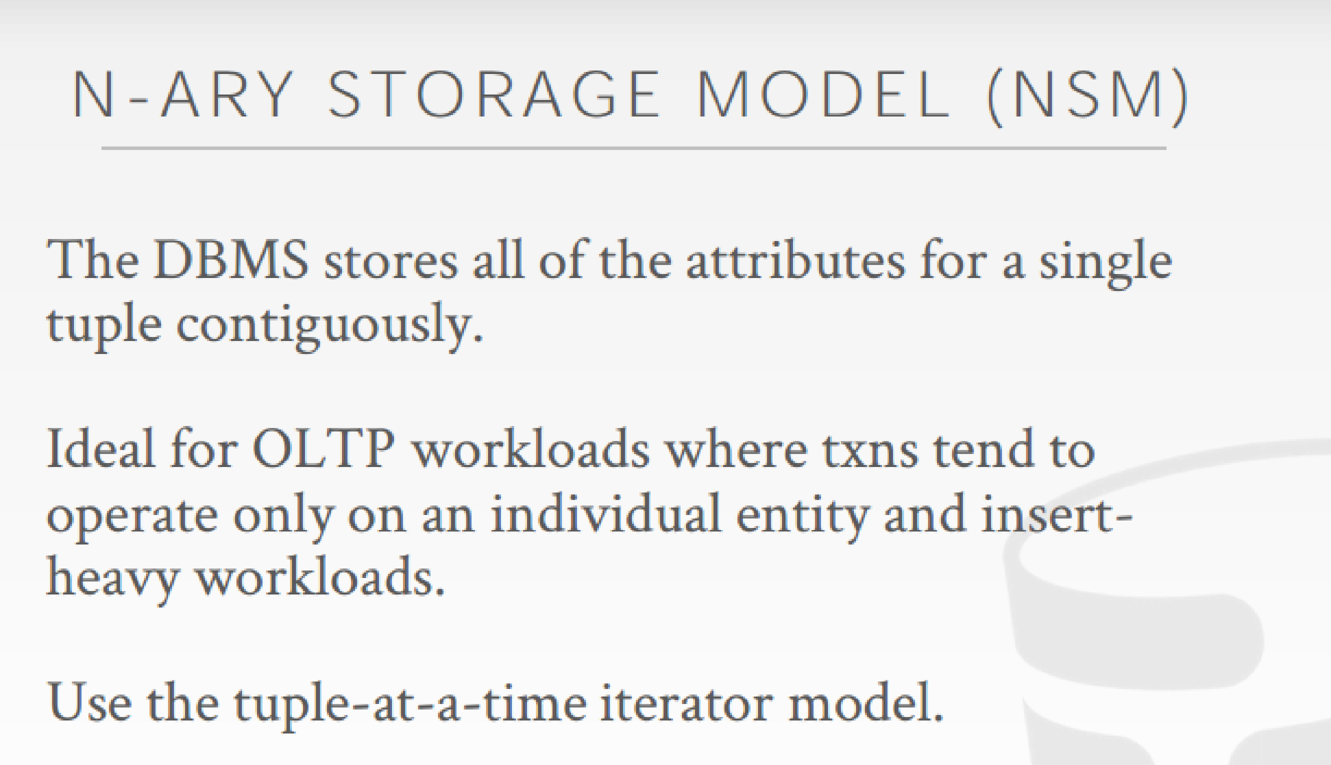
	Avg. Throughput
No Alignment	0.523 MB/sec
Padding	11.7 MB/sec
Padding + Sorting	814.8 MB/sec

Storage Models

- N-ARY STORAGE MODEL(NSM)

行式存储

一个tuple中所有的属性都连续存储在一起。适合OLTP因为OLTP通常涉及少量的tuple及tuple的所有attributes.



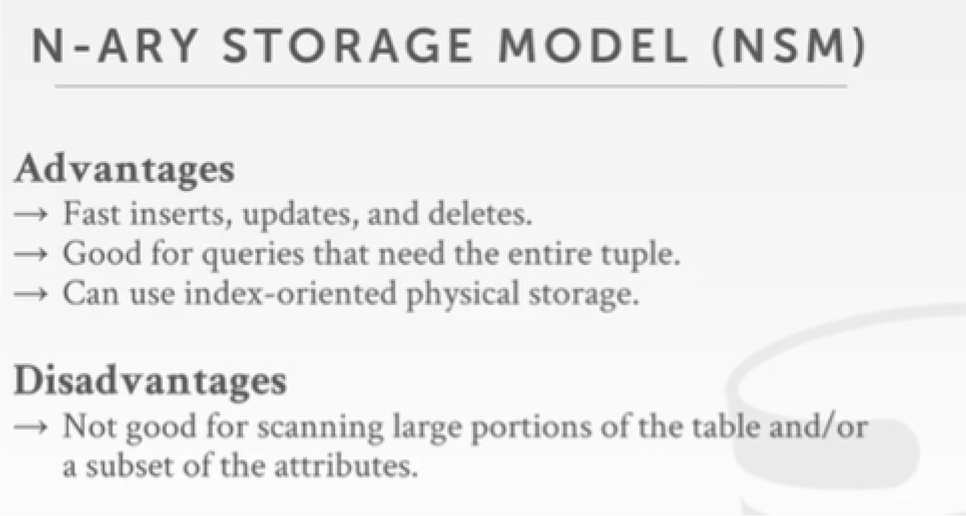
N-ARY STORAGE MODEL (NSM)

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.

但是不适合OLAP场景，因为OLAP通常是需要某几个列，而不是全部列，举个极端例子如果你只需要海量数据某一个列的数据来做分析，那么你需要把海量数据所有的tuple的所有attribute都加载出来。



N-ARY STORAGE MODEL (NSM)

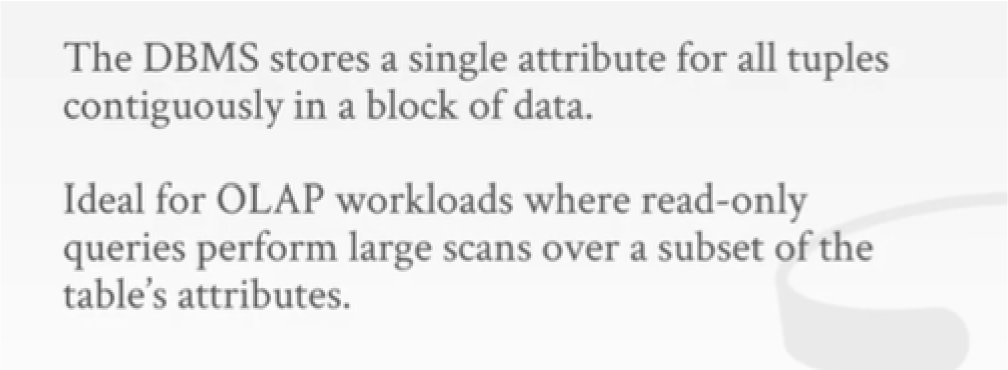
Advantages
→ Fast inserts, updates, and deletes.
→ Good for queries that need the entire tuple.
→ Can use index-oriented physical storage.

Disadvantages
→ Not good for scanning large portions of the table and/or a subset of the attributes.

- DECOMPOSITION STORAGE MODEL

列式存储，海量数据的某一列属性连续存储在一起，减少了把数据从索引加载到内存的waste work,因为你只需要读取你所用到的数据

OLAP场景是查询多，但对于增删改会比较慢

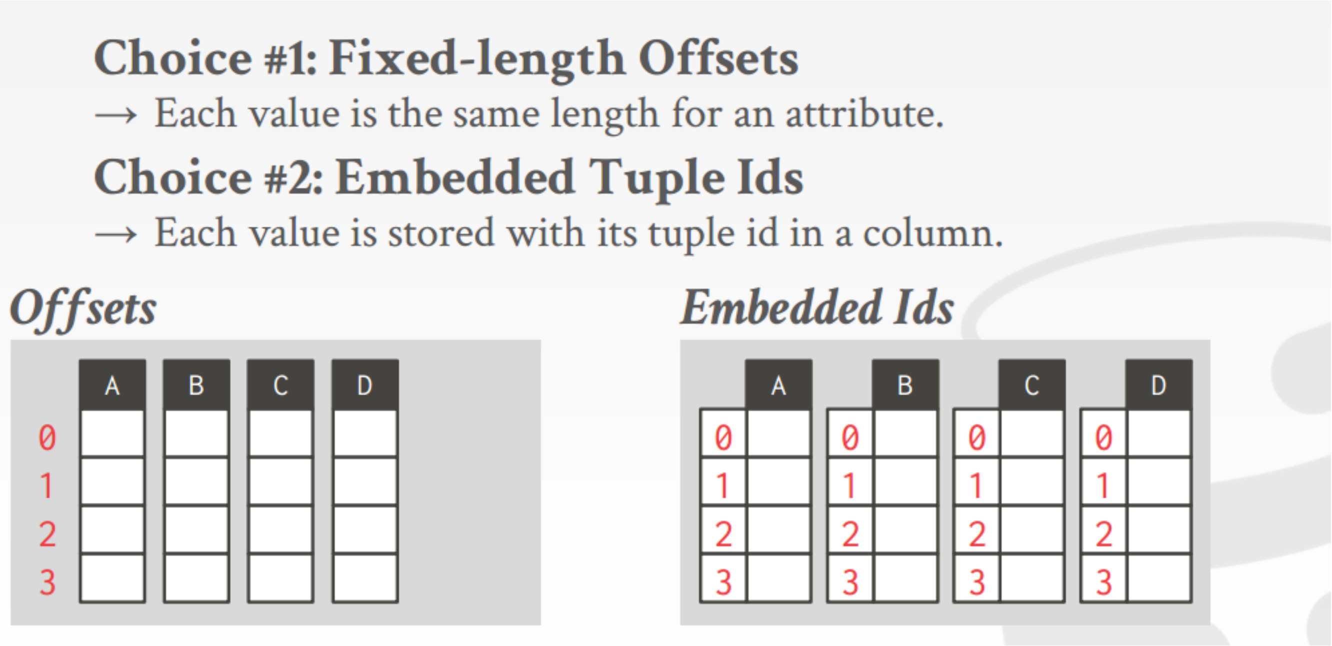


The DBMS stores a single attribute for all tuples contiguously in a block of data.

Ideal for OLAP workloads where read-only queries perform large scans over a subset of the table's attributes.

具体实现细节：

- Tuple Identification



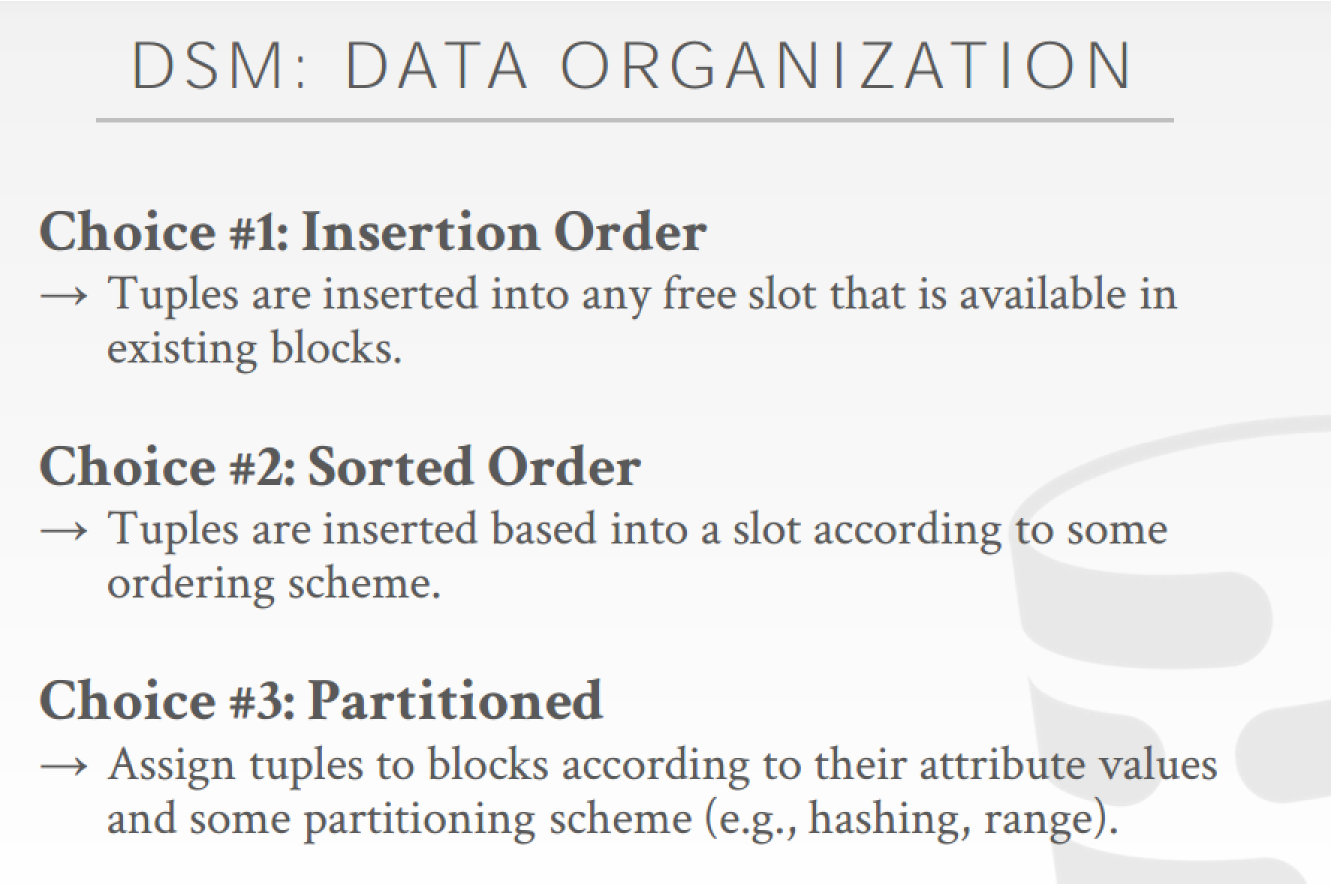
Choice #1: Fixed-length Offsets
→ Each value is the same length for an attribute.

Choice #2: Embedded Tuple Ids
→ Each value is stored with its tuple id in a column.

Offsets

Embedded Ids

- Data Organization



DSM: DATA ORGANIZATION

Choice #1: Insertion Order

→ Tuples are inserted into any free slot that is available in existing blocks.

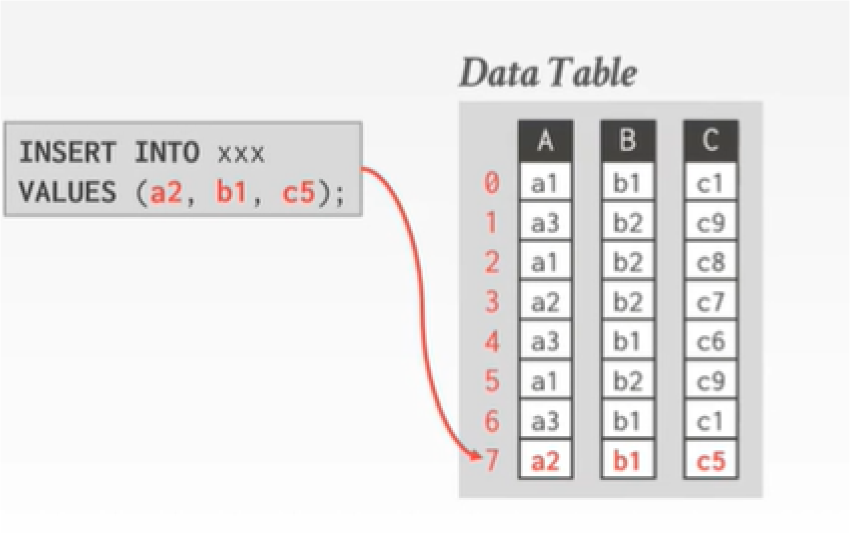
Choice #2: Sorted Order

→ Tuples are inserted based into a slot according to some ordering scheme.

Choice #3: Partitioned

→ Assign tuples to blocks according to their attribute values and some partitioning scheme (e.g., hashing, range).

Insert Order

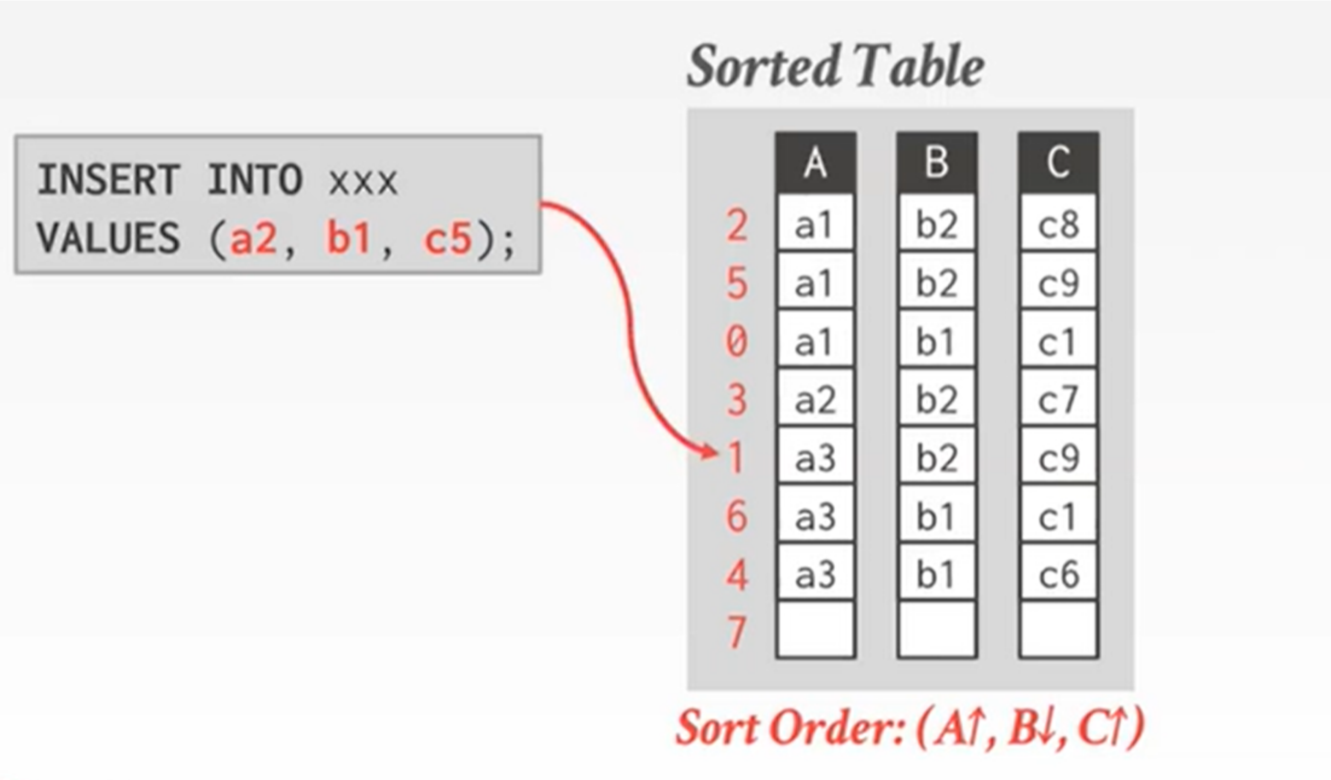


```
INSERT INTO xxx
VALUES (a2, b1, c5);
```

Data Table

	A	B	C
0	a1	b1	c1
1	a3	b2	c9
2	a1	b2	c8
3	a2	b2	c7
4	a3	b1	c6
5	a1	b2	c9
6	a3	b1	c1
7	a2	b1	c5

Sorted Order



```
INSERT INTO xxx
VALUES (a2, b1, c5);
```

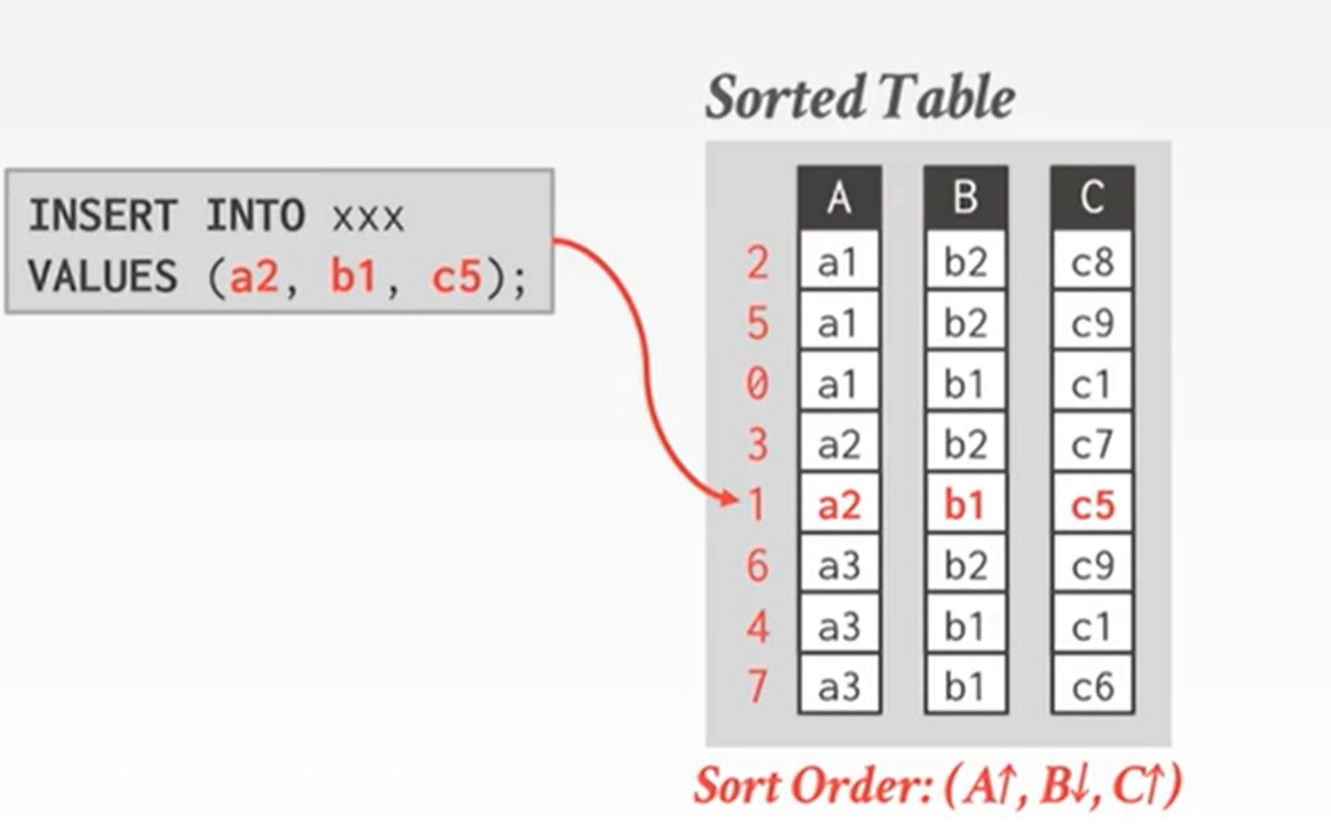
Sorted Table

	A	B	C
2	a1	b2	c8
5	a1	b2	c9
0	a1	b1	c1
3	a2	b2	c7
1	a3	b2	c9
6	a3	b1	c1
4	a3	b1	c6
7			

Sort Order: (A↑, B↓, C↑)

插入的时候需要保证Sorted Table的全局有序

这个时候可能需要把该tuple插入位置及之后的数据全部后移



```
INSERT INTO xxx
VALUES (a2, b1, c5);
```

Sorted Table

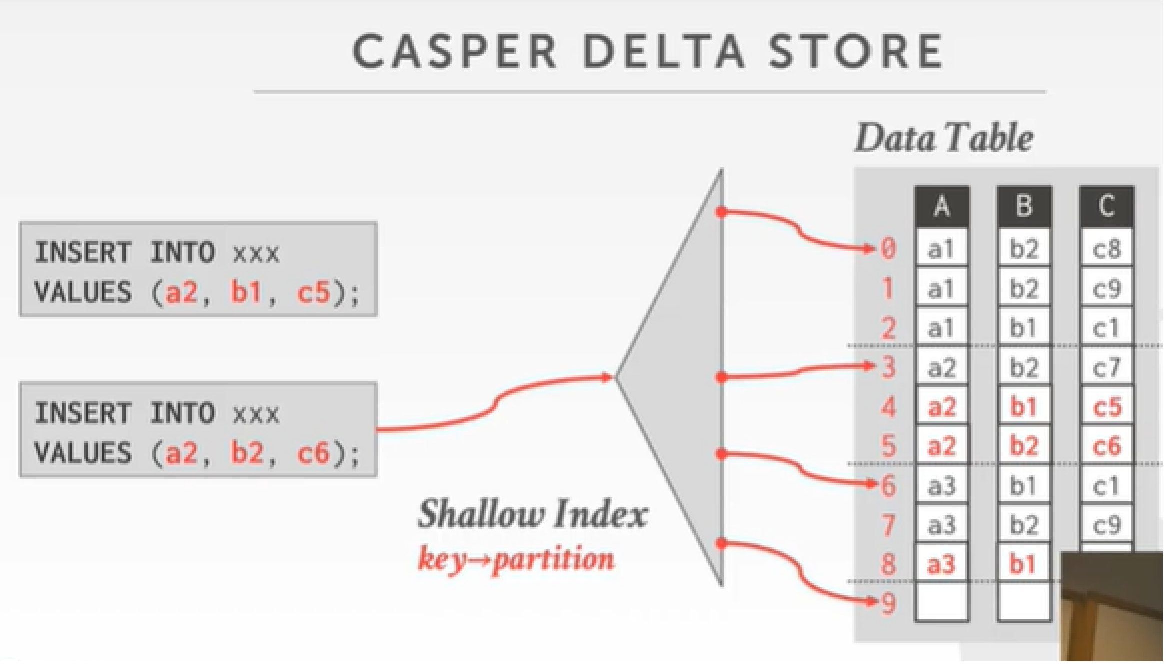
	A	B	C
2	a1	b2	c8
5	a1	b2	c9
0	a1	b1	c1
3	a2	b2	c7
1	a2	b1	c5
6	a3	b2	c9
4	a3	b1	c1
7	a3	b1	c6

Sort Order: (A↑, B↓, C↑)

Partitioned

一个二级索引分别指向不同paritition(可以按照Key来分)，插入数据的时候会定位到具体的partition.

那么寻找数据的时候只需要在这个数据的partition内查找，使得OLAP的查询效率得到提高。



CASPER DELTA STORE

```
INSERT INTO xxx
VALUES (a2, b1, c5);
```

```
INSERT INTO xxx
VALUES (a2, b2, c6);
```

Data Table

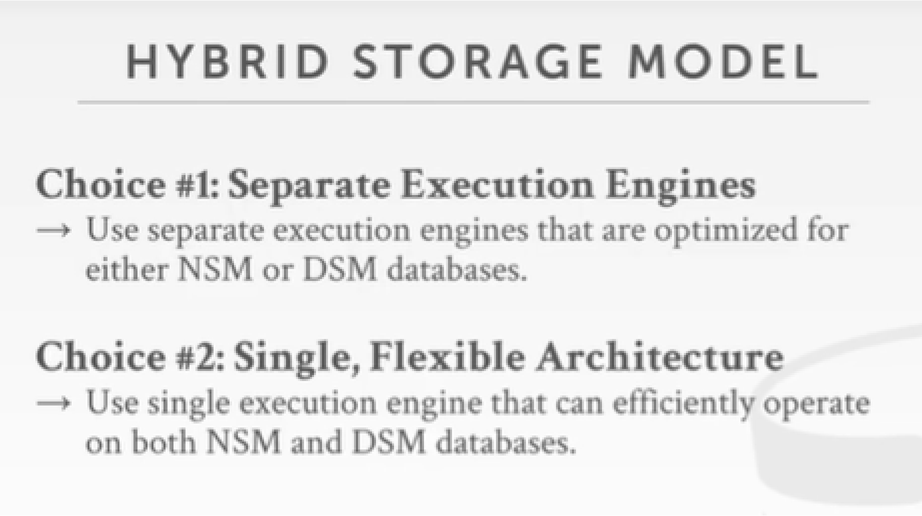
	A	B	C
0	a1	b2	c8
1	a1	b2	c9
2	a1	b1	c1
3	a2	b2	c7
4	a2	b1	c5
5	a2	b2	c6
6	a3	b1	c1
7	a3	b2	c9
8	a3	b1	
9			

但事实上我们可以结合两种storage model,因为可以把数据库的数据进行冷热划分:

hot data：刚刚insert到DB的数据, 很有可能最近还会被更新，适合增删改的OLTP，即行式存储。

cold data: 已经insert到DB很久的数据，被更新的几率已经不大，适合查的OLAP，即列式存储。

阿里云的AnalyticDB就是采用这种行列混存的形式存储冷热数据。



HYBRID STORAGE MODEL

Choice #1: Separate Execution Engines

→ Use separate execution engines that are optimized for either NSM or DSM databases.

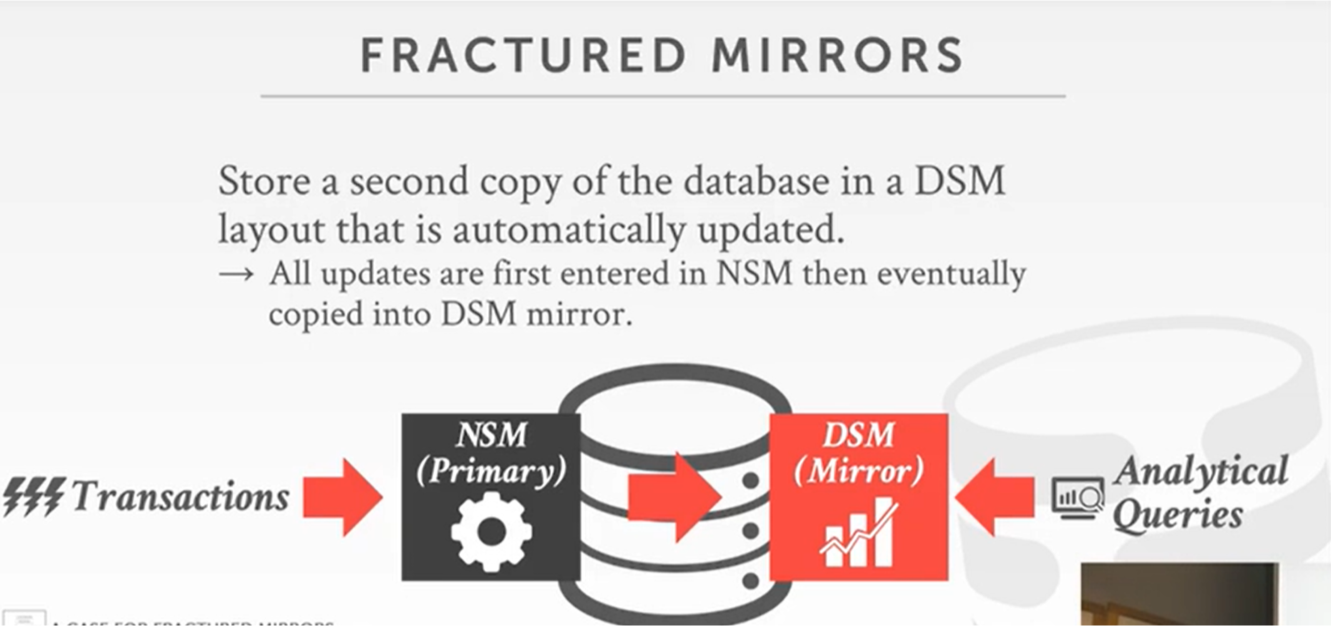
Choice #2: Single, Flexible Architecture

→ Use single execution engine that can efficiently operate on both NSM and DSM databases.

方案一：对行列存储分别使用不同的执行引擎

所有由事务插入到数据库的数据主要都是**行式存储。**

仅当分析性的Query来到要查询数据库数据的时候，要把行式存储的数据复制成**列式存储**的格式



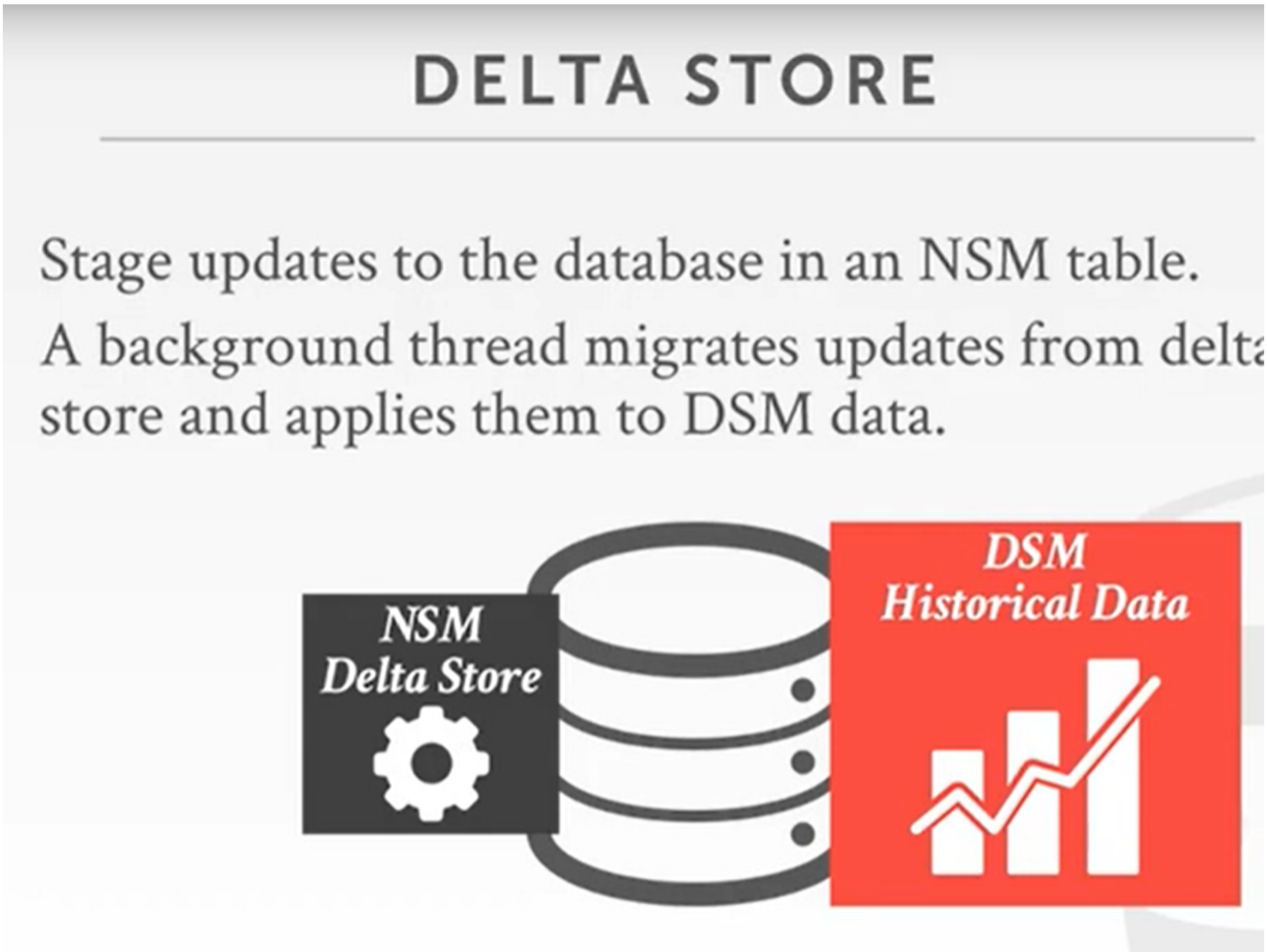
FRACTURED MIRRORS

Store a second copy of the database in a DSM layout that is automatically updated.

→ All updates are first entered in NSM then eventually copied into DSM mirror.

方案二：只用一套执行引擎

关于delta store可以参考这篇https://aboutsqlserver.com/2014/05/06/clustered-columnstore-indexes-exploring-delta-store-and-delete-bitmap/

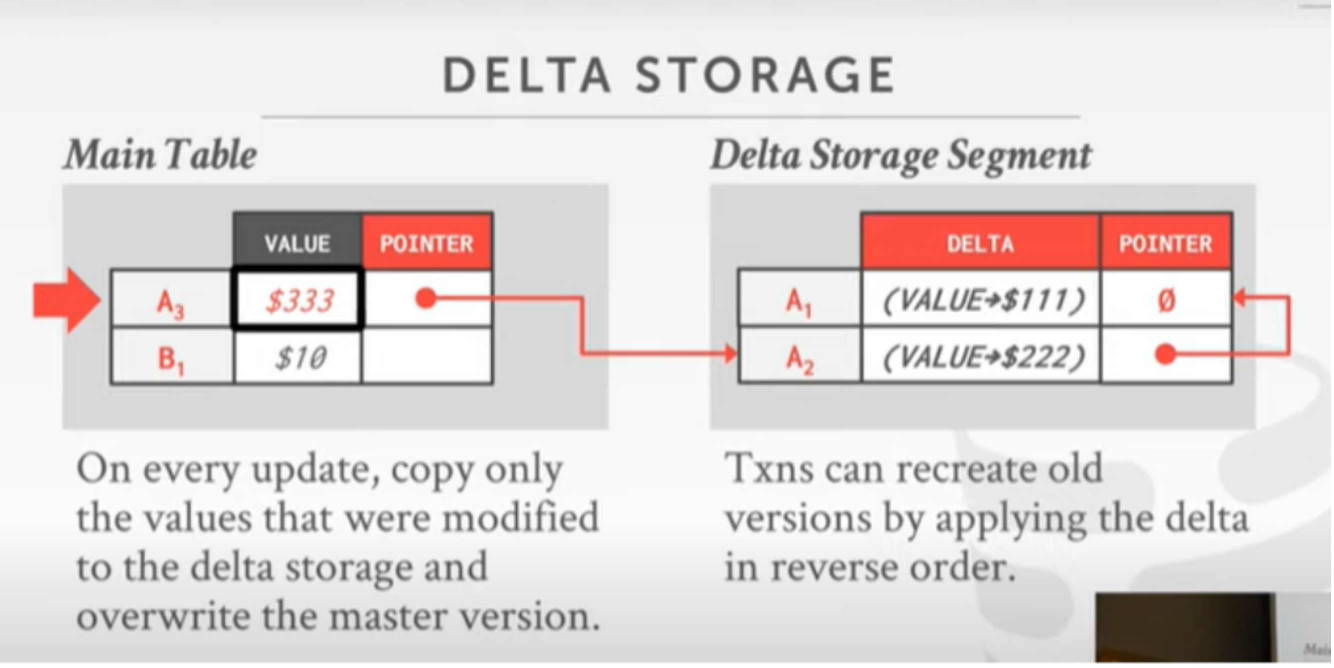


DELTA STORE

Stage updates to the database in an NSM table.

A background thread migrates updates from delta store and applies them to DSM data.

回忆一下第三节课讲的Delta Storage: main table只存储最新版本的数据，然后通过指针把历史版本数据连接在一起。



DELTA STORAGE

Main Table

	VALUE	POINTER
A_3	$333	●
B_1	$10	

Delta Storage Segment

	DELTA	POINTER
A_1	(VALUE→$111)	Ø
A_2	(VALUE→$222)	●

On every update, copy only the values that were modified to the delta storage and overwrite the master version.

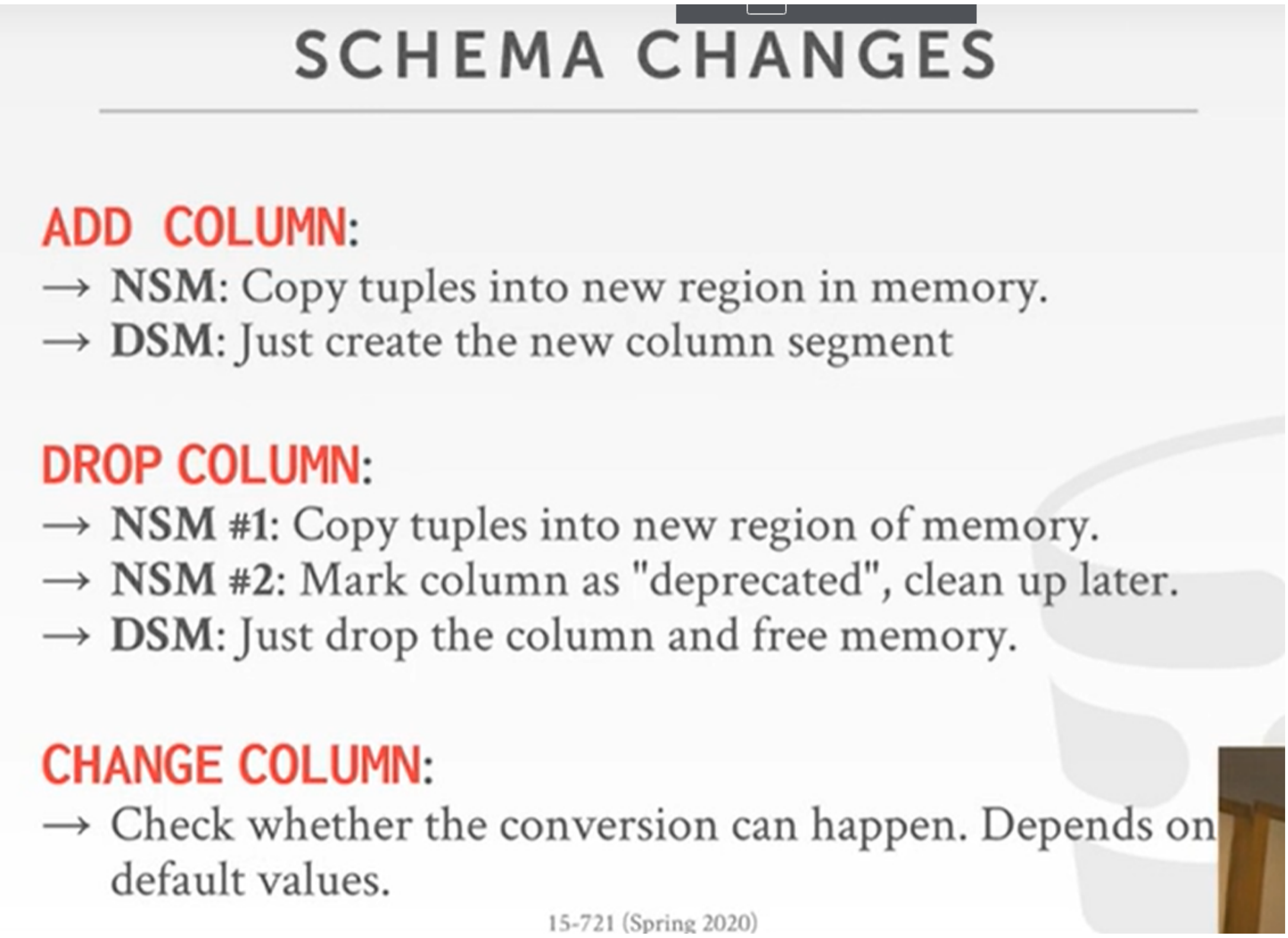
Txns can recreate old versions by applying the delta in reverse order.

System Catalogs

数据库的Catalog指的是什么？一个数据库系统会包括多个Catalog，每个Catalog下面又有多个Schema。每个Schema下面包含了DB的实例如table,view. 可以理解为命名空间层面上用于进行租户隔离的机制。

Schema Changes

NSM:行存 DSM:列存



SCHEMA CHANGES

ADD COLUMN:

→ **NSM**: Copy tuples into new region in memory.

→ **DSM**: Just create the new column segment

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 can happen. Depends on default values.