Tips ‘n Tricks with ColumnStore

Jim Tommaney
Alibaba Cloud

2006-2014 - InfiniDB Chief Architect/CTO
Tips ‘n Tricks with ColumnStore about Jim Tommaney

- 25+ years data architecture, modeling, tuning
- 2006-2014 Chief Architect/CTO for InfiniDB (now ColumnStore)
- Production: InfiniDB, MySQL, Oracle, Postgres, Redshift, Snowflake
- Verticals: Telecom, Web/Mobile Marketing, Genomics, Retail, Manufacturing
Redshift is described as "column-oriented"
HBase is also described as "column-oriented"

Redshift Architecture \(\equiv\) HBase Architecture

**Columnar:**
Vertica, Redshift, ColumnStore, InfoBright, Vectorwise, Snowflake

**Column Family:** BigTable, HBase, Cassandra (not this talk)

Additional discussion from Daniel Abadi:
Short Background on Columnar
Column Restriction and Projection

- Automatic Vertical Partitioning + Horizontal Partitioning
- Just-In-Time Materialization
Columnar Optimal Use Case

Columnar ~10x worse I/O
About 100x slower queries
About 100x slower inserts

Columnar ~10x better I/O
About 100x faster queries
About 100x faster inserts
Physical I/O cost is minimized

```
select l_linenumber, avg(l_quantity), count(*) from lineitem where l_discount < 0.1 group by 1;
```

```
<table>
<thead>
<tr>
<th>l_linenumber</th>
<th>avg(l_quantity)</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>25.502396</td>
<td>31177895</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25.502795</td>
<td>46753111</td>
</tr>
</tbody>
</table>
```

7 rows in set, 1 warning (10.191 sec)

```
select calgettrace()
```

```
*************************** 1. row ***************************
calgettrace():
Desc Mode Table TableOID ReferencedColumns PIO LIO PBE Elapsed Rows
BPS PM lineitem 3000 (l_discount,l_linenumber,l_quantity) 1171958 1173541 0 10.181 6489
```

```
select l_linenumber, avg(l_quantity), count(*) from lineitem where l_discount < 0.1 group by 1;
```

```
<table>
<thead>
<tr>
<th>l_linenumber</th>
<th>avg(l_quantity)</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>25.502396</td>
<td>31177895</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25.502795</td>
<td>46753111</td>
</tr>
</tbody>
</table>
```

7 rows in set, 1 warning (7.747 sec)

```
select calgettrace()
```

```
*************************** 1. row ***************************
calgettrace():
Desc Mode Table TableOID ReferencedColumns PIO LIO PBE Elapsed Rows
BPS PM lineitem 3000 (l_discount,l_linenumber,l_quantity) 1171958 1173541 0 10.181 6489
```

Physical I/O vs Cached

10.2 seconds PIO vs 7.7 seconds cached

1,171,958 blocks vs 0 blocks PIO
## Internal Column Sizes

<table>
<thead>
<tr>
<th>Internal Column Sizes</th>
<th>Column Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Byte</td>
<td>BOOLEAN, TINYINT, CHAR(1), VARCHAR(1)</td>
</tr>
<tr>
<td>2-Byte</td>
<td>SMALLINT, CHAR(2), VARCHAR(2), DECIMAL</td>
</tr>
<tr>
<td>4-Byte</td>
<td>INT, DECIMAL, FLOAT, DATE, CHAR(3 or 4), VARCHAR(3 or 4)</td>
</tr>
<tr>
<td>8-Byte</td>
<td>BIGINT, DECIMAL, DOUBLE, DATETIME, CHAR(5 - 8), VARCHAR(5 - 8)</td>
</tr>
<tr>
<td>8-Byte + Variable Length</td>
<td>CHAR(&gt;8), VARCHAR(&gt;8), TEXT, BLOB, ETC</td>
</tr>
</tbody>
</table>
Behind the scenes

MySQL Processes

Row storage

1 thread

Execution Mgr
1 to 8 threads

Parallel Workers
16 threads (or more)

Columnar storage

ColumnStore
Utility Statements
```sql
select calSetTrace(1);

select l_shipinstruct, count(*) from lineitem group by 1. /* < your query >. */

SELECT calGetTrace()

*************************** 1. row ***************************
calGetTrace():

<table>
<thead>
<tr>
<th>Desc</th>
<th>Mode</th>
<th>Table</th>
<th>TableOID</th>
<th>ReferencedColumns</th>
<th>PIO</th>
<th>LIO</th>
<th>PBE</th>
<th>Elapsed</th>
<th>Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPS</td>
<td>PM</td>
<td>lineitem</td>
<td>3000</td>
<td>(l_shipinstruct)</td>
<td>0</td>
<td>527380</td>
<td>0</td>
<td>10.901</td>
<td>3708</td>
</tr>
<tr>
<td>TAS</td>
<td>UM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.889</td>
<td>4</td>
</tr>
<tr>
<td>TNS</td>
<td>UM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.000</td>
<td>4</td>
</tr>
</tbody>
</table>

PM - Performance Module – Distributed, parallel processing
UM - User Module – Final aggregation, multi-threaded

PIO – Physical I/O - Blocks read from storage
LIO – Logical I/O - Blocks touched, from memory
PBE – Partition Blocks Eliminated – Blocks skipped with min/max meta-data check
```
sel**ect calflushcache();

select l_shipinstruct_code /*CHAR(1)*/ , count(*) c from lineitem group by l;

4 rows in set, 1 warning (3.113 sec)

*************************** 1. row ***************************
```
```calgettrace():
Desc Mode Table TableOID ReferencedColumns PIO LIO RBE Elapsed Rows
BPS PM lineitem 3000 (l_shipinstruct_code) 0 58598 0 3.105 3708
```
```
sel**ect calflushcache();

select l_shipinstruct_code /*CHAR(1)*/ , count(*) c from lineitem group by l;

4 rows in set, 1 warning (4.895 sec)

*************************** 1. row ***************************
```
```calgettrace():
Desc Mode Table TableOID ReferencedColumns PIO LIO RBE Elapsed Rows
BPS PM lineitem 3000 (l_shipinstruct_code) 58598 59314 0 4.887 3708
```
```
/* not useful for production */
```
Desc Lineitem (~1/2 billion rows, scale factor 80)

CREATE TABLE num (n tinyint(4)) ENGINE=Columnstore

select * from num;

select * from shipmode_fk_innodb;
select * from shipmode_fk_columnstore;

+----------+
| l_shipmode | l_shipcode |
+----------+
| AIR       | 1         |
| RAIL      | 4         |
| REG AIR   | 5         |
| MAIL      | 3         |
| TRUCK     | 7         |
| FOB       | 2         |
| SHIP      | 6         |
+----------+

| l_shipcode | tinyint(4) | added |
| l_comment_code | char(2) |
| l_shipinstruct_code | char(1) |
| l_shipdatetime | datetime |
| l_shipdate_yy | tinyint(4) |
| l_shipdate_mm | tinyint(4) |
| l_shipdate_yymm | smallint(6) |

+----------+
| l_shipcode | tinyint(4) | added |
| l_comment_code | char(2) |
| l_shipinstruct_code | char(1) |
| l_shipdatetime | datetime |
| l_shipdate_yy | tinyint(4) |
| l_shipdate_mm | tinyint(4) |
| l_shipdate_yymm | smallint(6) |
+----------+
Narrow column with aggregation (1-1 mapping)

<table>
<thead>
<tr>
<th>l_shipinstruct</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>120011699</td>
</tr>
<tr>
<td>TAKE BACK RETURN</td>
<td>120002249</td>
</tr>
<tr>
<td>COLLECT COD</td>
<td>120010978</td>
</tr>
<tr>
<td>DELIVER IN PERSON</td>
<td>120000203</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>l_shipinstruct</th>
<th>char(25)</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_shipinstruct_code</td>
<td>char(1)</td>
<td>added</td>
</tr>
</tbody>
</table>
Narrow column with aggregation (1-1 mapping)

Select `l_shipinstruct /*char(25)*/`, `count(*)` from lineitem group by `l_shipinstruct`;

<table>
<thead>
<tr>
<th>l_shipinstruct</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>120011699</td>
</tr>
<tr>
<td>TAKE BACK RETURN</td>
<td>120002249</td>
</tr>
<tr>
<td>COLLECT COD</td>
<td>120010978</td>
</tr>
<tr>
<td>DELIVER IN PERSON</td>
<td>120000203</td>
</tr>
</tbody>
</table>

4 rows in set, 1 warning (10.942 sec)

LIO = 527,380

Select `case l_shipinstruct_code` -> when 'C' then 'COLLECT COD' when 'D' then 'DELIVER IN PERSON' -> when 'N' then 'NONE' when 'T' then 'TAKE BACK RETURN' -> else 9 end instruct_code, c -> from ( select `l_shipinstruct_code /*CHAR(1)*/*`, `count(*)` c from lineitem group by `l_shipinstruct_code` ) a;

<table>
<thead>
<tr>
<th>instruct_code</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAKE BACK RETURN</td>
<td>120002249</td>
</tr>
<tr>
<td>DELIVER IN PERSON</td>
<td>120000203</td>
</tr>
<tr>
<td>NONE</td>
<td>120011699</td>
</tr>
<tr>
<td>COLLECT COD</td>
<td>120010978</td>
</tr>
</tbody>
</table>

4 rows in set, 1 warning (3.109 sec)

LIO = 58,598

About 3.5x faster
About 9x reduced LIO, PIO (if needed)

10.9 seconds vs 3.1 seconds

527k LIO vs 59k LIO
Narrow column as filter (1-1 mapping)

```sql
select l_shipinstruct /*char(25)*/, count(*) from lineitem
-> where l_shipinstruct = "TAKE BACK RETURN" group by l;
```

<table>
<thead>
<tr>
<th>l_shipinstruct</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>120011699</td>
</tr>
<tr>
<td>TAKE BACK RETURN</td>
<td>120002249</td>
</tr>
<tr>
<td>COLLECT COD</td>
<td>120010978</td>
</tr>
<tr>
<td>DELIVER IN PERSON</td>
<td>12000203</td>
</tr>
</tbody>
</table>

4 rows in set, 1 warning (7.204 sec)

LIO = 996,155

```sql
select case l_shipinstruct_code
-> when 'C' then 'COLLECT COD' when 'D' then 'DELIVER IN PERSON'
-> when 'N' then 'NONE' when 'T' then 'TAKE BACK RETURN' else 9 end instruct_code, c
-> from ( select l_shipinstruct_code /*CHAR(1)*/, count(*) c from lineitem
-> where l_shipinstruct_code = 'T' group by l Shipinstruct_code ) a;
```

<table>
<thead>
<tr>
<th>instruct_code</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAKE BACK RETURN</td>
<td>120002249</td>
</tr>
<tr>
<td>DELIVER IN PERSON</td>
<td>12000203</td>
</tr>
<tr>
<td>NONE</td>
<td>120011699</td>
</tr>
<tr>
<td>COLLECT COD</td>
<td>120010978</td>
</tr>
</tbody>
</table>

4 rows in set, 1 warning (2.108 sec)

LIO = 58,598

About 3.5x faster
About 17x reduced LIO

7.2 seconds vs 2.1 seconds
996k LIO vs 59k LIO
Deferred case statements

**Standard:**

```sql
select case l_shipcode when 1 then 'AIR' when 2 then 'FOB'
    when 3 then 'MAIL' when 4 then 'RAIL'
    when 5 then 'REG AIR' when 6 then 'SHIP'
    when 7 then 'TRUCK' else 9 end shipcode,
count(*) from lineitem group by 1;
```

**Nested Group By:**

```sql
select case l_shipcode when 1 then 'AIR' when 2 then 'FOB'
    when 3 then 'MAIL' when 4 then 'RAIL'
    when 5 then 'REG AIR' when 6 then 'SHIP'
    when 7 then 'TRUCK' else 9 end shipcode, c
from ( select l_shipcode, count(*) c
from lineitem group by 1 ) a;
```
Deferred case statements

Standard:

```sql
select case l_shipcode when 1 then 'AIR' when 2 then 'FOB'
  ->  when 3 then 'MAIL' when 4 then 'RAIL' when 5 then 'REG AIR'
  ->  when 6 then 'SHIP' when 7 then 'TRUCK' else 9 end shipcode,
  ->  count(*)
from lineitem
group by 1;
```

<table>
<thead>
<tr>
<th>shipcode</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR</td>
<td>68580321</td>
</tr>
<tr>
<td>RAIL</td>
<td>68587640</td>
</tr>
</tbody>
</table>

7 rows in set, 1 warning (28.043 sec) 480,000,000 case statements

Nested Group By:

```sql
select case l_shipcode when 1 then 'AIR' when 2 then 'FOB'
  ->  when 3 then 'MAIL' when 4 then 'RAIL' when 5 then 'REG AIR'
  ->  when 6 then 'SHIP' when 7 then 'TRUCK' else 9 end shipcode, c
from (select l_shipcode,
  ->  count(*)
c from lineitem
group by 1 ) a ;
```

<table>
<thead>
<tr>
<th>shipcode</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAIL</td>
<td>68587640</td>
</tr>
<tr>
<td>FOB</td>
<td>68575597</td>
</tr>
</tbody>
</table>

7 rows in set, 1 warning (3.172 sec) 7 case statements
Create table, insert few rows

```sql
CREATE TABLE `shipmode_fk_innodb` (  
    `l_shipmode` char(10) DEFAULT NULL,  
    `l_shipcode` tinyint(4) DEFAULT NULL  
) ENGINE=InnoDB;

Query OK, 0 rows affected (0.006 sec)

insert into shipmode_fk_innodb select distinct l_shipmode, l_shipcode  
    from lineitem where l_orderkey < 1000000;

Query OK, 7 rows affected, 1 warning (0.163 sec)
Records: 7  Duplicates: 0  Warnings: 0

CREATE TABLE `shipmode_fk_columnstore` (  
    `l_shipmode` char(10) DEFAULT NULL,  
    `l_shipcode` tinyint(4) DEFAULT NULL  
) ENGINE=Columnstore;

Query OK, 0 rows affected (0.165 sec)

insert into shipmode_fk_columnstore select distinct l_shipmode, l_shipcode  
    from lineitem where l_orderkey < 1000000;

Query OK, 7 rows affected, 1 warning (1.679 sec)
Records: 7  Duplicates: 0  Warnings: 0
```
Deferred join

**Standard:**

```sql
select dim.l_shipmode, count(*)
  from lineitem
  join shipmode_fk_columnstore dim
using (l_shipcode) group by 1;
```

**Nested Group By:**

```sql
select dim.l_shipmode, sum(c)
  from (select l_shipcode, count(*) c
           from lineitem group by 1
      ) L
  join shipmode_fk_columnstore dim
using (l_shipcode) group by 1;
```
Deferred join

**Standard:**

```
select dim.l_shipmode, count(*)
  -> from lineitem join shipmode_fk_columnstore dim using (l_shipcode)
  -> group by 1;
```

7 rows in set, 1 warning (10.721 sec) 480,000,000 rows joined

**Nested Group By :**

```
select dim.l_shipmode, sum(c)
  -> from ( select l_shipcode, count(*) c from lineitem group by 1 ) L
  -> join shipmode_fk_columnstore dim using (l_shipcode) group by 1;
```

7 rows in set, 1 warning (3.239 sec) 7 rows joined
Insert select many rows (10 million)

CREATE TABLE `lineitem_innodb` (
    `l_orderkey` int(11) DEFAULT NULL,
    ...
    `l_shipdate_yymm` smallint(6) DEFAULT NULL,
    KEY `i_l_shipdate` (`l_shipdate`),
    KEY `i_l_suppkey_partkey` (`l_partkey`,`l_suppkey`),
    KEY `i_l_partkey` (`l_partkey`),
    KEY `i_l_suppkey` (`l_suppkey`),
    KEY `i_l_receiptdate` (`l_receiptdate`),
    KEY `i_l_orderkey` (`l_orderkey`),
    KEY `i_l_orderkey_quantity` (`l_orderkey`,`l_quantity`),
    KEY `i_l_commitdate` (`l_commitdate`))

insert into lineitem_innodb select * from lineitem_source_Innodb;
Query OK, 10000000 rows affected (34 min 29.838 sec)
Records: 10000000  Duplicates: 0  Warnings: 0

insert into lineitem_Columnstore select * from lineitem_source_Columnstore;
Query OK, 10000000 rows affected, 1 warning (1 min 57.115 sec)
Records: 10000000  Duplicates: 0  Warnings: 0
Load Data Infile, cpimport 10 million

InnoDB
load data infile

load data infile '/home/mysql/lineitem_10m' into table lineitem_Innodb
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '
';

Query OK, 10000000 rows affected (33 min 14.988 sec)
Records: 10000000  Deleted: 0  Skipped: 0  Warnings: 0

ColumnStore
load data infile

load data infile '/home/mysql/lineitem_10m' into table
lineitem_columnstore FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '
';

Query OK, 10000000 rows affected (1 min 21.691 sec)
Records: 10000000  Deleted: 0  Skipped: 0  Warnings: 0

ColumnStore

cpimport -s ""," -E "" -l lineitem_10m tpch80 lineitem_columnstore

2019-05-24 01:49:23 (3164) INFO : No of Read Threads Spawned = 1
2019-05-24 01:49:23 (3164) INFO : No of Parse Threads Spawned = 3
Cross-Engine Join

```
select  dim.l_shipmode, count(*)
from    lineitem
join    shipmode_fk_innodb dim
using   (l_shipcode)
group by 1;
```

```
select  dim.l_shipmode, count(*)
from    lineitem
join    shipmode_fk_columnstore dim
using   (l_shipcode)
group by 1;
```
Cross-Engine Join: 7 rows x .48 billion

InnoDB dimension

```sql
select dim.l_shipmode, count(*)
  -> from lineitem join shipmode_fk_innodb dim using (l_shipcode)
  -> group by 1;
...
7 rows in set, 1 warning (9.899 sec) /* InnoDB dimension is faster */
```

ColumnStore dimension

```sql
select calgettrace()

*************************** 1. row ***************************
 calgettrace():
 Desc Mode Table TableOID ReferencedColumns PIO LIO PBE Elapsed Rows
  CES UM - - - - - - 0.000 7
  BPS PM lineitem 3000 (l_shipcode) 0 58598 0 9.883 6489
  HJS PM lineitem-dim 3000 - - - - - - -
  TAS UM - - - - - - 9.871 7
  TNS UM - - - - - - 0.000 7
```

```sql
select dim.l_shipmode, count(*)
  -> from lineitem join shipmode_fk_columnstore dim using (l_shipcode)
  -> group by 1;
...
7 rows in set, 1 warning (10.721 sec)
```

```sql
select calgettrace()

*************************** 1. row ***************************
 calgettrace():
 Desc Mode Table TableOID ReferencedColumns PIO LIO PBE Elapsed Rows
  BPS PM dim 3081 (l_shipcode,l_shipmode) 0 4 0 0.002 7
  BPS PM lineitem 3000 (l_shipcode) 0 58598 0 10.708 6489
  HJS PM lineitem-dim 3000 - - - - - - -
  TAS UM - - - - - - 10.690 7
  TNS UM - - - - - - 0.000 7
```
Cross-Engine Join + deferred join

select dim.l_shipmode, sum(c)
from ( select l_shipcode, count(*) c
    from lineitem group by 1 ) L
join shipmode_fk_innodb dim
using (l_shipcode) group by 1 ;

7 rows in set, 1 warning (3.223 sec)

select dim.l_shipmode, sum(c)
from ( select l_shipcode, count(*) c
    from lineitem group by 1 ) L
join shipmode_fk_columnstore dim
using (l_shipcode) group by 1 ;

7 rows in set, 1 warning (3.239 sec)
Behind the scenes

MySQL Processes

1 thread

ColumnStore

Execution Mgr
1 to 8 threads

Parallel Workers
16 threads (or more)

Read 7 Rows
Comment, add 2 byte field on leading

<table>
<thead>
<tr>
<th>l_comment</th>
<th>varchar(44)</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca</td>
<td></td>
<td></td>
</tr>
<tr>
<td>co</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>qu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>l_comment_code</th>
<th>char(2)</th>
<th>added</th>
</tr>
</thead>
</table>
Comment, add 2 byte field on leading...

<table>
<thead>
<tr>
<th>l_comment</th>
<th>varchar(44)</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_comment_code</td>
<td>char(2)</td>
<td>added</td>
</tr>
</tbody>
</table>

```
select count(*) from lineitem where l_comment like 'ul%';
+--------+
| count(*) |
+--------+
| 5458267 |
+--------+
1 row in set (25.753 sec)

select count(*) from lineitem where l_comment_code = 'ul';
+--------+
| count(*) |
+--------+
| 5458267 |
+--------+
1 row in set (3.707 sec)
```

25.8 seconds vs 3.7 seconds
**Cartesian hack**

```sql
select count(*) from shipmode_fk_innodb;
+----------+
| count(*) |
+----------+
| 7        |
+----------+

select count(*)
-> from shipmode_fk_innodb a
  -> **cross join** shipmode_fk_innodb b;
+----------+
| count(*) |
+----------+
| 49       |
+----------+

select count(*)
-> from shipmode_fk_columnstore a
  -> **cross join** shipmode_fk_columnstore b;
ERROR 1815 (HY000): Internal error: IDB-1000: 'a' and 'b' are not joined
```
Cartesian hack

```sql
select count(*)
  -> from shipmode_fk_columnstore a
  -> join shipmode_fk_columnstore b
  -> on (sign(a.l_shipcode) = sign(b.l_shipcode) );
+
| count(*) |
+----------+
|   49     |
+----------+
1 row in set (0.012 sec)
```

```sql
select * from shipmode_fk_columnstore ;
+
| l_shipmode | l_shipcode |
+------------+
| AIR        |     1     |
| RAIL       |     4     |
| REG AIR    |     5     |
| MAIL       |     3     |
| TRUCK      |     7     |
| FOB        |     2     |
| SHIP       |     6     |
+------------+
7 rows in set (0.007 sec)
```
CREATE TABLE num (
  n tinyint(4)
) ENGINE=Columnstore

select * from num;

<table>
<thead>
<tr>
<th></th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

```sql
select dim.l_shipmode, count(*)
from lineitem_innodb /* 10m */
join shipmode_fk_innodb dim
using (l_shipcode)
group by 1;
```

```sql
CREATE TABLE num (n tinyint(4)) ENGINE=Columnstore
```

```
select * from num;
```

<table>
<thead>
<tr>
<th></th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

```sql
select dim.l_shipmode, count(*)
from lineitem_columnstore /* 10m */
join shipmode_fk_columnstore dim
using (l_shipcode)
group by 1;
```

```sql
select dim.l_shipmode, count(*)
from lineitem_innodb /* 10m */
join shipmode_fk_innodb dim
using (l_shipcode)
join num
  on (sign(l_shipcode) = n)
group by 1;
```
Pull InnoDB into ColumnStore

InnoDB

select dim.l_shipmode, count(*)
    from lineitem_innodb
    where l_shipmode = 'N'
    join shipmode_fk_innodb dim
    using (l_shipcode)
    group by 1;

Native ColumnStore

select dim.l_shipmode, count(*)
    from lineitem_columnstore
    join shipmode_fk_columnstore dim
    using (l_shipcode)
    group by 1;

InnoDB into ColumnStore

select dim.l_shipmode, count(*)
    from lineitem_innodb
    join shipmode_fk_innodb dim
    join num
    on (sign(l_shipcode) = n)
    group by 1;

96 seconds 1st run
9 seconds 2nd run

0.5 seconds 1st run
0.33 seconds 2nd run

6.6 seconds 1st run
6.3 seconds 2nd run
Behind the scenes

MySQL Processes

ColumnStore

1 thread

Read 7 Rows
Read 10 million rows

Execution Mgr
1 to 8 threads

Parallel Workers
16 threads (or more)

InnoDB into ColumnStore
Views on top of InnoDB and ColumnStore

```
select min(l_shipdate), max(l_shipdate)
from lineitem_1month_innodb;
+-----------------+-----------------+
| min(l_shipdate) | max(l_shipdate) |
|-----------------+-----------------+
| 1995-01-01      | 1995-01-31      |
+-----------------+-----------------+
1 row in set (39.942 sec) (2.818 2nd run)
```

```
select min(l_shipdate), max(l_shipdate)
from lineitem_11month_columnstore;
+-----------------+-----------------+
| min(l_shipdate) | max(l_shipdate) |
|-----------------+-----------------+
| 1995-02-01      | 1995-12-31      |
+-----------------+-----------------+
1 row in set (0.862 sec) (0.650 2nd run)
```
Views on top of InnoDB and ColumnStore

<table>
<thead>
<tr>
<th>l_returnflag</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3092781</td>
</tr>
<tr>
<td>R</td>
<td>3091263</td>
</tr>
</tbody>
</table>

2 rows in set (3.403 sec)

Can we use the view with 4 - 5 second performance?

<table>
<thead>
<tr>
<th>l_returnflag</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12111401</td>
</tr>
<tr>
<td>N</td>
<td>42394320</td>
</tr>
<tr>
<td>R</td>
<td>12118452</td>
</tr>
</tbody>
</table>

3 rows in set (0.679 sec)
Select * Views on InnoDB + ColumnStore

**Experiment 1:**
create or replace view v_lineitem as
select * from lineitem_11m_columnstore where l_shipdate >= '1995-02-01'
union all
select * from lineitem_1m_innodb where l_shipdate < '1995-02-01';

**Experiment 2:**
create or replace view v_lineitem as
select * from lineitem_11m_columnstore where l_shipdate >= '1995-02-01'
union all
select * from ( select a.* from lineitem_1m_innodb_partitioned a 
join num on (n=sign(l_orderkey)) 
where l_shipdate < '1995-02-01' ) b;

) ENGINE=Innodb
PARTITION BY RANGE ( to_days(l_shipdate) )
{
    PARTITION p0 VALUES LESS THAN (to_days('1995-02-01')),
    PARTITION p1 VALUES LESS THAN (to_days('1995-03-01')),
    PARTITION p2 VALUES LESS THAN (to_days('1995-04-01')),
    PARTITION p3 VALUES LESS THAN (to_days('1995-05-01'))
};
InnoDB

ColumnStore

**Select * Views on InnoDB + ColumnStore**

```
select  l_returnflag, count(*) from v_lineitem where l_shipdate >= '1995-02-01' group by 1;
```

```
<table>
<thead>
<tr>
<th>l_returnflag</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12111401</td>
</tr>
<tr>
<td>N</td>
<td>42394320</td>
</tr>
<tr>
<td>R</td>
<td>12118452</td>
</tr>
</tbody>
</table>
```

3 rows in set (24.579 sec)

```
select  l_returnflag, count(*) from v_lineitem where l_shipdate < '1995-02-01' group by 1;
```

```
<table>
<thead>
<tr>
<th>l_returnflag</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3092781</td>
</tr>
<tr>
<td>R</td>
<td>3091263</td>
</tr>
</tbody>
</table>
```

2 rows in set (1 min 0.542 sec)

**Access is too slow!**

*(for these experiments)*

Expect times between 4 and 5 seconds.
Behind the scenes

MySQL Processes

ColumnStore

1 thread

Between 6 million and 66 million Rows

Execution Mgr
1 to 8 threads

Parallel Workers
16 threads (or more)

Select * Views
on InnoDB +
ColumnStore
Query specific views

create or replace view v_lineitem_query1 as

select l_returnflag, count(*) cnt
from lineitem_1m_innodb
where l_shipdate < '1995-02-01'
group by 1
union all

select l_returnflag, count(*) cnt
from lineitem_11m_columnstore
where l_shipdate >= '1995-02-01'
group by 1;
Query specific views

create or replace view v_lineitem_query1 as
select l_returnflag, count(*) cnt
from lineitem_1m_innodb
where l_shipdate < '1995-02-01'
group by 1
union all
select l_returnflag, count(*) cnt
from lineitem_11m_columnstore
where l_shipdate >= '1995-02-01'
group by 1;

select * from v_lineitem_query1 ;

<table>
<thead>
<tr>
<th>l_returnflag</th>
<th>cnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12111401</td>
</tr>
<tr>
<td>N</td>
<td>42394320</td>
</tr>
<tr>
<td>R</td>
<td>12118452</td>
</tr>
<tr>
<td>A</td>
<td>3092781</td>
</tr>
<tr>
<td>R</td>
<td>3091263</td>
</tr>
</tbody>
</table>

5 rows in set (4.360 sec)
Behind the scenes

MySQL Processes

Query Specific Views on InnoDB + ColumnStore

Aggregate 6 million, project 2 rows

Aggregate 66 million, project 3 rows

Union all 5 rows

1 thread

Execution Mgr 1 to 8 threads

Parallel Workers 16 threads (or more)

ColumnStore
Brief Overview of Joins

- All joins are hash-joins including inner, left, right, and subquery
- Small side(s) created first
  Single-pass stream large side joining all small sides

Best for large fact tables: small side is pushed to distributed servers

Small side similar size large side: small side at Execution Manager level

Diagram:
- ColumnStore Execution Manager
  - Server 1: 16 cores
  - Server 2: 16 cores
  - Server n: 16 cores
Recap, Questions

- Narrow columns are faster, more efficient
- Defer operations (joins, or complex expressions) after group by
- Joins applied after group by operations are nearly free
- Row-by-row is slow-by-slow for queries and loads (33 min vs 22 seconds)
- Sign(column) can be used to enable cartesian join (where appropriate)
- Dimension tables can be either ColumnStore or InnoDB
  - rule of thumb, anything under 1 million rows can be either
- $A \bowtie B \bowtie C$ resolved in ColumnStore when any of table is ColumnStore
- Select * views not effective, query specific views powerful
Rate My Session

Schedule
Timezone: Europe/Berlin +02:00

Details
Introducing gh-ost: triggerless, painless, trusted online schema migrations
- 11:20 → 12:10
- Matterhorn 2

Rate & Review
Tap a star to rate
Feedback (optional)
Anonymously
SUBMIT