Apache Phoenix: Transforming HBase into a SQL database

James Taylor
@JamesPlusPlus
http://phoenix.apache.org
About me

- Architect at Salesforce.com in Big Data group
  - Started Phoenix as internal project ~3 years ago
  - Open-source on Github ~1.5 years ago
  - Apache incubator for ~5 months
  - Graduated as Top Level Project in May 2014

- Engineer at BEA Systems
  - XQuery-based federated query engine
  - SQL-based complex event processing engine
Agenda

- What is Apache Phoenix?
- Why is it so fast?
- How does it help HBase scale?
- Roadmap
- Q&A
What is Apache Phoenix?
What is Apache Phoenix?

1. Turns HBase into a SQL database
   - Query Engine
   - MetaData Repository
   - Embedded JDBC driver
   - Only for HBase data
What is Apache Phoenix?

2. Fastest way to access HBase data
   - HBase-specific push down
   - Compiles queries into native HBase calls (no map-reduce)
   - Executes scans in parallel
```
SELECT * FROM t WHERE k IN (?, ?, ?)
```

<table>
<thead>
<tr>
<th>Phoenix</th>
<th>Stinger (Hive 0.11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.04 sec</td>
<td>280 sec</td>
</tr>
</tbody>
</table>

7,000x faster

* 110M row table
What is Apache Phoenix?

3. Lightweight
   - No additional servers required
   - 100% Java
   - Included in HDP 2.1 distribution
   - Available in Amazon EMR
   - Otherwise copy Phoenix jar into HBase lib directory on each RS
HBase Cluster Architecture

Client

Client finds RegionServer addresses in ZooKeeper

Client reads and writes rows by directly accessing the RegionServers

ZK Quorum
- ZK Peer
- ZK Peer
- ZK Peer

HMaster
- HMaster

RegionServer

RegionServer

RegionServer

HDFS

Master assigns regions and achieves load balancing
HBase Cluster Architecture

- **Phoenix**
  - Client
  - Client finds RegionServer addresses in ZooKeeper
  - Client reads and writes rows by directly accessing the RegionServers

- **ZK Quorum**
  - ZK Peer
  - ZK Peer
  - ZK Peer

- **HMaster**
  - Master assigns regions and achieves load balancing

- **RegionServer**
- **HDFS**
HBase Cluster Architecture

- **Client**
  - Phoenix
  - Reads and writes rows by directly accessing RegionServers
  - Finds RegionServer addresses in ZooKeeper

- **ZK Quorum**
  - ZK Peer

- **HMaster**
  - Master assigns regions and achieves load balancing

- **HDFS**

- **RegionServer**
  - Phoenix
  - Phoenix
What is Apache Phoenix?

4. Integration-friendly
   - Map to existing HBase table
   - Integrate with Apache Pig
   - Integrate with Apache Flume
   - Integrate with Apache Sqoop (wip)
What is Apache Phoenix?

1. Turns HBase into a SQL database
2. Fastest way to access HBase data
3. Lightweight
4. Integration-friendly
Why is Phoenix so fast?
Why is Phoenix so fast?

1. **HBase**
   - Fast, but “dumb” (on purpose)

2. **Data model**
   - Support for composite primary key
   - Binary data sorts naturally

3. **Client-side parallelization**

4. **Push down**
   - Custom filters and coprocessors
Phoenix Data Model

Phoenix maps HBase data model to the relational world
Phoenix Data Model
Phoenix maps HBase data model to the relational world
Phoenix Data Model

Phoenix maps HBase data model to the relational world
Phoenix Data Model

Phoenix maps HBase data model to the relational world

HBase Table

<table>
<thead>
<tr>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifier 1</td>
<td>Qualifier 2</td>
</tr>
<tr>
<td>KeyValue</td>
<td>Qualifier 3</td>
</tr>
</tbody>
</table>

Row Key 1
Phoenix Data Model

Phoenix maps HBase data model to the relational world.
Phoenix Data Model
Phoenix maps HBase data model to the relational world

<table>
<thead>
<tr>
<th>HBase Table</th>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Qualifier 1</td>
<td>Qualifier 2</td>
</tr>
<tr>
<td>Row Key 1</td>
<td>KeyValue</td>
<td></td>
</tr>
<tr>
<td>Row Key 2</td>
<td></td>
<td>Qualifier 3</td>
</tr>
<tr>
<td>Row Key 3</td>
<td></td>
<td>KeyValue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KeyValue</td>
</tr>
</tbody>
</table>
Phoenix Data Model

Phoenix maps HBase data model to the relational world

HBase Table

<table>
<thead>
<tr>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifier 1</td>
<td>Qualifier 2</td>
</tr>
<tr>
<td>Row Key 1</td>
<td>Key Value</td>
</tr>
<tr>
<td>Row Key 2</td>
<td>Key Value</td>
</tr>
<tr>
<td>Row Key 3</td>
<td>Key Value</td>
</tr>
</tbody>
</table>

Phoenix maps HBase data model to the relational world.
Phoenix Data Model
Phoenix maps HBase data model to the relational world

HBase Table
<table>
<thead>
<tr>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifier 1</td>
<td>Qualifier 2</td>
</tr>
<tr>
<td>Row Key 1</td>
<td>Row Key 2</td>
</tr>
<tr>
<td>KeyValue</td>
<td>KeyValue</td>
</tr>
<tr>
<td>Row Key 3</td>
<td></td>
</tr>
<tr>
<td>KeyValue</td>
<td>Key Value</td>
</tr>
</tbody>
</table>
Phoenix Data Model
Phoenix maps HBase data model to the relational world

Multiple Versions
Phoenix Data Model

Phoenix maps HBase data model to the relational world

Phoenix Table

<table>
<thead>
<tr>
<th>Row Key 1</th>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Qualifier 1</td>
<td>Qualifier 2</td>
</tr>
<tr>
<td></td>
<td>Key Value</td>
<td>Key Value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row Key 2</th>
<th></th>
<th>Qualifier 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Key Value</td>
<td>Key Value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row Key 3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Key Value</td>
<td></td>
</tr>
</tbody>
</table>
Phoenix Data Model

Phoenix maps HBase data model to the relational world

Phoenix Table

HBase Table

<table>
<thead>
<tr>
<th>Row Key 1</th>
<th>Column Family A</th>
<th>Column Family B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Qualifier 1</td>
<td>Qualifier 3</td>
</tr>
<tr>
<td></td>
<td>KeyValue</td>
<td>KeyValue</td>
</tr>
<tr>
<td>Row Key 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row Key 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key Value Columns
Phoenix Data Model

Phoenix maps HBase data model to the relational world.

Phoenix Table

HBase Table

Column Family A

Qualifier 1
KeyValue
Qualifier 2
KeyValue
Qualifier 3
KeyValue

Column Family B

Qualifier 3
KeyValue

Row Key 1
Row Key 2
Row Key 3

Primary Key Constraint

Key Value Columns
Example

Over metrics data for servers with a schema like this:

<table>
<thead>
<tr>
<th>SERVER METRICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>RESPONSE_TIME</td>
</tr>
<tr>
<td>GC_TIME</td>
</tr>
<tr>
<td>CPU_TIME</td>
</tr>
<tr>
<td>IO_TIME</td>
</tr>
</tbody>
</table>
Example

Over metrics data for servers with a schema like this:

<table>
<thead>
<tr>
<th>SERVER METRICS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>RESPONSE_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>GC_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CPU_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>IO_TIME</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
Example

DDL command looks like this:

```sql
CREATE TABLE SERVER_METRICS (
    HOST VARCHAR,
    DATE DATE,
    RESPONSE_TIME INTEGER,
    GC_TIME INTEGER,
    CPU_TIME INTEGER,
    IO_TIME INTEGER,
    CONSTRAINT pk PRIMARY KEY (HOST, DATE))
```
Example

With data that looks like this:

<table>
<thead>
<tr>
<th>HOST + DATE</th>
<th>RESPONSE_TIME</th>
<th>GC_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF1 1396743589</td>
<td>1234</td>
<td></td>
</tr>
<tr>
<td>SF1 1396743589</td>
<td>8012</td>
<td></td>
</tr>
<tr>
<td>SF1 1396743589</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF3 1396002345</td>
<td>2345</td>
<td></td>
</tr>
<tr>
<td>SF3 1396002345</td>
<td>2340</td>
<td></td>
</tr>
<tr>
<td>SF7 1396552341</td>
<td>5002</td>
<td>1234</td>
</tr>
<tr>
<td>SF7 1396552341</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Row Key
Example

With data that looks like this:

<table>
<thead>
<tr>
<th>HOST + DATE</th>
<th>RESPONSE_TIME</th>
<th>GC_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF1</td>
<td>1396743589</td>
<td>1234</td>
</tr>
<tr>
<td>SF1</td>
<td>1396743589</td>
<td>8012</td>
</tr>
<tr>
<td>SF3</td>
<td>1396002345</td>
<td>2345</td>
</tr>
<tr>
<td>SF3</td>
<td>1396002345</td>
<td>2340</td>
</tr>
<tr>
<td>SF7</td>
<td>1396552341</td>
<td>5002</td>
</tr>
<tr>
<td>SF7</td>
<td>1396552341</td>
<td>1234</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key Values**
Phoenix Push Down: Example

SELECT host, avg(response_time) 
FROM server_metrics 
WHERE date > CURRENT_DATE() - 7 
AND host LIKE 'SF%' 
GROUP BY host
Phoenix Push Down: Example

```
SELECT host, avg(response_time)
FROM server_metrics
WHERE date > CURRENT_DATE() - 7
AND host LIKE 'SF%'
GROUP BY host
```
Phoenix Push Down: Example

SELECT host, avg(response_time)
FROM server_metrics
WHERE date > CURRENT_DATE() - 7
AND host LIKE 'SF%'
GROUP BY host
Phoenix Push Down: Example

```
SELECT host, avg(response_time)
FROM server_metrics
WHERE date > CURRENT_DATE() – 7
AND host LIKE ‘SF%’
GROUP BY host
```
Phoenix Push Down: Example

SELECT host, avg(response_time) 
FROM server_metrics 
WHERE date > CURRENT_DATE() - 7 
AND host LIKE 'SF%' 
GROUP BY host
Phoenix Push Down

1. Skip scan filter
2. Aggregation
3. TopN
4. Hash Join
SELECT host, avg(response_time)
FROM server_metrics
WHERE date > CURRENT_DATE() – 7
AND host LIKE ‘SF%’
GROUP BY host
Phoenix Push Down: Skip scan

R₁

R₂

R₃

R₄
Phoenix Push Down: Skip scan
Client-side parallel scans

scan_1
scan_2
scan_3

R_1
R_2
R_3
R_4
Phoenix Push Down: Skip scan
Server-side filter
Phoenix Push Down: Skip scan
Server-side filter
Phoenix Push Down: Skip scan
Server-side filter

SKIP
Phoenix Push Down: Skip scan
Server-side filter

INCLUDE
Phoenix Push Down: Skip scan
Server-side filter
Phoenix Push Down: Skip scan
Server-side filter

INCLUDE
Phoenix Push Down: Skip scan
Server-side filter
Phoenix Push Down: Aggregation

SELECT host, avg(response_time) 
FROM server_metrics 
WHERE date > CURRENT_DATE() – 7 
AND host LIKE ‘SF%’
GROUP BY host
### Phoenix Push Down: Aggregation on server-side

Aggregate on server-side

#### SERVER METRICS

<table>
<thead>
<tr>
<th>HOST</th>
<th>DATE</th>
<th>KV1</th>
<th>KV2</th>
<th>KV3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF1</td>
<td>Jun 2 10:10:10.234</td>
<td>239</td>
<td>234</td>
<td>674</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 3 23:05:44.975</td>
<td>23</td>
<td>234</td>
<td>234</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 9 08:10:32.147</td>
<td>256</td>
<td>314</td>
<td>341</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 9 08:10:32.147</td>
<td>235</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 1 11:18:28.456</td>
<td>235</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 3 22:03:22.142</td>
<td>234</td>
<td></td>
<td>314</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 3 22:03:22.142</td>
<td>432</td>
<td>234</td>
<td>256</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 1 10:29:58.950</td>
<td>23</td>
<td>432</td>
<td></td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 2 14:55:34.104</td>
<td>314</td>
<td>876</td>
<td>23</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 3 12:46:19.123</td>
<td>256</td>
<td>234</td>
<td>314</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 3 12:46:19.123</td>
<td></td>
<td>432</td>
<td></td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 8 08:23:23.456</td>
<td>876</td>
<td>876</td>
<td>235</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 1 10:31:10.234</td>
<td>234</td>
<td>234</td>
<td>876</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 1 10:31:10.234</td>
<td>432</td>
<td>432</td>
<td>234</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 3 10:31:10.234</td>
<td></td>
<td></td>
<td>890</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 8 10:31:10.234</td>
<td>314</td>
<td>314</td>
<td>235</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 1 10:31:10.234</td>
<td>256</td>
<td>256</td>
<td>876</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 1 10:31:10.234</td>
<td>235</td>
<td>235</td>
<td>234</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 8 10:31:10.234</td>
<td>876</td>
<td>876</td>
<td>432</td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 9 10:31:10.234</td>
<td>234</td>
<td>234</td>
<td></td>
</tr>
<tr>
<td>SF3</td>
<td>Jun 3 10:31:10.234</td>
<td>432</td>
<td>276</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### SERVER METRICS

<table>
<thead>
<tr>
<th>HOST</th>
<th>AGGREGATE VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF1</td>
<td>3421</td>
</tr>
<tr>
<td>SF2</td>
<td>2145</td>
</tr>
<tr>
<td>SF3</td>
<td>9823</td>
</tr>
</tbody>
</table>
Phoenix Push Down: TopN

```
SELECT host, date, gc_time
FROM server_metrics
WHERE date > CURRENT_DATE() - 7
AND host LIKE 'SF%'
ORDER BY gc_time DESC
LIMIT 5
```
Phoenix Push Down: TopN
Client-side parallel scans

\[ \text{scan}_1 \]
\[ R_1 \]

\[ \text{scan}_2 \]
\[ R_2 \]

\[ \text{scan}_3 \]
\[ R_3 \]

\[ \text{scan}_4 \]
\[ R_4 \]
Phoenix Push Down: TopN
Each region holds N rows
Phoenix Push Down: TopN
Each region holds N rows

\[ \text{scan}_2 \]
Phoenix Push Down: TopN
Each region holds N rows

R_1

R_2

R_3

R_4

\text{scan}_3
Phoenix Push Down: TopN Client-side final merge sort

### SERVER METRICS

<table>
<thead>
<tr>
<th>HOST</th>
<th>DATE</th>
<th>GC_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF3</td>
<td>Jun 2 10:10:10.234</td>
<td>22123</td>
</tr>
<tr>
<td>SF5</td>
<td>Jun 3 23:05:44.975</td>
<td>19876</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 9 08:10:32.147</td>
<td>11345</td>
</tr>
<tr>
<td>SF2</td>
<td>Jun 1 11:18:28.456</td>
<td>10234</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 3 22:03:22.142</td>
<td>10111</td>
</tr>
</tbody>
</table>

- **Scan**
  - Scan$_1$
  - Scan$_2$
  - Scan$_3$
Phoenix Push Down: TopN Secondary Index

CREATE INDEX gc_time_index
ON server_metrics (gc_time DESC, date DESC)
INCLUDE (response_time)

<table>
<thead>
<tr>
<th>GC_TIME_INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC_TIME</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>HOST</td>
</tr>
<tr>
<td>RESPONSE_TIME</td>
</tr>
</tbody>
</table>
Phoenix Push Down: TopN Secondary Index

CREATE INDEX gc_time_index
ON server_metrics (gc_time DESC, date DESC)
INCLUDE (response_time)
Phoenix Push Down: TopN Secondary Index

- Original query doesn’t change
- Phoenix rewrites query to use index table
- All referenced columns must exist in index table for it to be considered
- Stats coming soon!
Phoenix Push Down: Hash Join

SELECT m.*, i.location
FROM server_metrics m
JOIN host_info i ON m.host = i.host
WHERE m.date > CURRENT_DATE() - 7
AND i.location = 'SF'
ORDER BY m.gc_time DESC
LIMIT 5
Phoenix Push Down: Hash Join
Separate LHS and RHS

SELECT m.*, i.location
FROM server_metrics m
JOIN host_info i ON m.host = i.host
WHERE m.date > CURRENT_DATE() - 7
AND i.location = 'SF'
ORDER BY m.gc_time DESC
LIMIT 5
Phoenix Push Down: Hash Join
Separate LHS and RHS

```
SELECT m.*, i.location
FROM server_metrics m
JOIN host_info i ON m.host = i.host
WHERE m.date > CURRENT_DATE() - 7
AND i.location = 'SF'
ORDER BY m.gc_time DESC
LIMIT 5
```
Phoenix Push Down: Hash Join
Separate LHS and RHS

**LHS**
SELECT *
FROM server_metrics
WHERE date > CURRENT_DATE() – 7
ORDER BY gc_time DESC
LIMIT 5

**RHS**
SELECT location
FROM host_info
WHERE location = ‘SF’
Phoenix Push Down: Hash Join
Execute & broadcast RHS to each RS
Phoenix Push Down: TopN Server-side map lookup during scan

LHS
- scan_1
- scan_2
- scan_3
- scan_4

RHS

R_1

R_2

R_3

R_4
How does Phoenix help HBase scale?
How does Phoenix help HBase scale?

1. Phoenix allows multiple tables to share same physical HBase table
   - Updateable VIEW
   - Multi-tenant TABLE + tenant-specific VIEW
   - Support for secondary indexes on VIEWs
How does Phoenix help HBase scale?

2. HBase wants small # of big tables instead of large # of small tables
   - Each region for each column family of each table consumes resources
Phoenix Shared Tables: VIEW

CREATE TABLE event (  
type CHAR(1),  
event_id BIGINT,  
created_date DATE,  
created_by VARCHAR,  
CONSTRAINT pk PRIMARY KEY (type, event_id));

CREATE VIEW web_event (  
referrer VARCHAR) AS  
SELECT * FROM event  
WHERE type='w';  

• Includes columns from TABLE  
• Cannot define PK  
• Updateable if only equality expressions separated by AND
Phoenix Shared Tables: VIEW
Same physical HBase table

<table>
<thead>
<tr>
<th>EVENT</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAT_EVENT</td>
<td>‘c’</td>
</tr>
<tr>
<td>MOBILE_EVENT</td>
<td>‘m’</td>
</tr>
<tr>
<td>PHONE_EVENT</td>
<td>‘p’</td>
</tr>
<tr>
<td>WEB_EVENT</td>
<td>‘w’</td>
</tr>
</tbody>
</table>
Phoenix Shared Table: MULTI_TENANT

CREATE TABLE event (  
tenant_id VARCHAR,  
type CHAR(1),  
event_id BIGINT,  
created_date DATE,  
created_by VARCHAR,  
CONSTRAINT pk PRIMARY KEY (tenant_id, type, event_id))  
MULTI_TENANT=true;

First PK column identifies tenant ID
CREATE VIEW web_event (referrer VARCHAR) AS
SELECT * FROM event
WHERE type='w';

DriverManager.connect("jdbc:phoenix:localhost;tenantId=me");

CREATE VIEW my_web_event AS
SELECT * FROM web_event;

Tenant-specific view

Tenant-specific connection
Phoenix Shared Tables: MULTI_TENANT
Same physical HBase table

<table>
<thead>
<tr>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenant_id = 'aaa'</td>
</tr>
<tr>
<td>tenant_id = 'aab'</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>tenant_id = 'zzz'</td>
</tr>
</tbody>
</table>
Phoenix Shared Tables: MULTI_TENANT
Same physical HBase table

<table>
<thead>
<tr>
<th>EVENT</th>
<th>tenant_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAT_EVENT</td>
<td>type = ‘c’</td>
</tr>
<tr>
<td>MOBILE_EVENT</td>
<td>type = ‘m’</td>
</tr>
<tr>
<td>PHONE_EVENT</td>
<td>type = ‘p’</td>
</tr>
<tr>
<td>WEB_EVENT</td>
<td>type = ‘w’</td>
</tr>
</tbody>
</table>
Phoenix Shared Tables: MULTI_TENANT

- Tenant-specific connection may only see and operate on *their* data
  - MetaData APIs honor this
  - Phoenix automatically manages scan ranges
- Primary key constraint of base table may not be changed
  - Indexes in separate shared table may be added to a VIEW
- DDL operations restricted
  - No ALTER of base table
  - No DROP of columns referenced in WHERE clause
Phoenix Roadmap

- Derived/nested tables (in 3.1/4.1)
- Local Indexes (in 4.1)
- Transactions
- More Join strategies
- Correlated sub-queries
- Cost-based query optimizer
- OLAP extensions
Thank you!
Questions/comments?