Apache Phoenix

We put the SQL back in NoSQL
http://phoenix.incubator.apache.org

James Taylor  @JamesPlusPlus
Maryann Xue  @MaryannXue
Eli Levine  @teleturn
About James

- Engineer at Salesforce.com in BigData group
  - Started Phoenix as internal project ~2.5 years ago
  - Open-source on Github ~1.5 years ago
  - Apache incubator for past 5 months

- 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 0.13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.04 sec</td>
<td>280 sec</td>
</tr>
</tbody>
</table>

* 110M row table

7,000x faster
What is Apache Phoenix?

3. Lightweight
   - No additional servers required
   - Bundled with Hortonworks 2.1
   - 100% Java
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**
  - Master assigns regions and achieves load balancing

- **RegionServer**

- **HDFS**
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

RegionServer

ZK Quorum
  ZK Peer
  ZK Peer
  ZK Peer

HMaster

HMaster

Client finds RegionServer addresses in ZooKeeper

Client reads and writes rows by directly accessing the RegionServers

Master assigns regions and achieves load balancing

HDFS

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 maps HBase data model to the relational world

**Phoenix Data Model**

<table>
<thead>
<tr>
<th>HBase Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Family A</td>
</tr>
<tr>
<td>Qualifier 1</td>
</tr>
<tr>
<td>Row Key 1</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.

<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>Key Value</td>
<td>Key Value</td>
</tr>
<tr>
<td>Row Key 1</td>
<td>Qualifier 3</td>
</tr>
<tr>
<td>Key Value</td>
<td>Key Value</td>
</tr>
<tr>
<td>Row Key 2</td>
<td></td>
</tr>
</tbody>
</table>
**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>Key Value</td>
<td></td>
</tr>
<tr>
<td>Row Key 2</td>
<td></td>
<td>Key Value</td>
</tr>
<tr>
<td>Row Key 3</td>
<td>Key Value</td>
<td>Key Value</td>
</tr>
</tbody>
</table>
Phoenix Data Model

Phoenix maps HBase data model to the relational world
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>KeyValue</td>
</tr>
<tr>
<td>Row Key 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

HBase Table

Column Family A

Row Key 1
Qualifier 1
KeyValue
Row Key 2
Qualifier 2
KeyValue
Row Key 3
Qualifier 3
KeyValue

Column Family B

KeyValue

Multiple Versions
Phoenix Data Model

Phoenix maps HBase data model to the relational world

Phoenix Table

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>KeyValue</td>
</tr>
<tr>
<td>Row Key 1</td>
<td>Qualifier 3</td>
</tr>
<tr>
<td>Row Key 2</td>
<td></td>
</tr>
<tr>
<td>Row Key 3</td>
<td></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

- Column Family A
  - Qualifier 1
    - Key Value
  - Qualifier 2
    - Key Value
  - Qualifier 3
    - Key Value

- Column Family B
  - Qualifier 3
    - Key Value

Phoenix Table

- 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>
<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

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:

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</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>
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>
</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
Phoenix Push Down: Skip scan

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

\[ R_1 \]

\[ R_2 \]

\[ R_3 \]

\[ R_4 \]
Phoenix Push Down: Skip scan
Client-side parallel scans

scan₁

R₁

scan₂

R₂

scan₃

R₃

R₄
Phoenix Push Down: Skip scan
Server-side filter

<table>
<thead>
<tr>
<th>SKIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
Phoenix Push Down: Skip scan
Server-side filter
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: Skip scan
Server-side filter

INCLUDE
Phoenix Push Down: Skip scan
Server-side filter

INCLUDE

INCLUDE

INCLUDE

INCLUDE
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
Aggregate on server-side

<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>674</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>23</td>
</tr>
<tr>
<td>SF1</td>
<td>Jun 3 22:03:22.142</td>
<td>234</td>
<td>314</td>
<td>23</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>432</td>
<td></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></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>
</tbody>
</table>

<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

scan₁

R₁

scan₂

R₂

scan₃

R₃

R₄
Phoenix Push Down: TopN
Each region holds N rows

scan_1
Phoenix Push Down: TopN
Each region holds N rows

R_1
R_2
R_3
R_4

scan_2

[Diagram showing four regions R_1 to R_4 with a scan operation indicated by scan_2]
Phoenix Push Down: TopN
Each region holds N rows

R₁
R₂
R₃
R₄

scan₃
Phoenix Push Down: TopN Client-side final merge sort

Scan_1
Scan_2
Scan_3

<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>
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

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

- 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
- Local Indexing coming soon!
- 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 host, location  
FROM host_info  
WHERE location = 'SF'
Phoenix Push Down: Hash Join
Execute & broadcast RHS to each RS
Phoenix Push Down: Hash Join
Server-side map lookup during scan

LHS

scan_1
R_1

scan_2
R_2

scan_3
R_3

scan_4
R_4

RHS
Phoenix Push Down: Hash Join
Derived Tables and Filter Rewrite

```sql
SELECT i.host, i.location, m.res, m.gc
FROM host_info i
JOIN (SELECT host, avg(response_time) res, avg(gc_time) gc FROM server_metrics GROUP BY host) AS m
ON i.host = m.host
WHERE i.location != 'NJ'
AND (m.res >= 10000 OR m.gc >= 2000)
```
Phoenix Push Down: Hash Join
Filters Pushed Down and Rewritten for RHS

LHS
SELECT host, location
FROM host_info
WHERE location != 'NJ'

RHS
SELECT host, avg(response_time), avg(gc_time)
FROM server_metrics
GROUP BY host
HAVING avg(response_time) >= 10000
OR avg(gc_time) >= 2000

SELECT host, avg(response_time),
FROM server_metrics
GROUP BY host
HAVING avg(response_time) >= 10000
OR avg(gc_time) >= 2000
Phoenix Push Down: Hash Join Multiple Joins and Sub-joins

```
SELECT *
FROM server_metrics m
JOIN (host_info h JOIN location_info l
       ON h.location = l.location)
       ON m.host = h.host
WHERE m.date > CURRENT_DATE() - 7
AND l.user_count >= 200000
```
Phoenix Push Down: Hash Join
Recursively Separate LHS and RHS

Outer LHS
SELECT *
FROM server_metrics
WHERE date > CURRENT_DATE() - 7

Outer RHS
Inner LHS
SELECT *
FROM host_info

Inner RHS
SELECT *
FROM location_info
WHERE user_count >= 200000
Phoenix Push Down: Hash Join
Iterative Execution of Multiple Joins

Iterative execution of multiple joins with broadcast and scan operations.
Phoenix Push Down: Hash Join
Star-join Optimization

SELECT *
FROM server_metrics m
JOIN host_info h
ON m.host = h.host
JOIN location_info l
ON h.location = l.location
WHERE m.date > CURRENT_DATE() - 7
AND l.user_count >= 200000
Phoenix Push Down: Hash Join
Separate LHS and Multiple Parallel RHS

LHS
SELECT *
FROM server_metrics
WHERE date >
CURRENT_DATE() – 7

RHS 1
SELECT *
FROM host_info

RHS 2
SELECT *
FROM location_info
WHERE
user_count >= 200000
Phoenix Push Down: Hash Join
Star-join Execution of Multiple Joins

Scan RHS 1
RS₁

Scan RHS 2
RS₁
RS₂

Broadcast

Scan LHS
RS₁
RS₂
RS₃
RS₄
New in Phoenix 3: Shared Tables

• HBase wants small # of big tables instead of large # of small tables

• Two types of shared tables in Phoenix 3:
  • Views
  • Tenant-specific Tables
Views

- Multiple Phoenix tables share same physical HBase table
- Inherit parent table’s PK, KV columns
- Updateable Views
- Secondary Indexes on Views
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
Views

<table>
<thead>
<tr>
<th>EVENT</th>
<th>type = 'c'</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAT_EVENT</td>
<td></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>
Tenant-Specific Tables

- Tenant data isolation and co-location
- Built using Views
- Uses tenant-specific Connections
Tenant-Specific Tables

Step 1: Create multi-tenant base table

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
Tenant-Specific Tables

Step 2: Create tenant-specific tables

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 Tables

<table>
<thead>
<tr>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- tenant_id = ‘me’
- tenant_id = ‘you’
- ...
- tenant_id = ‘them’
## Tenant-Specific Tables

<table>
<thead>
<tr>
<th>EVENT</th>
<th>PER tenant_id</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAT_EVENT</td>
<td></td>
<td>‘c’</td>
</tr>
<tr>
<td>MOBILE_EVENT</td>
<td></td>
<td>‘m’</td>
</tr>
<tr>
<td>PHONE_EVENT</td>
<td></td>
<td>‘p’</td>
</tr>
<tr>
<td>WEB_EVENT</td>
<td></td>
<td>‘w’</td>
</tr>
</tbody>
</table>
Tenant-Specific Tables

- Tenant-specific connection may only see and operate on *their* data
- Inherit parent table’s PK, KV columns
- Tenant-specific secondary indexes
- Restrictions:
  - No ALTER base table
  - No DROP columns used in PK and where clause
  - PK columns same as parent
Shared Tables Future Work

- Allow shared tables to extend parent’s PK
- Support more complex WHERE clauses for updatable views
- Support projecting subset of columns to View
Phoenix Roadmap

- Local Indexes
- Transactions
- More Join strategies
- Cost-based query optimizer
- OLAP extensions
Thank you!
Questions/comments?