

# Apache Phoenix



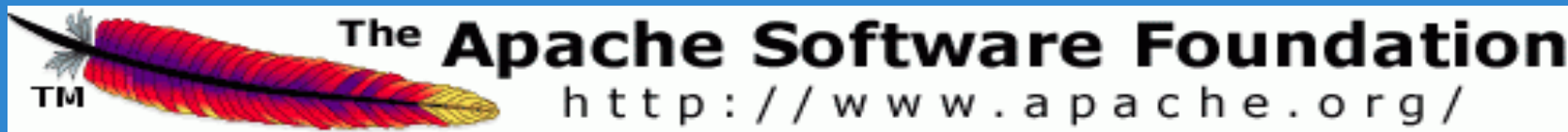
*We put the SQL back in NoSQL*

<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 (?, ?, ?)

| Phoenix  | Stinger (Hive 0.13) |
|----------|---------------------|
| 0.04 sec | 280 sec             |

7,000x faster

\* 110M row table



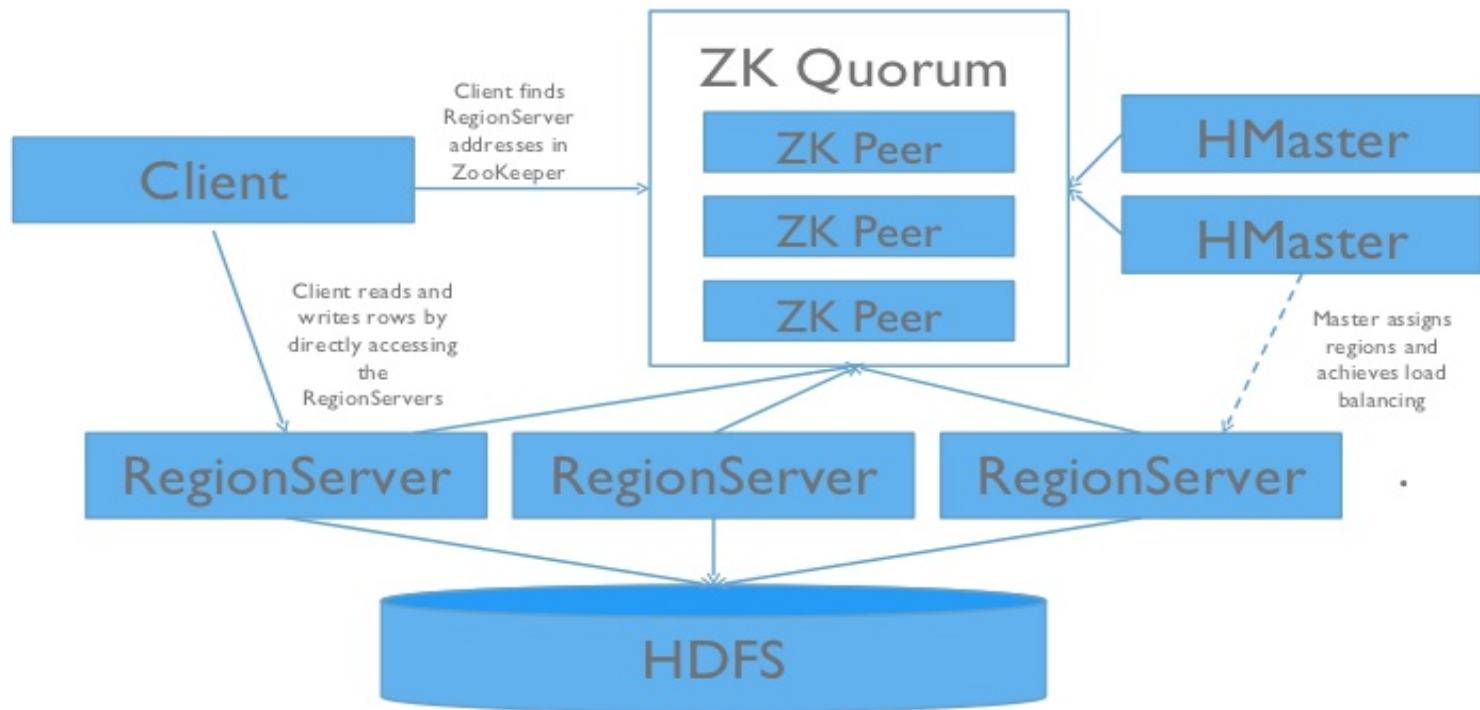
# What is Apache Phoenix?

## 3. Lightweight

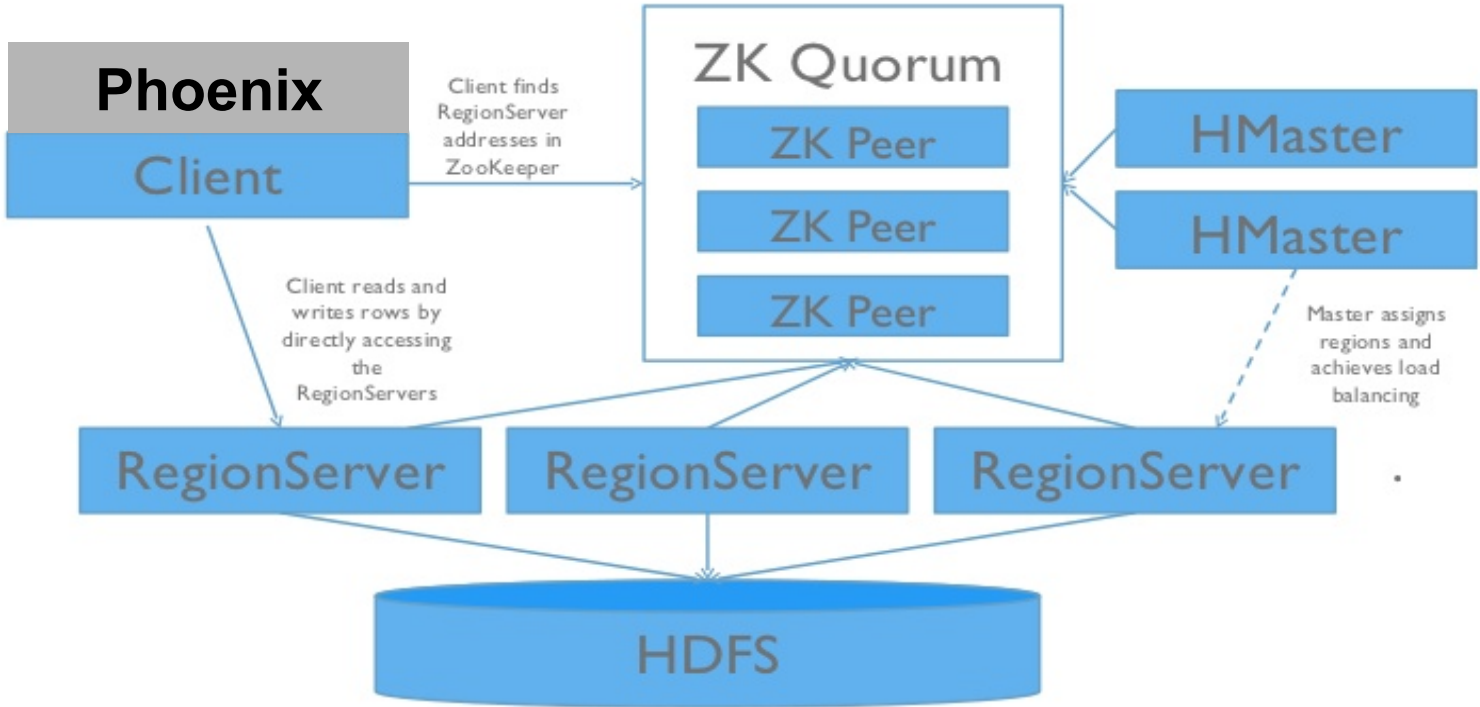
- No additional servers required
- Bundled with Hortonworks 2.1
- 100% Java



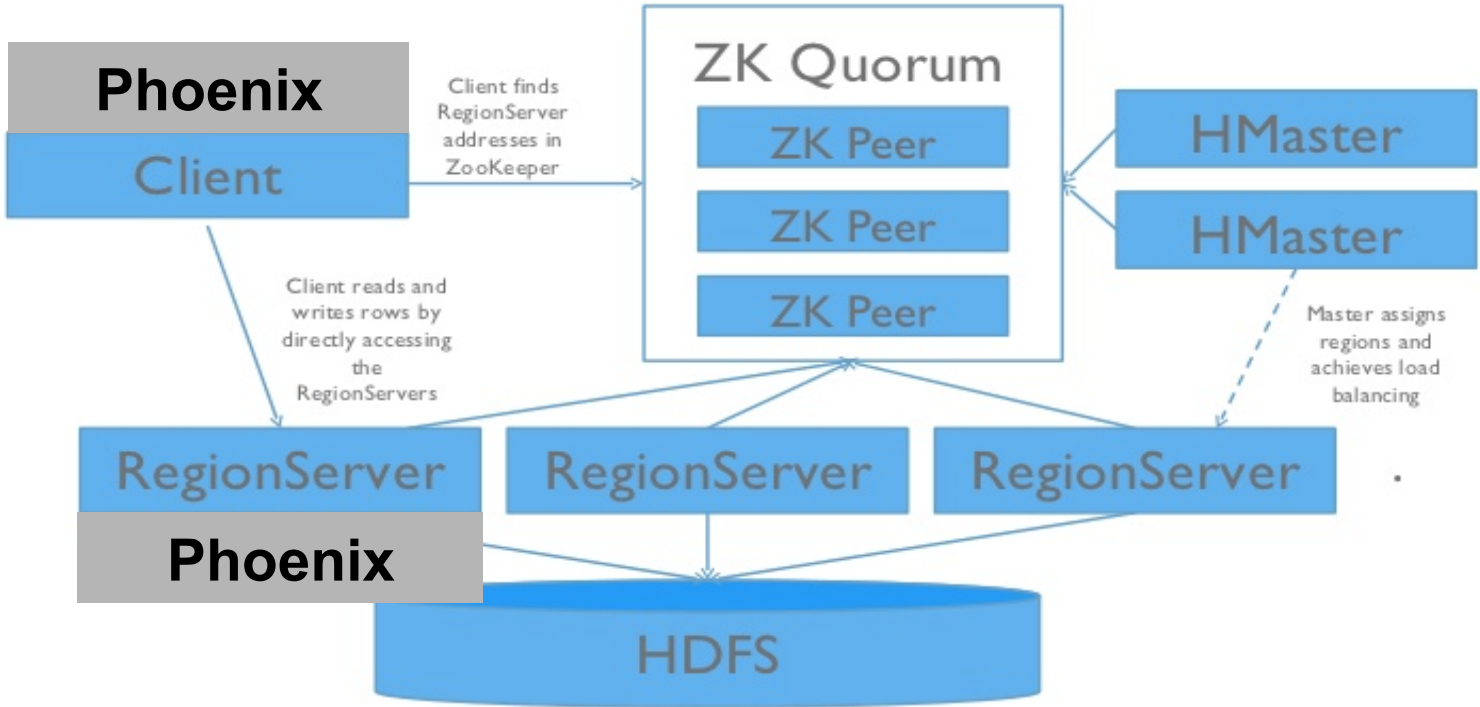
# HBase Cluster Architecture



# HBase Cluster Architecture



# HBase Cluster Architecture





# 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

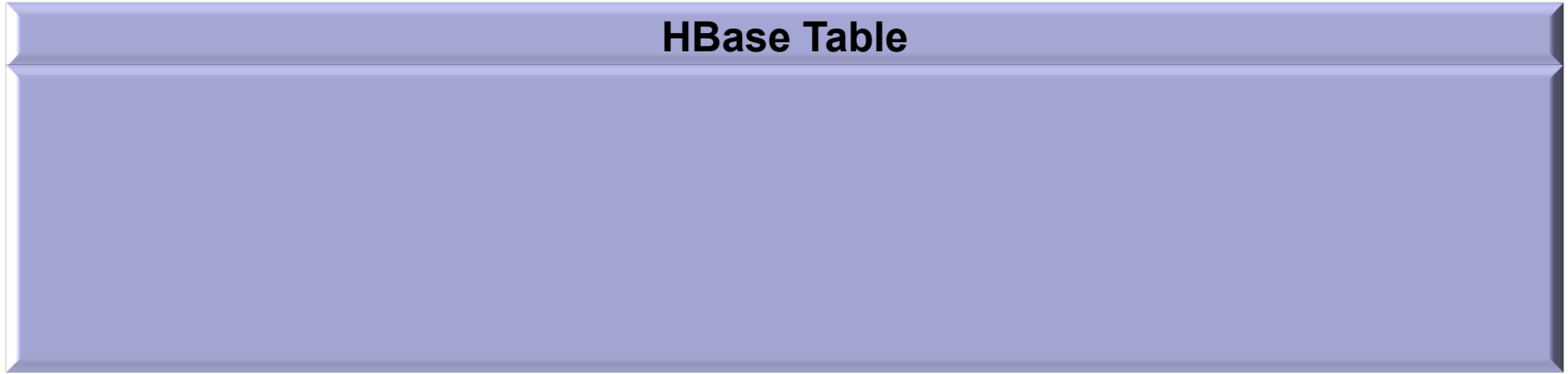
## 3. Client-side parallelization

## 4. Push down

- Custom filters and coprocessors

# Phoenix Data Model

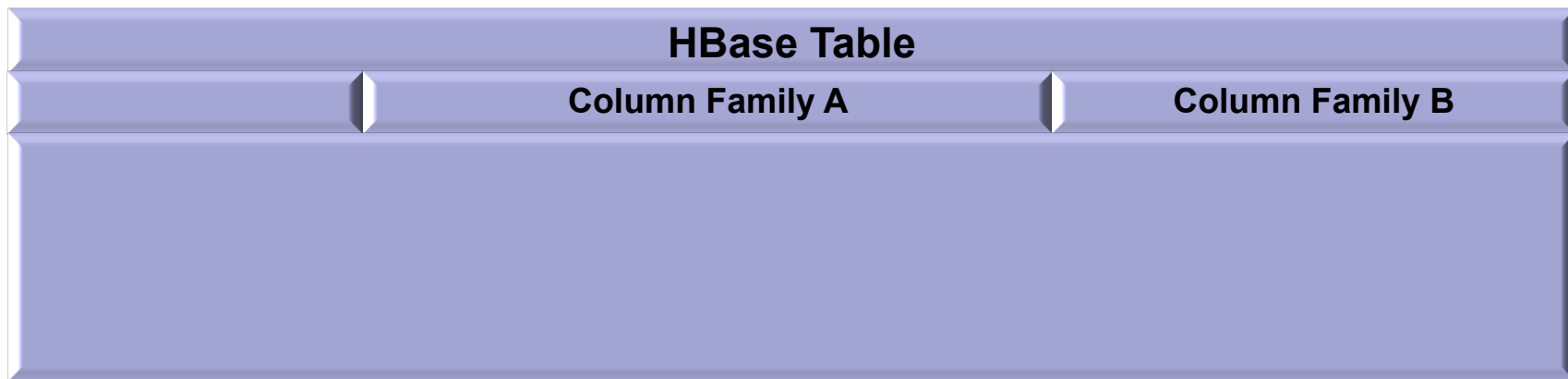
Phoenix maps HBase data model to the relational world





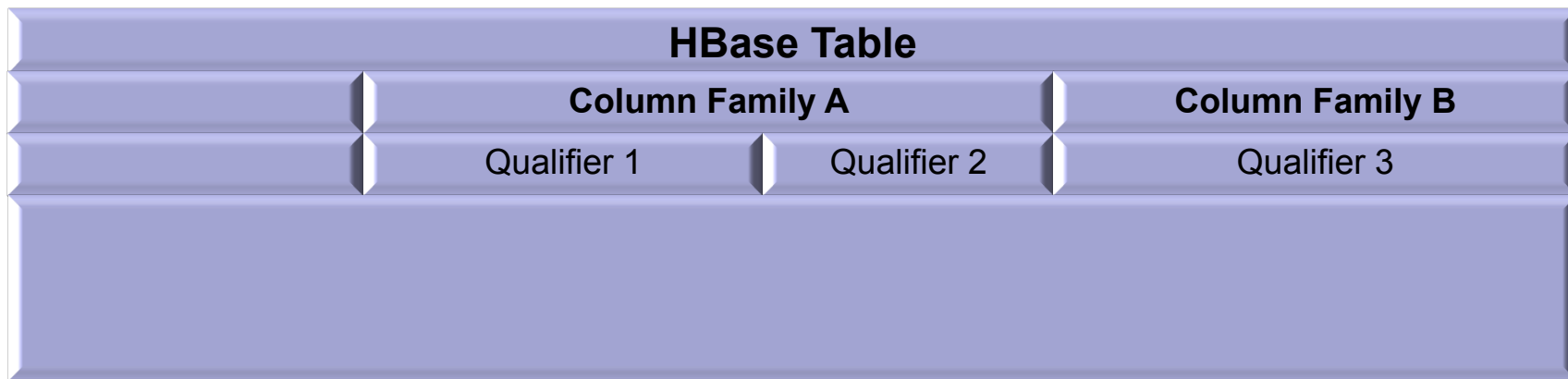
# Phoenix Data Model

Phoenix maps HBase data model to the relational world



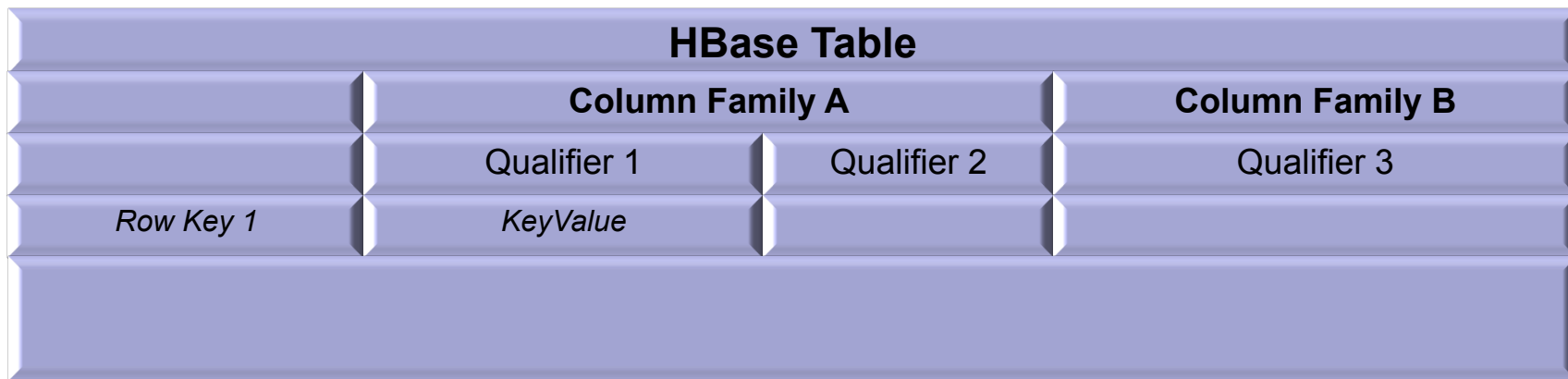
# Phoenix Data Model

Phoenix maps HBase data model to the relational world



# Phoenix Data Model

Phoenix maps HBase data model to the relational world



# Phoenix Data Model

Phoenix maps HBase data model to the relational world

| HBase Table      |                 |                 |                 |
|------------------|-----------------|-----------------|-----------------|
|                  | Column Family A |                 | Column Family B |
|                  | Qualifier 1     | Qualifier 2     | Qualifier 3     |
| <i>Row Key 1</i> | <i>KeyValue</i> |                 |                 |
| <i>Row Key 2</i> |                 | <i>KeyValue</i> | <i>KeyValue</i> |

# Phoenix Data Model

Phoenix maps HBase data model to the relational world

| HBase Table      |                 |                 |                 |
|------------------|-----------------|-----------------|-----------------|
|                  | Column Family A |                 | Column Family B |
|                  | Qualifier 1     | Qualifier 2     | Qualifier 3     |
| <i>Row Key 1</i> | <i>KeyValue</i> |                 |                 |
| <i>Row Key 2</i> |                 | <i>KeyValue</i> | <i>KeyValue</i> |
| <i>Row Key 3</i> | <i>KeyValue</i> |                 |                 |

# Phoenix Data Model

Phoenix maps HBase data model to the relational world

| HBase Table      |                 |                 |                 |
|------------------|-----------------|-----------------|-----------------|
|                  | Column Family A |                 | Column Family B |
|                  | Qualifier 1     | Qualifier 2     | Qualifier 3     |
| <i>Row Key 1</i> | <i>KeyValue</i> |                 |                 |
| <i>Row Key 2</i> |                 | <i>KeyValue</i> | <i>KeyValue</i> |
| <i>Row Key 3</i> | <i>KeyValue</i> |                 |                 |

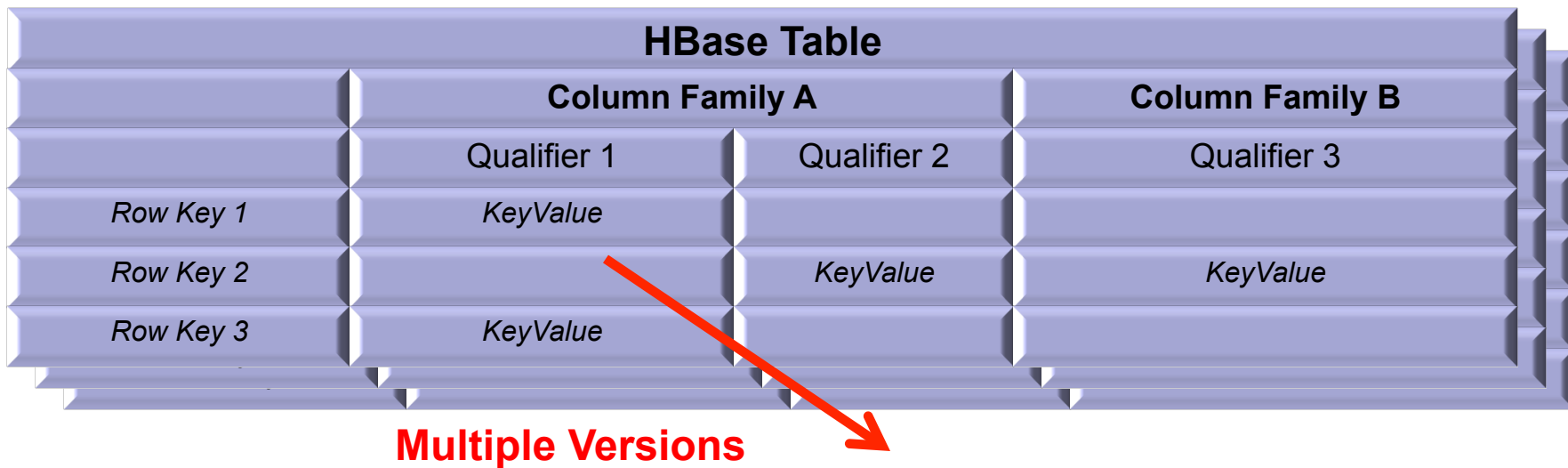
# Phoenix Data Model

Phoenix maps HBase data model to the relational world

| HBase Table      |                 |                 |                 |
|------------------|-----------------|-----------------|-----------------|
|                  | Column Family A |                 | Column Family B |
|                  | Qualifier 1     | Qualifier 2     | Qualifier 3     |
| <i>Row Key 1</i> | <i>KeyValue</i> |                 |                 |
| <i>Row Key 2</i> |                 | <i>KeyValue</i> | <i>KeyValue</i> |
| <i>Row Key 3</i> | <i>KeyValue</i> |                 |                 |

# Phoenix Data Model

Phoenix maps HBase data model to the relational world

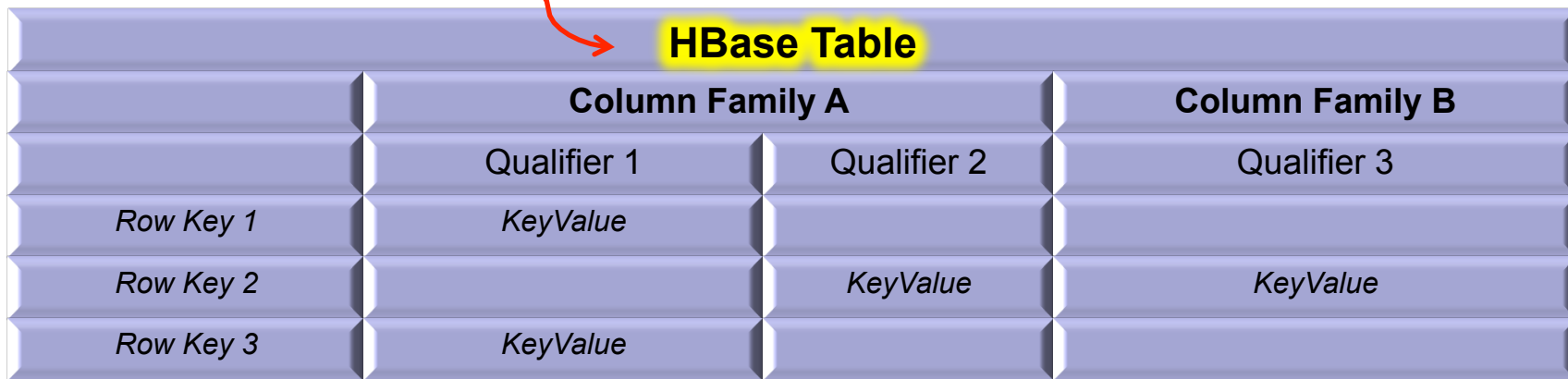




# Phoenix Data Model

Phoenix maps HBase data model to the relational world

**Phoenix Table**

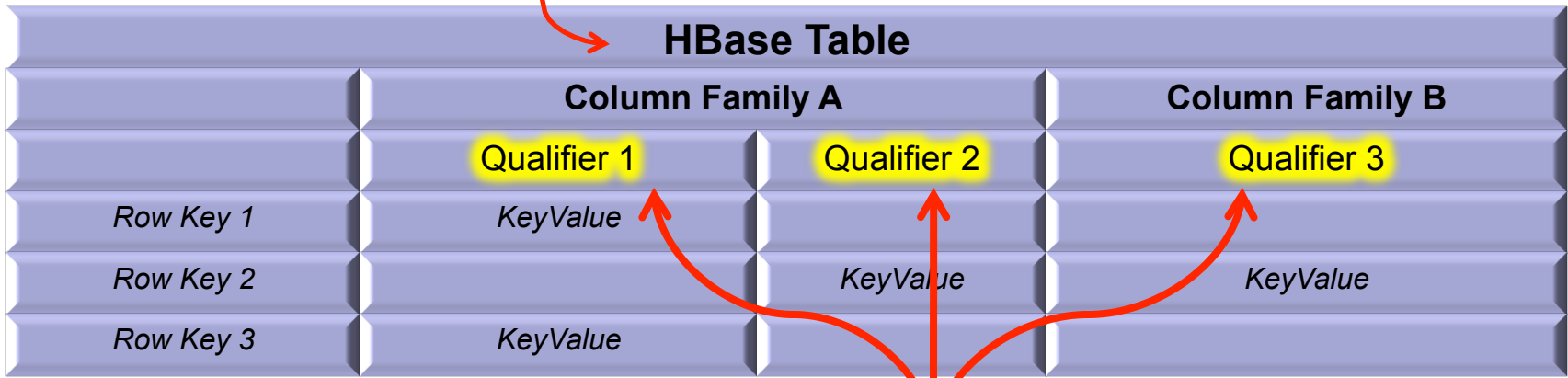


| HBase Table |                 |             |                 |
|-------------|-----------------|-------------|-----------------|
|             | Column Family A |             | Column Family B |
|             | Qualifier 1     | Qualifier 2 | Qualifier 3     |
| Row Key 1   | KeyValue        |             |                 |
| Row Key 2   |                 | KeyValue    | KeyValue        |
| Row Key 3   | KeyValue        |             |                 |

# Phoenix Data Model

Phoenix maps HBase data model to the relational world

**Phoenix Table**

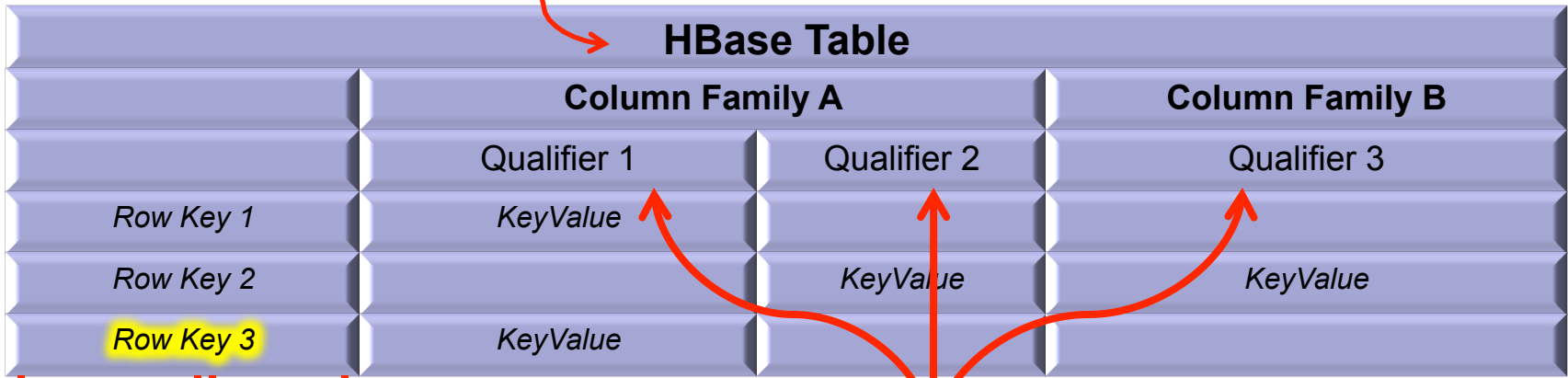


**Key Value Columns**

# Phoenix Data Model

Phoenix maps HBase data model to the relational world

## Phoenix Table




**Primary Key Constraint**

**Key Value Columns**

# Example

Over metrics data for servers with a schema like this:

| SERVER METRICS |         |
|----------------|---------|
| HOST           | VARCHAR |
| DATE           | DATE    |
| RESPONSE_TIME  | INTEGER |
| GC_TIME        | INTEGER |
| CPU_TIME       | INTEGER |
| IO_TIME        | INTEGER |

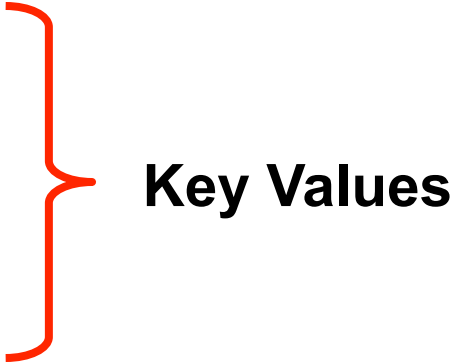


Row Key

# Example

Over metrics data for servers with a schema like this:

| SERVER METRICS |         |
|----------------|---------|
| HOST           | VARCHAR |
| DATE           | DATE    |
| RESPONSE_TIME  | INTEGER |
| GC_TIME        | INTEGER |
| CPU_TIME       | INTEGER |
| IO_TIME        | INTEGER |



**Key Values**

# 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:

| SERVER METRICS |            |               |         |
|----------------|------------|---------------|---------|
| HOST + DATE    |            | RESPONSE_TIME | GC_TIME |
| SF1            | 1396743589 | 1234          |         |
| SF1            | 1396743589 |               | 8012    |
| ...            |            |               |         |
| SF3            | 1396002345 | 2345          |         |
| SF3            | 1396002345 |               | 2340    |
| SF7            | 1396552341 | 5002          | 1234    |
| ...            |            |               |         |



**Row Key**

# Example

With data that looks like this:

| SERVER METRICS |            |               |         |
|----------------|------------|---------------|---------|
| HOST + DATE    |            | RESPONSE_TIME | GC_TIME |
| SF1            | 1396743589 | 1234          |         |
| SF1            | 1396743589 |               | 8012    |
| ...            |            |               |         |
| SF3            | 1396002345 | 2345          |         |
| SF3            | 1396002345 |               | 2340    |
| SF7            | 1396552341 | 5002          | 1234    |
| ...            |            |               |         |



**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
```

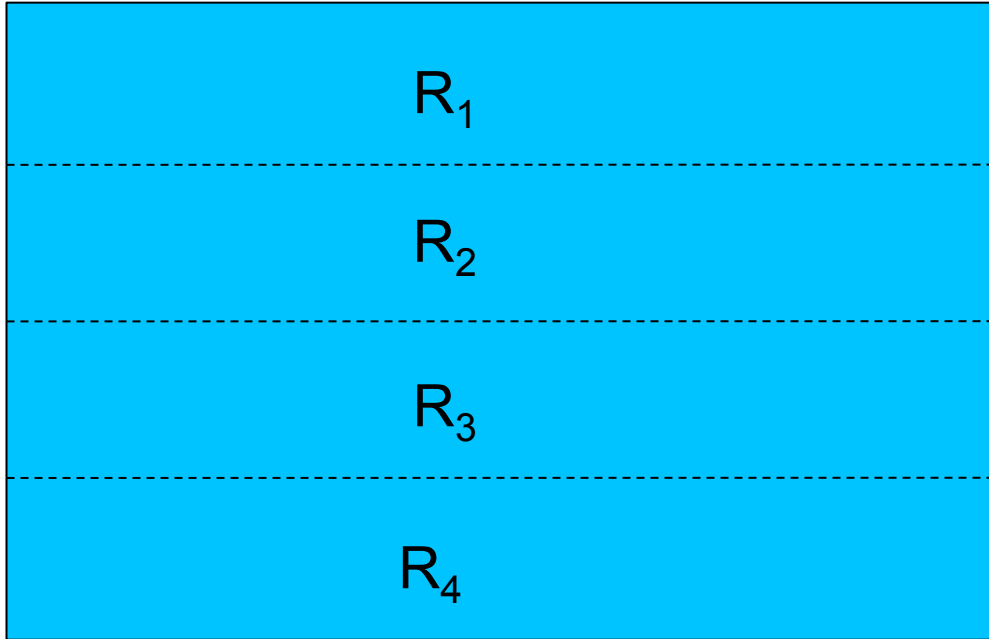
# Phoenix Push Down

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
```

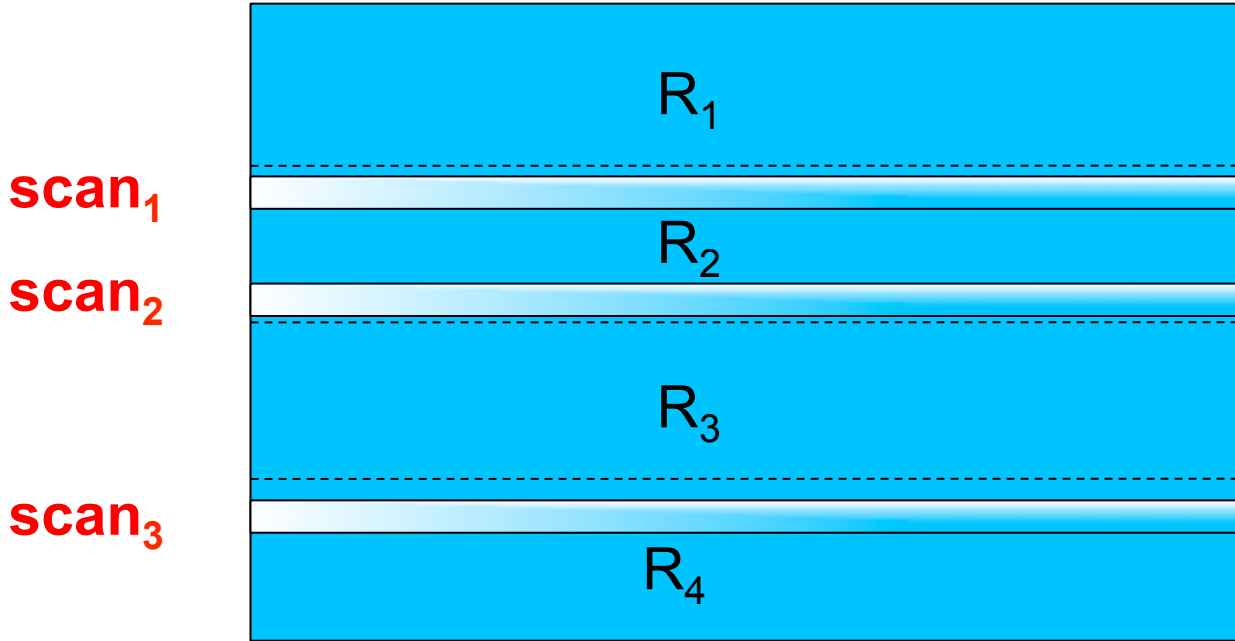
# Phoenix Push Down: Skip scan





# Phoenix Push Down: Skip scan

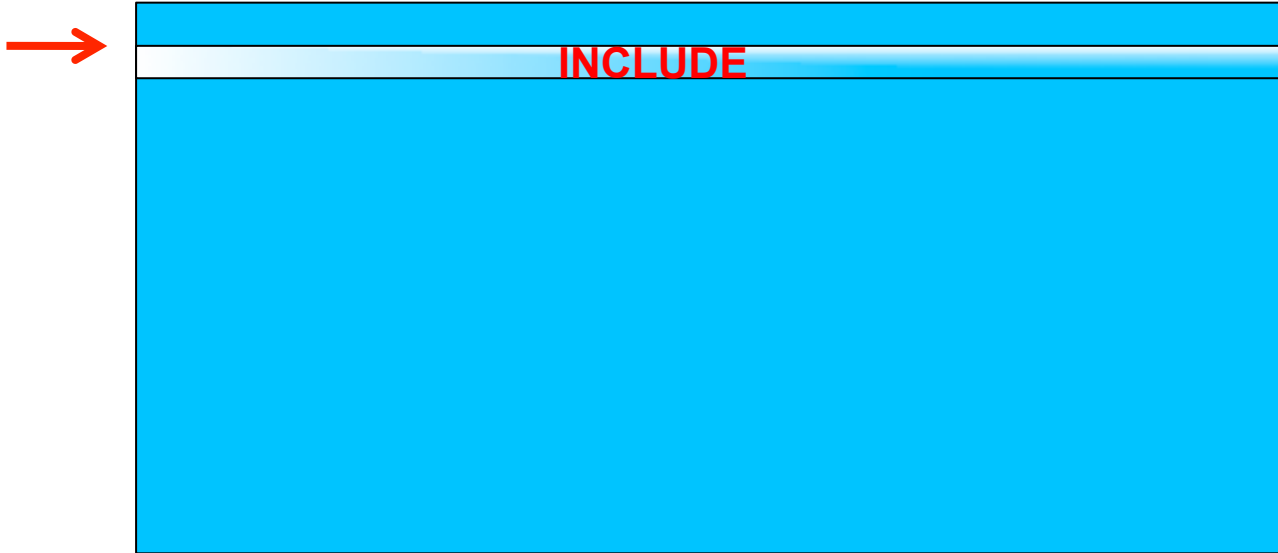
## Client-side parallel scans



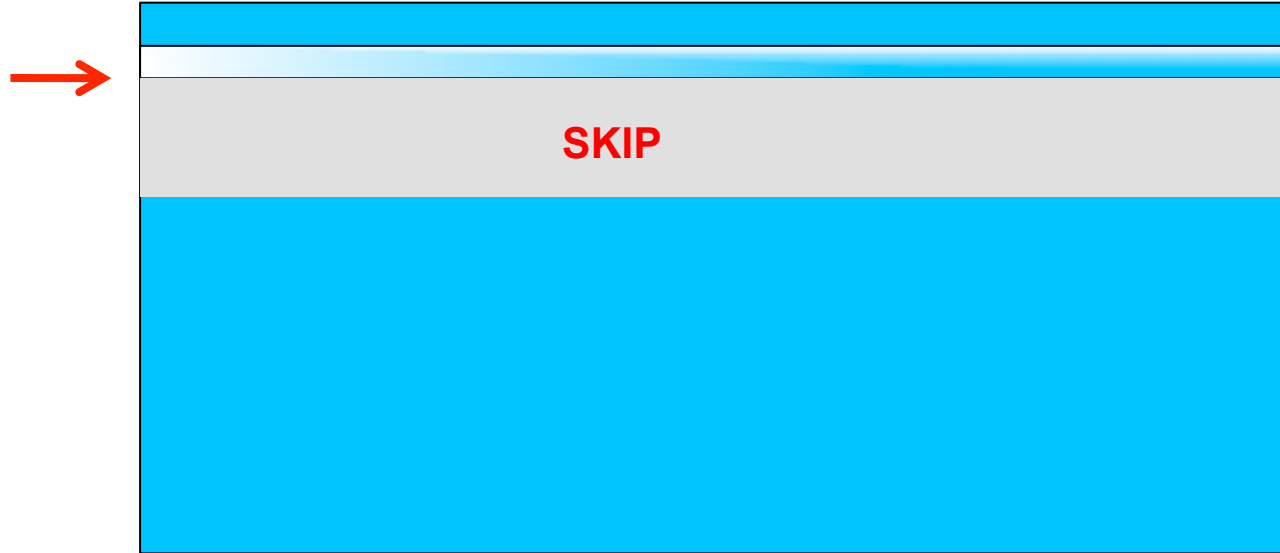
# Phoenix Push Down: Skip scan Server-side filter



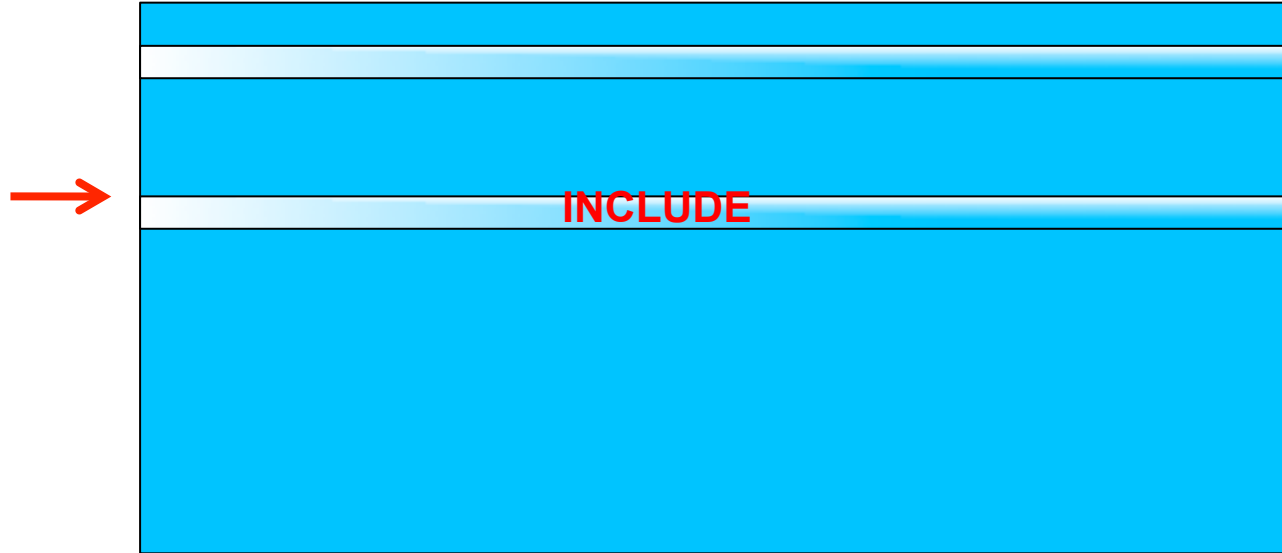
# Phoenix Push Down: Skip scan Server-side filter



# Phoenix Push Down: Skip scan Server-side filter



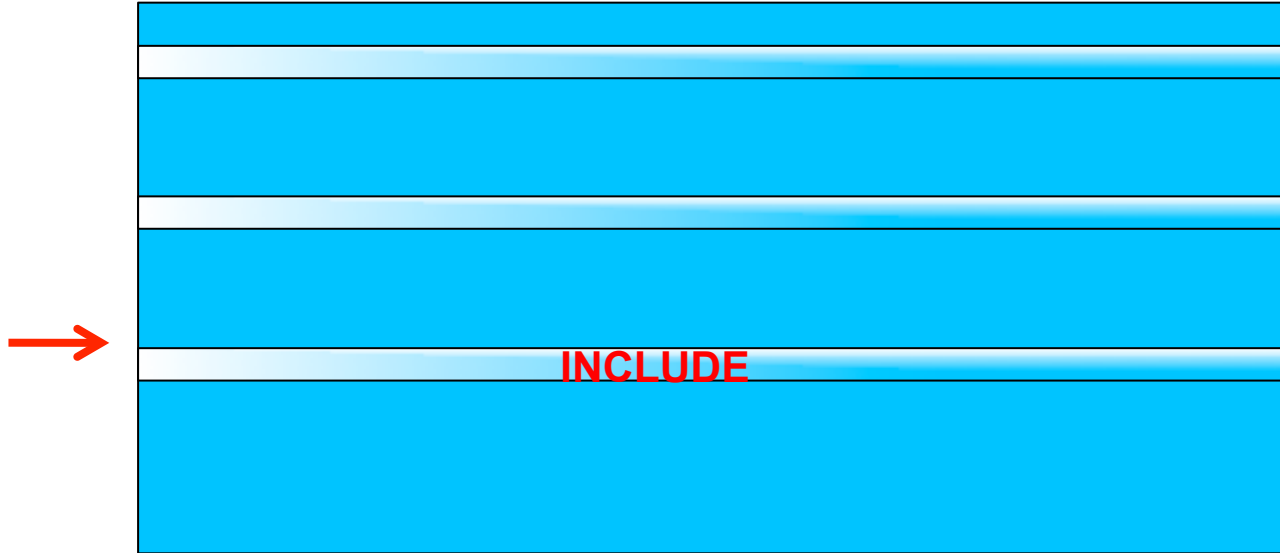
# Phoenix Push Down: Skip scan Server-side filter



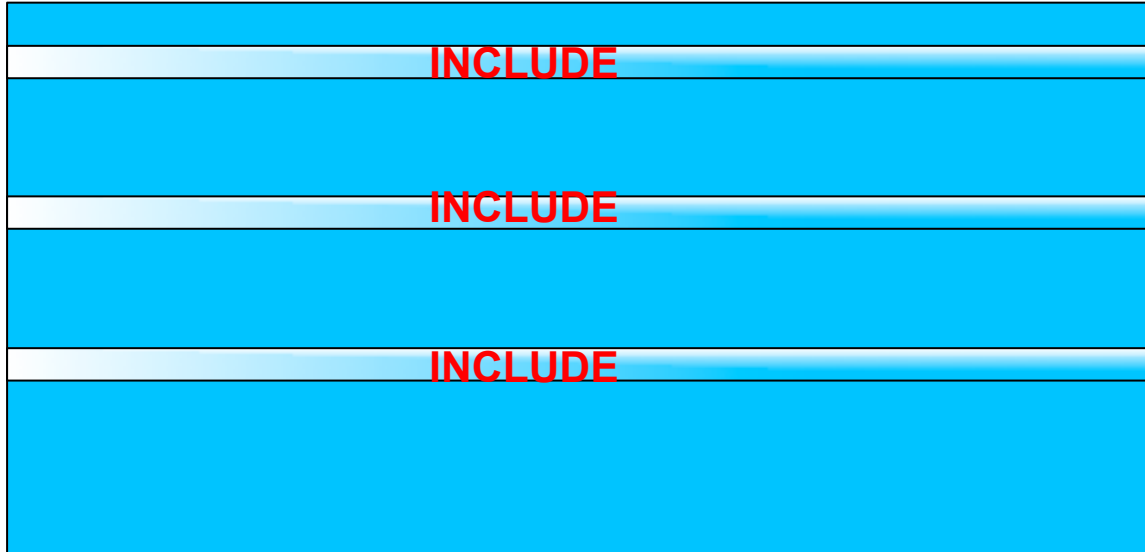
# Phoenix Push Down: Skip scan Server-side filter



# Phoenix Push Down: Skip scan Server-side filter



# 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 Aggregate on server-side

| SERVER METRICS |                    |                 |                 |                 |
|----------------|--------------------|-----------------|-----------------|-----------------|
| HOST           | DATE               | KV <sub>1</sub> | KV <sub>2</sub> | KV <sub>3</sub> |
| SF1            | Jun 2 10:10:10.234 | 239             | 234             | 674             |
| SF1            | Jun 3 23:05:44.975 |                 | 23              | 234             |
| SF1            | Jun 9 08:10:32.147 | 256             | 314             | 341             |
| SF1            | Jun 9 08:10:32.147 | 235             | 256             |                 |
| SF1            | Jun 1 11:18:28.456 |                 | 235             | 23              |
| SF1            | Jun 3 22:03:22.142 | 234             |                 | 314             |
| SF1            | Jun 3 22:03:22.142 | 432             | 234             | 256             |
| SF2            | Jun 1 10:29:58.950 | 23              | 432             |                 |
| SF2            | Jun 2 14:55:34.104 | 314             | 876             | 23              |
| SF2            | Jun 3 12:46:19.123 | 256             | 234             | 314             |
| SF2            | Jun 3 12:46:19.123 |                 | 432             |                 |
| SF2            | Jun 8 08:23:23.456 | 876             | 876             | 235             |
| SF2            | Jun 1 10:31:10.234 | 234             | 234             | 876             |
| SF3            | Jun 1 10:31:10.234 | 432             | 432             | 234             |
| SF3            | Jun 3 10:31:10.234 |                 | 890             |                 |
| SF3            | Jun 8 10:31:10.234 | 314             | 314             | 235             |
| SF3            | Jun 1 10:31:10.234 | 256             | 256             | 876             |
| SF3            | Jun 1 10:31:10.234 | 235             |                 | 234             |
| SF3            | Jun 8 10:31:10.234 | 876             | 876             | 432             |
| SF3            | Jun 9 10:31:10.234 | 234             | 234             |                 |
| SF3            | Jun 3 10:31:10.234 |                 | 432             | 276             |
| ...            | ...                | ...             | ...             | ...             |



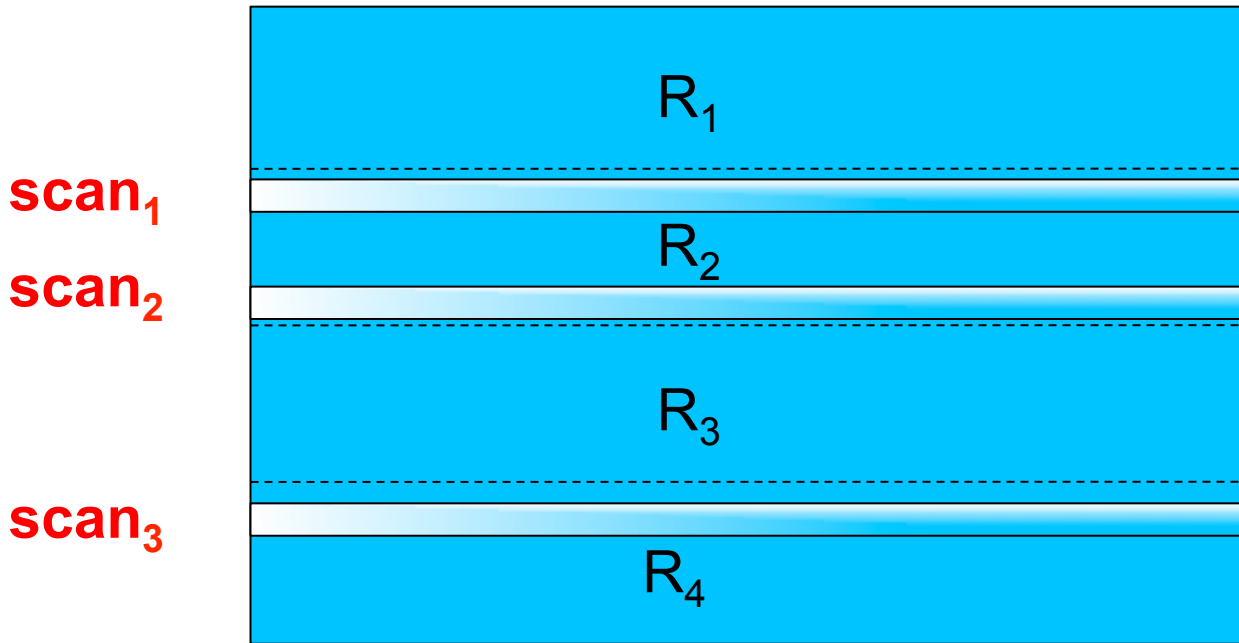
| SERVER METRICS |                  |
|----------------|------------------|
| HOST           | AGGREGATE VALUES |
| SF1            | 3421             |
| SF2            | 2145             |
| SF3            | 9823             |

# 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
```

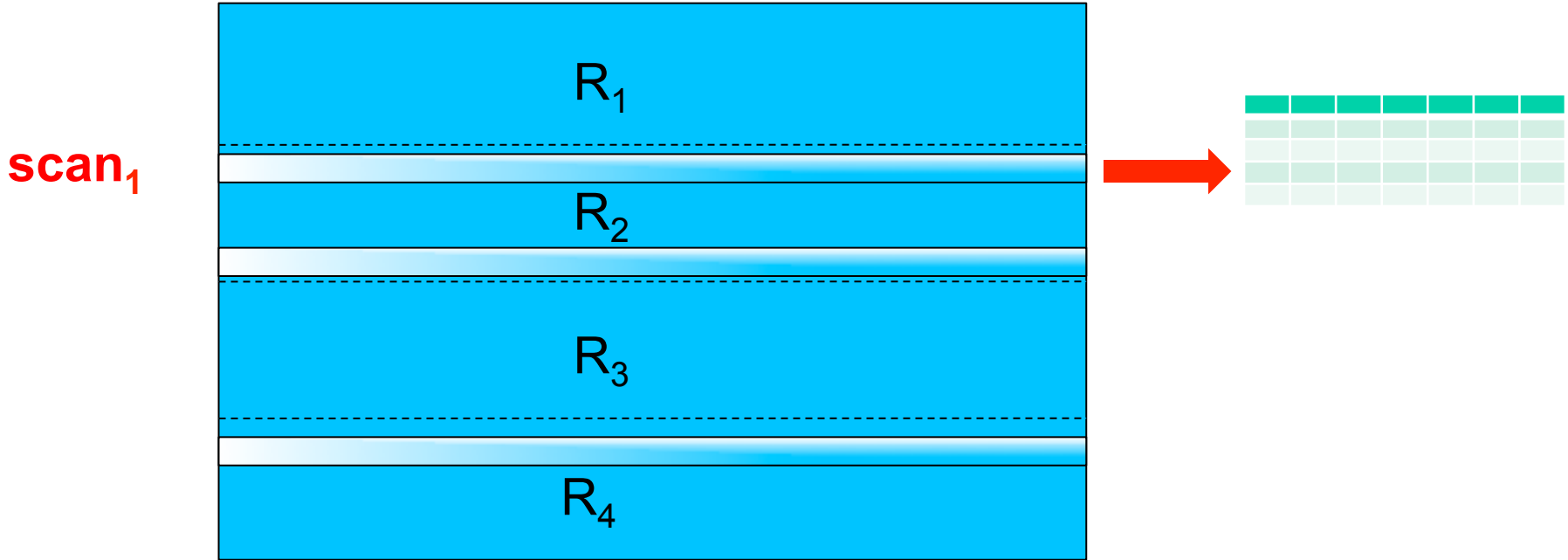
# Phoenix Push Down: TopN

## Client-side parallel scans



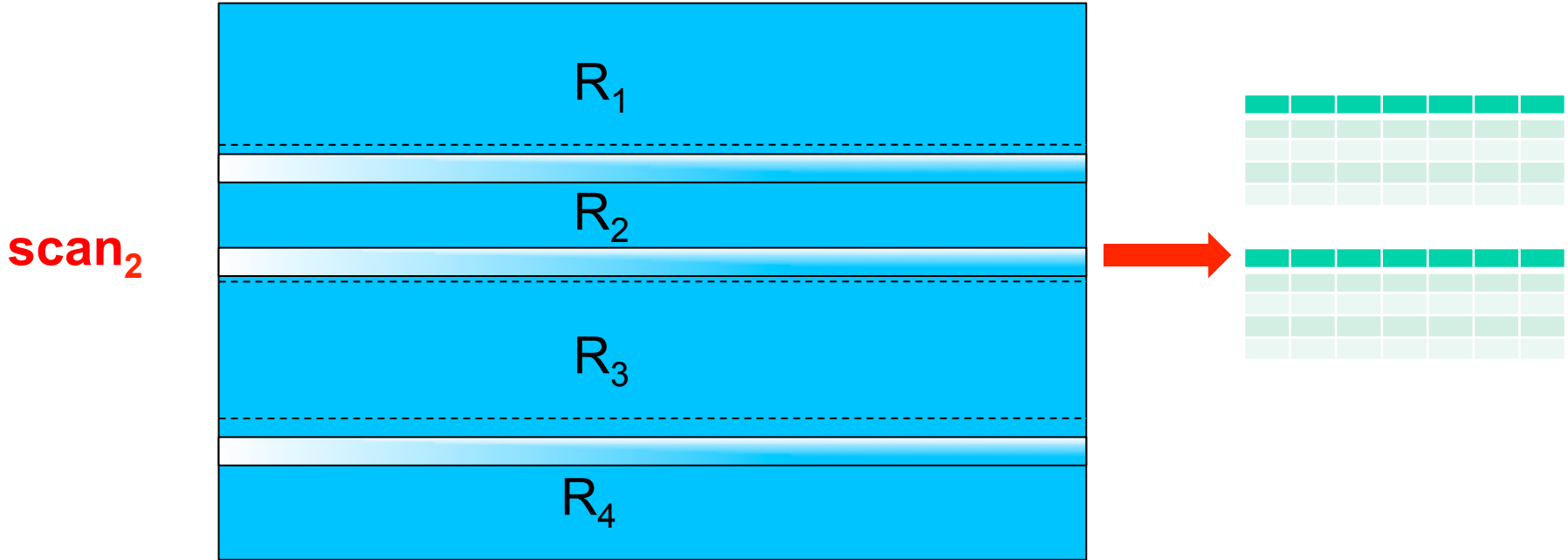
# Phoenix Push Down: TopN

Each region holds N rows



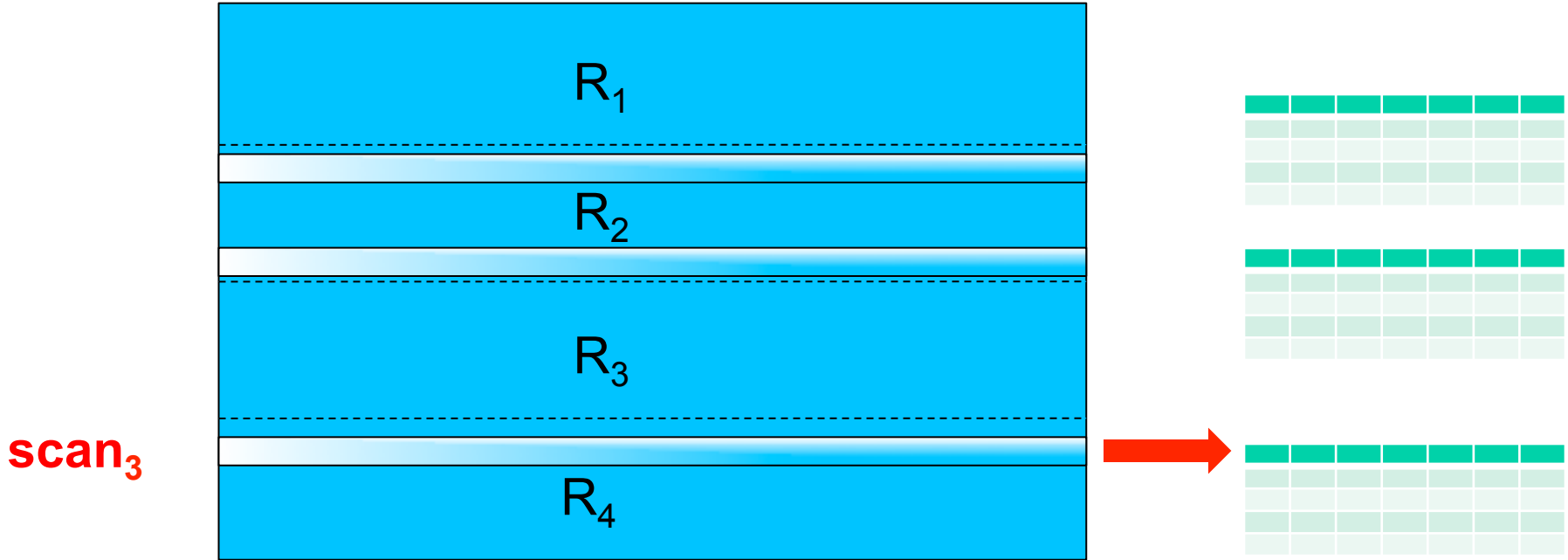
# Phoenix Push Down: TopN

## Each region holds N rows



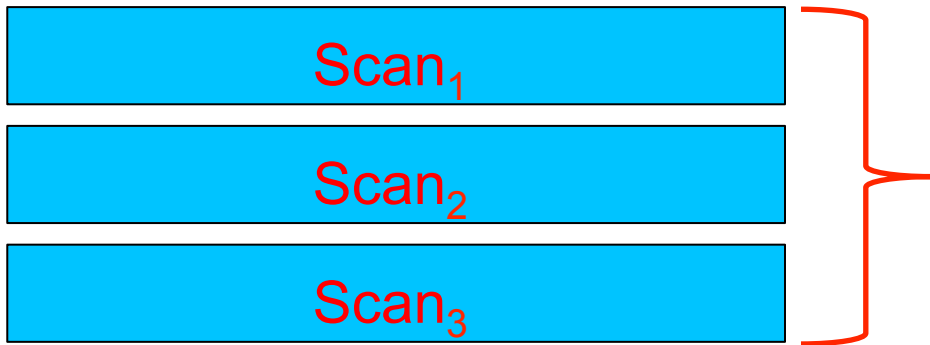
# Phoenix Push Down: TopN

Each region holds N rows



# Phoenix Push Down: TopN

## Client-side final merge sort



| SERVER METRICS |                    |         |
|----------------|--------------------|---------|
| HOST           | DATE               | GC_TIME |
| SF3            | Jun 2 10:10:10.234 | 22123   |
| SF5            | Jun 3 23:05:44.975 | 19876   |
| SF2            | Jun 9 08:10:32.147 | 11345   |
| SF2            | Jun 1 11:18:28.456 | 10234   |
| SF1            | Jun 3 22:03:22.142 | 10111   |



# Phoenix Push Down: TopN Secondary Index

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

| GC_TIME_INDEX |         |
|---------------|---------|
| GC_TIME       | INTEGER |
| DATE          | DATE    |
| HOST          | VARCHAR |
| RESPONSE_TIME | INTEGER |



Row Key

# Phoenix Push Down: TopN Secondary Index

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

| GC_TIME_INDEX |         |
|---------------|---------|
| GC_TIME       | INTEGER |
| DATE          | DATE    |
| HOST          | VARCHAR |
| RESPONSE_TIME | INTEGER |

 Key Value

# 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

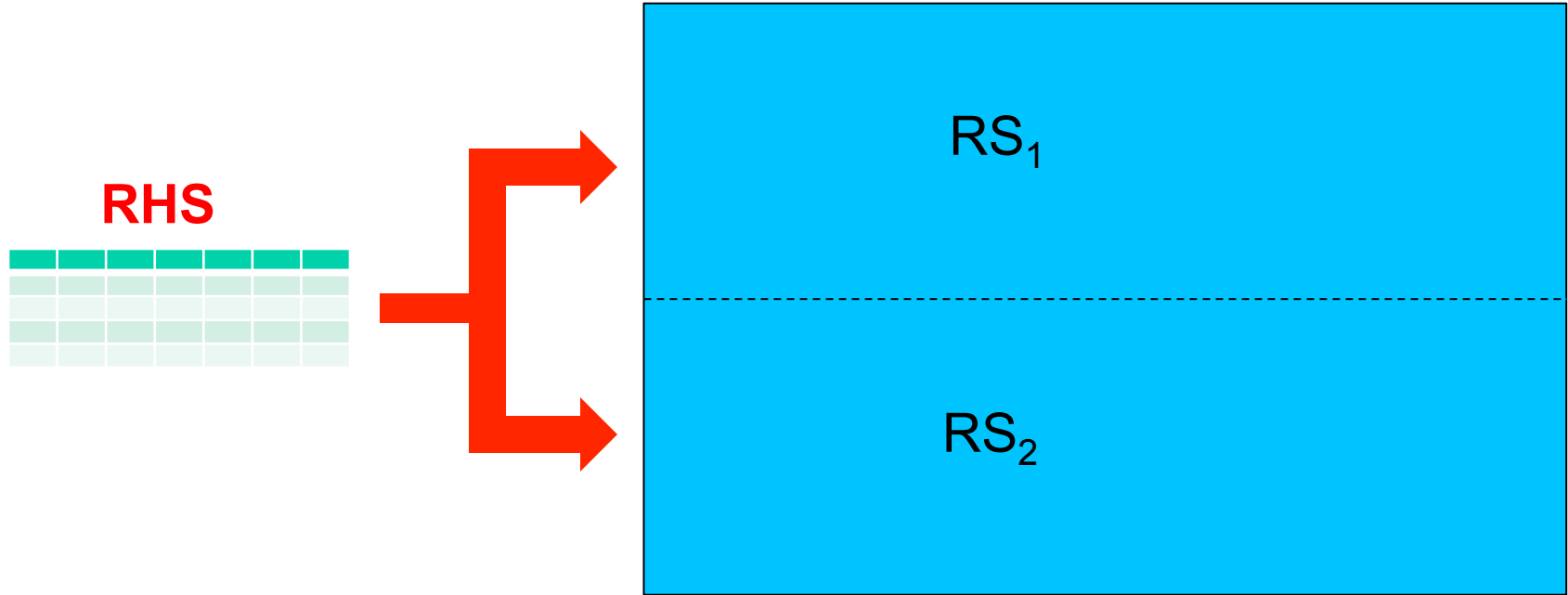
### LHS

```
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'
```

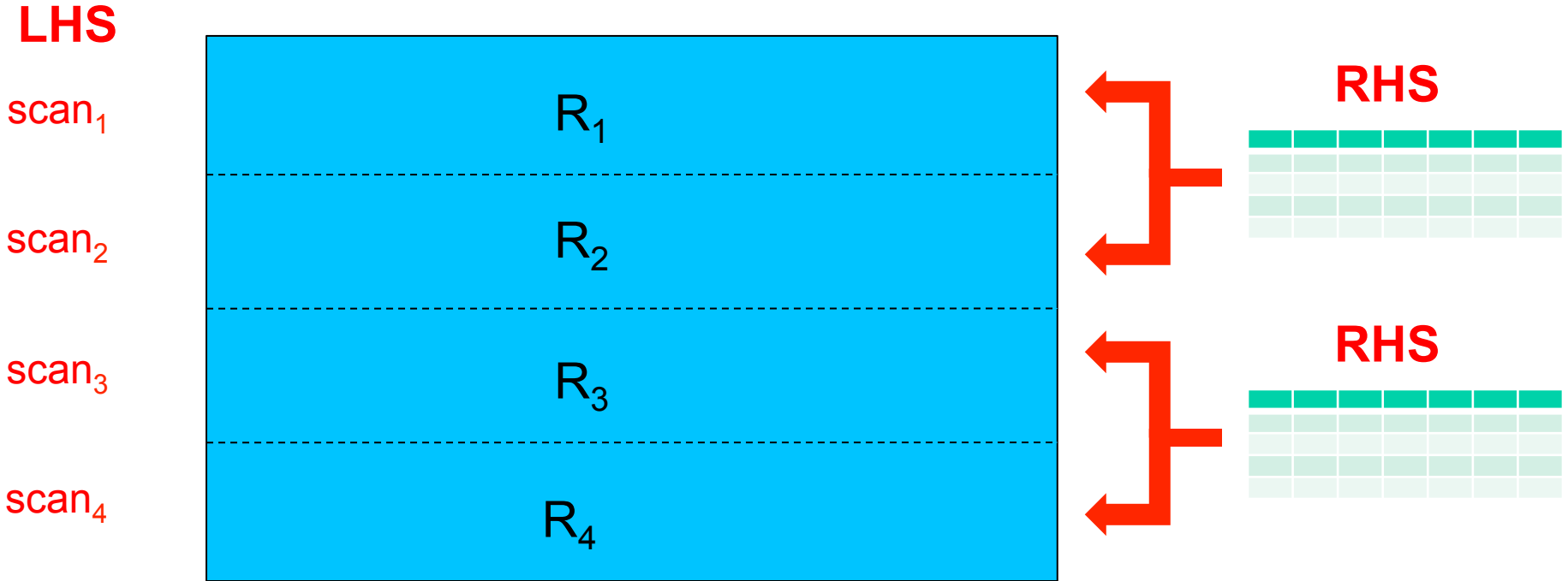
# Phoenix Push Down: Hash Join Execute & broadcast RHS to each RS





# Phoenix Push Down: Hash Join

## Server-side map lookup during scan



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

```
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
```

# 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

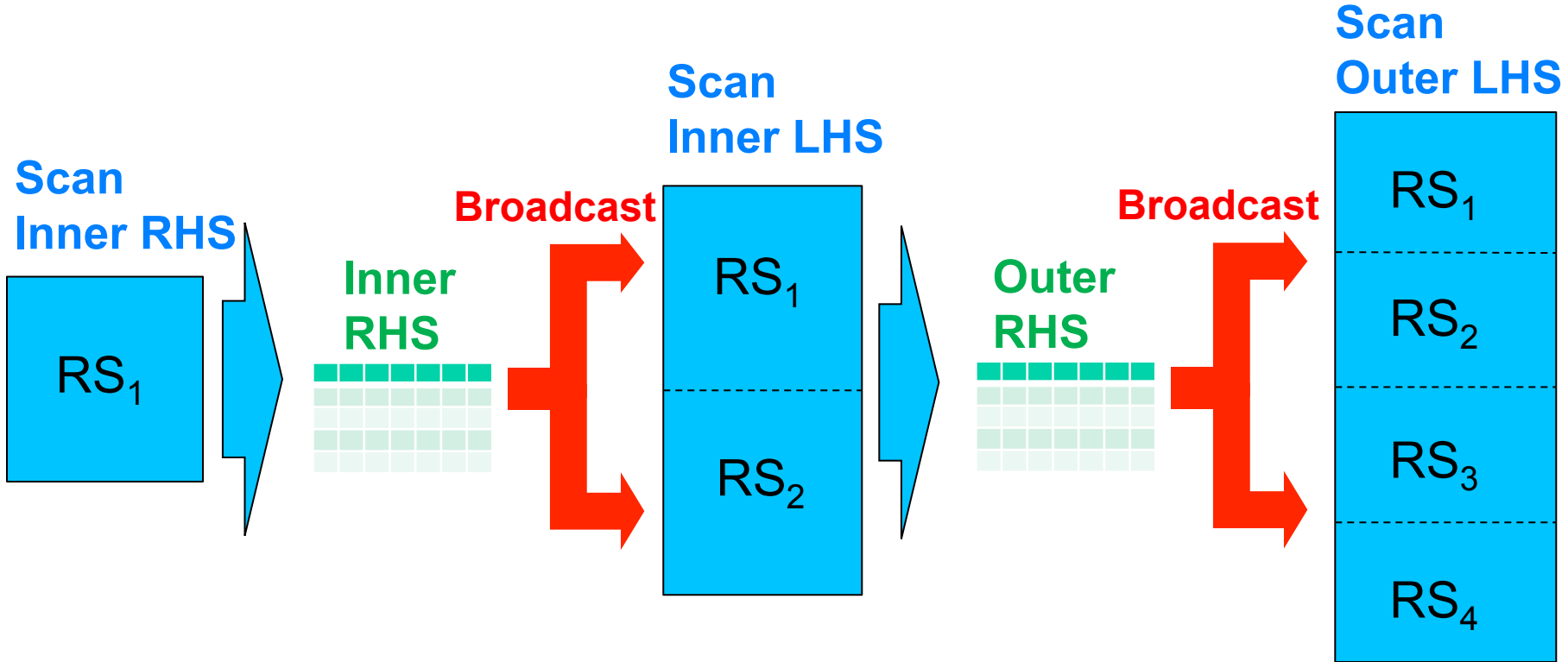
```
SELECT *  
FROM host_info
```

#### Inner RHS

```
SELECT *  
FROM location_info  
WHERE  
user_count >= 200000
```

# Phoenix Push Down: Hash Join

## Iterative Execution of Multiple Joins



# 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
```

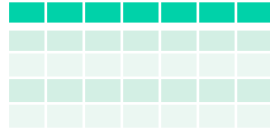


# Phoenix Push Down: Hash Join Star-join Execution of Multiple Joins

Scan RHS 1



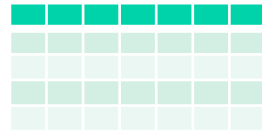
RHS 1



Scan RHS 2



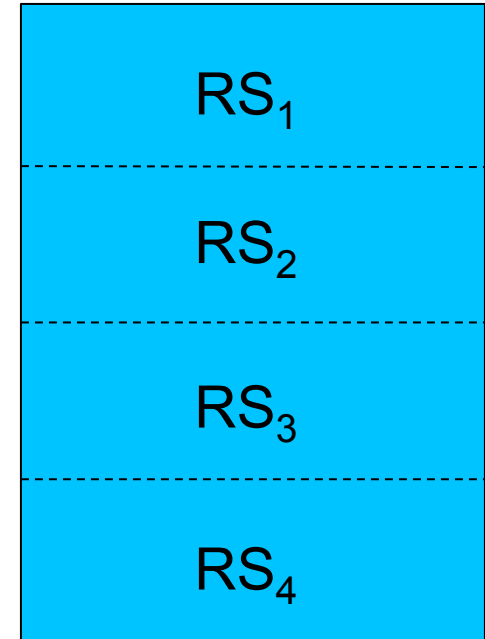
RHS 2



Broadcast



Scan LHS





# 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



# 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





# 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, } First PK column identifies tenant ID  
  type CHAR(1),  
  event_id BIGINT,  
  created_date DATE,  
  created_by VARCHAR,  
  CONSTRAINT pk PRIMARY KEY (tenant_id, type, event_id))  
MULTI_TENANT=true;
```



# Tenant-Specific Tables

Step 2: Create tenant-specific tables

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

Tenant-specific  
connection

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

```
CREATE VIEW my_web_event AS  
SELECT * FROM web_event;
```

Tenant-specific view





# Tenant-Specific Tables

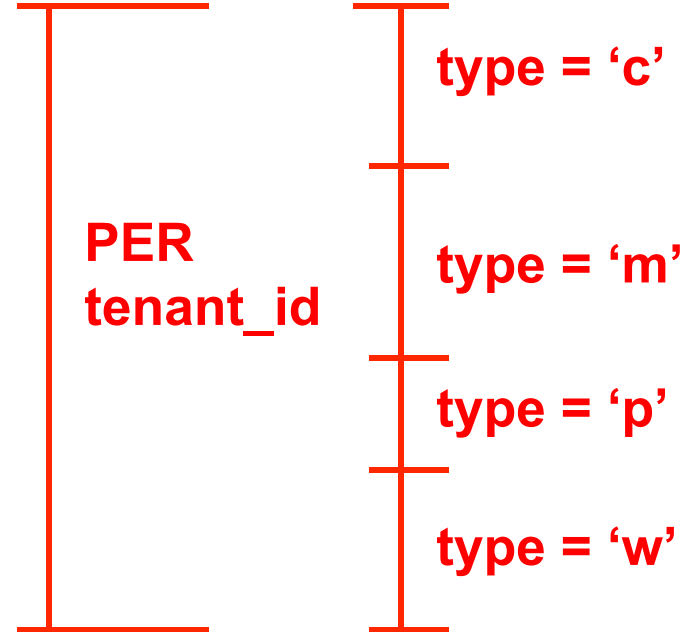
| EVENT |
|-------|
|       |
|       |
|       |
|       |
|       |
|       |
|       |
|       |
|       |
|       |

tenant\_id = 'me'  
tenant\_id = 'you'  
...  
tenant\_id = 'them'



# Tenant-Specific Tables

| EVENT        |
|--------------|
| CHAT_EVENT   |
| MOBILE_EVENT |
| PHONE_EVENT  |
| WEB_EVENT    |





# 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?**