Project Seeds

Languages & Runtimes for Big Data

Reminder

- Homework 1: Database Cracking
  - Read the paper (linked from the course page)
  - Submit 2 discussion points (strength and weakness of the work) or make a counterargument to someone else's points via Disqus
  - If you're uncomfortable using Disqus, email me (with [CSE-662] in the subject line)
- Disqus thread started for group formation

Types of Projects

- Data Quality
- Query Processing
- Index Structures
- Pocket Scale Data

Checkpoint Expectations

- Checkpoint 1: Project Description (Due by 11:59 PM Sept. 26)
  - What is the specific challenge that you will solve?
  - What metrics will you use to evaluate success?
  - What deliverables will you produce?
- Checkpoint 2: Progress Report (Due by 11:59 PM Oct. 22)
  - What challenges have you overcome so far?
  - How does your existing work compare to other, similar approaches?
  - How have your goals changed from checkpoint 1?
  - What challenges remain for you to overcome?
- Checkpoint 3: Final Report (Due by 11:59 PM Dec. 3)
  - What specific challenge did you solve?
  - How does your final solution compare to other, similar approaches?

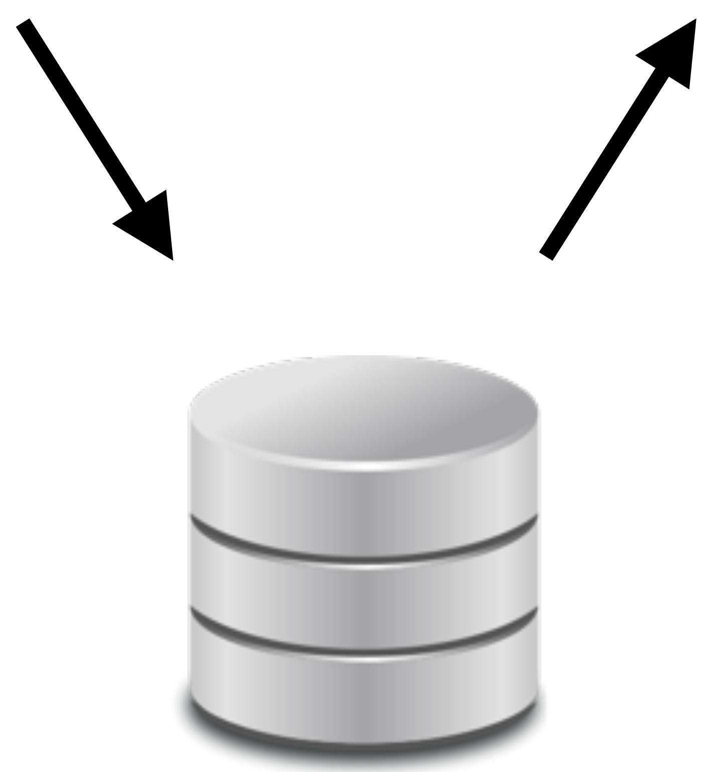
Deferred Constraint-Based Data Validation

Constraint

Temperature Changes at < 5˚ C/Hr
One Unique SS# Per Person
Weight Variance < 20lb

Constraint Violations

{ <12:45, 20˚C>, <13:45, 30˚C> }
{ <12345, “Alice”>, <12345, “Bob”> }
{ <“Jan”, 160lb>, <“Feb”, 180lb>, <“Mar”, 220lb> }



Deferred Constraint-Based Data Validation

Query

Average Temperature Over the Past Week

What’s Bob’s SS#?

What was the weight in Feb?

Answer

25˚C

12345

180 lb

Constraint Violations

{ <12:45, 20˚C>, <13:45, 30˚C> }

{ <12345, “Alice”>, <12345, “Bob”> }

{ <“Jan”, 160lb>, <“Feb”, 180lb>, <“Mar”, 220lb> }

Deferred Constraint-Based Data Validation

Query

Average Temperature Over the Past Week
What's Bob's SS#?
What was the weight in Feb?

Answer

25˚C ± 3˚
12345 or ?
180 lb ± 40 lb

Constraint Repairs

Deferred Constraint-Based Data Validation

- **Language**: SQL + (Scala or Java)
- **First Steps**: Read up on constraint repair and triggers.
- **Expected Outcomes**: I give you a query, you tell me which rows/cells are complicit in a constraint violation.

Query Sampling Optimizer

Uncertain Data

< Spot, { Alive | Dead } >

```
SELECT COUNT(*) FROM Cats
WHERE State = 'Alive';
```

```
   COUNT
-----------
 { 0 | 1 }
```

Query Sampling Optimizer

Uncertain Data

World 1: < Spot, Alive >

World 2: < Spot, Dead >

```
SELECT COUNT(*) FROM Cats
WHERE State = 'Alive';
```

```
 WORLD | COUNT
-------+---------
   1   |   1
   2   |   0
```

Query Sampling Optimizer

WORLD	Cat	State
1	Spot	Alive
2	Spot	Dead

```
SELECT COUNT(*) FROM Cats
WHERE State = 'Alive'
GROUP BY WORLD;
```

WORLD	COUNT
1	1
2	0

Query Sampling Optimizer

1 cat = 2 worlds

2 cats = 4 worlds

10 cats = 1024 worlds

…

n cats = 2^N worlds

Query Sampling Optimizer

Idea: Sample from the worlds

Query Sampling Optimizer

Interleaved:

WORLD	Cat	State
1	Spot	Alive
2	Spot	Dead

Tuple Bundle:

Cat	State
Spot	[Alive, Dead]

or

Cat	State_1	State_2
Spot	Alive	Dead

Query Sampling Optimizer

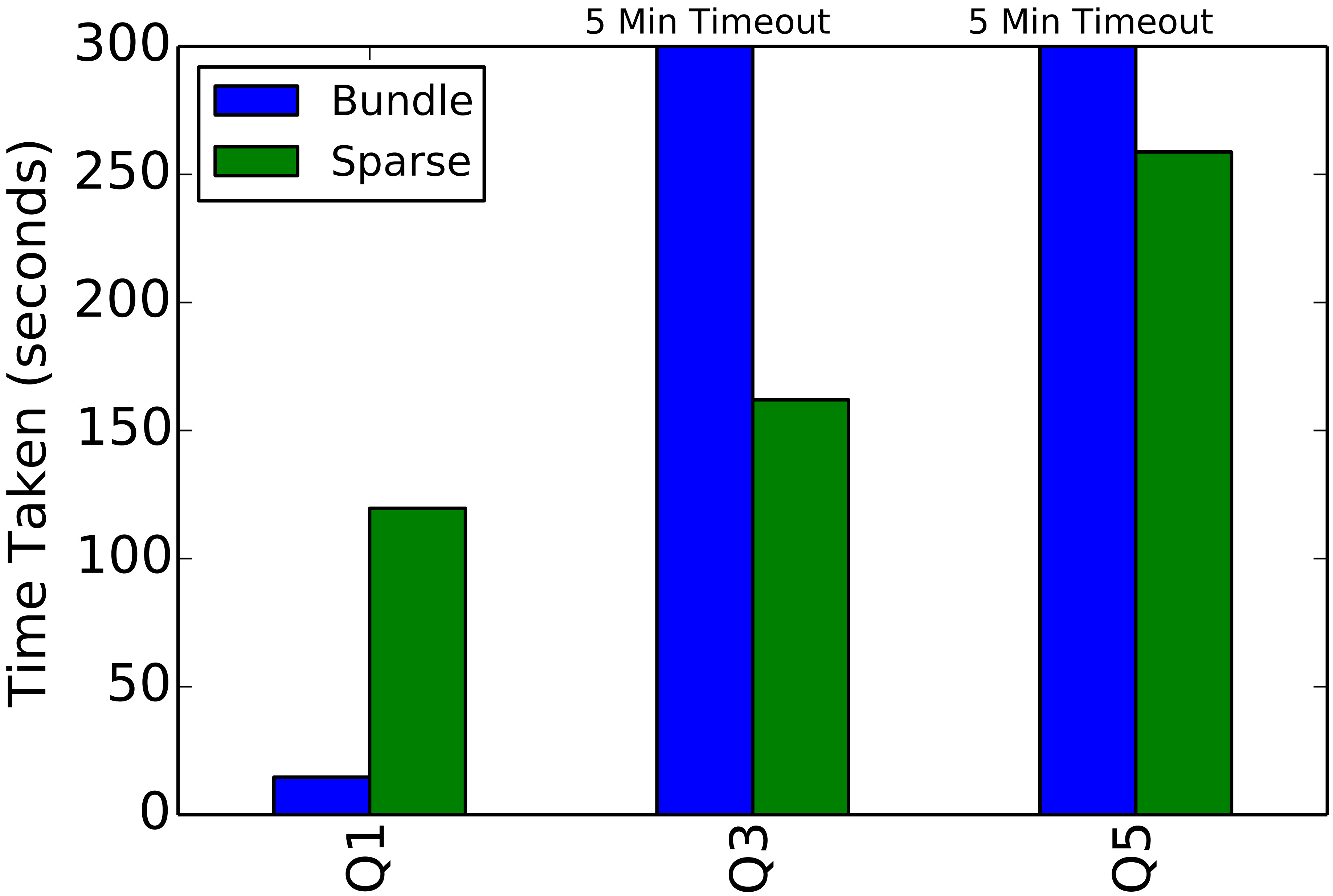
Interleaved:

```
SELECT COUNT(*) FROM Cats
WHERE State = 'Alive'
GROUP BY WORLD;
```

Tuple Bundle:

```
SELECT
   SUM(
     CASE WHEN State_1 = 'Alive' THEN 1
          ELSE 0 END) AS COUNT_1,
   SUM(
     CASE WHEN State_2 = 'Alive' THEN 1
          ELSE 0 END) AS COUNT_2
FROM Cats;
```

a



5 Min Timeout
5 Min Timeout
Bundle
Sparse
Time Taken (seconds)
300
250
200
150
100
50
0
Q1
Q3
Q5

Query Sampling Optimizer

- **Language**: RA + Scala
- **First Steps**: Install Mimir and get it to compile
- **Expected Outcomes**: I give you a query and you give me a sampling-based execution plan for it.

Explaining Offset-Outliers

```
SELECT Neighborhood, Week, COUNT(*)
FROM PoliceComplaints
WHERE Type = 'Noise'
```

Why so many?

Neighborhood	Week	COUNT
Black Rock	1	53
Black Rock	2	10
Amherst	1	5
Amherst	2	6
Elmwood	1	10
Elmwood	2	9

Explaining Offset-Outliers

e.g., There were fewer noise complaints
that week everywhere else.

# of noise complaints
in all of Buffalo is stable

Black Rock, Week 1
is counterbalanced by
a dip elsewhere

"What's Normal" → **"How's this different from normal"**

Explaining Offset-Outliers

<u>“What’s Normal”</u>

For all X:

f(X) ≈

```
SELECT g, COUNT(*)
FROM Data
WHERE c = X
GROUP BY g
```

Explaining Offset-Outliers

<u>“What’s Normal”</u>

For all Cities C:
f(C) ≈

```
SELECT week, COUNT(*)
FROM NoiseComplaints
WHERE city = C
GROUP BY week
```

Explaining Offset-Outliers

<u>"What's Normal"</u>

For all Cities C:

f(C) =

```
SELECT AVG(count) FROM (
  SELECT week, COUNT(*) AS count
  FROM …
);
```

```
SELECT week, COUNT(*)
FROM NoiseComplaints
WHERE city = C
GROUP BY week
```

Explaining Offset-Outliers

```
SELECT neighborhood, city, week, COUNT(*)
FROM NoiseComplaints
GROUP BY week
```

Why so many?

Neighborhood	City	Week	COUNT
Black Rock	BUF	1	53
Black Rock	BUF	2	10
Amherst	BUF	1	5
Amherst	BUF	2	6
Elmwood	BUF	1	3
Elmwood	BUF	2	9

...

Explaining Offset-Outliers

Question 1: Is the overall situation “normal”?

(Are there more noise complaints than usual in Buffalo?)

Question 2: Is the cell abnormally high (or low)?

(Are there more noise complaints in Black Rock compared to the average week?)

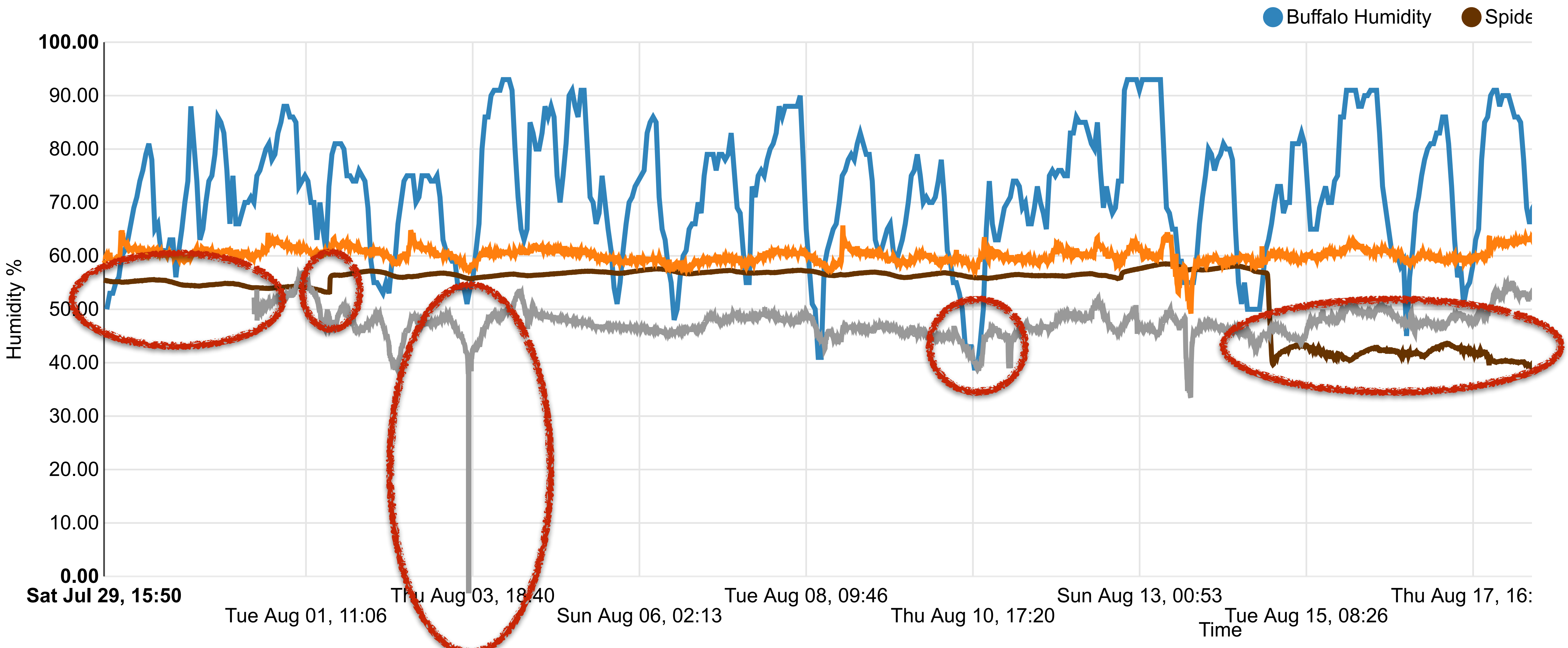
Question 3: What counterbalances the cell?

(Are there other neighborhoods where noise complaints dropped that week?)

Explaining Offset-Outliers

- **Language**: SQL + [Your Choice]
- **First Steps**: Write a piece of code to execute aggregate SQL queries with varying sets of group-by terms.
- **Expected Outcomes**: I give you a dataset and a set of stability constraints on that data, and you give me a set of explanations for outliers.

Physical Layouts for Forked Data



Physical Layouts for Forked Data

Just because something is an outlier doesn't mean that the data should be removed.

… but now you need to keep track of multiple "versions" of the data.

Physical Layouts for Forked Data

Query A: Lookup key K in version V

Query B: Lookup keys in range $[K_1, K_2]$ in version V

Query C: Find all versions with keys in range $[K_1, K_2]$

Query D: Find all keys in range $[K_1, K_2]$ with identical values in all versions

Query E: Find all keys in range $[K_1, K_2]$ with at least one version-based difference.

Physical Layouts for Forked Data

<u>Naive 1: Version Tuples</u>

Faster for querying
all versions (C, D, E)

<u>Naive 2: Version Tables</u>

(or indexes)

Faster for querying
one version (A, B)

Physical Layouts for Forked Data

- **Language**: [Your Choice – C/C++ Suggested]
- **First Steps**: Implement a simple B+ tree in your language of choice.
- **Expected Outcomes**: A data store that supports efficient point/range queries across branches, forking, and both batch and single-branch updates.

Adaptive Multidimensional Indexing

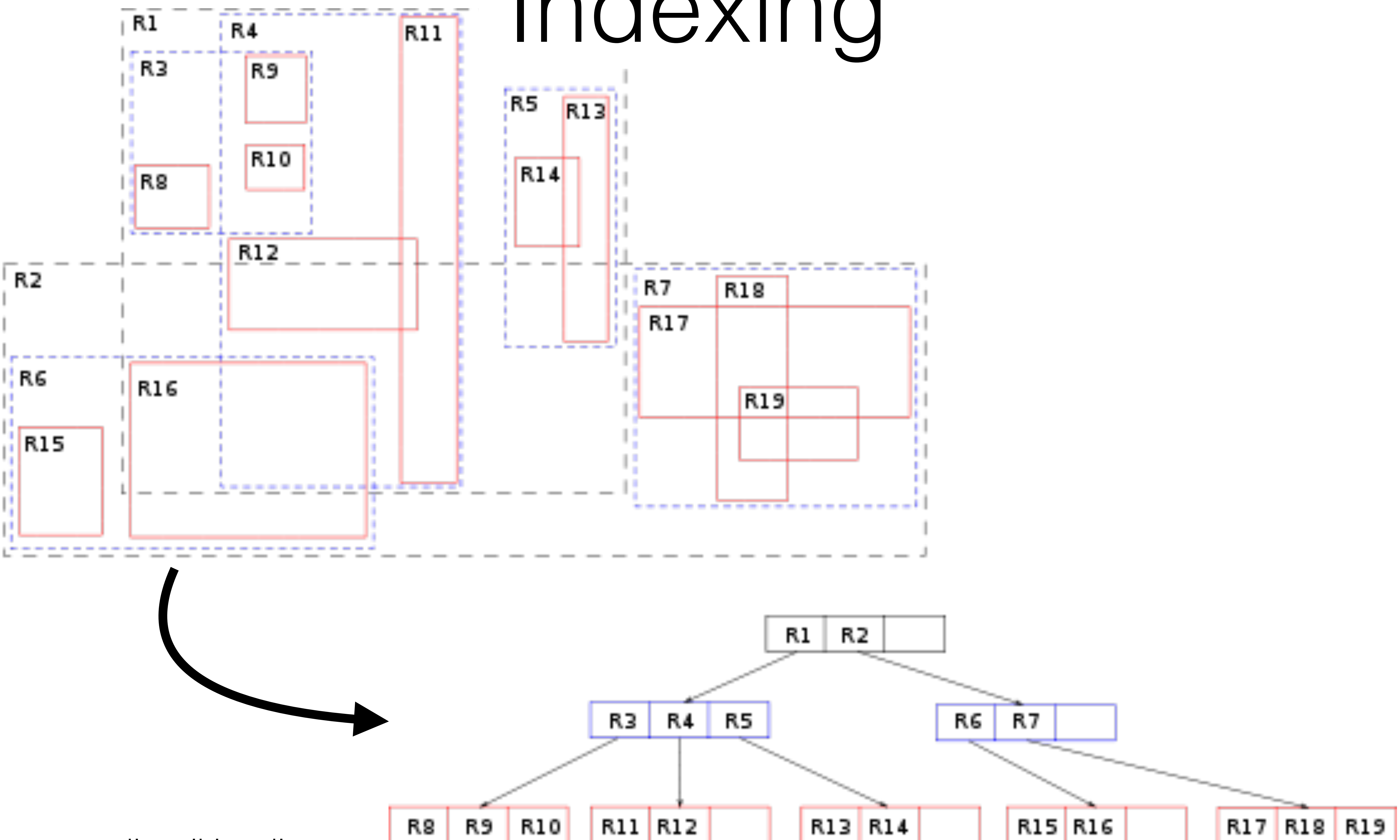


image credit: wikipedia

Adaptive Multidimensional Indexing

Problem: How to subdivide records?
(there's no globally ideal sort order)

Approach 1: Take a hint from the query workload.
(Use query boundaries as partition points)

Approach 2: Keep learning from the query workload.
(Repartition data according to query boundaries)

Adaptive Multidimensional Indexing

- **Language**: [Your Choice – C/C++ Suggested]
- **First Steps**: Implement a simple R* tree in your language of choice.
- **Expected Outcomes**: A 2-dimensional cracker index, ideally supporting dynamic repartitioning as workloads change.

Mimir on SparkSQL

Mimir on SparkSQL

Relational Algebra	**Spark DataFrames**
Relation	DataFrame
Project	R.map { tuple => … }
Select	R.filter { tuple => … }
Aggregate	R.groupBy().[…]
Join	R.flatMap { tupleR => S.map { tupleS => ... } }
Union	R.union(S)

Mimir on SparkSQL

Devil in the Details

Implementing User-defined functions and aggregates

Spark is Read-Only (Mimir needs metadata)

Dynamically compiling maps, filters, etc…

Schema management

Mimir on SparkSQL

- **Language**: Scala
- **First Steps**: Get Mimir compiling
- **Expected Outcomes**: A version of mimir backed by SparkSQL, with an independent metadata store.

In-Class Assignment

- Form a group of 4 as a project group for the duration of the semester
- Come up will a clever group name
- Challenge: form a group with people you do not know or do not know well