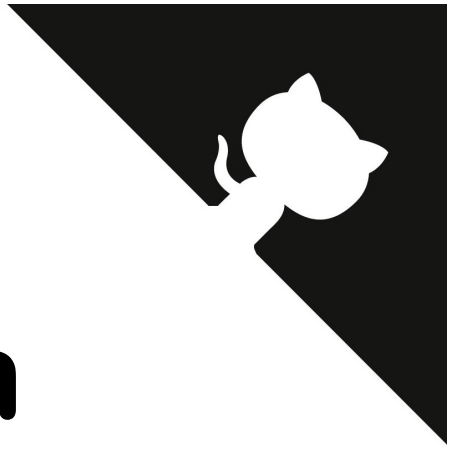


The Decomposition of Databases™

@mhelmich – 07/20/2017

https://github.com/mhelmich/carbon-copy

Databases are Magicians

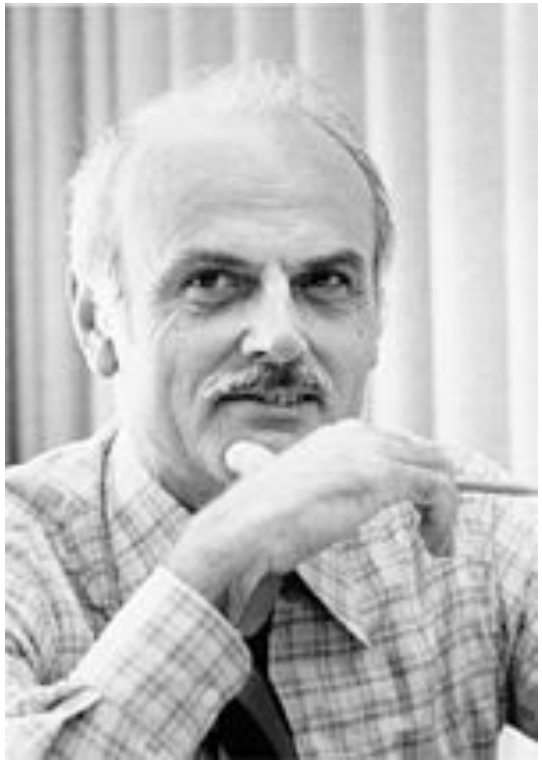
"Magic seems to break the laws of physics but it's an illusion [… relational databases are magicians] Because they decouple 'what you want' from 'how you get it'"[*1]

- The core illusion relational databases entertain us with is that we actually don't know how and where the data is stored
  - And we don't need to know in order to retrieve data

*1 https://medium.com/salesforce-engineering/the-architecture-files-ep-4-the-database-is-a-magician-b951945ea5b8

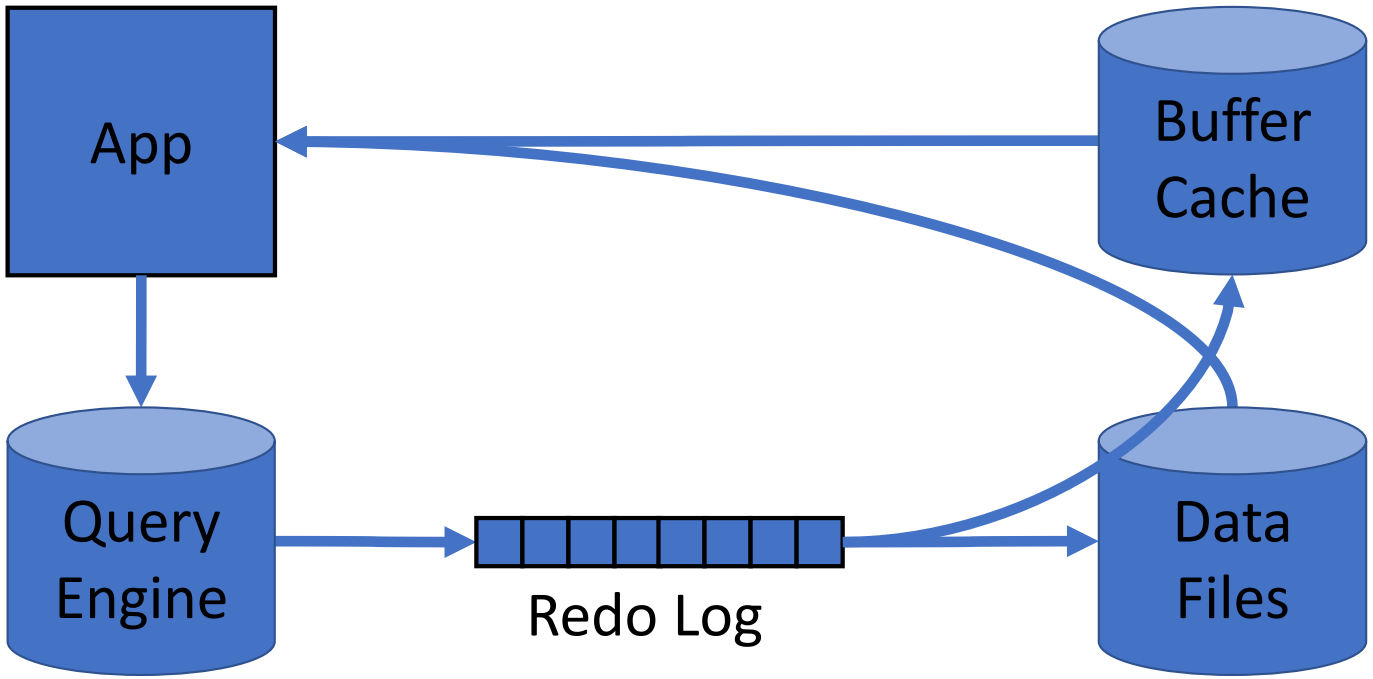
A History Lesson

- Data was structured in a hierarchical way
  - Imagine this much like a JSON file
  - Code would read like a description of how to get the data you want
- Until Dr. Frank Codd came around
  - Combining set theory and graph theory to propose a declarative way of describing ”the data you want”

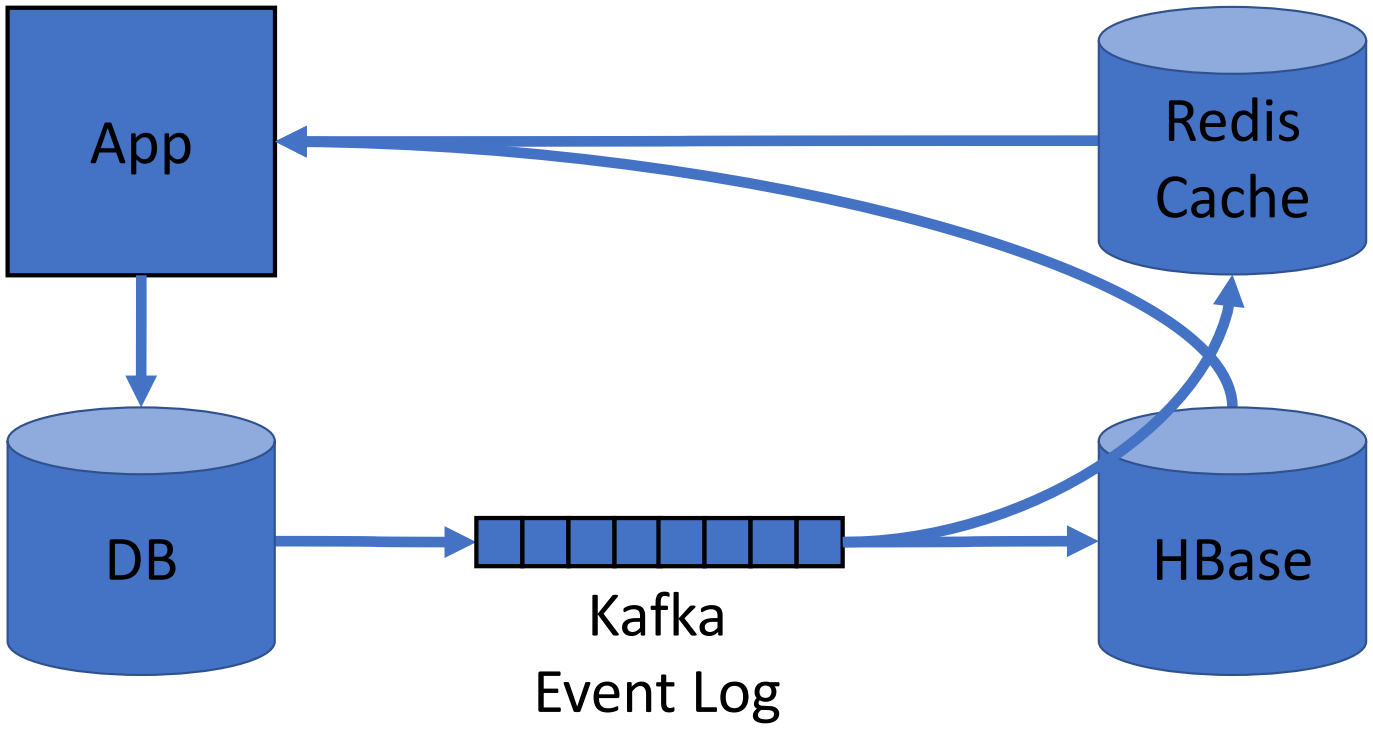


https://en.wikipedia.org/wiki/Edgar_F._Codd#/media/File:Edgar_F_Codd.jpg

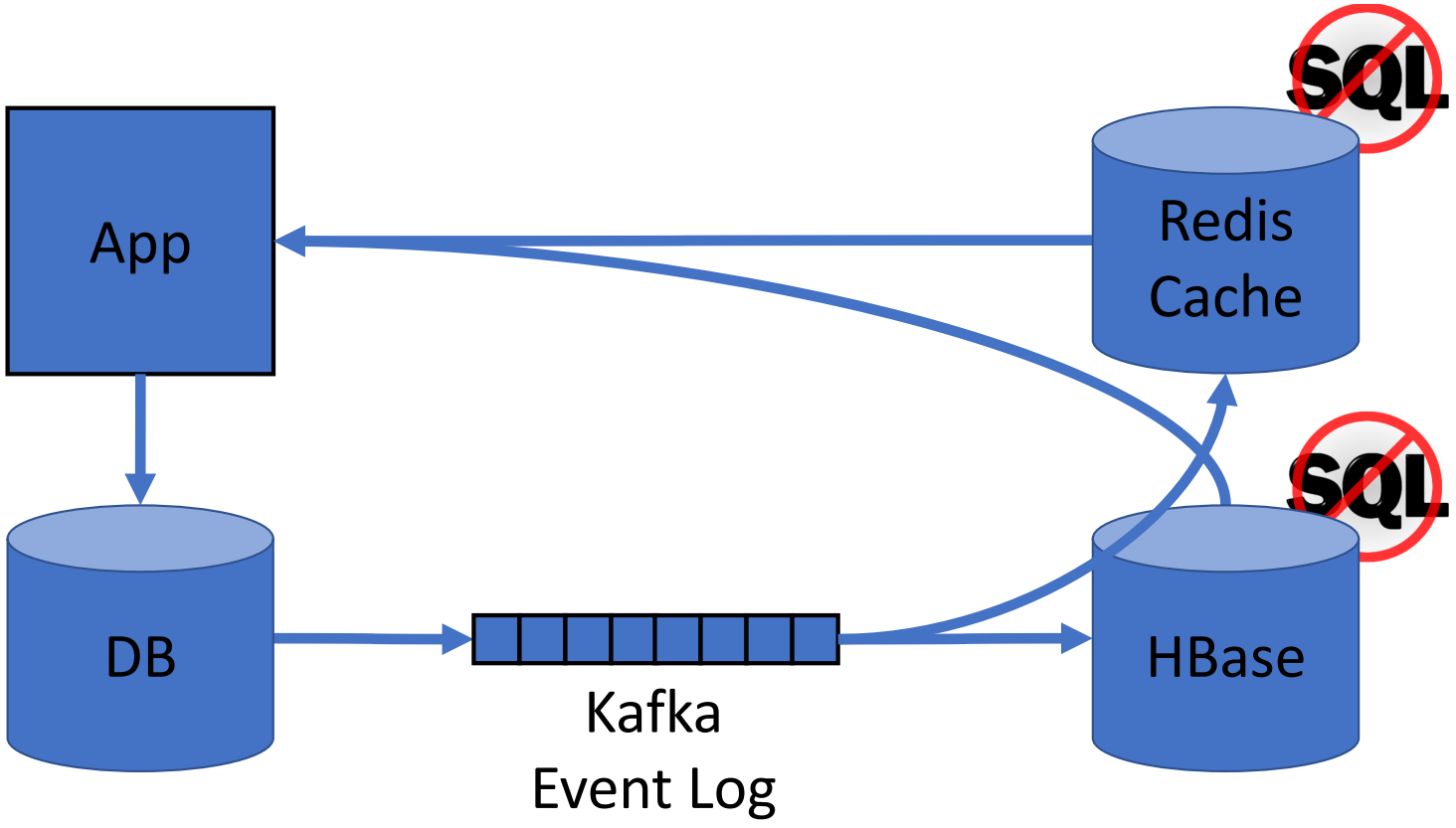
Recap: How do databases work again?



The Decomposition of Databases



The Decomposition of Databases



https://apigee.com/about/blog/technology/nosql-noproblem-mapping-your-sql-thinking-nosql

The Decomposition of Databases

```
Jedis jedis = new Jedis();
Map<String, Double> scores = new HashMap<>();

scores.put("PlayerOne", 3000.0);
scores.put("PlayerTwo", 1500.0);
scores.put("PlayerThree", 8200.0);

scores.keySet().forEach(player -> {
    jedis.zadd("ranking", scores.get(player), player);
});

String player = jedis.zrevrange("ranking", 0, 1).iterator().next();
long rank = jedis.zrevrank("ranking", "PlayerOne");
```

http://www.baeldung.com/jedis-java-redis-client-library

The Decomposition of Databases

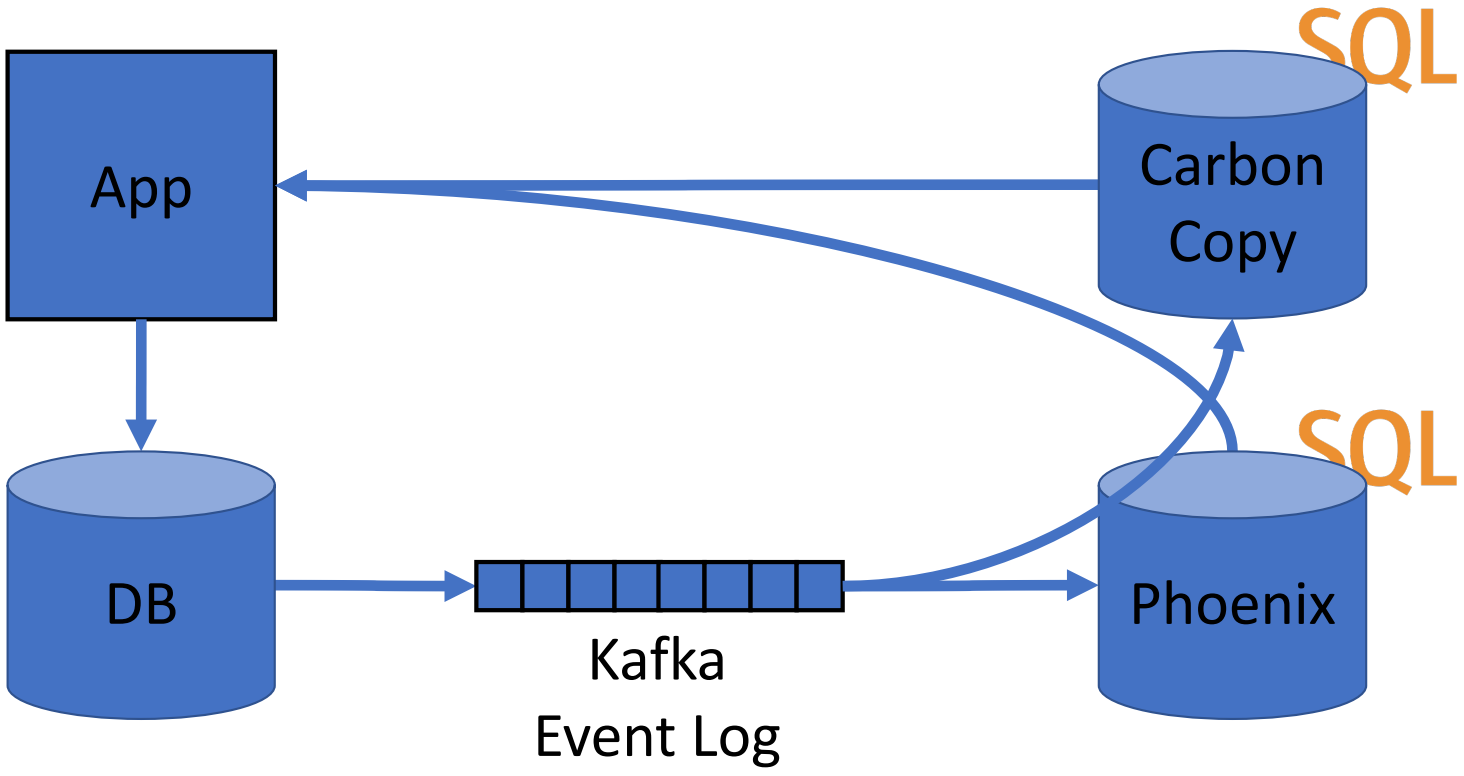
player	score
PlayerOne	3000.0
PlayerTwo	1500.0
PlayerThree	8200.0

```
SELECT player FROM ranking ORDER BY score LIMIT 1;
SELECT ROWNUMBER() FROM ranking WHERE name = 'PlayerOne' ORDER BY score;
```

Enter Carbon Copy

- An in-memory cache that speaks SQL
  - Tell me “what you want” not “how to get it”
  - Ships with its own JDBC driver
- Based on two design concepts
  - Data placement
  - Minimum coordination

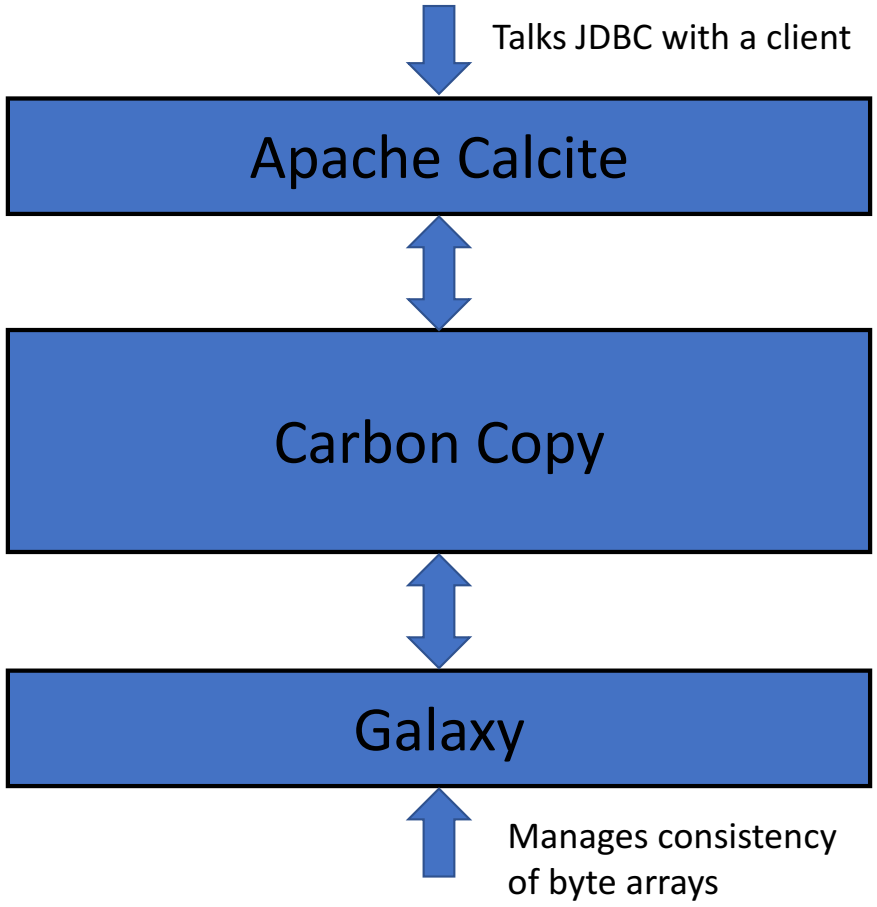
The Decomposition of Databases



https://cdn.codementor.io/assets/topic/category_header/sql-5850f823f464b56f4c150db67dae5081.png

Carbon Copy

- Calcite providing JDBC interface, query planning in a box
- Carbon Copy building complex data structures, managing data placement and query distribution
- Galaxy providing a consistency framework for byte[]



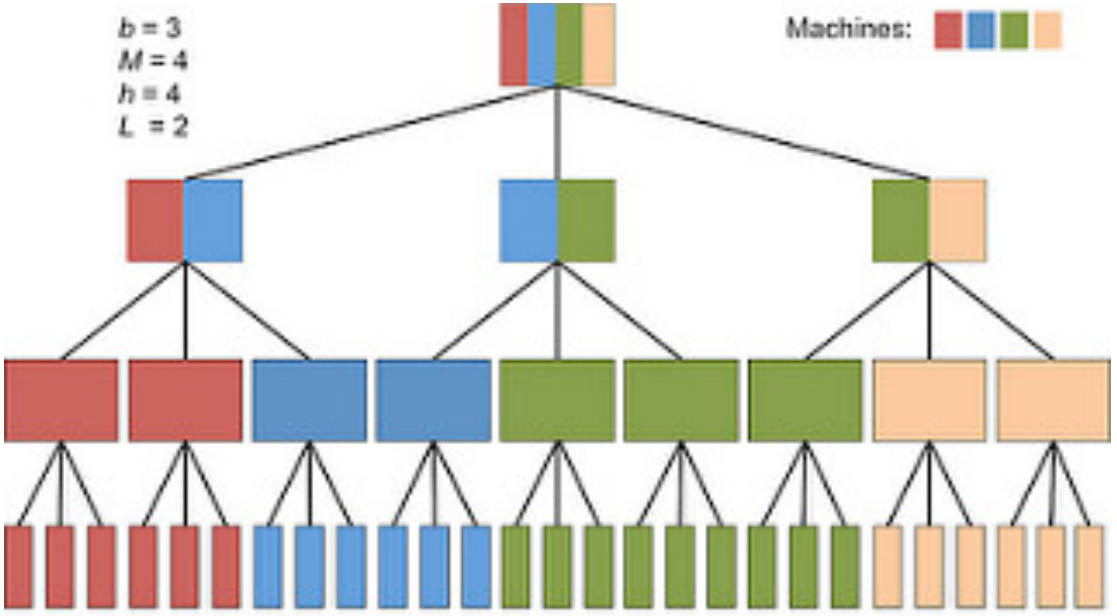
Behind the Scenes
Byte Arrays to distributed Indexes

- Started with byte[]
  - DataBlock is the simplest data structure
  - Just 32kb long linked lists in a byte array

- DataBlocks are composing complex data structures
  - Like hashes and btrees and tables and indexes

Behind the Scenes Distributed BTrees

- Two fundamental design principles explained
  - Minimum coordination
  - Data placement
    - Code is moved to data



A tree distributed with Galaxy

http://highscalability.com/blog/2012/8/20/the-performance-of-distributed-data-structures-running-on-a.html

Next Up

- Performance and scalability
- Feed the query optimizer
  - Collecting stats
  - Data sampling
- Monitoring
- Automatic generation of indexes based on usage patterns

Thank you!

- Fork me on github: https://github.com/mhelmich/carbon-copy
- Questions?