NOSQL DATA MODELS

NoSQL taxonomy

- Key-Value stores
- Column stores
- Document stores

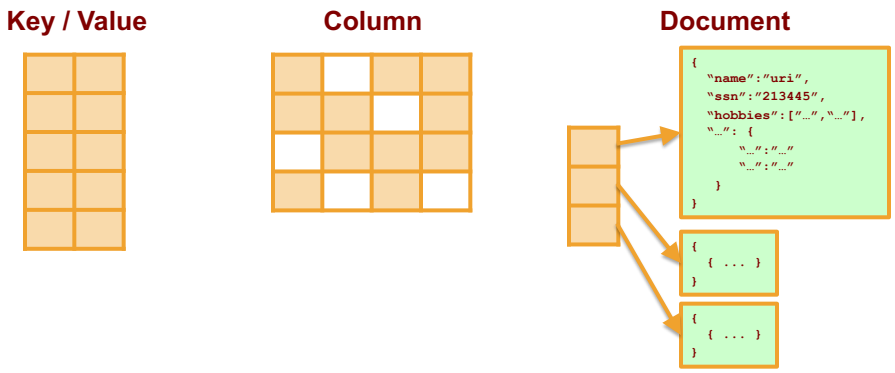
NoSQL taxonomy

- Key-Value stores — Amazon Dynamo
- Column stores — Google Bigtable, Cassandra
- Document stores — CouchDB, MongoDB, SimpleDB

4

4

A Universe of Data Models

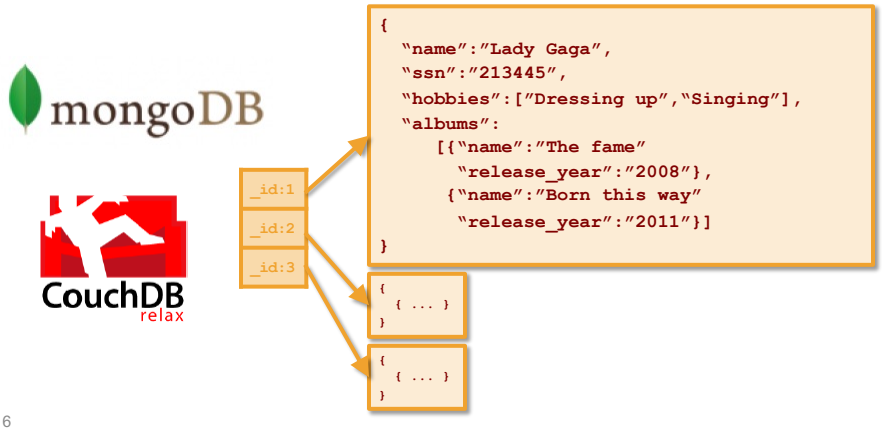


5

5

Document

- Think JSON (or BSON, or XML)



6

6

Key/Value

- Have the key? Get the value
  - Map/Reduce (sometimes)
  - Good for
    - cache aside (e.g. Hibernate 2nd level cache)
    - Simple, id based interactions (e.g. user profiles)
- In most cases, values are opaque

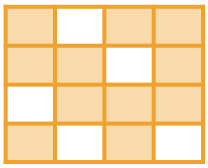
K1	V1
K2	V2
K3	V3
K4	V1

7

7

Column Based

- Mostly derived from Google's BigTable papers
- One giant table of rows and columns
  - Column == pair (name and a value, sometimes timestamp)
  - Table is sparse:
    (#rows) × (#columns) ≥ (#values)

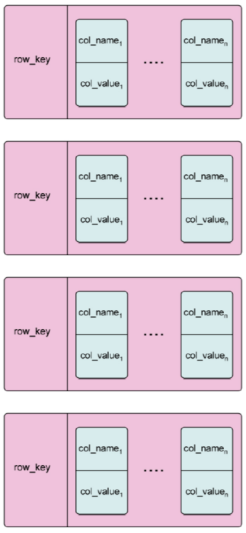


8

8

Column Based

- Query on row key
  - Or column value (aka secondary index)
- Good for a constantly changing, (albeit flat) domain model



9

9

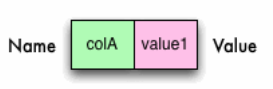
BIGTABLE: COLUMN-BASED STORE

10

10

Data Model

A single column

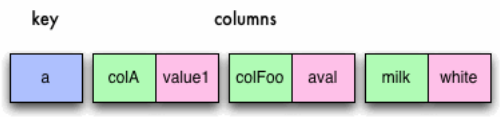


11

11

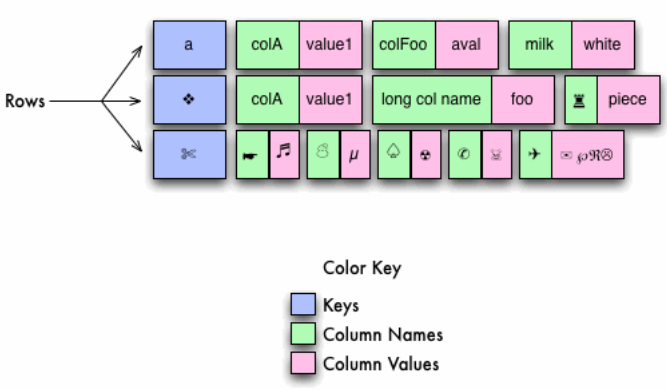
Data Model

A single row



12

Data Model



13

Data Model - Vocabulary

- Keyspace – like namespace for unique keys.
- Column Family – very much like a table… but not quite.
- Key – a key that represents row (of columns)
- Column – representation of value with:
  - Column name
  - Value
  - Timestamp

Data Model - Column Family

- Similar to SQL tables
- Groups of column keys
- Hundreds of static column families
- Syntax is family:key, e.g., Language:English, Language:German, etc

Data Model - Rows

- Primary key for objects
- All keys are arbitrary length binaries

```
Users:                                  CF
    foo:                                 ROW
        emailAddress: foo@bar.com,      COLUMN
        webSite: http://bar.com         COLUMN
    bar:                                ROW
        emailAddress: bug@example.org   COLUMN
Stats:                                  CF
    foo:                                 ROW
        visits: 243                     COLUMN
```

16

16

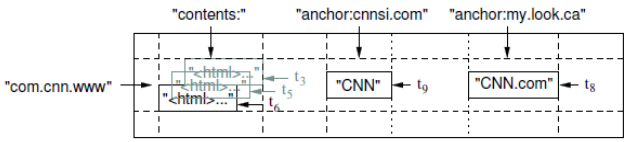
Data Model Summary

- Keyspace
  - ColumnFamily
    - Row (indexed)
      - Key
      - Columns
        - Name (sorted)
        - Value

17

Bigtable

- A sparse, distributed persistent multi-dimensional sorted map.
- The map is indexed by a row key, column key, and a timestamp.
- Output value is array of bytes.
  - (row: byte[ ], column: byte[ ], time: int64) → byte[ ]



18

18

Rows

- Read/write of data under a single row is atomic
- Data ordered by row key
- Row range dynamically partitioned
- **Tablet**: partition (row range)
  - Unit of distribution and load-balancing
- Objective: make read operations single-sited!
  - E.g., In Webtable: com.google.maps instead of maps.google.com.

19

19

Timestamps

- 64 bit integers
- Items in a cell stored in decreasing timestamp order
- Application specifies how many versions (n) of data items are maintained in a cell.
  - Bigtable garbage collects obsolete versions.

20

20

Application 1: Google Analytics

- Webmaster stats:
  - Number of unique visitors per day
  - Page views per URL per day,
  - Percentage of users that made a purchase given that they earlier viewed a specific page.
- How?
  - JavaScript embedded in web pages, records:
    - User identifier
    - The page being fetched

21

21

Application 1: Google Analytics

- Raw click table (~ 200 TB)
  - A row for each end-user session
  - Row name includes website name and time of session
  - Clustering of sessions that visit the same web site
- Summary table (~ 20 TB)
  - Predefined summaries for each web site.
  - Generated from raw click table by periodic MapReduce jobs.
  - Row name includes website's name
  - Column family is aggregate summaries

Application 2: Personalized Search

- Records user queries and clicks across Google
- Users browse their search histories
- Personalized search results
- One Bigtable:
  - Row name is userid
  - Column family for each action type, e.g., web queries, clicks.
  - User profiles are generated using MapReduce.
  - Replicated geographically to reduce latency and increase availability.