Automated Schema Design for NoSQL Databases

Michael J. Mior

Supervised by Kenneth Salem
For a given workload, construct the optimal physical schema to answer queries in the workload for a given storage budget
What is NoSQL?

- Not SQL?
- NoACID?

- Not only SQL
Types of NoSQL Databases

- Document stores
- Key-value stores
- Graph databases
- ...
- Wide column stores
Cassandra data model

<table>
<thead>
<tr>
<th>Column Family</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row A</strong></td>
</tr>
<tr>
<td>Column 1</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td><strong>Row B</strong></td>
</tr>
<tr>
<td>Column 2</td>
</tr>
<tr>
<td>Value</td>
</tr>
</tbody>
</table>
Cassandra data model

Queries specify

A set of row keys
  and either
A list of column keys
  or
A prefix of the column keys with a range
Cassandra schema design

No single obvious mapping from the conceptual schema
Cassandra schema design

- Multiple choices of logical and physical schema are possible for a given conceptual schema
- Different choices are more suitable for different queries
- Harder to add additional structures in the future
Cassandra schema design

**POIs**

<table>
<thead>
<tr>
<th>POI17</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Liberty Bell”</td>
</tr>
</tbody>
</table>

**Hotels**

<table>
<thead>
<tr>
<th>Hotel3</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“Holiday Inn”</td>
</tr>
</tbody>
</table>

**HotelToPOI**

<table>
<thead>
<tr>
<th>Hotel3</th>
<th>POI17</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(null)</td>
<td></td>
</tr>
</tbody>
</table>

**POIToHotel**

<table>
<thead>
<tr>
<th>POI17</th>
<th>Hotel3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(null)</td>
<td></td>
</tr>
</tbody>
</table>
Cassandra schema design

POIToHotel

<table>
<thead>
<tr>
<th>POIID</th>
<th>HotelID</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>(null)</td>
<td>…</td>
</tr>
</tbody>
</table>

HotelToPOI

<table>
<thead>
<tr>
<th>HotelID</th>
<th>POIID</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotel3</td>
<td>POI17</td>
<td>“Liberty Bell”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HotelID</th>
<th>POIID</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Holiday Inn”</td>
<td>POI17</td>
<td>…</td>
</tr>
</tbody>
</table>
Challenges

- Storage costs
- View maintenance
- Transactions
- Data distribution
Relational schema design

- Commercial database vendors have existing tools
  - Microsoft AutoAdmin for SQL Server
  - DB2 Design Advisor
  - Oracle SQL Tuning Advisor

- These tools don’t solve all the problems for NoSQL databases

- Usually an existing schema is assumed, and tools suggest secondary indices and materialized views
Relational schema design

Conceptual schema → Logical schema → Physical schema

```
CREATE TABLE Hotel(...);
CREATE TABLE POI(....);
CREATE TABLE HotelToPOI(...,
    FOREIGN KEY (HotelID)
    REFERENCES Hotel(HotelID)
    FOREIGN KEY (POOID)
    REFERENCES POI(POIID)
);
CREATE MATERIALIZED VIEW POIsWithHotels AS SELECT POI.*, Hotel.* FROM POI, HotelToPOI, Hotel
    WHERE HotelToPOI.HotelID = Hotel.HotelID
    AND HotelToPOI.POIID = POI.POIID;
```

Mapping from conceptual to logical schema is straightforward
Cassandra schema design

No single obvious mapping from the conceptual schema
Abstract example
Get all points of interests near hotels a guest has stayed at

```
SELECT Name FROM POI WHERE
    POI.HotelToPOI.
    Hotel.Room.Reservation.
    Guest.GuestID = ?
```
Index paths

- Queries can “skip” entities on a path using an index
- Indices to the final entity are equivalent to materialized views
- Schema design is now the selection of these indices
Problem definition

Input:
- ER diagram with entity counts and cardinalities
- Weighted queries over the diagram
- Storage constraint
- Cost model for target DB

Output:
- Suggested index configuration
- Plans for executing each query
Proposed solution

1. Enumerate possible useful indices for all queries

2. Determine the benefit of a given index for each query

3. Select the optimal indices for a given space constraint
Future Work

- Retargeting to other NoSQL DBs
- Automatic implementation of query plans
- Handling workloads which include updates
- Richer query language
- Exploitation of DBMS-specific features
Summary

- Schema design in NoSQL databases presents unique challenges
- A workload-driven approach is necessary
- Conceptual schemata and queries are viable abstractions for NoSQL schema design tools
Query language

SELECT [attributes] FROM [entity]
WHERE [path](=|<|<=|>|>=) [value] AND ...
ORDER BY [path] LIMIT [count]

Conjunctive queries with simple predicates and ordering

Higher level queries can be composed by the application
Input

- Workload
- Index Enumerator
  - Index benefit analysis
- Query Planner

Output

- Index Selection
  - Selected Indices
- Query Plans
Implementation

- Focus on Cassandra as target system
- Simple cost model for read-only in-memory workloads
- DBMS-independent index enumerator and query planner
- Index selection ILP implemented with GLPK
Evaluation - TBD

- Re-implement existing workloads in the target DBMS (e.g. RUBiS, RUBBoS)
- Develop new benchmarks
- Comparison with a human DBA
- Random ER diagrams/queries with realistic properties