**Developing the Cassandra Data Model**

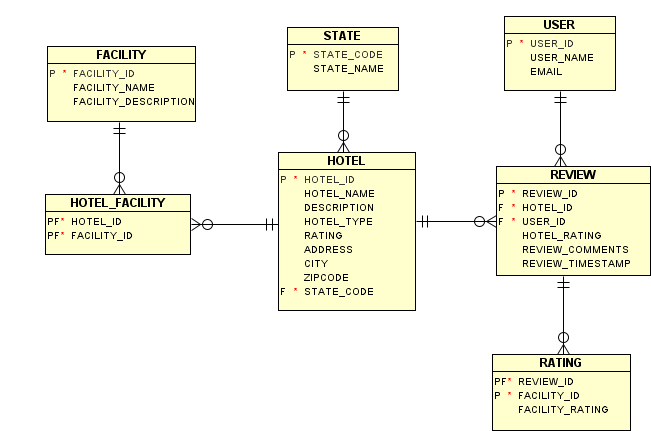
There are two key inputs that will be used to design the Cassandra data model.

1. The relational model of the data elements.
2. The various ways the data will be queried by the application.

We choose to start with the relational model in order to ensure completeness in terms of all the data elements that need to be considered for the solution on Cassandra and in capturing the relationships between them.

Once the preliminary data elements have been captured, the nature of the queries and Cassandra’s architecture will drive the rest of the design. The translations that will be applied on the first level model are targeted at achieving optimum query performance with the right levels of de-normalization implemented.

1. **Design the Entities, Attributes, Keys and Relationships**
2. Model the data using the relational modeling approach.



1. Categorize the entities.

|  |  |
| --- | --- |
| **Entity Category** | **Description** |
| Primary Parent | These are entities that are not a child of any other entity. |
| Simple Join | These are entities that are used to maintain many to many relationships. The resultant entity must not have any additional attributes other than the primary keys of the parent entities. |
| Child | These are entities that have a many to one relationship with another entity.  The primary key of the parent entity is either defined as a foreign key column or is part of an aggregate primary key. |

|  |  |
| --- | --- |
| **Entity** | **Category** |
| State | Primary parent. |
| User | Primary parent. |
| Facility | Primary parent. |
| Hotel | Child. |
| Hotel Facility | Simple Join. Queried by hotel. |
| Review | Child.  A hotel can have multiple reviews.  A user can give multiple reviews. |
| Rating | Child.  A hotel review can have multiple ratings, one per facility. |

1. Apply translations to the relational model to derive the column families and the row keys.
   1. For every Primary Parent entity
      1. Define a Standard Column Family.
      2. The Row Key of the Column Family will be the primary key of the entity. A composite key can be implemented using the Aggregate Key feature in Cassandra.
   2. For every Simple Join entity
      1. Define a Standard Column Family.
      2. The Row Key of the Column Family will be the primary key of the parent entity that will be mostly queried on.
      3. Define valueless columns to store the list of the primary key of the other parent entity.
   3. For every Child entity
      * 1. Define a Standard Column Family.
        2. Define a secondary index on the columns that are the primary key of the parent entity.

**The first level model:**

1. **Designing for Optimal Access**
2. Identify the primary search paths for all the entities.

|  |  |
| --- | --- |
| **Entity** | **Search paths** |
| State | Not queried for |
| User | By User |
| Facility | Not queried for |
| Hotel Facility | By Hotel |
| Hotel | By State Name  By Hotel Name |
| Review | By Hotel  By User |
| Rating | By Hotel |

1. Identify column families that are not searched for via their row key.

For every such column family

1. Define as many super column families as the number of search paths.
2. The super column row key will be the search key.
3. The sub column row key will be a unique identifier for the sub group.
4. For column families with secondary indexes.

If the cardinality of the values in this column is low, retain the secondary index on the column

Else

Define a new column family (materialized view) with the search key being the row key.

Define valueless columns to store the list of the row key of such a column family.

Remove the secondary index in such a column family.

The secondary index on State Code in Column Family Hotel has low cardinality and can be retained.

1. For super column families that have search columns within the sub column, move the sub column to the super column row key (aggregate key).
2. For column families that are do not have any search paths, if they do not have many columns, consider replacing the references to the row keys of these column families in other column families with all the columns.
3. Categorize the columns based on access – high or low. Consider separating columns that are least accessed from column families where other columns are highly accessed.
4. **Validate the model:** Identify as many queries upfront as possible and verify that there are column families that have row keys that match the search keys of these queries.

**The Cassandra Data Model:**