Adam Burton

RDBMS Normalizer

Overview

This program is a normalizer for relational database schema. It takes an argument of the name of a file with a single relation schema. It then prompts for the desired normal form, which it normalizes the table to. The resulting relations are outputted to a file called "normalized_schema.txt" (or, the user can specify a different output location with a second file name argument). If normalized up to BCNF, the output file will not contain table data values. If normalized to 4NF or 5NF, relations that were affected by these normalizations may contain table data values.

Relation format

The input relation must follow a certain syntax. The output relations will follow the same syntax. An example follows:

```
Table: MyTableName
Attributes: attr1:INTEGER, attr2:VARCHAR(255), attr3:VARCHAR(255), attr4:INTEGER
Primary Key: {attr1, attr2}
Candidate Keys: {attr2, attr3}, {attr4}
Multi-Valued Attributes: attr2
Functional Dependencies:
{attr1, attr2} ->> {attr3} | {attr4}
{attr1} -> {attr3}
```

Normalization Logic

First Normal Form

First Normal Form requires the separation of all multi-valued attributes into their own relations. All multivalued attributes are explicitly named in the input. The normalizer takes the list of multi-valued attributes and creates a new table for each of them.

If the attribute is the sole determinant of a functional dependency in the original table, the new table consists of the determinant and the attribute. Otherwise, the new table consists of the original table's primary key and the attribute. Any functional dependencies containing only primary key attributes are kept.

Second Normal Form

Second Normal Form requires the separation of all partial dependencies. into their own relations. In other words, all non-prime attributes must be fully functionally dependent on the primary key, not partially dependent on any candidate keys, and any attributes in violation of this rule must be separated, based on whatever attributes they are dependent on.

For each functional dependency, the normalizer checks if the determinant contains prime attributes but not the primary key. If a partial key determinant is detected, it then checks if the dependent attributes are non-prime. Any located non-prime attributes are added to a list of affected attributes, which are to be separated into a different relation. The new relation contains the functional dependency's determinant, which is set as the primary key, and any affected attributes. The functional dependency is transferred into the relation, along with any functional dependencies based that contain only attributes in the new relation.

Third Normal Form

Third Normal Form requires the separation of transitive dependencies into their own relations. In other words, for every functional dependency, the determinant must be a functional dependency, or the dependent must contain only prime attributes.

For each functional dependency, the normalizer checks if the determinant is not equal to the primary key and the dependent contains non-prime attributes. If this condition is met, the functional dependency violates 3NF and must be separated into its own relation. The determinant of the dependency becomes the primary key, and the dependents are the other attributes. The dependent attributes are removed from the original relation. The functional dependency is transferred to the new relation and removed from the original relation.

Boyce-Codd Normal Form

Boyce-Codd Normal Form is a more strict variation of 3NF that is violated whenever the determinant is not a superkey.

For each functional dependency, the normalizer checks if the determinant is not equal to the primary key. If this condition is met, the functional dependency violates BCNF and must be separated into its own relation. The determinant of the dependency becomes the primary key, and the dependents are the other attributes. The dependent attributes are removed from the original relation. The functional dependency is transferred to the new relation and removed from the original relation.

Fourth Normal Form

Fourth Normal Form requires the removal of multi-valued dependencies into their own relations. If a multi-valued dependency (identified by the user) is identified, attributes must be separated into their own tables such that the determinant no longer multi-determines.

For each relation in the table with 3 or more attributes, the normalizer displays the schema and existing functional dependencies before asking them to input any additional multi-valued dependencies. If the relation contains multi-valued dependencies after this step, the normalizer prompts the user to enter the table's data and uses the data to validate the multi-valued dependencies before storing it in the relation.

The normalizer then asks the user which dependent set they would like to separate, and then separates that dependency. This continues until the relation no longer contains any multi-valued dependencies.

Fifth Normal Form

Fifth Normal Form breaks large tables down even further by identifying join dependencies and separating based on them.

For each relation with 3 or more attributes, the normalizer asks the user to enter table data (if it is not already known from the 4NF step). Then, the normalizer attempts every possible combination of splitting the relation into two new relations, checking for valid join dependencies. When a valid join dependency is found, the relations are split.