Normalization

by Sudheer Sharma - Wednesday, January 21, 2009

http://dwhnotes.com/data-base/normalization

What do you think of this post?

Awesome (2) Interesting (2) Useful (2)

Normalization is the process of efficiently organizing data in the database. Normalized data must be in "relational" data structure.

Why "Normalize" ?

- Flexibility – Supports many ways to look at the data
- Data Integrity – Modification Anomalies (Insert, Delete, Update)
- Efficiency – Eliminate redundant data and save space

When Not to Normalize

- Want to keep tables simple so user can make his own queries

- Avoid processing multiple tables

- Archiving records

- If no need to perform complex queries

- Flatten and store in one or more tables

Normal Forms

Database community has developed a series of steps for ensuring that databases are normalized. Let's explore the normal forms.

In practical applications, you'll often see 1NF, 2NF, 3NF, and 4NF. 5NF is very rarely seen.

Rules of Data Normalization

1NF	**Eliminate Repeating Groups** - Make a separate table for each set of related attributes, and give each table a primary key.
2NF	**Eliminate Redundant Data** - If an attribute depends on only part of a multi-valued key, remove it to a separate table.
3NF	**Eliminate Columns Not Dependent On Key** - If attributes do not contribute to a description of the key, remove them to a separate table.
BCNF	**Boyce-Codd Normal Form** - If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.
4NF	**Isolate Independent Multiple Relationships** - No table may contain two or more 1:n or n:m relationships that are not directly related.
5NF	**Isolate Semantically Related Multiple Relationships** - There may be practical constrains on information that justify separating logically related many-to-many relationships.
ONF	**Optimal Normal Form** - a model limited to only simple (elemental) facts, as expressed in Object Role Model notation.
DKNF	**Domain-Key Normal Form** - a model free from all modification anomalies.

First Normal Form (1NF): Eliminating repeating groups

- Eliminate duplicative columns from the same table
- Create separate tables for each group of related data and identify each row with a unique column or set of columns (primary key)

Second Normal Form (2NF): Eliminate redundant data

- Meet all the requirements of First Normal Form
- Remove subsets of data that apply to multiple rows of a table and place then in a separate tables
- Create relationships between these new tables and their predecessors thru the use of foriegn keys

Third Normal Form (3NF): Eliminate column not dependant on Key

- Meet all the requirements of Second Normal Form
- Remove column that are not dependant upon primary key.

Boyce-Codd Normal Form (BCNF): If there are non-trivial dependancies between candidate key attributes, seperate them out in distinct tables

A relation R is said to be in BCNF if whenever X -> A holds in R, and A is not in X, then X is a candidate key for R.

BCNF covers very specific situations where 3NF misses inter-dependencies between non-key (but candidate key) attributes.

Typically, any relation that is in 3NF is also in BCNF. However, a 3NF relation won't be in BCNF if (a) there are multiple candidate keys, (b) the keys are composed of multiple attributes, and (c) there are common attributes between the keys.

Fourth Normal Form (4NF): Isolate independant multiple relationships

- Meet all the requirements of Third Normal Form
- A relation is in 4NF it has no multi-valued dependencies.

Fifth Normal Form (5NF): Isolate symentaically related multiple relationships

OK, now let's modify the original business diagram and add a link between the books and the software tools, indicating which books deal with which software tools, as shown below.

Initial business request

This makes sense after the discussion on Rule 4, and again we may be tempted to resolve the multiple M:M relationships into a single association, which would now violate 5th normal form. The ternary association looks identical to the one shown in the 4th normal form example, and is also going to have trouble displaying the information correctly. This time we would have even more trouble because we can't show the relationships between books and software unless we have a member to link to, or we have to add our favorite dummy member record to allow the record in the association table.

Incorrect solution

The solution, as before, is to ensure that all M:M relationships that are independent are resolved independently, resulting in the model shown below. Now information about members and books, members and software, and books and software are all stored independently, even though they are all very much semantically related. It is very tempting in many situations to combine the multiple M:M relationships because they are so similar. Within complex business discussions, the lines can become blurred and the correct solution not so obvious.

Correct 5th normal form

Optimal Normal Form (ONF): A model limited to only simple (elemental) facts, as expressed in object role model notaion.

At this point, we have done all we can with Entity-Relationship Diagrams (ERD). Most people will stop here because this is usually pretty good. However, another modeling style called Object Role Modeling (ORM) can display relationships that cannot be expressed in ERD. Therefore there are more normal forms beyond 5th. With Optimal Normal Form (OMF)

It is defined as a model limited to only simple (elemental) facts, as expressed in ORM.

Domain-Key Normal Form (DKNF):

This level of normalization is simply a model taken to the point where there are no opportunities for modification anomalies.

1. "If every constraint on the relation is a logical consequence of the definition of keys and domains"
2. Constraint "a rule governing static values of attributes"
3. Key "unique identifier of a tuple"
4. Domain "description of an attribute's allowed values"

- A relation in DK/NF has no modification anomalies, and conversely.
- DK/NF is the ultimate normal form; there is no higher normal form related to modification anomalies
- Defn: A relation is in DK/NF if every constraint on the relation is a logical consequence of the definition of keys and domains.
- Constraint is any rule governing static values of attributes that is precise enough to be ascertained whether or not it is true
- E.g. edit rules, intra-relation and inter-relation constraints, functional and multi-valued dependencies.
- Not including constraints on changes in data values or time-dependent constraints.
- Key – the unique identifier of a tuple.
- Domain: physical and a logical description of an attributes allowed values.
- Physical description is the format of an attribute.
- Logical description is a further restriction of the values the domain is allowed
- Logical consequence: find a constraint on keys and/or domains which, if it is enforced, means that the desired constraint is also enforced.
- Bottom line on DK/NF: If every table has a single theme, then all functional dependencies will be logical consequences of keys. All data value constraints can them be expressed as domain constraints.
- Practical consequence: Since keys are enforced by the DBMS and domains are enforced by edit checks on data input, all modification anomalies can be avoided by just these two simple measures.

What do you think of this post?

PDF generated by Kalin's PDF Creation Station