Boyce-Codd Normal Form

Third Normal Form is defined with reference to a selected primary key. Functional dependencies may be broken even though a schema is maintained in 3NF. For example:

Sample_Schema

A	*B*	*C*	*D*	*X*	*Y*
77	smith	alf	pat	megan	371
65	smith	jim	eva	fred	83
47	smith	jim	ada	bob	197

Here we assume that the primary key attributes are $\{A, B\}$ and that the attribute set $\{B, C, D\}$ is also a candidate key. It is possible for there to be a FD $\{B, C\} \rightarrow X$ without breaking 3NF, since this dependency is on the values of a pair of {key, non-key} attributes. (NO, I've not come up with a convincing story to account for these dependencies)

How can we fix this gap, and hopefully establish FD-based criteria which do not depend on arbitrary choice of key?

Definition

Let be ***R*** a relation defined over attributes { A_i | $1 \le i \le n$ }.
A proper subset of $k < n$ attributes forms a ***determinant*** if some other attribute of ***R*** , *C* say, is functionally dependent on the values taken by these k attributes.

Boyce-Codd Normal Form

A database schema (in the Relational Model of Data) is in ***Boyce-Codd Normal Form*** (***BCNF***) if ***in every relation of the database, every determinant is a key***.

Intuition behind *BCNF*

IF some set of k attributes determines ***one*** other attribute

THEN the set determines ***every*** other attribute

HENCE any set of k values determines a ***unique tuple***

If this condition holds for every determinant, then there can be no breach of ***any*** functional dependency, since any values associated with those attributes will arise in a unique tuple.

Hence certainly ***BCNF*** implies ***3NF*** .

So far we have considered functional (many-1) dependencies only when looking at breaches of natural semantics. Are there similar considerations for many-many relationships ?

4th and 5th Normal Forms

We have so far only considered functional dependencies: the values taken by a set of $k < n$ attributes determines a unique value of some other attribute. ***BCNF*** requires that the ***determinant*** is a ***key***. If we break ***BCNF*** we run the risk of storing a determined value in more than one place.

4th (and 5th) Normal Forms are related to ***multi-valued*** dependencies, another way in which data can be stored redundantly. If we (equi-)join two ***many−many*** relations

$$R \subset X \times Y \quad \text{and} \quad S \subset X \times Z$$

on join attributes X , then the result $R * S$ may well be in ***BCNF***, but still suffer from update/insertion anomalies.

Example

QANTAS Airways run a fleet of Boeing 747s. Individual aircraft have been purchased over a number of years, and differ in payload, seating capacity and range: thus each aircraft flies only some of the ***QANTAS*** routes, being flown by particular crews. Spare parts are held at major airports visited by ***QANTAS*** aircraft, but only those required for the models that actually fly the relevant routes.

Jumbo_Fleet

aircraft	crew captain	spares depot
City of Brisbane	Capt. Thomas	Auckland
City of Brisbane	Capt. Thomas	Tullamarine
City of Brisbane	Capt. West	Auckland
City of Brisbane	Capt. West	Tullamarine
City of Melbourne	Capt. West	Amsterdam
City of Melbourne	Capt. West	Singapore
City of Melbourne	Capt. West	Tullamarine
City of Swan Hill	Capt. Smith	Auckland
City of Swan Hill	Capt. Smith	Faaa
City of Swan Hill	Capt. Smith	Tullamarine
City of Swan Hill	Capt. Thomas	Auckland
City of Swan Hill	Capt. Thomas	Faaa
City of Swan Hill	Capt. Thomas	Tullamarine

This relation is ***all key***, and is therefore in ***BCNF***. On the other hand, it is evident that data is stored redundantly. Indeed, it appears that for each aircraft there is a set of crews who have trained on aircraft of that type, and that spares for each type of aircraft are held at specific depots.

Multi-valued Dependencies (*MVD*)

Given a relation ***R*** with sets of attributes ***X*** , ***Y*** and ***Z*** , the ***multi-valued dependency*** ***X*** $\rightarrow\rightarrow$ ***Y*** holds in ***R*** if and only if the set of ***Y***-values occurring for given values of attributes in {***X***, ***Z***} is independent of the values from the set ***Z*** .

4th Normal Form (*4NF*)

A relation ***R*** is in 4th Normal Form if and only if, whenever there is a ***MVD*** in ***R***, ***say*** ***X*** $\rightarrow\rightarrow$ ***Y***, then all attributes of ***R*** are functionally dependent on ***X***.

An equivalent definition of *4NF*

A relation ***R*** is in 4th Normal Form

if ***R*** is in ***BCNF***

and all ***MVDs*** in ***R*** are in fact ***FDs***.

What it means for a relation to be in 4th Normal Form is explained well in the book by Ullman and Widom.

Note that the expression ***multi-valued dependency*** explains what is really going on. If we could regard the attributes spares_depots and qualified_crews as ***set-valued*** , then they would indeed be ***FDs***. That would take us outside ***1NF*** ; we require the power of ***NF2*** DBMS, see section 5.

A presentation of the data in *4NF*

Crews_for_aircraft

aircraft	crew captain
City of Brisbane	Capt. Thomas
City of Brisbane	Capt. West
City of Melbourne	Capt. West
City of Swan Hill	Capt. Smith
City of Swan Hill	Capt. Thomas

Spares_for_aircraft

aircraft	spares depot
City of Brisbane	Auckland
City of Brisbane	Tullamarine
City of Melbourne	Amsterdam
City of Melbourne	Singapore
City of Melbourne	Tullamarine
City of Swan Hill	Auckland
City of Swan Hill	Faaa
City of Swan Hill	Tullamarine

These relations are ***all key***, and therefore in ***BCNF***.

A presentation of the data as an *NF2* schema

Crews_for_aircraft

aircraft	qualified crews
	captain
City of Brisbane	{Thomas, West}
City of Melbourne	{West}
City of Swan Hill	{Smith, Thomas}

Spares_for_aircraft

aircraft	spares depots
	airfield
City of Brisbane	{Auckland, Tullamarine}
City of Melbourne	{Amsterdam, Singapore, Tullamarine}
City of Swan Hill	{Auckland, Faaa, Tullamarine}

Nested relations for displaying the same information.

Algorithms for establishing *3NF* and *4NF*

Suppose given a database application for which a relational schema is required. First, represent the information that is to be recorded, using your favourite formalism, such as an ***entity-attribute-relationship*** diagram. Next, write down all the attributes that are to be recorded in the database. We have seen a number of different criteria that will help to eliminate redundancy and guarantee semantic integrity.

Bernstein's algorithm will establish a relational presentation in ***3NF*** , given the set of attributes and a specification of the ***functional dependencies*** $X \rightarrow Y$ that hold between subsets. The algorithm runs in ***polynomial time***, and it is often used when there is a need to design a complex schema.

The situation is rather different where ***4NF*** is concerned. In the same way it is possible to specify all the ***multi-valued dependencies*** $X \rightarrow\rightarrow Y$ that hold between subsets of the attributes. Unfortunately the best algorithm that has been developed (the ***Chase***) is a search procedure, and it is not surprising that it has been shown to be ***NP-complete***.

Join dependencies and *5NF*

The example that we gave of a relation that breached ***4NF*** was constructed by joining two ***all-key*** relations. We took two relations having attribute sets {***X***, ***Y***} and {***X***, ***Z***} respectively, in each case forming the key of the relation. The result of performing ***equi-join*** on attributes ***X*** was a relation ***R*** with attributes {***X***, ***Y***, ***Z***} that breaks ***4NF***. Suppose now that ***A*** = ***X*** ∪ ***Y*** and ***B*** = ***X*** ∪ ***Z*** . Then ***R*** is the ***equi-join*** of its projection on ***A*** with its projection on ***B***.

Definition of join dependency (*JD*)

Relation ***R*** satisfies the ***join dependency*** * (***X***, ***Y***, . . . ***Z***) if and only if ***R*** is equal to the ***equi-join*** of its projections on the given attribute subsets ***X***, ***Y***, . . . ***Z*** of ***R*** , ***in all possible populations of the database***.

Definition of *5NF* (*Projection-Join / NF*)

Relation ***R*** is in ***5NF*** if, whenever ***R*** satisfies some ***join dependency*** * (***X***, ***Y***, . . . ***Z***) , each set of projection attributes ***X***, ***Y***, . . . ***Z*** etc. contains a candidate key.

4NF is the special case in which the definition of join dependency is restricted to two sets of projection attributes.

Are Normal Forms a good thing ?

NOT obvious !

1. Decomposition may lead to poor performance

   get the semantics right first, then tune the performance by caching or whatever

   self-maintaining materialised views ?

2. Automated decomposition may generate unnatural database designs

   large-scale design will require tools, but the schema generated may need tuning

3. Decomposition may break referential integrity

   use the ***FOREIGN KEY*** directive of ***SQL*** to enforce the links via the schema