Devere Anthony Weaver
COP4710 - Database Systems
Chapter 3 ER Textbook Notes

3.1 - Intro

This chapter delves deeper into Edgar Codd's concept of a relational database. The main idea here is that one should care that data is in a suitable form and not actually worry about the manner in which records are physically linked. To do so, data must be normalized which is achieved via the notion of *functional dependencies.*

3.2 - Fundamental Relational Database

The two-dimensional table

- data is arranged into rows and columns with one piece of data in each cell
- each cell in a column has the same data type and same semantics
- pretty much what you're used to seeing at this point

atomic, atomicity - the cell contains one fact and only one fact

- more than one fact in a single data cell is said to contain a *repeating group* and thus is nonatomic

table - two-dimensional arrangement of data cells that contains only atomic data

first normal form (1NF) - Codd, if you have a table, then the data is in the first normal form

relations - populated tables

attributes - columns in a table

Checkpoint 3.1

1. Is the following arrangement of data a table?

Ans: No, it is not. The third column of the second row contains a repeating group. Thus, it is not a table.

2. Is the following arrangement of data a table?

Ans: No, it is not. The second row has mixed up the columns therefore it doesn't adhere to the definition of a table.

3. Is the following arrangement of data a table?

Ans: Yes, this is a table.

4. Is the following arrangement of data a table?

Ans: No, it is not a table. The second record, third column contains an inappropriate value for color attribute.

5. Is the following arrangement of data a table?

Ans: No, it is not a table. The second record, second column has an inappropriate value for the Make attribute.

6. What does R(A,B,C,D,E,F) mean?

Ans: This means that the attributes A,B,C,D,E, and F all relate to each other. In database terms, this means they all exist as a table (1NF) since they are all atomic.

3.3 - Relational Database and Sets

Codd viewed a relational database as containing sets of rows. This is important since sets in the mathematical sense are unordered and don't contain duplicate values.

Applied to rows in a database, there are now duplicate rows in a database (duplicate records), and there is no order among the rows (they aren't presumed to be sorted in any way).

Any duplicate rows in a relational database violate the notion of a set and thus is an invalid relational database. It also violates the sense of being able to identify one and only one row from a primary key.

3.4 Functional Dependency

functional dependency (FD) - a relationship of one attribute in a relation to another

- one field in a record has a relationship with another field in the same record (old terminology)
- in a database it is sometimes the case that one attribute may *define* another
- in the case when one attribute is *defined* or *identified* by another attribute, we say that attribute is *functionally dependent* on the defining attribute

The idea of a FD is to define one field as an anchor from which one can always find a single value for another field

- this is the idea of a *primary key,* we want to find a primary key such that all data in a record depend on the primary key alone (it acts as a unique identifier)

A FD is obtained from the user by the designer when building a database

The notation for a FD is of the form:

- X -> Y
- "X defines Y" or "X implies Y"

Checkpoint 3.2

1. Does all the data conform to EmpNo -> Name?

Ans: Yes, the EmpNo is unique to each record despite the name.

2. Does the fact that the data conforms to the proposed FD prove that the FD is in fact true?

Ans: It does not prove the FD is true; however, this is how the FD is defined by the user.

3. Does all the data conform to EmpNo -> Name?

Ans: I would say that EmpNo is acting as a unique identifier; however, in the second record there is a repeating group. So, I'm not entirely sure if this would still be a FD since technically this isn't a table.

4. Does all the data conform to EmpNo -> Name?

Ans: Yes. Each record has a unique identifier to be used as a primary key. Name is functionally dependent on EmpNo.

NOTE: Find a resource to check the highlighted answers.

3.5 - Non-1NF to 1NF

When dealing with data that is non-1NF, one technique is to create separate tables in order to get the data in 1NF. To do so, first select a primary key that will uniquely identify the rows. Then, create separate tables in which the first table is in 1NF containing the primary key along with the atomic data. Then the next table must contain the primary key along with the values for the repeating groups in the original table until all tables are in 1NF.

3.6 - The Second Normal Form

Tables may be valid, but if no FD are defined with the table, the only FD that can be assumed are reflexive ones which are always true and correct in the mathematical sense; however, they aren't always particularly meaningful.

FD *must* be defined with a table. If they are defined, they will always appear on the left hand side of a FD, also called a primary key.

- in a file description, primary keys are often placed first even if they aren't the first column in the table

Two or more attributes used as keys results in a key that is *concatenated.*

A symptom of a design problem is when redundancy appears

- in the book example, we needed to update data in multiple cells when we made a simple change to one of the taxi cabs
- this is a bad design if we are dealing with tables with thousands or millions or row, we will need to go in and update the necessary data (time-consuming and error-prone)

Anomalies

Anomalies occur in three forms for relational databases:

- *update anomaly* - an update is not a simple update to the table but ends up requiring multiple updates due to data redundancy
- *insert anomaly* - occurs when there are issues inserting data into the table because of constraints
    - ex: attempting to insert a record where the primary key is NULL, by the *entity integrity constraint*, any part of a primary key can't be NULL (createse contradiction)
- *delete anomaly* - causes data to be deleted beyond that which was intended

Non-2NF to 2NF

partial dependency - occurs when you have a concatenated key and you have an attribute in the table that is dependent on only part of the key

- tables with partial dependencies are NOT in second normal form

second normal form (2NF) - for a table to be 2NF, it has to have a primary key with no partial dependencies

When dealing with a table that is non-2NF, it should be decomposed into two tables, each containing data dependent only on the primary key.

- break down the relation into two different relations until all nonkey attributes depend only on the primary key

When this decomposition is implemented correctly, an update to one value should only occur in one location. This prevents redundancy and hence update anomalies.

- doing this can also eliminate delete and insertion anomalies since deleting or inserting a record will not disturb the other tables and accidentally delete extra information or duplicate information when inserting

Checkpoint 3.4

1. Given AB -> CDE, is R(A,B,C,D,E) in 2NF?

Ans: Yes, this relation is in 2NF. All of the non-key attributes depend entirely on the concatenated primary key.

2. Given B -> ACDE, is R(A, B, C, D, E) in 2NF?

Ans: Yes, this relation is in 2NF. It is in 1NF, it has a primary key, and there are no partial dependencies.

3. Given AB -> CD and B -> E, is R(A,B,C,D,E) in 2NF?

Ans: No, this relation is not 2NF. There exist partial dependencies.

4. False. It is possible that some of the non-key attributes don't depend on the primary key, despite it being in 1NF.

3.7 - The Third Normal Form

transitive dependency - think back to the properties of relations from Set Theory and Mathematical Logic

- can cause redundancy which in turn can cause anomalies (update, insert, delete)

The third normal form demands there exists *no transitive dependencies* as these transitive dependencies cause anomalies

- to fix these dependencies, decompose the 3NF table into tables that depend only on the key of the table
- the issue is when transitive dependencies exists in a *single table*

3.8 - The Equijoin Operation

While tables are decomposed, they can easily be reconstructed with the equijoin operation of relational calculus which is implemented in SQL

- so while a normalized table may have been decomposed, the original table can be re-formed from the decomposed ones

A decomposed table can be reconstructed by joining two or more tables by their common attribute.

** This is only touched upon in this text, be sure to use the practical text/other resources to get a further understanding of this. **

3.9 - Some Functional Dependency Rules

A general process of bringing data to 3NF is

- find the *minimal key*
- determine if the relation is in 3NF
    - decompose until it is

Some rules:

1. reflexive
2. augmentation - if A->B, the AC -> B
    a. adding information to the LHS doesn't enhance the FD, it is superfluous as long as the RHS stays the same
3. decomposition - if A->BCD, then A->B, A->C, and A->D
4. union - if A->B and A->C, the A->BC (the inverse of the decomposition rule)
5. transitive - if A->B and B->C, then A->C
6. subset

** Look more into this, I’m sure it will come up later in the course though. **

3.10 - The Boyce Codd Normal Form

The suggestion is that all tables be in 3NF, when they are, there is likely to be no redundancy, no anomalies, BUT there may be cases where 3NF is not good enough.