FUNCTIONAL DEPENDENCIES & NORMALIZATION

Reference

- Elmasri & Navathe – (Chapter 15)

The Big Picture

1. In certain cases, a schema can allow for data to be duplicated **redundantly** in the database.
2. In order to fix the schema, we need to analyze it using **functional dependencies** (a form of business rules; i.e. ICs) as a tool.
3. Analysis of the schema will reveal whether it is in **1NF, 2NF, 3NF, or BCNF** (whereby these normal forms are successively better in that they permit less and less redundancy.)
4. In order to fix a schema, we **decompose** it.
5. The ***decomposition*** must, at the least, be **lossless**.

Now we'll cover the terminology and concepts mentioned above.

1. Update Anomalies … they are caused by Redundancy

Example: **STUDENTS (<u>sNum</u>, sName, sAddress, fNum, fName, fOffice)**

- **Insertion anomalies**
  - We can't add a faculty who doesn't have any advisees.
- **Deletion anomalies**
  - We can't keep the faculty information if their last advisee is deleted.
- **Modification anomalies:**
  - If a faculty who has, say 20, advisees changes office then 20 rows have to be updated. Problems that might arise then are:
    - Possible DB consistency.
    - Inefficiency.

- The above anomalies are (collectively) called **update anomalies**.
- The purpose of normalization is to identify and remove (or control) redundancy.

2. Functional dependency

Given a relation R having two sets of attributes X and Y, we say that X **functionally determines** Y, written as: $\boldsymbol{X \rightarrow Y}$, if (*in every legal state of R*) it is the case that whenever two tuples have the same value for X, they also have the same value for Y.

- Intuitively, X acts a key for Y (*though it doesn't have to be minimal*).
- Notice that every candidate key, K, determines every other attribute.
- An FD is a semantic constraint of the application, and not of a particular database state.

Example:

Suppose that the Boats table (in our SAILORS DB) has another attribute called logKeeperRating. In that case, a functional dependency would be:

bid → logkeeperRating

Discussion:

In our database, is this FD true? **bname → logkeeperRating**

Example

Advising (Gnum, Name, Address, advisorNum, advisorName, OfficeNum)

Suppose that an application rule says: *An advisor can have no more than one office or name*.

Then the FDs are:

advisorNum → advisorName, OfficeNum
GNum → Name, Address, advisorNum, advisorName, OfficeNum *(we don't explicitly write this FD since the key determines every attribute.)*

Example

(Suppose that in a database only the student's latest grade in any repeated course is to be recorded.)

Grades (sNum, courseNum, grade, address)
Suppose that an application rule says: *A student has at most one address*

Then an FD is: sNum → address

Discussion: What is the difference in meaning between?

sNum → address   vs.   address → sNum

3. The different types of dependencies and normal forms

- The types of FDs that might exist in a relation:

R (<u>A, B</u>, C, D, E, F)

1. AB → CDEF … necessary … AB is a key.
2. B → C … problematic … partial dependency.
3. E → F … problematic … transitive dependency.
4. E → B … problematic … BC (Boyce-Codd) dependency.

- Example of a <u>partial</u> dependency: *Refer to the Sailors database*
  – What is the problem if RESERVATIONS was designed as:

  RESERVATIONS (<u>bID, forDate</u>, sID, onDate, *boatRate*)

- Example of a <u>transitive</u> dependency: *Refer to the Sailors database*
  – What is the problem if Boats was designed as:

  BOATS (<u>bID</u>, bName, color, rate, length. logKeeper, *logKeeperRating*)

Several Normal Norms can be defined depending upon the non-key FDs that they ***<u>disallow</u>***.
BCNF is usually sufficient in most practical situations.

A Summary Table of Normal Forms

If a relation has a key then it is, at least in1NF. It can also be in higher normal forms if it doesn't have non-key dependencies as per the following table.

	Has partial Dependency	Has Transitive Dependency	Has BC Dependency
2NF	No	-	-
3NF	No	No	-
BCNF	No	No	No

Conclusion

For a relation to be in BCNF, every attribute must be dependent upon the key, the whole key, and nothing but the key (*for every candidate key*.)

Class Exercises

Determine the ***highest*** normal form (1NF, 2NF, 3NF, or BCNF) for each one of the following relations. (Notice that the primary key of each relation is underlined.)

	Relations	Answers
1	**R (<u>A, B</u>, C, D, E)** … *Given the functional dependencies:* **A → E** *and* **C → B**	
2	**R (<u>A, B</u>, C, D, E)** … *Given the functional dependencies:* **D → E** *and* **C → B**	
3	**R (<u>A, B</u>, C, D, E)** … *Given the functional dependency:* **E → B**	
4	**R (<u>A, B</u>, C, D, E)**	
5	**Work (<u>EmployeeNumber</u>, ProjNumber, projLocation)** *Given that a project is located in one location only.*	
6	**Work (<u>EmployeeNumber</u>, ProjNumber, projLocation)** *Given that each location has one project only.*	
7	**Work (<u>EmployeeNumber, ProjNumber</u>, projLocation)** *Given that each location has one project only.*	

4. Normalization

- Normalization involves decomposing a relation into two or more relations such that each resulting relations has no no-key FDs.

- Two properties of decomposition are:
  - **Losslessness** … *it is a matter of correctness, and must be met*
  - **Dependency-preservation** … *it is a matter of efficiency. We'll not discuss it further.*

Lossless Decompositions

Motivation

The natural join of the relations that result from the decomposition must reconstruct the original relation precisely.

Example of a non-lossless decomposition

A relation and a possible decomposition into two relations:

T

S#	C#		Sec#	F#
11	CS1		01	21
10	CS1		02	23
11	CS8		01	21

⇒

T1

S#	C#
11	CS1
10	CS1
11	CS8

+

T2

C#	Sec#	F#
CS1	01	21
CS1	02	23
CS8	01	21

But the natural join (on C#) of the resulting two relations as per:

```
Select * From T1, T2, WHERE T1.C# = T2.C#;
```

produces spurious tuples.

S#	C#	Sec#	F#	
11	CS1	01	21	
11	CS1	02	23	***Spurious tuple*** … *It is not in the original relation.*
10	CS1	01	21	***Spurious tuple*** … *It is not in the original relation.*
10	CS1	02	23	
11	CS8	01	21	

How to test whether a decomposition is lossless

A sufficient (*though not necessary*) condition is:

The set of attributes that is common to both relations must contain a key of at least one of the two relations.

5. Examples of normalization

Generally:

1. <u>If the relation doesn't have BC dependencies</u> (examples1, 2, and 3), then create one relation for every FD. Choose, as the key for the created relation, the left-hand side of the FD <u>(called the **determinant**)</u>; and link the resulting relations through foreign keys if necessary.

2. <u>If the relation does have BC dependencies</u> (example 4), then apply Armstrong's pseudo transitive axiom first; and then decompose using guideline-1 above.

<u>Example-1</u>

RECORDS_1 (<u>S#, C#, sem, year</u>, F#, Sname, Cname, grade, crHrs)

Given the following FDs: S# → Sname
C# → Cname, crHrs
C#, sem, year → F#

Is **RECORDS_1** in: 1NF? _____ 2NF? _____ 3NF? _____ BCNF? _____
Identify some problems:
Decomposition into BCNF:

<u>Example-2</u>

RECORDS_2 (<u>S#, C#, sem, year</u>, F#, Sname, Cname, crHrs)

Given the following FDs: S# → Sname
C# → Cname, crHrs
C#, sem, year → F#

Is **RECORDS_2** in: 1NF? _____ 2NF? _____ 3NF? _____ BCNF? _____
Identify some problems:
Decomposition into BCNF:

<u>Example-3</u>

TEACHES (<u>C#, sem, year</u>, F#, Fname, Foffice)

Given the following FD: F# $\rightarrow$ Fname, Foffice

Is TEACHES in: 1NF? _____ 2NF? _____ 3NF? _____ BCNF? _____

Identify some problems:
Decomposition into BCNF:

Example-4 … *this one has a BC dependency; so we'll need to apply Armstrong's pseudo-transitive axiom first.*

ADVISES (S#, Major, Advisor, MajorGPA) … *an example from McFadden*
Given the following FD: **Advisor → Major**
(i.e. an advisor advises, at most, in one major; and a major may have several advisors.)

Is ADVISES in: 1NF? _____ 2NF? _____ 3NF? _____ BCNF? _____

Question: What is the problem if an advisor in a major changes?

Question: What is the problem if a new major is set up but has no students yet?

Conclusion: 3NF can still have pmalies.

- Solution? … Seek BCNF.
- But first … a little detour

A detour: Consider the following ***incorrect*** decomposition that uses the simple procedure we used in examples 1, 2, and 3 above.

ADVISOR (Advisor, Major)
STUDENT (S#, Major, MajorGPA)

What is its highest normal form for ADVISOR? ______

What is its highest normal form for STUDENT? ______

Is the decomposition lossless? ________

Solution (using a 2-step process):

First, apply Armstrong's pseudo transitive rule $\boxed{\textbf{\textit{If } } X \rightarrow Y \textbf{\textit{ and }} WY \rightarrow Z \textbf{\textit{ then }} WX \rightarrow Z}$

-

Second, decompose the reformulated relation

-
-

Are the resulting relations in BCNF? ____________________
Is the decomposition lossless? ____________________

5. Normalization and De-normalization

- In an actual design we may **denormalize** some relations after they have been normalized.
    - Denormalization introduces **controlled redundancy**
    - Denormalization is motivated by performance issues.
    - Discuss: Controlled Vs. Uncontrolled Redundancy.