UNIT V

Functional dependency

In a given table, an attribute *Y* is said to have a functional dependency on a set of attributes *X* (written $X \rightarrow Y$) if and only if each *X* value is associated with precisely one *Y* value.

For example, in an "Employee" table that includes the attributes "Employee ID" and "Employee Date of Birth", the functional dependency {Employee ID} → {Employee Date of Birth} would hold. It follows from the previous two sentences that each {Employee ID} is associated with precisely one {Employee Date of Birth}.

Full functional dependency

An attribute is fully functionally dependent on a set of attributes X if it is:

- functionally dependent on X, and
- not functionally dependent on any proper subset of X. {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, because it is also dependent on {Employee ID}.Even by the removal of {Skill} functional dependency still holds between {Employee Address} and {Employee ID}.

Transitive dependency

A transitive dependency is an indirect functional dependency, one in which $X \rightarrow Z$ only by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$.

Trivial functional dependency

A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.

Multivalve dependency

A multivalued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows.

Join dependency

A table *T* is subject to a join dependency if *T* can always be recreated by joining multiple tables each having a subset of the attributes of *T*.

Normal form	Defined by	In	Brief definition	Description
1NF	First normal form	Two versions: E.F. Codd (1970), C.J. Date (2003)	1970and 2003	The domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.
2NF	Second normal form	E.F. Codd	1971	No non-prime attribute in the table is functionally dependent on a proper subset of any candidate
3NF	Third normal form	Two versions: E.F.	1971and 1982	Every non-prime attribute is non-transitively dependent on every candidate key in the

Normal form	Defined by	In	Brief definition	Description
		Codd (1971), C. Zaniolo (1982)		table. The attributes that do not contribute to the description of the primary key are removed from the table. In other words, no transitive dependency is allowed.
EKNF	Elementary Key Normal Form	C. Zaniolo	1982	Every non-trivial functional dependency in the table is either the dependency of an elementary key attribute or a dependency on a superkey
BCNF	Boyce–Codd normal form	Raymond F. Boyceand E.F. Codd	1974	Every non-trivial functional dependency in the table is a dependency on a superkey
4NF	Fourth normal form	Ronald Fagin	1977	Every non-trivial multivalued dependency in the table is a dependency on a superkey
5NF	Fifth normal form	Ronald Fagin	1979	Every non-trivial join dependency in the table is implied by the superkeys of the table
DKNF	Domain/key normal form	Ronald Fagin	1981	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints
6NF	Sixth normal form	C.J. Date, Hugh Darwen, and Nikos Lorentzos	2002	Table features no non-trivial join dependencies at all (with reference to generalized join operator)

UNF: A table that contains 1/more repeating gps.

1NF: Each cell must have one value & no repeating gps.

2NF: Every non-primary key Attribute is fully functionaly dependent on primary key. i.e. Remove
partially dependency.

3NF: Dependent on primary key. i.e. Remove transitive dependency.

BCNF: Boyce Codd Normal Form: Every determenent is a candidate key.

4NF---->5NF: Higher Normal Form

Modification Anomalies

Once our E-R model has been converted into relations, we may find that some relations are not properly specified. There can be a number of problems:

Deletion Anomaly: Deleting one fact or data point from a relation results in other information being lost.

Insertion Anomaly: Inserting a new fact or tuple into a relation requires we have information from two or more entities – this situation might not be feasible.

Update Anomaly: Updating one fact in a relation requires us to update multiple tuples.

Here is a quick example to illustrate these anomalies: A company has a Purchase Order form:

Normalization Process

- Relations can fall into one or more categories (or classes) called Normal Forms
- Normal Form: A class of relations free from a certain set of modification anomalies.
- Normal forms are given names such as:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
  - Fourth normal form (4NF)
  - Fifth normal form (5NF)
  - Domain-Key normal form (DK/NF)
- These forms are cumulative. A relation in Third normal form is also in 2NF and 1NF.
- The Normalization Process for a given relation consists of:
  a. Specify the Key of the relation
  b. Specify the functional dependencies of the relation.
     Sample data (tuples) for the relation can assist with this step.
  c. Apply the definition of each normal form (starting with 1NF).
  d. If a relation fails to meet the definition of a normal form, change the relation (most often by splitting the relation into two new relations) until it meets the definition.
  e. Re-test the modified/new relations to ensure they meet the definitions of each normal form.

In the next set of notes, each of the normal forms will be defined along with an example of the normalization steps.

First Normal Form (1NF)

A relation is in first normal form if it meets the definition of a relation:

- Each attribute (column) value must be a single value only.
- All values for a given attribute (column) must be of the same type.
- Each attribute (column) name must be unique.
- The order of attributes (columns) is insignificant
- No two tuples (rows) in a relation can be identical.
- The order of the tuples (rows) is insignificant.
  - If you have a key defined for the relation, then you can meet the unique row requirement.
  - Example relation in 1NF (note that key attributes are underlined):
  - STOCKS (Company, Symbol, Headquarters, Date, Close_Price)

Company	Symbol	Headquarters	Date	Close Price
Microsoft	MSFT	Redmond, WA	09/07/2013	23.96
Microsoft	MSFT	Redmond, WA	09/08/2013	23.93
Microsoft	MSFT	Redmond, WA	09/09/2013	24.01
Oracle	ORCL	Redwood Shores, CA	09/07/2013	24.27
Oracle	ORCL	Redwood Shores, CA	09/08/2013	24.14

Oracle	ORCL	Redwood Shores, CA	09/09/2013	24.33

Second Normal Form (2NF)
A relation is in second normal form (2NF) if all of its non-key attributes are dependent on all of the *key*.
Another way to say this: A relation is in second normal form if it is free from partial-key dependencies
Relations that have a single attribute for a key are automatically in 2NF.
This is one reason why we often use artificial identifiers (non-composite keys) as keys.
In the example below, Close Price is dependent on Company, Date
The following example relation *is not* in 2NF:
STOCKS (Company, Symbol, Headquarters, Date, Close_Price)

Company	Symbol	Headquarters	Date	Close Price
Microsoft	MSFT	Redmond, WA	09/07/2013	23.96
Microsoft	MSFT	Redmond, WA	09/08/2013	23.93
Microsoft	MSFT	Redmond, WA	09/09/2013	24.01
Oracle	ORCL	Redwood Shores, CA	09/07/2013	24.27
Oracle	ORCL	Redwood Shores, CA	09/08/2013	24.14
Oracle	ORCL	Redwood Shores, CA	09/09/2013	24.33

Company	Symbol	Headquarters
Microsoft	MSFT	Redmond, WA
Oracle	ORCL	Redwood Shores, CA

FD1: Symbol → Company, Headquarters

STOCK_PRICES relation:

Symbol	Date	Close Price
MSFT	09/07/2013	23.96
MSFT	09/08/2013	23.93
MSFT	09/09/2013	24.01
ORCL	09/07/2013	24.27
ORCL	09/08/2013	24.14
ORCL	09/09/2013	24.33

FD1: Symbol, Date → Close Price

In checking these new relations we can confirm that they meet the definition of 1NF (each one has well defined unique keys) and 2NF (no partial key dependencies).

Third Normal Form (3NF)
A relation is in third normal form (3NF) if it is in second normal form and it contains no *transitive dependencies*.
Consider relation R containing attributes A, B and C. R(A, B, C)
If A → B and B → C then A → C
Transitive Dependency: Three attributes with the above dependencies.
Example: At CUNY:

Course_Code → Course_Number, Section
Course_Number, Section → Classroom, Professor

Consider one of the new relations we created in the STOCKS example for 2nd normal form:

Company	Symbol	Headquarters
Microsoft	MSFT	Redmond, WA
Oracle	ORCL	Redwood Shores, CA

The functional dependencies we can see are:
FD1: Symbol → Company
FD2: Company → Headquarters
so therefore:
Symbol → Headquarters

This is a transitive dependency.
This gives us the following sample data and FD for the new relations

Company	Symbol
Microsoft	MSFT
Oracle	ORCL

FD1: Symbol → Company

Company	Headquarters
Microsoft	Redmond, WA
Oracle	Redwood Shores, CA

FD1: Company → Headquarters

Again, each of these new relations should be checked to ensure they meet the definition of 1NF, 2NF and now 3NF.
Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF if every determinant is a candidate key.
- Recall that not all determinants are keys.
- Those determinants that are keys we initially call *candidate keys*.
- Eventually, we select a single candidate key to be *the key* for the relation.

- Consider the following example:
  - Funds consist of one or more Investment Types.
  - Funds are managed by one or more Managers
  - Investment Types can have one more Managers
  - Managers only manage one type of investment.
- Relation: FUNDS (FundID, InvestmentType, Manager)

FundID	InvestmentType	Manager
99	Common Stock	Smith
99	Municipal Bonds	Jones
33	Common Stock	Green
22	Growth Stocks	Brown
11	Common Stock	Smith

Fourth Normal Form (4NF)
A relation is in fourth normal form if it is in BCNF and it contains no *multivalued dependencies*.
Multivalued Dependency: A type of functional dependency where the determinant can determine more than one value.
More formally, there are 3 criteria:

1. There must be at least 3 attributes in the relation. call them A, B, and C, for example.
2. Given A, one can determine multiple values of B.
3. Given A, one can determine multiple values of C.
4. B and C are independent of one another.

Book example:
Student has one or more majors.
Student participates in one or more activities.

StudentID	Major	Activities
100	CIS	Baseball
100	CIS	Volleyball
100	Accounting	Baseball
100	Accounting	Volleyball
200	Marketing	Swimming

FD1: StudentID →→ Major
FD2: StudentID →→ Activities

Portfolio ID	Stock Fund	Bond Fund
999	Janus Fund	Municipal Bonds
999	Janus Fund	Dreyfus Short-Intermediate Municipal Bond Fund

999	Scudder Global Fund	Municipal Bonds
999	Scudder Global Fund	Dreyfus Short-Intermediate Municipal Bond Fund
888	Kaufmann Fund	T. Rowe Price Emerging Markets Bond Fund

A few characteristics:

- No regular functional dependencies
- All three attributes taken together form the key.
- Latter two attributes are independent of one another.
- Insertion anomaly: Cannot add a stock fund without adding a bond fund (NULL Value). Must always maintain the combinations to preserve the meaning.
  - Stock Fund and Bond Fund form a multivalued dependency on Portfolio ID.
  - PortfolioID →→ Stock Fund
  - PortfolioID →→ Bond Fund

Resolution: Split into two tables with the common key:

Portfolio ID	Stock Fund
999	Janus Fund
999	Scudder Global Fund
888	Kaufmann Fund

Portfolio ID	Bond Fund
999	Municipal Bonds
999	Dreyfus Short-Intermediate Municipal Bond Fund
888	T. Rowe Price Emerging Markets Bond Fund

Fifth Normal Form (5NF)

Also called "Projection Join" Normal form.

There are certain conditions under which after decomposing a relation, it cannot be reassembled back into its original form.

We don't consider these issues here.

Domain Key Normal Form (DK/NF)

A relation is in DK/NF if every *constraint* on the relation is a logical consequence of the definition of *keys*and *domains*.

Constraint: An rule governing static values of an attribute such that we can determine if this constraint is True or False. Examples:

- Functional Dependencies
- Multivalued Dependencies
- Inter-relation rules

- Intra-relation rules

However: Does *Not* include time dependent constraints.

Key: Unique identifier of a tuple.

Domain: The physical (data type, size, NULL values) and semantic (logical) description of what values an attribute can hold.
There is no known algorithm for converting a relation directly into DK/NF.

What is Normalization?
Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

Summary of the Normal Forms
The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and3NF along with the occasional 4NF. Fifth normal form is very rarely seen and won't be discussed in this article.

Before we begin our discussion of the normal forms, it's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible ramifications they could have on your system and account for possible inconsistencies. That said, let's explore the normal forms.

First Normal Form (1NF)
First normal form (1NF) sets the very basic rules for an organized database:
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 (the primary key).

Second Normal Form (2NF)
Second normal form (2NF) further addresses the concept of removing duplicative data:
Meet all the requirements of the first normal form.
Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
Create relationships between these new tables and their predecessors through the use of foreign keys.

Third Normal Form (3NF)
Third normal form (3NF) goes one large step further:
Meet all the requirements of the second normal form.
Remove columns that are not dependent upon the primary key.

Boyce-Codd Normal Form (BCNF or 3.5NF)
The Boyce-Codd Normal Form, also referred to as the "third and half (3.5) normal form", adds one more requirement:

Meet all the requirements of the third normal form.
Every determinant must be a candidate key.

Fourth Normal Form (4NF)
Finally, fourth normal form (4NF) has one additional requirement:

Meet all the requirements of the third normal form.
A relation is in 4NF if it has no multi-valued dependencies.

Remember, these normalization guidelines are cumulative. For a database to be in 2NF, it must first fulfill all the criteria of a 1NF database.