Database Normalization

Mohua Sarkar, Ph.D

Software Engineer

California Pacific Medical Center

415-600-7003

sarkarm@sutterhealth.org

Definition

- A **database** is an organized collection of data whose content must be *quickly* and *easily*
  - Accessed
  - Managed
  - Updated
- A **relational database** is one whose data are split up into **tables**, sometimes called **relations.**

What Is Database Normalization?

- Cures the ‘SpreadSheet Syndrome’.
- Store only the minimal amount of information.
- Remove redundancies.
- Remove anomalies.
- Restructure data

Concept of normalization and the most common normal forms.

- Originally developed by E.F. Codd in 1970. He then wrote a paper in 1972 on “Further Normalization of the Data Base Relational Model”.
- Normal forms reduce the amount of redundancy and inconsistent dependency within databases.
- Codd proposed three normal forms and through the years two more have been added.
- Normalization organizes the data into tables where each item is a row and the attributes of the item are in columns.

There are two goals of the normalization process:

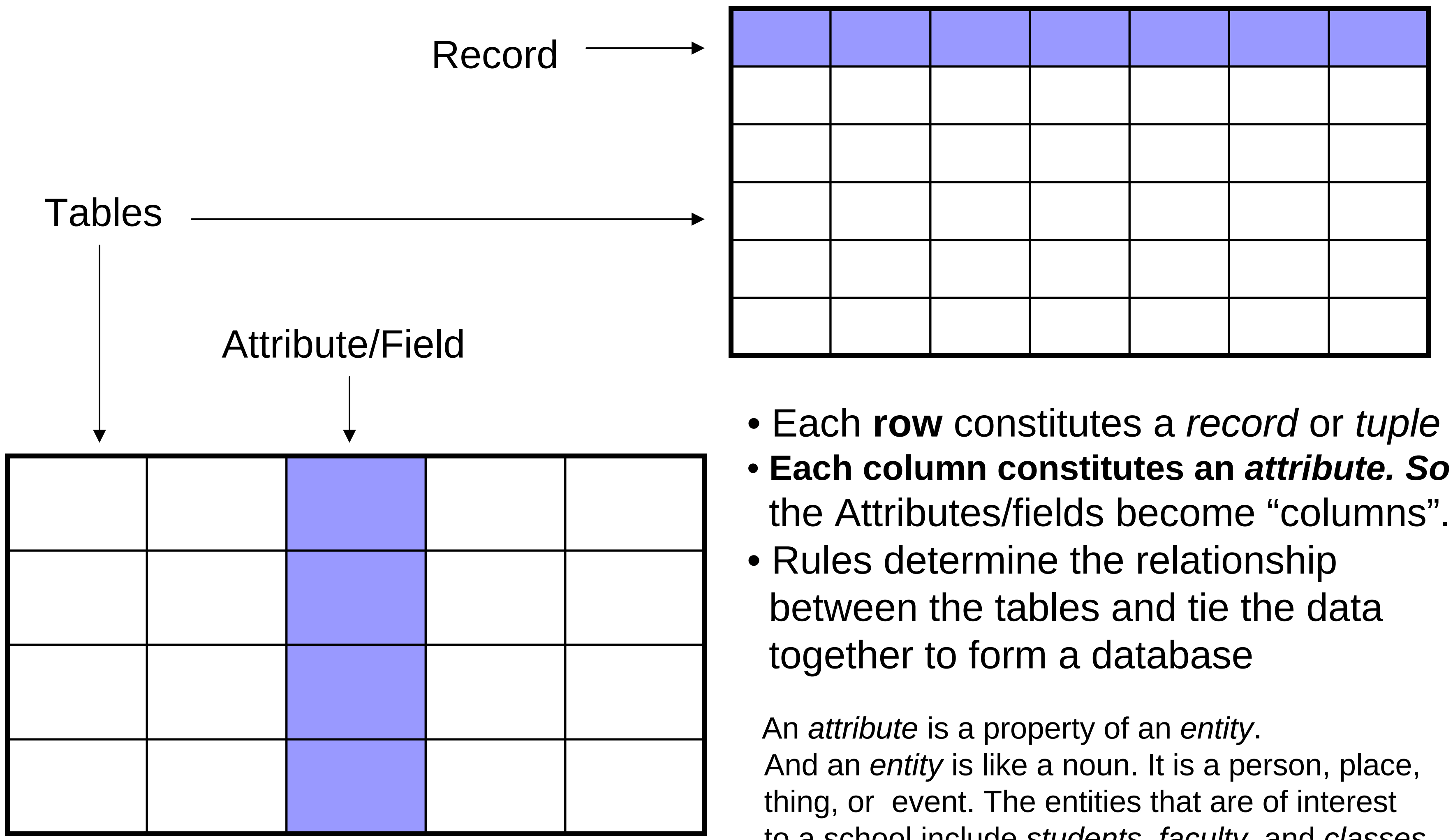
- eliminate redundant data (for example, storing the same data in more than one table) and
- ensure data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

To begin

Lets first, memorize the 3 normal forms so that you can recite them in your sleep. The meaning will become clear as we go.

- No repeating elements or groups of elements
- No partial dependencies on a concatenated key
- No dependencies on non-key attributes

Parts of a database



- Each **row** constitutes a *record* or *tuple*
- **Each column constitutes an *attribute. So*** the Attributes/fields become "columns".
- Rules determine the relationship between the tables and tie the data together to form a database

An *attribute* is a property of an *entity*.
And an *entity* is like a noun. It is a person, place, thing, or event. The entities that are of interest to a school include *students*, *faculty*, and *classes.*

Normalization is the process for assigning attributes to entities.

Description of Normalization

- Thus Normalization is the process of organizing and designing a data model to efficiently store data in a database. The end result is that redundant data is eliminated, and only data related to the attribute is stored within the table.

- Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. For example: A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

There are a few rules for database normalization. Each rule is called a "normal form."

The Normal Forms

- Four most commonly used normal forms are first (1NF), second (2NF) and third (3NF) normal forms, and Boyce–Codd normal form (BCNF).
- **First Form**: *sets the very basic rules for an organized database*
  - Eliminate replicated data in tables
  - Create separate tables for each set of related data
  - Identify each set of related data with a primary key
  - No partial functional dependencies.

The Zero Form

- No rules have been applied
- Where most people start (and stop)
- No room for growth
- Usually wastes space

Contacts

Name	Company	Address	Phone1	Phone2	Phone3	ZipCode
Joe	ABC	123	5532	2234	3211	12345
Jane	XYZ	456	3421			14454
Chris	PDQ	789	2341	6655		14423

First Normal Form

- Eliminate repeating columns in each table
- Create a separate table for each set of related data
- Identify each set of related data with a primary key
- All attributes are single valued & non-repeating

Contacts

Id	Name	Company	Address	Phone	ZipCode
1	Joe	ABC	123	5532	12345
1	Joe	ABC	123	2234	12345
1	Joe	ABC	123	3211	12345
2	Jane	XYZ	456	3421	14454
3	Chris	PDQ	789	2341	14423
3	Chris	PDQ	789	6655	14423

Benefits: Now we can have infinite phone numbers or company addresses for each contact.

Drawback: Now we have to type in everything over and over again. This leads to inconsistency, redundancy and wasting space. Thus, the second normal form…

- **Second Form**
  - Create separate tables for sets of values that apply to multiple records.
  - Relate the tables with a foreign key.
  - Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

Second Normal Form

- Create separate tables for sets of values that apply to multiple records
- Relate these tables with a “foreign key”.
- addresses the concept of removing duplicative data:
- Meet all the requirements of the first normal form.
- Create relationships between these new tables and their pre decessors through the use of foreign keys.
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.

People

Id	Name	Company	Address	Zip
1	Joe	ABC	123	12345
2	Jane	XYZ	456	14454
3	Chris	PDQ	789	14423

PhoneNumbers

PhoneID	Id	Phone
1	1	5532
2	1	2234
3	1	3211
4	2	3421
5	3	2341
6	3	6655

- **Third Form**
  - Eliminate fields that do not depend on the primary key.
  - Each non-primary key attribute must be dependent ***only*** on primary key

Third Normal Form

- Eliminate fields that do not depend on the primary key.

People

Id	Name	AddressID
1	Joe	1
2	Jane	2
3	Chris	3

PhoneNumbers

PhoneID	Id	Phone
1	1	5532
2	1	2234
3	1	3211
4	2	3421
5	3	2341
6	3	6655

Address

AddressID	Company	Address	Zip
1	ABC	123	12345
2	XYZ	456	14454
3	PDQ	789	14423

Is this enough? Codd thought so…
What about “many to many”?

- **Fourth Form** : *also called Boyce Codd Normal Form (BCNF)*
  - In many-to-many relationships, independent entities cannot be stored in the same table.
  - A relation is in 4NF if it has no multi-valued dependencies.
  - 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.

- **Fifth Form**:
  - The “very esoteric” one that is probably not required to get the most out of your database.
  - “The original table must be reconstructed from the tables into which it has been broken down.”
  - The rule ensures that you have not created any extraneous columns and all the tables are only as large as they need to be. The rule do exist, but are rarely considered in practical design.

Disregarding these rules may result in less than perfect database design, but should not affect functionality.

Steps in Normalization

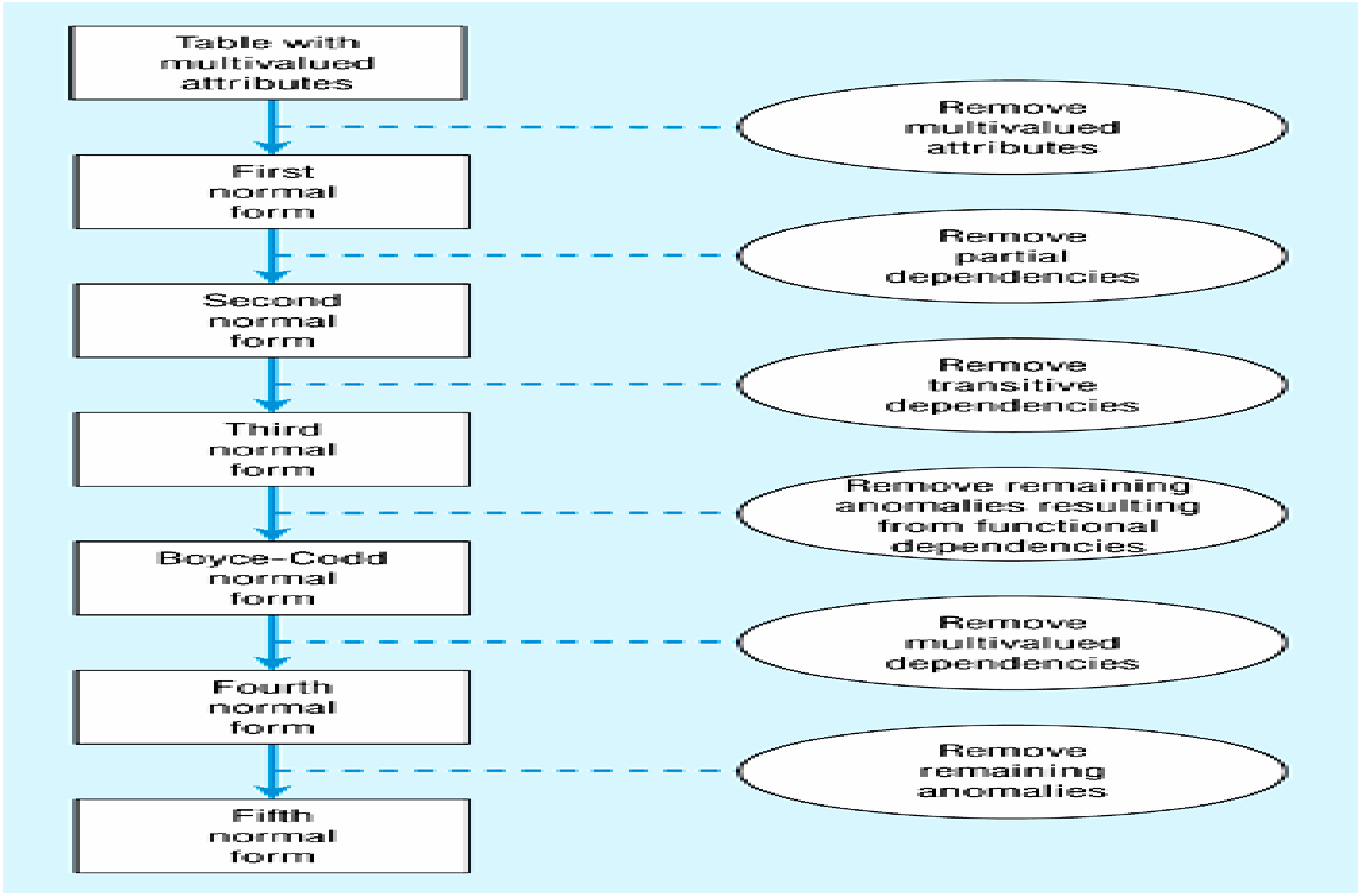


Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form
Remove remaining anomalies resulting from functional dependencies
Boyce-Codd normal form
Remove multivalued dependencies
Fourth normal form
Remove remaining anomalies
Fifth normal form

Why normalize?

- Properly executed, the normalization process eliminates uncontrolled data redundancies, thus eliminating the data anomalies and the data integrity problems that are produced by such redundancies.
- It does not eliminate data redundancy; instead, it produces the carefully *controlled* redundancy that lets us properly link database tables.
- Increases the integrity of the data
- Improves efficiency
- Although normalization can be hard, it is worth it in the long run.

What do I need to remember?

- Keep normalization in mind.
- Don’t replicate data in a table.
- If you break the rules, know why you are breaking the rules and do it for a good reason.

What are the Benefits of Database Normalization?

- Improved data integrity!
  - No INSERT or UPDATE anomalies.
- Decreased storage requirements!

  No redundant data stored. (strings vs. ints * millions of rows)
- Faster search performance!
  - Smaller file for table scans.
  - More directed searching.

THANK YOU