Step 4: Normalization

The process discussed previously to transform E-R diagrams to relation schema often results in well-structured relations that avoid anomalies, but not always.

A **normal form** is a state of a set of relations that requires that certain rules regarding relationships between the attributes of the relations are satisfied.

If the relations satisfy the rules, the relations are said to be in the particular normal form.

Normalization is a process transforming relations into a particular normal form by *successively reducing relations with anomalies* to produce smaller, better-structured relations that satisfy the rules of the normal form.

The goals of normalization are to:

- Minimize data redundancy to avoid anomalies and conserve storage space.
- Simplify the enforcement of referential integrity constraints.
- Make it easier to maintain data (CRUD - create, remove, update, delete rows)
- Provide a better design for future growth.

First Normal Form

Relations are in **first normal form (1NF)** if all multivalued attributes have been removed, so that each attribute in each row has a single value.

Our relations should already be in first normal form.

Second Normal Form

A **functional dependency** is a constraint between two attributes or two sets of attributes. For any relation, attribute B is functionally dependent on attribute A, denoted A -> B, if for every unique instance of A, the value of A determines the value of B.

$R = AXB, A \rightarrow B$

(1, 2)
(1, 2)
(2, 7)
(2, 7)

$R = AXB$, (no functional dependency)

(1, 2)
(1, 2)
(2, 7)
(2, 5)

Below are examples of functional dependencies.

VIN → Make, Model
ISBN → Title, Publisher, FirstAuthor
SSN → FirstName, DOB

The bold attributes in the examples below may or may not be functional dependencies, depending on the rules the organization has established for updating the table.

VIN → Make, Model, **Color**
ISBN → Title, Publisher, FirstAuthor, **Cost**
SSN → FirstName, DOB, **LastName**

The attribute on the left side of the arrow of a functional dependency is called a **determinant**.

A **candidate key** is an attribute or combination of attributes that uniquely identifies a row in a relation. A relation can have multiple candidate keys and we choose our primary keys from the candidate keys.

Candidate keys satisfy the following properties.

- Unique Identification - For every row, the value of the key must be unique. This implies that all non-key attributes are functional dependent on the key.
- Non-redundancy - No attribute in the key can be deleted without destroying the property of unique identification.

Consider the following contrived example.

EMPLOYEE-COURSE (EmployeeID, FirstName, LastName, CourseTitle, DateCompleted)

The functional dependencies are:

EmployeeID → FirstName, LastName
EmployeeID, CourseTitle →DateCompleted

Since EmployeeID & CourseTitle are the only candidate keys, they make up the primary key of the relation.

A **partial functional dependency** exists when a non-key attribute is dependent on part (but not all) of the the primary key.

In the above example, FirstName and LastName are non-key attributes that are dependent on part (only EmployeeID) of the primary key.

A relation is in **second normal form (2NF)** if it is in first normal form and contains no partial functional dependencies.

To convert a relation with partial dependencies to second normal form we do the following:

1. Create a new relation for each primary key attribute that is a determinant in a partial dependency. That attribute is the primary key in the new relation.
2. Move the non-key attributes that are only dependent on this primary key attribute from the old relation to the new relation.

Continuing the example, we start with the following relation and functional dependencies:

EMPLOYEE-COURSE (EmployeeID, FirstName, LastName, CourseTitle, DateCompleted)
EmployeeID → FirstName, LastName
EmployeeID, CourseTitle →DateCompleted

- Since EmployeeID is a primary key attribute that is a determinant in a partial dependency, we create a new relation with EmployeeID as the primary key.

EMPLOYEE-COURSE (EmployeeID, FirstName, LastName, CourseTitle, DateCompleted)
EMPLOYEE(EmployeeID)

- We then move FirstName and LastName from EMPLOYEE_COURSE to EMPLOYEE since they are non-key attributes that are only dependent on EmployeeID.

EMPLOYEE-COURSE (EmployeeID, CourseTitle, DateCompleted)
EMPLOYEE(EmployeeID, FirstName, LastName)