Rules of Functional Dependencies

In this lesson, we will take a look at the different rules of FDs.

WE'LL COVER THE FOLLOWING

Armstrong's axioms

Armstrong's axioms are a set of inference rules used to infer all the functional dependencies on a relational database. They were developed by William W. Armstrong in 1974.

We will denote a set of attributes by the letters X, Y, and Z. Also, we will represent the union of two sets of attributes X and Y by *XY* instead of the usual $X \cup Y$; this notation is rather standard in database theory when dealing with sets of attributes.

We will now highlight the three primary rules:

Axiom of reflexivity

This axiom says, if Y is a subset of X, then X determines Y.

> If $Y \subseteq X$ then , $X \rightarrow Y$

For example, `Address` is composed of more than one piece of information; i.e. `House_No`, `Street`, and `State`. So according to the axiom of reflexivity `Address` (X) $\rightarrow$ `House_No` (Y) as `House_No` $\subseteq$ `Address`.

Axiom of augmentation

The axiom of augmentation, also known as a **partial dependency**, says if X determines Y, then XZ determines YZ for any Z.

> If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

The axiom of augmentation says that every non-key attribute must be fully dependent on the whole composite primary key. To get a better understanding, look at the example below:

Std_Id	Course_Id	Name	Address	Age	Grade	Date_Completed
1	CS-100	Bob	777 Brockton Avenue, Abington MA 2351	22	A-	2019-10-09
1	PHY-101	Bob	777 Brockton Avenue, Abington MA 2351	22	C-	2019-10-10
2	CS-100	Jack	30 Memorial	20	B+	2019-10-12

			Drive, Avon MA 2322			

In the table above, we can see that `Std_Id` and `Course_Id` form a composite key (as the combination of these two attributes can be used to identify each tuple uniquely). However, we can also observe that the `Name`, `Address`, and `Age` attributes are only dependent on the `Std_Id`, not on the whole `Std_Id` and `Course_Id` composite key.

This situation is not desirable because every non-key attribute has to be fully dependent on the PK not just part of it. In this situation, student information is only partially dependent on the PK (`Std_Id`) which violates the axiom of augmentation.

We will discuss the solution to this problem when we study normalization in the next chapter.

Axiom of transitivity

The axiom of transitivity, also known as a **transitive dependency**, says if X determines Y, and Y determines Z, then X must also determine Z.

> If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

Let's consider the following example:

Std_Id	Std_Name	Address	Age	Prog_Id	Prog_Name
1	Bob	777 Brockton Avenue, Abington MA 2351	22	CS-200	Introduction to Programming

1	Bob	777 Brockton Avenue, Abington MA 2351	22	PHY-100	Modern Physics
2	Jack	30 Memorial Drive, Avon MA 2322	20	CS-300	Advanced Programming

The table above has information not directly related to the student; for instance, `Prog_Name` is not dependent on `Std_Id`; it's dependent on `Prog_Id`.

This situation is not desirable because a non-key attribute (`Prog_Name`) depends on another non-key attribute (`Prog_Id`), which results in transitive dependency.

Again we will discuss the solution to this problem when we study normalization in the next chapter.

In the next lesson, we will look at how to display these dependencies in diagrammatic form.