Module -4
Functional Dependencies & Normalization for Relational Databases

Functional Dependence:

- The attribute B is functionally dependent on A if A determines B. "A determines B" indicates that knowing the value of attribute A means that we can look up (determine) the value of attribute B. In this case, attribute A is the called the determinant.
- For example, knowing the *PROD_CODE* in the **PRODUCT** table in Figure 1 means we can look up the product's description, product price and so on. The notation we use for "A determines B" is:

A → B if A determines B
A → B, C, D if A determines B, C & D

Using the above example, we have

PROD_CODE → *PROD_DESCRIPT*

Table Name: PRODUCT
Primary Key: *PROD_CODE*
Foreign Key: *VEND_CODE*

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw Hammer	12.95	23	232
123-21UUY	Chain Saw	189.49	4	235
QER-34256	Sledge Hammer	18.63	6	231
SRE-657OG	Rat-Tail File	2.99	15	232
77x-3245Q	Steel Tape	6.79	8	235

Figure 1: PRODUCT Table

- As another example, note that *PROD_PRICE* attribute is functionally dependent on *PROD_CODE* since, for example, *PROD_CODE* value 001278-AB determines the *PROD_PRICE* value 12.95 (that is, *PROD_CODE* determines *PROD_PRICE*).
- As a rule, note that the primary key determines every other attribute in the table.
- **Question**: is *PROD_CODE* functionally dependent on *VEND_CODE*?

- Normalization

- Normalization is a technique used to design table structures in which data redundancies are minimized/eliminated.

SUNIL G L SVIT

- Normalization works through a series of stages called normal forms. The first three stages are described as first normal form (1NF), second normal form (2NF), and third normal form (3NF).
- From a structural point of view, 2NF is better than 1NF, and 3NF is better than 2NF.
- For most business database design purposes, 3NF is as high as we need to go in the normalization process.
- You should not assume that the highest level of normalization is always the most desirable. Generally, the higher the normal form, the more joins are required to produce a specified output and the more slowly the database responds to end user demands. Therefore, you will occasionally be expected to denormalize some table structures to avoid performance degradation.
- Denormalization is the reverse process of normalization and it produces a lower normal form; that is, 3NF can be denormalized to 2NF and 2NF can be denormalized to 1NF.

- The Need For Normalization

- To understand the benefits of normalization, let us consider a poorly designed table of a contracting company with many data redundancies as shown in Figure 2.

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	J. Arbough	E. Engineer	$84.50	23.8
		101	J. News	DB Designer	$105.00	19.4
		105	A. Johnson	DB Designer	$105.00	35.7
		106	W. Smithfield	Programmer	$35.75	12.6
		102	D. Senior	S. Analyst	$96.75	23.8
18	Amber Wave	114	A. Jones	A. Designer	$48.10	24.6
		105	A. Johnson	DB Designer	$105.00	20.0
		118	J. Frommer	G. Support	$18.36	45.3
		104	A. Ramoras	S. Analyst	$96.75	32.4
22	Rolling Tide	105	A. Johnson	DB Designer	$105.00	64.7
		104	A. Ramoras	S. Analyst	$96.75	48.4
		113	D. Joenbrood	A. Designer	$48.10	23.6
		106	W. Smithfield	Programmer	$35.75	12.8
25	Starflight	107	M. Alonzo	Programmer	$35.75	24.6
		115	T. Bawangi	S. Analyst	$96.75	45.8
		101	J. News	DB Designer	$105.00	56.3
		105	A. Johnson	DB Designer	$105.00	30.0

Figure 2: Poorly Designed Table

- The table in Figure 2 contains information about projects and employees working on these projects. This table does not conform to the Relational Database Model we learned in Chapter 5 for the following reasons:

    1. It contains information about two entities: **PROJECT** and **EMPLOYEE**.
    2. The project number (*PROJ_NUM*) is intended to be the primary key, or at least a part of a primary key, but it contains nulls.

SUNIL G L SVIT

3. The table contains data redundancies: *EMP_NAME*, *JOB_CLASS*, *CHG_HOUR*, etc.
4. Table entries invite data inconsistencies. For example, *JOB_CLASS* value "E. Engineer" in one row may be accidentally entered as E.Engineer in another row causing data inconsistencies.
5. The data redundancies yield the following anomalies (anamoly means that change is made in multiple places):

    a. **Update anomalies**: Modifying the *CHG_HOUR* for employee number (*EMP_NUM*) 105 for the Evergreen project for example requires updating the *CHG_HOUR* for the same employee in the other projects.
    b. **Insertion anomalies**: Employee data for employee number 105 is inserted many times in the table.
    c. **Deletion anomalies**: If employee 105 quits, not only he will be deleted from the Evergreen project, but he will also be deleted from the other projects (that is, a total of 4 deletions are required).

- Conversion To First Normal Form (1NF)

- A table is said to be in 1NF if it satisfies the following conditions:

1. It contains a primary key to uniquely identify each row.
2. There are no repeating groups. In other words, each row/column intersection can contain one and only one value, not a set of values. Note that the table in Figure 2 contains repeating groups because any project number (*PROJ_NUM*) can have a group of several data entries. For example, the Evergreen project's entries are shown in Figure 3:

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	J. Arbough	E Engineer	$84.50	23.8
		101	J. News	DB Designer	$105.00	19.4
		105	A. Johnson	DB Designer	$105.00	35.7
		106	W. Smithfield	Programmer	$35.75	12.6
		102	D. Senior	S. Analyst	$96.75	23.8

Figure 3: Example Repeating Group

3. All attributes are dependent on the primary key. Note that the primary key determines any other attribute in the table.

- Note that all tables conforming to the Relational Model we learned in Chapter 5 are in 1NF by default.

- Figure 4 below displays the conversion of the table in Figure 2 to 1NF by satisfying the above three conditions.

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	J. Arbough	E Engineer	$84.50	23.8
15	Evergreen	101	J. News	DB Designer	$105.00	19.4
15	Evergreen	105	A. Johnson	DB Designer	$105.00	35.7
15	Evergreen	106	W. Smithfield	Programmer	$35.75	12.6
15	Evergreen	102	D. Senior	S. Analyst	$96.75	23.8
18	Amber Wave	114	A. Jones	A. Designer	$48.10	24.6
18	Amber Wave	105	A. Johnson	DB Designer	$105.00	20.0
18	Amber Wave	118	J. Frommer	G. Support	$18.36	45.3
18	Amber Wave	104	A. Ramoras	S. Analyst	$96.75	32.4
22	Rolling Tide	105	A. Johnson	DB Designer	$105.00	64.7
22	Rolling Tide	104	A. Ramoras	S. Analyst	$96.75	48.4
22	Rolling Tide	113	D. Joenbrood	A. Designer	$48.10	23.6
22	Rolling Tide	106	W. Smithfield	Programmer	$35.75	12.8
25	Starflight	107	M. Alonzo	Programmer	$35.75	24.6
25	Starflight	115	T. Bawangi	S. Analyst	$96.75	45.8
25	Starflight	101	J. News	DB Designer	$105.00	56.3
25	Starflight	105	A. Johnson	DB Designer	$105.00	30.0

Figure 4: The Table in Figure 1 in 1NF

- Note that the table in Figure 4 satisfies the three conditions. The first condition is satisfied because the table has a composite primary key consisting of the key attributes *PROJ_NUM* and *EMP_NUM*. The second condition is satisfied because the table no longer contains repeating groups. The third condition is satisfied because any attribute is dependent on the primary key. For example, *EMP_NAME* is dependent on the primary key *PROJ_NUM*, *EMP_NUM* because if we know *PROJ_NUM* and *EMP_NUM* (15 and 103 for example), we can determine *EMP_NAME* (J. Arbough).

- Dependencies between attributes for the table in Figure 4 can be identified with the help of the dependency diagram as shown in Figure 5 below:

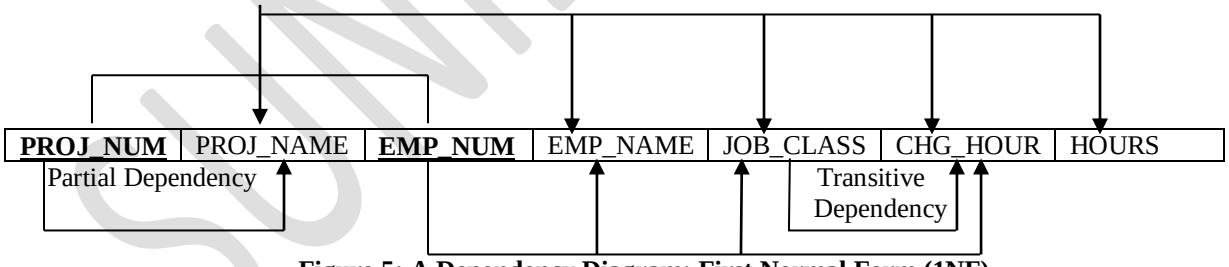


Figure 5: A Dependency Diagram: First Normal Form (1NF)

- Note the following about the dependency diagram in Figure 5:

1. The primary key is bold and underlined.
2. The arrows above attributes include all desirable dependencies. These dependencies can be expressed as follows:

PROJ_NUM, EMP_NUM → PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS
PROJ_NUM → PROJ_NAME
EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR

JOB_CLASS → CHG_HOUR

3. The arrows below the dependency diagram indicate less desirable dependencies:

   a. Partial Dependencies: This applies to composite primary keys (a composite primary key is a primary key that consists of more that one attribute) where the value of an attribute is only dependent on part of the primary key. For example, *PROJ_NAME* is dependent on *PROJ_NUM*, which forms part of the primary key (if we know *PROJ_NUM*, we can determine *PROJ_NAME*). Dependencies based on only a part of a composite primary key are called partial dependencies.
   b. Transitive Dependencies: Looking at Figure 5, note that *CHG_HOUR* is dependent on *JOB_CLASS.* Because neither *CHG_HOUR* nor *JOB_CLASS* is a key attribute, we have a condition known as transitive dependency. In other words, a transitive dependency is a dependency of one nonkey attribute on another nonkey attribute.

- Conversion To Second Normal Form (2NF)

- A table is said to be in second normal form (2NF) if it satisfies the following conditions:

1. It is in 1NF.
2. It includes no partial dependencies; that is, no attribute is dependent on a portion of the primary key.

- Let us convert the table in Figure 4 from 1NF to 2NF. To do that, follow these steps:

1. Write each key attribute (recall from Chapter 2 that a key attribute is an attribute that is part of the key) on a separate line, then write the primary key on the last line:

   PROJ_NUM
   EMP_NUM
   PROJ_NUM EMP_NUM

   The attributes on each line will become keys in a new table. In other words, the original table is now split into three tables. We'll call these tables **PROJECT**, **EMPLOYEE** and **ASSIGN** respectively.
2. The attribute(s) on each line will form the primary key for the new tables. The rest of the attributes for each table can be determined from the dependency diagram:

   PROJECT(*PROJ_NUM*, *PROJ_NAME*) because in the dependency diagram in Figure 5, *PROJ_NUM → PROJ_NAME*
   EMPLOYEE(EMP_NUM, *EMP_NAME, JOB_CLASS, CHG_HOUR)* because in the dependency diagram in Figure 5, *EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR*

SUNIL G L SVIT

ASSIGN(*PROJ_NUM*, *EMP_NUM*, *HOURS)* because in the dependency diagram in Figure 5, *PROJ_NUM*, *EMP_NUM* → *HOURS*

- Note that the new tables are in 2NF because each table is in 1NF and none of the tables contain partial dependencies. The dependency diagram for the new tables is shown in Figure 6:

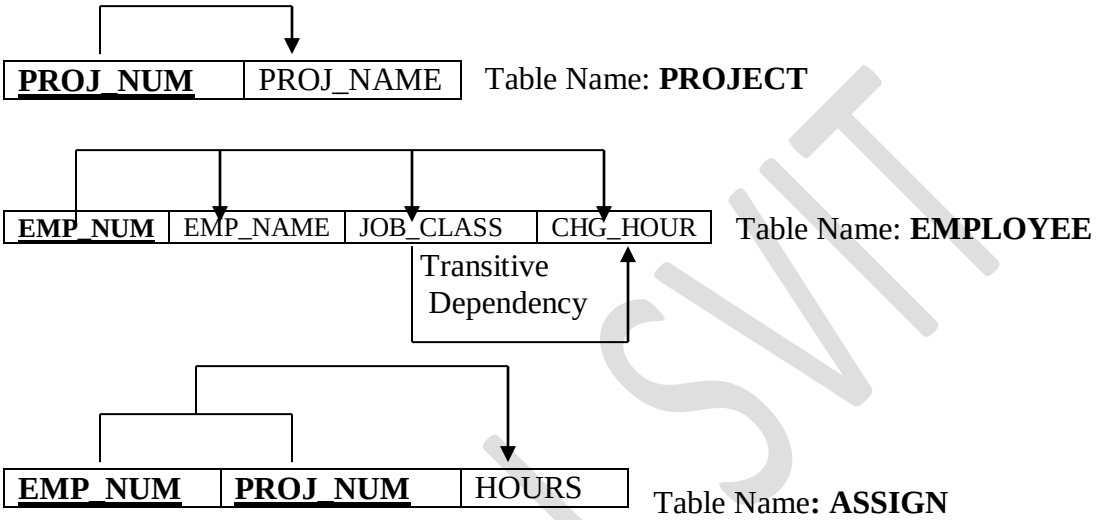


Figure 6: Second Normal Form (2NF) Conversion Results

- Note that the conversion to 2NF did not eliminate the transitive dependency (the conversion to 3NF will eliminate it). This dependency causes data redundancy if multiple employees have the same *JOB_CLASS* and it will therefore create the data anomalies we learned in Chapter 1.

- Because a partial dependency applies only to tables with composite primary keys, a table whose primary key consists of only a single attribute must automatically be in 2NF if it is in 1NF.

- Conversion To Third Normal Form (3NF)

- To eliminate the transitive dependency, the table must be converted to 3NF. A table is said to be in 3NF if it satisfies the following two conditions:

1. It it is in 2NF.
2. It contains no transitive dependencies.

- Note that in Figure 6, the **EMPLOYEE** table is the one that contains transitive dependency:

SUNIL G L SVIT

EMPLOYEE(EMP_NUM, *EMP_NAME, JOB_CLASS, CHG_HOUR)*

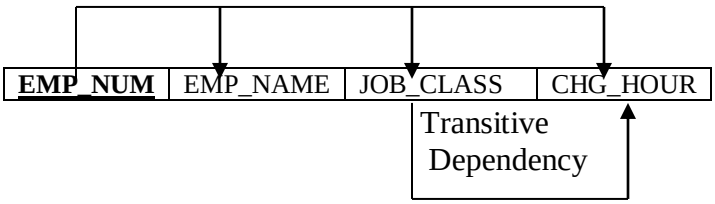


Table Name: **EMPLOYEE**

- To eliminate the transitive dependency, simply move the attributes causing data redundancy (namely *JOB_CLASS* and *CHG_HOUR*) into a new table called **JOB**. Then in the **EMPLOYEE** table, define a foreign key that will be used to link it to the **JOB** table. In this case, the foreign key will be *JOB_CLASS*. The result of the conversion to 3NF will be the following tables:

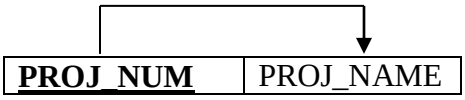


Table Name: **PROJECT**

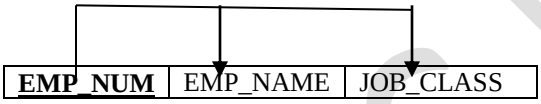


Table Name: **EMPLOYEE**

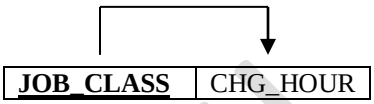


Table Name: **JOB**

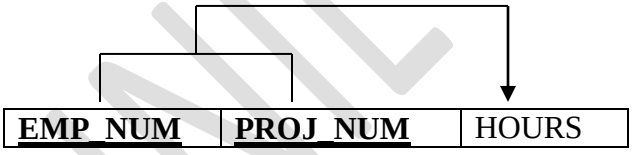


Table Name**: ASSIGN**

- Note that the conversion to 3NF has eliminated the original **EMPLOYEE** table's transitive dependency; the tables are now said to be in third normal form (3NF).

- The Boyce-Codd Normal Form (BCFN)

- A table T is said to be in Boyce-Codd Normal Form (BCFN) if every determinant in the table is a superkey of T.
- Note that every table in BCNF is also in 3NF, however, a table in 3NF is not necessarily in BCNF (see following example).
- A table that is in 3NF violates BCNF if it contains a nonkey that is the determinant of a key attribute. This situation can be illustrated using the example in Figure 7 below:

SUNIL G L SVIT

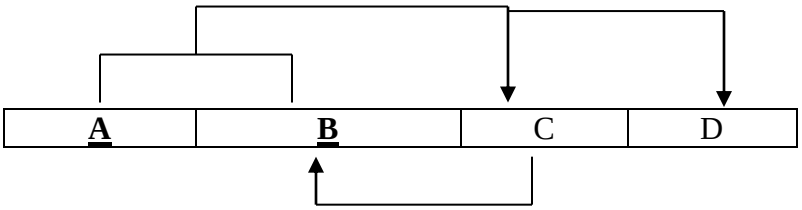


Figure 7: A Table That is In 3NF But Not In BCNF

- Note the following functional dependencies about Figure 7:

  A, B → C, D
  C → B

- Note that the table structure in Figure 7 has no partial dependencies nor does it contain transitive dependencies (Note that C → B indicates that a nonkey attribute determines a key attribute and this dependency is neither partial nor transitive).
- Therefore, the table structure in Figure 7 is in 3NF but not in BCNF because of the dependency C → B.
- To convert the table structure in Figure 7 to BCNF, follow the following procedure:

  1. Change the primary key to A, C and create table containing A, C and D.
  2. Create another table containing C as the primary key and B. Note that C will also serve as the foreign key to link this table to the table created in step 1.

- The conversion to BCNF is shown in Figure 8.

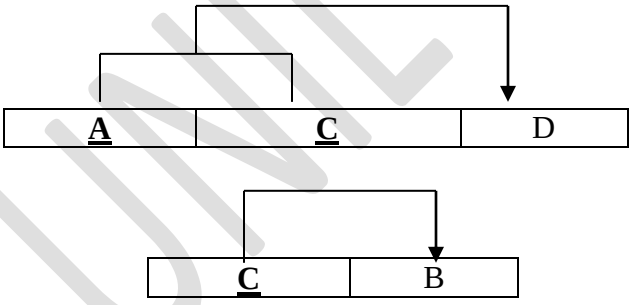


Figure 8: The Decomposition of a Table Structure To Meet BCNF Requirements

SUNIL G L SVIT

- Let us apply this procedure to the table in Figure 9.

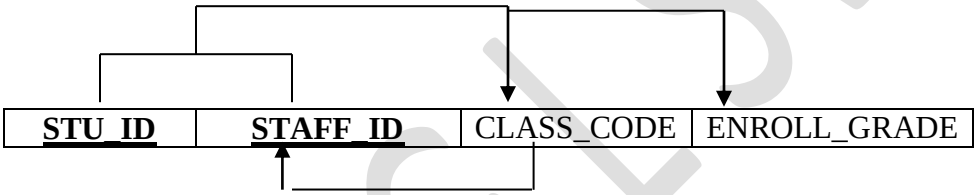
Primary Key: STU_ID,STAFF_ID

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

Figure 9: Sample Data For a BCNF Conversion

- Note the following dependencies in Figure 9:

  STU_ID, STAFF_ID → CLASS_CODE, ENROLL_GRADE
  CLASS_CODE → STAFF_ID



- The table structure above is in 3NF but not BCNF because *CLASS_CODE* is a determinant but not a superkey key. Why? To convert this table structure, we apply the above procedure by creating two tables. The first table will contain *STU_ID*, *CLASS_CODE* and *ENROLL_GRADE* (with *STU_ID* and *CLASS_CODE* as the primary key). The second table will contain the attributes *CLASS_CODE* and *STAFF_ID* with *CLASS_CODE* as the primary key. The result of the conversion to BCNF is shown in Figure 10 below.

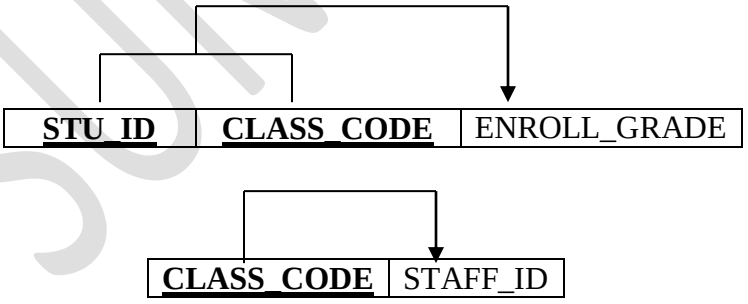


Figure 10: Conversion To BCNF

SUNIL G L SVIT