Normal Forms

When determining whether a particular relation is in **normal form**, we must examine the FDs between the attributes in the relation. The relational notation which we will be using takes the form

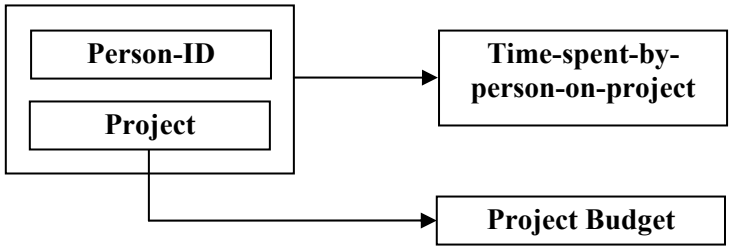
$$\mathbf{R1} = (\{\mathbf{X, Y, Z}\}, \{\mathbf{X \rightarrow Y, X \rightarrow Z}\})$$

and is made up of two components – firstly, the set of attributes in the relation and secondly a set of FDs that apply between these attributes.

Note: The FDs between attributes in a relation are obviously important when determining the relation's key. A relation key uniquely identifies a row, hence a key of a relation uniquely determines the values of the non-prime attributes, and therefore a full FD exists from the key to the nonprime attributes. **It is with full FDs whose determinants are *not* keys of a relation that problems arise.**

In general, we can observe that **if some FD $\mathbf{A \rightarrow B}$ holds for a relation *R*, and A is not a relation key, then relation *R* will involve some redundancy**.

For example, again examine the relation **ASSIGN**. In this relation, the key is {**Person-ID**, **Project**}. However, the attribute **Project-Budget** depends on only part of the key (i.e. on **Project**). The fact that an attribute does not depend on the *complete* relation key is an undesirable property of the relation and leads to anomalies.



Because Project is only part of the relation key, it is not necessarily unique. Project-Budget only depends on Project, so it is possible to give the same project two different budgets, which is contrary to the intended logic of the data and creates redundancy in instances of this relationship.

Conversion of relations to **normal form** requires choosing how the relation should be decomposed into relations that do not contain such undesirable dependencies. We will now examine four different levels of normal forms.

First Normal Form (1NF)

A relation is in the first normal form (**1NF**) if all the domains are *simple*, that is, if none of the attributes are themselves relations.

Relations that have non-atomic domains are not in first normal form and are called **unnormalised** relations – the only example of an unnormalised relation which we have encountered thus far is the relation LIVED-IN, which contained the non-simple (non-atomic) attribute **Residence**.

RELATION: LIVED-IN

Person	**Residence**
Martha	**City** / **Date-Moved-In** New York / 030303 Boston / 070705
Jack	**City** / **Date-Moved-In** Boston / 040593 Philadelphia / 070699

An unnormalised relation is normalised by replacing the non-simple domains with simple ones. Therefore, the 1st normalised form of the LIVED-IN relation would appear as follows:

RELATION: LIVED-IN

Person	**City**	**Date-Moved-In**
Martha	New York	030303
Martha	Boston	070705
Jack	Boston	040593
Jack	Philadelphia	070699

Second Normal Form (2NF)

A relation **R** is in second normal form (**2NF**) if it is in 1NF and every non-prime attribute in **R** is fully functionally dependent on every relation key of **R**.

We may rephrase this by saying that a relation is *not* in 2NF if it is possible to determine a non-prime attribute in the relation with only part of a relation key. Also, any relation not in 1NF is automatically not in 2NF.

The relation **ASSIGN** is in 1NF because all domains are atomic, but is not in second normal form because the nonprime attribute **Project-Budget** is not fully dependent on the relation key {**Person-ID**, **Project**}, but instead is fully dependent on a subset of the relation key – i.e. the attribute **Project**. In order to bring **ASSIGN** to second normal form, we must *decompose* it into two relations, **PROJECT** and **ASSIGN_1**, each of which is in 2NF and appears as follows:

RELATION: PROJECT

Project	Project-Budget
P1	32
P2	40
P3	27

RELATION: ASSIGN_1

Person-ID	Project	Time-spent-by-person-on-project
S75	P1	7
S75	P2	3
S79	P1	4
S79	P3	1
S80	P2	5

Exercise: Check that PROJECT and ASSIGN_1 are in 2NF by drawing an FD diagram for each relation.

We can use an informal (non-mathematical method) to decompose small tables into 2NF. For tables with many attributes we need to use mathematical algorithms (later).

Identify any functional dependencies whose determinant is not the full relation key. For each FD identified:

1. Create a new relation containing the determinant. The determinant is the relation key of the new relation.
2. Move the determined attributes from the original table to the new table.
3. The determinant of the FD in the original table becomes a Foreign Key.

Exercise 8

(a) Draw an FD diagram for the relation below:

R = {part_no, part_description, part_price, supplier_id, supplier_address}

The following FDs hold on **R**:

part_no $\rightarrow$ part_description
supplier_id $\rightarrow$ supplier_address
{part_no, supplier_id} $\rightarrow$ part_price

(b) Explain why this relation is not in 2NF.

(c) Normalise the relation to 2NF.

(d) Identify the relation keys and foreign keys in the new relations.

(e) Is your decomposition lossless? Explain how you determined this.

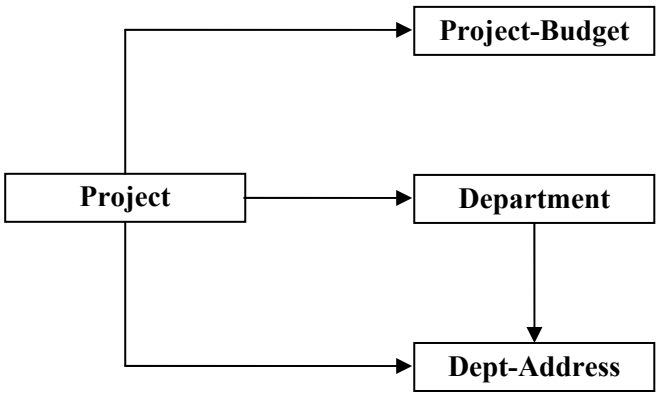
Third Normal Form (3NF)

It is possible to have relations that are in 2NF but that still contain redundancies. As an example, consider the relation **PROJECTS** below.

RELATION: PROJECTS

Project	Project-Budget	Department	Dept-Address
P1	32	Construct	20 Main St.
P2	40	Construct	20 Main St.
P3	27	Construct	20 Main St.
P4	17	Build	8 Fifth Ave.

The FDs between the attributes of **PROJECTS** are:



In this relation, the FDs demonstrate that each project is worked on in only one department, while a department may hold multiple projects. Each department has only one address. There is only one attribute in the relation key – namely, **Project**. Both **Department** and **Dept-Address** are fully functionally dependent on **Project**, and therefore **PROJECTS** is in 2NF. However, the relation does have one undesirable property – a department's address may be stored more than once, as there is an FD between **Department** and **Dept-Address** (both of which are non-prime attributes).

Third Normal Form (3NF) eliminates such redundancies from relations.

A relation **R** is in **3NF** if it has the following properties:

1. The relation **R** is in 2NF,

2. The non-prime attributes are *mutually independent* – i.e. no non-prime attribute is functionally dependent on another non-prime attribute.

An alternative definition of 3NF is:

- A relation **R** is in 3NF if it is in 2NF and also
- every non-prime attribute is *non-transitively* dependent on every relation key of **R**

The information in **PROJECTS** can be represented as 3NF relations by decomposing **PROJECTS** into the two relations **DEPTS** and **PROJECTS_1** as shown:

RELATION: DEPTS

Department	**Dept-Address**
Construct	20 Main St.
Build	8 Fifth Ave.

RELATION: PROJECTS_1

Project	**Project-Budget**	**Department**
P1	32	Construct
P2	40	Construct
P3	27	Construct
P4	17	Build

Both of these relations are in 3NF. The department addresses are stored only once.

The same (informal) procedure as before can be used to decompose small tables into 3NF. For tables with many attributes we need to use mathematical algorithms (later).

Identify any functional dependencies whose determinant is not the full relation key (e.g. any determinants which are non-prime attributes). For each FD identified:

1. Create a new relation containing the determinant. The determinant is the relation key of the new relation.
2. Move the determined attributes from the original table to the new table.
3. The determinant of the FD in the original table becomes a Foreign Key.

Note that the procedure above will decompose a relation whilst **preserving the functional dependencies**. That is the determinant and determined values will remain in the same relation, for each FD in the original relation. This is a desirable property of the decomposition process.

We can state the overall objectives of normalisation as:

- **Eliminate/reduce redundancy and possibilities for update anomalies (i.e. bring relation to higher normal form)**
- **Decomposition must be lossless**
- **Dependencies should be preserved**

Note: It is always possible to meet these three criteria when normalising up to 3NF.

To achieve a higher normal form than 3NF, we may have to sacrifice objective 3.

We should never break objective 2. when normalising a relation.

Exercise 9

Consider the following relations and the FDs between the attributes in each relation. For each relation:

(a) Draw an FD diagram showing the FDs,

(b) State all the relation keys in the relation,

(c) State the highest normal form (1NF, 2NF or 3NF) of the relation.

R1 = ({W,X,Y,Z}, {XY $\rightarrow$ Z, Y $\rightarrow$ W})
R2 = ({A,B,C,D}, {A $\rightarrow$ BC, C $\rightarrow$ D})
R3 = ({K,L,M,N}, {KL $\rightarrow$ N, K $\rightarrow$ M})
R4 = ({P,Q,R,S}, {P $\rightarrow$ Q, Q $\rightarrow$ R, R $\rightarrow$ S})

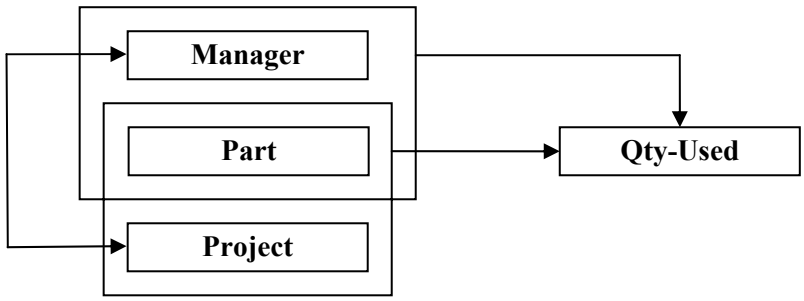
Boyce-Codd Normal Form (BCNF)

In our examples of 2NF and 3NF, we have seen how we can eliminate certain undesirable redundancies in a relation. There are other types of redundancies which can exist but will pass the criteria for 2NF/3NF. For example, examine relation **USE**:

RELATION: USE

Project	**Manager**	**Part**	**Qty-Used**
P1	Carson	Hammer	10
P1	Carson	Drill	15
P2	Gregg	Drill	22
P2	Gregg	Saw	34
P2	Gregg	Hammer	50
P3	White	Drill	37

This relation stores the parts used by both projects and project managers. Each project has one manager and each manger manages only one project. The FDs for this relation are:



There are two relation keys of USE – {**Project**, **Part**} and {**Manager**, **Part**}. The keys overlap because **Part** appears in both, whereas **Project** and **Manager** appear in one relation key only.

The relation **USE** is in both 2NF and 3NF. It has only one non-prime attribute, **Qty-Used**, which is fully functionally dependent on each of the two relation keys and not dependent on any other non-prime attribute. However, it does have some undesirable properties:

- The manager of each project is stored more than once,
- A project's manager cannot be stored until the project has ordered some parts,
- A project cannot be entered unless that project's manager is known,
- If a project's manager is changed, some *n* tuples must also be changed.

The reason for these undesirable properties is that relation **USE** has two relation keys that overlap. There is also a dependency between the two attributes **Project** and **Manager**, both of which appear in one relation key only. To identify these types of undesirable dependencies, the **Boyce-Codd Normal Form** (BCNF) is defined:

A relation **R** is in **BCNF** if, for every *non-trivial* FD $\mathbf{X} \rightarrow \mathbf{Y}$ in **R,**

X is a superkey of **R**

Relation **USE** does not satisfy this condition, as it contains the two FDs,

$$\mathbf{Manager} \rightarrow \mathbf{Project}$$
$$\mathbf{Project} \rightarrow \mathbf{Manager}$$

but neither **Manager** nor **Project** is a superkey.

The relation **USE** can be decomposed into the following two relations which meet the BCNF requirement:

USE (**Project**, **Part**, **Qty-Used**)
PROJECTS (**Project**, **Manager**)

Note a simple difference between 3NF and BCNF – in the definition of BCNF each FD $\mathbf{X} \rightarrow \mathbf{Y}$, **X** *must always* be a superkey. For 3NF, however, **X** must be a superkey *only* if **Y** is not part of any other relation key (i.e. if **Y** is non-prime).

Note that it is not always possible to find a dependency preserving BCNF decomposition: Example: $R = (\{J, K, L\}, \{JK \rightarrow L, L \rightarrow K\})$ has no dependency-preserving BCNF decomposition. **Exercise**: Draw an FD diagram and check that this relation is in 3NF.

Exercise 10

Consider the following relations and the FDs that hold in each relation.
For each relation:

(a) Draw an FD diagram showing the FDs,

(b) State all the relation keys in the relation,

(c) State the highest normal form (1NF, 2NF, 3NF, BCNF) of the relation.

$R1 = (\{T,U,V\}, \{T \rightarrow U, U \rightarrow T, T \rightarrow V\})$

$R2 = (\{W,X,Y,Z\}, \{WY \rightarrow XZ, Z \rightarrow W, YZ \rightarrow W\})$

$R3 = (\{A,B,C,D\}, \{AB \rightarrow D, AC \rightarrow D\})$

$R4 = (\{K,L,M,N\}, \{KL \rightarrow N, KM \rightarrow NL, L \rightarrow M\})$

$R5 = (\{A,B,C,D\}, \{A \rightarrow BC, A \rightarrow D, A \rightarrow E\})$