**Normalization**

Normalisationis a process which we analyze and alter a database relation in order to get more concise and organized data structures. Normalised data is stable and has a natural structure. We call a relation normalized if:

* it does not contain any redundancy
* it does not cause maintenance problems
* it is an accurate representation of the data

Relations that aren't normalised contain non-atomic attributes and therefore can contain redundant information. Detailed planning of the ERM can help creating normalised relations. The following steps will explain how existing relations can be normalised step by step.

# Dependencies

In order to be able to normalise a relation according to the three normal forms, we must first understand the concept of dependency between attributes within a relation.

**Functional dependency:**

If A and B are attributes of relation R, B is functionally dependenton A (denoted A --> B), if each value of A in R is associated with exactly one value of B in R.

Example:

| **ID** | **Name** |
| --- | --- |
| S1 | Meier |
| S2 | Weber |

The attribute Name is functionally dependent of attribute ID (ID --> Name).

**Identification key:**

If every attribute B of R is functionally dependent of A, than attribute A is a primary key.

Beispiel:

| **ID** | **Name** | **Surname** |
| --- | --- | --- |
| S1 | Meier | Hans |
| S2 | Weber | Ueli |

Attribute ID is the identification key

**Full functional dependency:**

We talk about full functional dependency if attribute B is functional dependent on A, if A is a composite primary key and B is not already functional dependent on parts of A.

Beispiel:

| **IDStudent** | **Name** | **IDProfessor** | **Grade** |
| --- | --- | --- | --- |
| S1 | Meier | P2 | 5 |
| S2 | Weber | P1 | 6 |

The attribute Grade is fully functional dependent on the attributes IDStudent and IDProfessor.

**Transitive dependency:**

If A determines B and B determines C then C is determined by (dependent on) A. We write A --> B and B --> C but not B --> A.

Example:

| **ID** | **Name** | **Konto\_Nr** | **Bank\_Code\_No** | **Bank** |
| --- | --- | --- | --- | --- |
| L1 | Meier | 1234-5 | 836 | UBS |
| L2 | Weber | 5432-1 | 835 | CS |

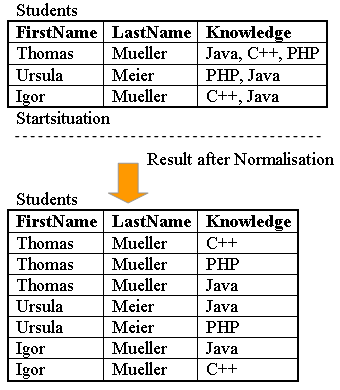
There is a transitive dependency between Bank\_Code\_No and Bank because Bank\_Code\_No is not the primary key of the relation.

# First normal form (1NF)

**First normal form:**

A relation is in first normal form if every attribute in every row can contain only one single (atomic) value.

A university uses the following relation:  
  
Student(Surname, Name, Skills)  
  
The attribute Skills can contain multiple values and therefore the relation is not in the first normal form.  
  
But the attributes Name and Surname are atomic attributes that can contain only one value.

Example First normal form

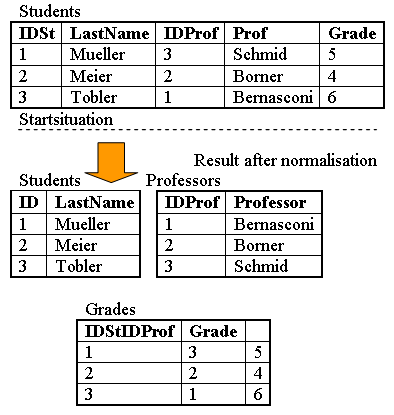
To get to the first normal form (1NF) we must create a separate tuple for each value of the multivalued attribute

# Second normal form (2NF)

**Second normal form:**

A relation is in second normal form if it is in 1NF and every non key attribute is fully functionally dependent on the primary key.

A university uses the following relation:  
  
Student(IDSt, StudentName, IDProf, ProfessorName, Grade)  
  
The attributes IDSt and IDProf are the identification keys.  
All attributes a single valued (1NF).  
  
The following functional dependencies exist:  
  
1. The attribute ProfessorName is functionally dependent on attribute IDProf (IDProf --> ProfessorName)  
  
2. The attribute StudentName is functionally dependent on IDSt (IDSt --> StudentName)  
  
3. The attribute Grade is fully functional dependent on IDSt and IDProf (IDSt, IDProf --> Grade)

Example Second normal form

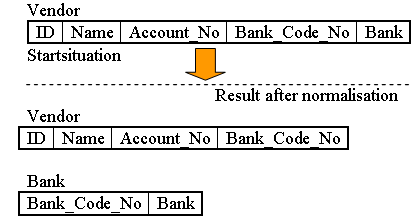
The table in this example is in first normal form (1NF) since all attributes are single valued. But it is not yet in 2NF. If student 1 leaves university and the tuple is deleted, then we loose all information about professor Schmid, since this attribute is fully functional dependent on the primary key IDSt. To solve this problem, we must create a new table Professor with the attribute Professor (the name) and the key IDProf. The third table Grade is necessary for combining the two relations Student and Professor and to manage the grades. Besides the grade it contains only the two IDs of the student and the professor. If now a student is deleted, we do not loose the information about the professor.

# Third normal form (3NF)

**Third normal form:**

A relation is in third normal form if it is in 2NF and no non key attribute is transitively dependent on the primary key.

A bank uses the following relation:  
  
Vendor(ID, Name, Account\_No, Bank\_Code\_No, Bank)  
  
The attribute ID is the identification key. All attributes are single valued (1NF). The table is also in 2NF.  
  
The following dependencies exist:  
  
1. Name, Account\_No, Bank\_Code\_No are functionally dependent on ID (ID --> Name, Account\_No, Bank\_Code\_No)  
  
2. Bank is functionally dependent on Bank\_Code\_No (Bank\_Code\_No --> Bank)

Example Third normal form

The table in this example is in 1NF and in 2NF. But there is a transitive dependency between Bank\_Code\_No and Bank, because Bank\_Code\_No is not the primary key of this relation. To get to the third normal form (3NF), we have to put the bank name in a separate table together with the clearing number to identify it.