**Database Normalisation**

**Normalisation:** process of no loss decomposition

Normalisation is the process of organising the columns and tables of a relational database to eliminate data redundancy and ensure data integrity. Normalisation usually involves dividing large table sets into smaller, less redundant tables and defining relationships between them.

Normalisation is to design a database so that duplicate and redundant data is avoided. If some piece of data are duplicated in several places in the database, there is the risk that it is updated in one place but not the other, leading to data corruption.

Normalisation aims to:

* Remove redundant data
* Limit the use of NULL values
* Remove opportunities for loss of data

and improve database maintenance and resolve database anomalies:

* Insert
* Update
* Delete

**Redundant data**

Data that appears in more than one place.

**NULL**

NULLs are markers indicating the lack of a value, which is confusing to work with. NULLs are not equivalent to zero. NULL values cannot be removed entirely from a database as they define optionality.

**Functional Dependency**

Normalisation has one semantic concept; functional dependence. The primary key determines all other data items. Non key items must depend on the key, the whole key and nothing but the key.

**DRY**

Normalisation does not directly apply to anything outside of databases. The principles of normalisation are quite specific to relational databases. However the general underlying theme, that you shouldn't have duplicate data, can be applied broadly in software engineering. This is the DRY principle (Don’t Repeat Yourself).

**Normal Form Levels**

There are a number of normalisation levels from 1st normal form through Boyce Codd level to 15th normal form. Each normal form describes how to remove of some specific problem, usually related to redundancy. 3NF is appropriate for most professional applications. The problem solved by normal forms above 3NF are quite rare problems, so chances are that your schema is already in 5NF.

Note: You can remember the first three normal forms by the terms

1NF| repeating\* or 1NF| no repeats

2NF| redundant 2NF| the whole key

3NF| non-dependent 3NF| nothing but the key

**First Normal form (1NF)**

Each data items is an [atomic](http://en.wikipedia.org/wiki/First_normal_form#Atomicity) values and the value of each attribute contains only a single value from that domain. An atomic value is a value that cannot be divided.

Example

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **surname** | **petname** | **type** |
| 12 | BROWN | TOM | CAT |
|  | | JERRY | MOUSE |
| 13 | BLUE | MICKEY | MOUSE |
|  | | SCOOBY | DOG |

Problem: petname and type have multiple values.

Solution: petname and type need to be moved to an additional relation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **customer\_id** | **surname** |  | **customer\_id** | **petname** | **type** |
| 12 | BROWN |  | 12 | TOM | CAT |
| 13 | BLUE |  | 12 | JERRY | MOUSE |
|  |  |  | 13 | MICKEY | MOUSE |
|  |  |  | 13 | SCOOBY | DOG |

**Second Normal form (2NF)**

Second normal requires all non-key items to be fully functional dependent on the whole of the primary key, not just part of the primary.

Example

|  |  |  |
| --- | --- | --- |
| **order\_number** | **product\_code** | **product\_name** |
| 1001 | P456 | pen |
| 1002 | P123 | pencil |
| 1003 | P321 | ruler |

Problem: Product name is dependent on product code only, order number isn’t needed to determine the product name.

Solution: Product needs to be moved to an additional relation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **order\_number** | **product\_code** |  | **product\_code** | **product\_name** |
| 1001 | P456 |  | P456 | pen |
| 1002 | P123 |  | P123 | pencil |
| 1003 | P321 |  | P321 | ruler |

**Third Normal form (3NF)**

Third normal form requires there to be no transitively functional dependency. All non key items should depend fully on the primary key and nothing else.

Example

|  |  |  |  |
| --- | --- | --- | --- |
| **award** | **year** | **winner** | **releasedate** |
| MOST PROMISING GAME | 2011 | MICKEY MOUSE CASTLE OF ILLUSION | 21-APR-2000 |
| BEST CHARACTER | 2014 | ANGERY BIRDS | 21-NOV-2010 |
| BEST KINECT GAME | 2015 | SPORTS RIVALS | 5-MAY-2001 |

Problem: release date is transitively dependant on award via winner. Winner determines release date.

Solution winner needs to be moved to an additional relation

|  |  |  |
| --- | --- | --- |
| **award** | **year** | **winner** |
| MOST PROMISING GAME | 2011 | MICKEY MOUSE CASTLE OF ILLUSION |
| BEST CHARACTER | 2014 | ANGRY BIRDS |
| BEST KINECT GAME | 2015 | SPORTS RIVALS |

|  |  |
| --- | --- |
| **winner** | **releasedate** |
| MICKEY MOUSE CASTLE OF ILLUSION | 21-APR-2000 |
| ANGRY BIRDS | 21-NOV-2010 |
| SPORTS RIVALS | 5-MAY-2001 |

The principles of 3NF should be applied to all professional databases whether the process is carried out or the data reviewed.

* No repeating data
* Non key items depend on the whole key and only the key
* There is no redundant data
* There is minimal use of NULL
* There is no opportunity for loss of data

**Codds Normal Form Definitions**

**1NF**

The domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

**2NF**

No non-prime attribute in the table is functionally dependent on a proper subset of any candidate key**3NF**

Every non-prime attribute is non-transitively dependent on every candidate key in the table. The attributes that do not contribute to the description of the primary key are removed from the table. In other words, no transitive dependency is allowed.