**Normalization**

Normalization is a set of rules to allocate data to tables in such a way as to minimize the redundancy and to minimize certain anomalies related to insert ,delete and update of data.

**Anomalies**

A major objective of data normalization is to avoid modification anomalies.

These come in three flavors :

1. An insertion anomaly is a failure to place information into all the places in the database. In a properly normalized database, information about a new entry needs to be inserted into only one place in the database.
2. A deletion anomaly is a failure to remove information about an existing database entry from all places. In a properly normalised database, the entry needs to be deleted from only one.
3. A update anomaly is a failure to update information about an existing database entry from all places.

All three kinds of anomalies are highly undesirable, since their occurrence constitutes corruption of the database.

**First Normal Form**

1. A Relations is said to be in First Normal Form (1NF) when it contains no multi-valued attribute, i.e. every attribute must contain at most one data value.
2. In other words a table is in first normal form if all the key attributes have been defined and it contains no repeating groups

Example : Let us assume an STUDENT table as below

|  |  |  |
| --- | --- | --- |
| SID | SNAME | COURSE |
| 101 | Raja | C, C++ |
| 102 | Roja | C, Java |
| 103 | Pooja | Oracle, Java |

In this table we have multi value attribute in the course column.

|  |  |  |
| --- | --- | --- |
| SID | SNAME | COURSE |
| 101 | Raja | C |
| 101 | Raja | C++ |
| 102 | Roja | C |
| 102 | Roja | Java |
| 103 | Pooja | Oracle |
| 103 | Pooja | Java |

Now we have eliminated multi value attributes using first normal form.

**Second Normal Form**

A relation is said to be in 2nd Normal Form if

* The relation is in 1st Normal Form
* Every non key attribute(cust\_address,cust\_contact) is fully functionally dependent on the primary key

(i.e. there are no partial dependencies).

1. Anomalies can occur when attributes are dependent on only part of a multi-attribute (composite) key.

2. A relation is in second normal form when all non-key attributes are dependent on the

whole key. That is, no attribute is dependent on only a part of the key.

3. Any relation having a key with a single attribute is in second normal form

Take the following table structure as an example:

Order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| order\_id | Cust\_id | cust\_address | cust\_contact | order\_date | order\_total |
| 1 | 101 | Chennai | 9092112345 | 01-02-2017 | 1000 |
| 2 | 101 | Chennai | 9092112345 | 01-03-2017 | 2000 |

Here we should realize that cust\_address and cust\_contact are functionally dependant

On the whole key. To make this table 2NF these attributes must be removed and placed

somewhere else.

**Order\_cust**

|  |  |  |  |
| --- | --- | --- | --- |
| order\_id | cust\_id | order\_date | order\_total |
| 1 | 101 | 01-02-2017 | 1000 |
| 2 | 101 | 01-03-2017 | 2000 |

**Customer**

|  |  |  |
| --- | --- | --- |
| cust\_id | cust\_address | cust\_contact |
| 101 | Chennai | 9092112345 |

So what problems can occur if we do not make it into 2nd Normal Form?

It can result into

**Update anomaly**: If you want to update the address of a customer, need to update in all orders made by this customer, if update is missed at any place then data will be inconsistent.

**Insert anomaly** : there is no way to add a customer information he/she has not made any order.

**Third Normal Form**

The table is said to be in third normal form

1. Meets 1NF and 2NF

2. Does not contain columns(attributes) that not fully dependent on the primary key

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Empno | Ename | Gender | Salary | Annual\_sal | Dname | Dhead |
| 101 | Raja | Male | 1000 | 12000 | IT | Ram |
| 102 | Roja | Female | 2000 | 24000 | HR | Sam |
| 103 | Pooja | Female | 3000 | 36000 | IT | Ram |
| 104 | Vanaja | Female | 2500 | 30000 | HR | Sam |

Here Annual\_sal is not fully dependent on the primary key column and it can be calculated from salary column on the fly so we can remove this Annual\_sal to get 3NF.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| empno | Ename | Gender | Salary | Dept\_Id |
| 101 | Raja | Male | 1000 | 1 |
| 102 | Roja | Female | 2000 | 2 |
| 103 | Pooja | Female | 3000 | 1 |
| 104 | Vanaja | Female | 2500 | 2 |

|  |  |  |
| --- | --- | --- |
| Dept\_Id | Dname | Dhead |
| 1 | IT | Ram |
| 2 | HR | Sam |

**Boyce-Codd Normal Form**

1. It is next to the third normal form
2. It was developed by Boyce and E.F.Codd
3. When there is more than one candidate keys ( atleast one composite key) with 3NF

we go for BCNF.

4. It is possible to have a table with 3NF & BCNF

with 3NF & without BCNF

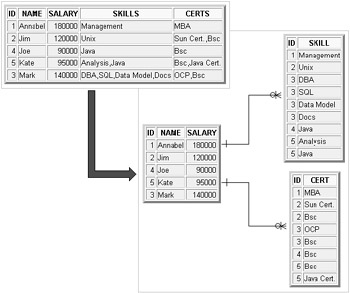
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | **Sno** | **HOD** |  |  |  |
|  | Student\_subject | |  |  | 1 | Zakaria |  |  |  |
|  | **Sno** | **Subject** | **HOD** |  | 2 | Ramanujam |  |  |  |
|  | 1 | Electronics | Zakaria |  | 1 | Ramanujam |  |  |  |
|  | 2 | Maths | Ramanujam |  |  |  |  |  |  |
|  | 1 | Maths | Ramanujam |  |  |  | **HOD** | **Subject** |  |
|  |  |  |  |  |  |  | Zakaria | Electronics |  |
|  |  |  | |  | | --- | |  | |  |  |  | Ramanujam | Maths |  |
|  |  |  |  |  |  |  | Ramanujam | Maths |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**Fourth Normal Form**

To be in Fourth Normal Form,

1. Table must be in Boyce-Codd Normal Form.

2. Table may not contain more than one multi-valued attribute.



Fourth Normal Form eliminates independent Many-to-Many relationships between columns

**Fifth Normal Form**

**Normalization and Tuning**

What does Normalization have to do with tuning? There are a few simple guidelines to follow and things to watch out for:

1. Too little Normalization leads to too much duplication of data and thus your database will get too big. Too much data will lead to slow access times due to having too much disk space to search through.
2. Incorrect Normalization should be obvious: convoluted and complex application code with nowhere to go but a rewrite or the garbage pile, the final resting place of many failed commercial applications.
3. Too much Normalization leads to over complex SQL code which can be difficult, if not impossible, to tune. I tend to pass by 4th and 5th Normal Forms and think very carefully about the usefulness of 3rd Normal Form derived entities.
4. Quite often databases are designed without knowledge of applications. The data model could be built on a purely theoretical basis. Later in the development cycle, the application not only has difficulty using a highly granular data model but the data may be structured totally differently to that of the application. One possible answer is that both development and administration people should be involved in data modeling. Another possibility is that the data model supports the application and thus should be built with all knowledge in hand. This is of course impossible. Thus, it should be acceptable to alter the data model at least during the development process, possibly substantially. Most of the problems with relational database model tuning are Normalization related. Normalization should be simple because it is simple!