# Database

Talking about database, everyone will talk about tables and stored procedures (that are functions which includes a bunch of SQLs that performs a specific task). Actually, tables are entities. What is entity? You can think of database entity like Java Object. For instance a “Car” which include attributes and methods. A database entity is a table which includes attributes that we refer to fields, triggers, primary or composite keys. Database entities like Java OO has relationship that is each entity is related to another entity (of course sometimes can be standalone). Types of relationship includes “one to one” relationship, “one to many” relationship and “many to many” relationship. Each will be discussed. How to define an entity is based on the business/User requirement. Let’s take an example of Taobao website. Today I logged in the Taobao website with my mobile phone number and discovered that my name **is not displayed correctly** and the ordering items shown are not the ones I purchased. From this incident, lets don’t talk about the cause of why this happened, we talk about some business rules. First how do we identify a Taobao user? I may guess by his/her personal Id (HKID/China ID) or by mobile phone. Second how does he/she login and what happens when he/she successfully logins. From the business requirement, what are the entities and relationships? We need to think like we programming Java OO, if you were writing a program about user login, I assume there will be an object “User”. That’s right “User” will be the entity (table), second “Login”. Then what are the relationship about these two entities? (one to one, one to many or many to many?), what are these relationships means?

## One to One relationship

Login

User

1

1

From the above diagram, one user corresponds to one and only one login.

## One to Many Relationship

Login

User

1

0 to \* \*

From the above diagram, one user corresponds to more than one login.

## Many to Many Relationship

Login

User

1 to \* \*

0 to \* \*

From the above diagram. Two or more users with the same identity (same HKID/CHINA ID/Mobile Phone) has more than one login. Seems the cause of my case above.

After illustrating all the relationships. Which is more being the case? The answer is “one to many”. Why? One to One relationship seems feasible, if **only** the userlogin once a time but never again. I assume if I am a Taobao user, I will frequently login to the system. For “many to many” also seems feasible but not realistic. How do I identify a particular user? If several users hold the same identify. Talking about this case, we need first understand what is database “Normal Forms” and “Primary / Composite” keys. It is hard to just look at the entity diagram (Actually there is a term **Entity Relationship Diagram or ER Diagram for short**). Let’s filled up with some data.

**Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Mobile phone Number | Name | Date Of Birth | Agenda |
| 0 | 13138628002 | Alan Tang | 21-09-1979 | Male |
| 1 | 66966379 | Sing Lun | 12-09-1980 | Male |
| 2 | 64960648 | Fun Yueng | 19-08-1949 | Female |
| 3 | 13138628002 | 林浩天 (林罗吴) | 05-04-2000 | Male |

**Login**

Name Includes Brackets

|  |  |  |
| --- | --- | --- |
| Id | Mobile phone Number | Login Time |
| 0 | 13138628002 | 2023-04-21 10:30 AM |
| 1 | 64960648 | 2023-04-23 11:30 AM |
| 2 | 64960648 | 2023-04-24 12:30 AM |
| 3 | 13138628002 | 2023-04-24 10:30 AM |

For Simplicity now we have filled up the tables with some data. As we talk above, we prefer “**One to Many**” relationship, that means one and only one user can login to the system multiple times. By looking at both tables above, can we achieve this? Seems possible for the user “Fun Yueng”, but not for the user with the mobile phone number “13138628002”, why? Because by looking at the mobile phone “13138628002” we cannot identify which user login the system. For example, if the manager raises up a question one day by asking who login the system on the 24th of April. There will be 2 possible answers one is the user with “64969648” and the other is “13138628002”. But looking carefully, for the mobile phone “13138628002”, you cannot identify which user “Alan” or “林浩天” login. That means that in order to fulfil the requirement of “**One to Many**” relationship, we need to have some guidelines, that is what we call “**Database Normal Form**”. Which will be talk about in the next section.

## First Normal Form (1NF)

Many websites or text books have their meanings of first normal form, but they are all similar. A database is said to be first normal form if it satisfied the following conditions:

* **Rule 1 (Atomic Values)**− Every column of a table should contain only atomic values. An atomic value is a value that cannot be divided further.
* **Rule 2 (No Repeating Groups)**− There are no repeating groups of data. This means a table should not contain repeating columns.

The problem seems to occur from the user’s table. We will then focus on the user’s table. Does the users table violate rule number one. The answer is “Yes”, why? Let’s take a look at the **name** column field. Rule number 1 suggested that every column should only contain one atomic value and the value cannot be divided further. However, by looking at the name filed we cannot define which user has the surname for instance call Mr Tang, although you can have some searching algorithm to do so. That means in order to achieve rule number 1, we need to separate the name column to “Surname” and “Other or Given Name”. There are 2 ways to do that one is to add additional columns (suitable for adding only limited number of columns) to users table and the other is to add a separate “Name” entity.

Then the User table will be split up to the 2 tables

**Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Mobile phone Number | NameId | Date Of Birth | Agenda |
| 0 | 13138628002 | 0 | 21-09-1979 | Male |
| 1 | 66966379 | 3 | 12-09-1980 | Male |
| 2 | 64960648 | 1 | 19-08-1949 | Female |
| 3 | 13138628002 | 2 | 05-04-2000 | Male |

Name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | SurName | MiddleName | OtherName | NickName |
| 0 | Tang |  | Alan |  |
| 1 | Yeung |  | Fun |  |
| 2 | 林 |  | 浩天 | 小豬 |
| 3 |  |  | SingLun |  |

Now if we want to ask which user has the surname of “Mr Tang”. We could easily find out. By using the below SQL statement:

Select \* from users u, name n where u.nameid = n.id and n.surname = “Tang”

That will satisfy rule number 1.

Now we have a new entity Name. The Entity Diagram will be:

Name

Users

1

1

After satisfied rule number 1, lets jump to rule 2. The definition of rule number 2 states that there should no repeating groups of data, that is no repeating columns. Let take a look at our user’s table. Do we violate this constraint. The answer is “Yes”. Why? Let’s take a look at the user data.

**Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Mobile phone Number | NameId | Date Of Birth | Agenda |
| 0 | 13138628002 | 0 | 21-09-1979 | Male |
| 1 | 66966379 | 3 | 12-09-1980 | Male |
| 2 | 64960648 | 1 | 19-08-1949 | Female |
| 3 | 13138628002 | 2 | 05-04-2000 | Male |

As discussed previously in order to **identify** a single person, we need to have an attribute to identify it. From the Users table, the possible fields are “**Id**” and “**Mobile Phone Number**”. That’s right. 2 columns can identify it. Which one should be used? The answer is “Phone Number” not the “Id”. Why there is an “Id” column, if it cannot identify a single user? Cause this is the matter of what type of database you are working with. Some database (don’t count on which brand such as Oracle, MS SQL, Access....), when creates a table, it will automatically add an “id” field at the beginning of the table, and that id field will automatically increment by 1, each time when a record is created. It will not duplicate. People sometimes use as a key field to identify each single record, but not for this case. What attributes beastly identify a single user apart from the “id” field? (cause id field is not a user attribute), you cannot make a user to memorized the id number, the only attribute he knows is his/Her’s “phone number” or even his/her document “id” if have. How do we handle this situation, we cannot have 2 keys to identify a user at the same time. Yes, there is, that is what we call “**unique key**” not “**primary key**”, “primary key” default is predefined as an “id” field, we cannot or supposed to revoke it. However, we can set a unique key for the “Mobile Phone” number. Primary keys default comes with “**indexes**” which can be think of a “**hash table**” which facilitate searching. When create an “Unique Key”, you can also specify an index with it. Another concern is “Disaster and Backup Recovery”. First ask a question, what happens if an employee of a company delete some of the users records in the user table. Ask yourself, how will you recover it? The answer might be inserting the user records from the backup that does not exists in the user table. Right! what happens when you insert all the records or partial records that already exists in the user table? As told before the “**id**” field increments automatically, how do I know the “id” field from the backup is correspond to the “id” filed of the user’s table. That is why “Unique Key” comes into play. The “Unique Key” is a constraint, not just a simple field with index, when inserting records, the database system will check where the “phone number’, such as my case “13138628002” is exists or not in the user table. If not exists, create or insert, or else the database system will not allow you and will prompt you with an error message together with an error code (usually not friendly message). Now back to the case, is there any way to eliminate only one user that’s correspond to the phone number “13138628002”, “NO”, this must be handle manually, the staff of the company must contact both users and clarify who is the owner of this “Phone Number”.

To find all the duplicates, use the following syntax:

Select count(\*), phoneNo from users group by phoneNo

**Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Mobile phone Number | NameId | Date Of Birth | Agenda |
| 0 | 13138628002 | 0 | 21-09-1979 | Male |
| 1 | 66966379 | 3 | 12-09-1980 | Male |
| 2 | 64960648 | 1 | 19-08-1949 | Female |

After further verification or confirmation, the mobile phone number is related to the user “Alan Tang”. The user “林浩天” is a human fault and needs to be deleted from the user’s table.

The final ER (Entity Relationship) diagram will be:

ER Diagram of Users Login

Users

Name

1

1

1

0 to \*

Login

## Second Normal Form (2NF)

To be second normal form, you need to satisfied the following conditions:

* **Rule 1 -** To be First Normal Form
* **Rule 2 -** Has no partial dependency. That is, all non-key attributes are fully dependent on a primary key.

In order to demonstrate second normal form. Let’s add an address field to the original Users table.

**Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Mobile phone Number | Name | Date Of Birth | Agenda | Address |
| 0 | 13138628002 | Alan Tang | 21-09-1979 | Male | Wong Tai Sin |
| 1 | 66966379 | Sing Lun | 12-09-1980 | Male | Chuk Yuen |
| 2 | 64960648 | Fun Yueng | 19-08-1949 | Female | Tusen Wan |
| 3 | 13138628002 | 林浩天 (小豬) | 05-04-2000 | Male | Wong Tai Sin |

As have talk about first normal form. For simplicity let jump directly to second normal form. In order to satisfied 2NF, it must not have partial dependency for all the non-key attributes. From the users table above, do we violate the rule? The answer is “Yes”, the “Address” field. As you can see for the user with the phone no “13138628002” has the address of “Wong Tai Sin”. That means the non-key attribute address field “Wong Tai Sin” **partial depends** on the key field mobile phone number “13138628002”. How about agenda for instance, the answer is “No”, why? Because it is **fully dependent**, it is either male or female. What is the main problem to this type of violation? The answer is address updating. If now the user of “13138628002” address changes to “China”. You will have to update all the records of the address field to “China” for the user “13138628002” correctly hopping without mistyping or misspelling issues. To handle this situation the best approach is to add a separate “Address” entity.

**Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Mobile phone Number | Name | Date Of Birth | Agenda | AddressId |
| 0 | 13138628002 | Alan Tang | 21-09-1979 | Male | 0 |
| 1 | 66966379 | Sing Lun | 12-09-1980 | Male | 1 |
| 2 | 64960648 | Fun Yueng | 19-08-1949 | Female | 2 |
| 3 | 13138628002 | 林浩天 (小豬) | 05-04-2000 | Male | 0 |

**Address**

|  |  |
| --- | --- |
| Id | Address Name |
| 0 | Wong Tai Sin |
| 1 | Chuk Yuen |
| 2 | Tusen Wan |

Now the partial Entity Relationship Diagram Becomes.

Address

Users

1

1

The overall ER Diagram Becomes:

ER Diagram Of Users Login

Name

Address

Users

1

1

1

1

Login

1

0 to \*

By the time of witting, there are totally up to 6 normal forms. Promise I tell you; you will not use all normal forms because each deal with its own condition or specification, up to second normal form is already enough. If you still want to know more about it you can click on the link below:

* [Database normalization - Wikipedia](https://en.wikipedia.org/wiki/Database_normalization)

## Many To Many Relationship

Back to the relationship, so far we have discussed “one to one”, “one to many” relationship and discussed how to deal with it. What about “many to many” relationship? Talking about my Taobao case, its seems suit the situation of “many to many” relationship, which I want to discuss here. I will use the “**Normalized**” tables, which at least satisfy up to second normal form and I will add 2 entities “Order” and “Products” entities to it.

ER Diagram of Orders Products

Products

Orders

1 to \* \*

1 to \* \*

Above is the partial ER diagram of Order Products. Let’s create some tables and feeds some data to it.

Orders

|  |  |
| --- | --- |
| Order No | ProductNo |
| A001 | P001 |
| A002 | P001 |
| A003 | P003 |
| A001 | P004 |

Products

|  |  |  |
| --- | --- | --- |
| ProductId | ProductDescription | BrandName |
| P001 | Lenovo Laptop | Lenovo |
| P002 | Microsoft Surface Laptop | Microsoft |
| P003 | Dell Laptop | Dell |
| P004 | Nvida Laptop | Nvida |

After creating the tables and feeding in the data. What is the problem with this relationship? First, I assume both tables are up to second normal form, that means the fields “OrderNo” and “ProductId” are unique. Now If I am a manager, I would like to raise up a question, by look this order “A001”, can I find what products he has purchase? The answer “Yes”, by looking his orderNo, we could find out that he has purchase items “P001” and “P004”. Now another question, by taking the product “P001”, I want to know whether this client has purchase the item “P001”, even it does not have an order number? How will you treat this situation? I think many sales will take the product no “P001” and start to scan the whole customers’ orders database until they find the order “A001” and check the order list to see if they can find this item “P001”. You see that’s the problem of “many to many”. How to treat this situation? The solution is when dealing “many to many”, break up the relationship to “one to many” by adding a intermedia table between it.

ER Diagram Of User Orders Products

Orders

Products

UserOrderProduct

Users

As discussed, when dealing with “many to many” relationship, you need to break up the relationship to “one to many” by adding an intermedia table between it. Now we have added an additional table “UserOrderProduct”, let feed in some data to it.

UserOrderProduct

|  |  |  |
| --- | --- | --- |
| UserId | OrderNo | ProductNo |
| 0 | A001 | P001 |
| 1 | A001 | P001 |
| 2 | A003 | P003 |
| 0 | A001 | P004 |
| 0 | A002 | P001 |

The Complete ER Diagram

Products

Orders

Name

UserOrderProduct

Users

Login

Address