**Mohammed Mahin Ibnay Mamun (346584)**

**Unit 18 Task 1**

**Evaluate the role of relational data structures:**

## **Relation**

The term database in computing can be defined as a well-ordered collection of electronically accessed and stored data. Small databases are found in a file system in contrast, databases which are larger are hosted on cloud storage or on clusters of the computer. Definition of relation in this context (relational data structure) is an entire table on a database.

Many times, a relation is found on each table. A database consists of multiple categories but sometimes only one. This is found in Attributes Aswell as in columns. The rows in the table can be defined as a tuple if not record. These are built with a unique piece of data like a key.

Advantages

* Easy to use
* Less duplicate values
* Secure

Disadvantages

* Cost
* Problems in maintenance
* Lack of capacity and space

## **Attribute**

Attribute is the term given for a certain index of data which is used to narrate a piece of entity. A characteristic is also an attribute. This is typically seen in tables of data. Attributes can also be found in a database field; this refers to an individual cell in a table using rows and columns. There are 4 common types of attributes, which are:

Multivalued attributes – this type of attribute accepts more than one piece of data. An explain of Multivalued attributes is if somebody has more than 1 contact number or email address.

Key attribute – is a key, number, or character that which is used to specify an item. Number of sales can be an example of this. A strong entity is created when a table uses only one key attribute. Nevertheless, most tables will consist of more than one key attribute.

Composite attribute – this is when one attribute is used to build up multiple other attributes. A notable example of this can be address because it will take 2 attributes. Number and street name.

(Source of research: [What is an Attribute in Database Systems? - Definition from Techopedia](https://www.techopedia.com/definition/1164/attribute-database-systems))

## **Domain**

A set of values which have the capability to be stored into a table via a column is a defined as a database domain. The domain can also be used to show the values in a row if it is displayed in the form of a table. integers, Booleans, characters, and other data types can all be stored as a domain. As a user, you can input simple of complex domains. Along as they meet the data type requirements, they will be valid.

Domains can later be amended using various techniques. A few common examples incorporate modifying, deletion, and insertion, these methods will be included below in this document with a more in-depth explanation.

If a user was intended to generate a domain, then a command line will need to be created and ran. As we are adding to a database, SQL will be necessary. Below is an example which I have created with a description to help you understand.

CREATE DOMAIN E\_Age INT (2) NOT NULL;

The above line of text can be run as a command in SQL for a database. It starts off with create domain, this is a simple function. Then I have wrote E\_Age which is an attribute followed by an integer data type vale with 2 digits. This means E\_Age can have a 2-digit value as it stands for employee age in this example. Finally, NOT NULL is a function which is used to return a value if it does not meet the requirements for this example a 2-digit integer value. If user types in characters, then null function will be operated.

CREATE DOMAIN E\_Present BOOLEAN(1) NOT NULL;

Here is another example, the only difference is the name of the attribute, data type and the number of digits. This time the user can only input Boolean values which are: (yes/no, true / false, 0/1 or y/n). Because the limit has been set to 1, user will input either 0/1 or they will input y/n as they are one character. If the requirement is not met, then the input will be returned.

The following line below is another important aspect of domains in SQL. This line is used to check the input which is assigned by the user. This can also be referred to as a filter.

CREATE DOMAIN E\_Code INT (4) NOT NULL CHECK (VALUE > 0);

[Domain in DBMS - Scaler Topics](https://www.scaler.com/topics/domain-in-dbms/)

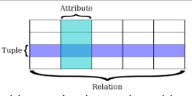
## **Tuple**

In relation databased, the expression tuple is a singular row which is used to show a unique individual record. Tuples are remarkably like lists, and they are sorted. However, do not allow changes from users. The input for a tuple might be data or even parameters.

[What is a Tuple (Database)? - Definition from Techopedia](https://www.techopedia.com/definition/1251/tuple-database)

[What Is A Tuple In A Table](https://www.knowledgewow.com/what-is-a-tuple-in-a-table-of-database/) Of Database? - Knowledge WOW

Below is a screenshot to show an example of what these will look like. As you can see in the screenshot below, a table is displayed and there are 3 specific areas highlighted these are: attribute, tuple, and relation. This image clearly shows us that tuples in simple term is referred to as a row.



There are two types of tuples:

* Logic tuples

This type of tuple is the data represented in memory. This source of storage for data is only temporary kept in advance of it being written onto a disk.

* Physical tuples

On the other hand, physical tuples are slightly different. In contrast to logical tuples which is only temporary store, this type of tuple is the confirmed data which has been placed in a database.

[Tuple in DBMS - GeeksforGeeks](https://www.geeksforgeeks.org/tuple-in-dbms/)

## **Degree**

In a database, the term "degree" simply refers to the number of columns that are present. You will see a sizable number of attributes as well if your database table has a lot of degree information. Understanding the properties allows us to determine the degree of the table. In contrast, the degree would be low if a table has only a few features, which is done to keep the table simple.

Being able to understand the significance of a database degree as humans is crucial since it enables one to realise how much data can be kept. As was already indicated, if the degree is low, the amount of data we can contribute is also low. On the other side, a high degree will produce more data, which might be seen as a disadvantage because it will entangle the table.

Personally, a low degree table is more efficient; it should only contain the information that is necessary.

## **Cardinality**

The link between two tables is crucial in relational databases. This demonstrates how entities in several tables are comparable to one another. This method can be helpful; for instance, if you wanted to create a new table using the same data, it can be a quick and effective approach to ensure that it is accurate.

There are three crucial elements that together demonstrate the significance of cardinality:

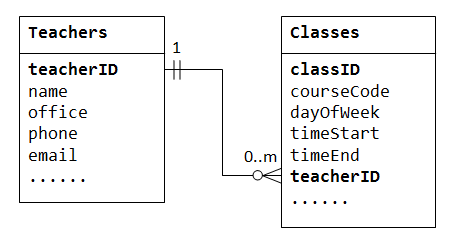
* Cardinality is necessary for normalisation. Utilising normalisation techniques aims to minimise repetition and errors while maintaining simplicity and clarity. The degree of normalisation is influenced by the cardinality.
* Data sorting is carried done through data modelling. Additionally, this is required for cardinality. Additionally, it is helpful for database storage.
* Data integrity is a crucial factor since it establishes if two tables' entities are connected to one another. The number of records in each table is then checked by Cardinality to see whether any matches exist.

## **One-to-one**

One to one relationship is common in databases. One of the simplest linkages, two tables are needed. A one-to-one link exists if a given piece of information can also be found on the other table. Here are some examples of one-to-one relationships, which are common.

* Email + account address
* Town + capital city
* Age + date of birth

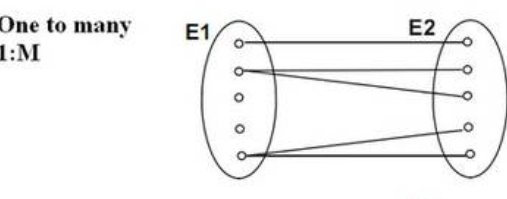
An er diagram is a simple tool for demonstrating this approach. This is made up of two tables with a line connecting them.



[What Is a One-to-One Relationship in a Database? | Vertabelo Database Modeler](https://www.vertabelo.com/blog/one-to-one-relationship-in-database/)

**One-to-many**

A record or records in another table are associated to every record in the first table. One consumer, for instance, might place numerous orders. Primary key relationship is another name for a one-to-many relationship.

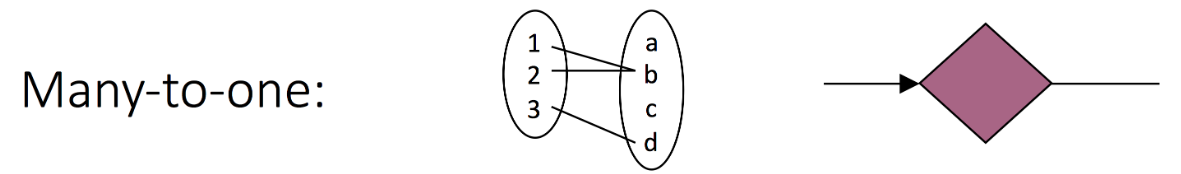


Consider the relationship between a teacher and the classes they teach as a practical illustration of this. Although a teacher may teach multiple courses, there will be differences in the interactions between the teachers and the courses.

Divide the tables into two, then use a foreign key to link them together to form a one-to-many relationship. This is the procedure that will be used to create a one-to-many relationship.

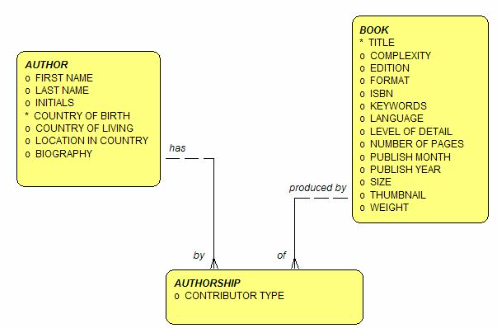
**Many-to-one.**

There is a direct relationship between every record in one table and every record in the other. For instance, there can only be one customer per order. This phrase is in the same vein as the one before it, which is called one to many. The perspective is what distinguishes this, though. A many-to-one relationship, on the other hand, allows for the linking of numerous rows in the local database to a single row in another table. Below is a diagram showing how this can be drew out. From looking at the image provided, you can tell it is extremely like one to many.



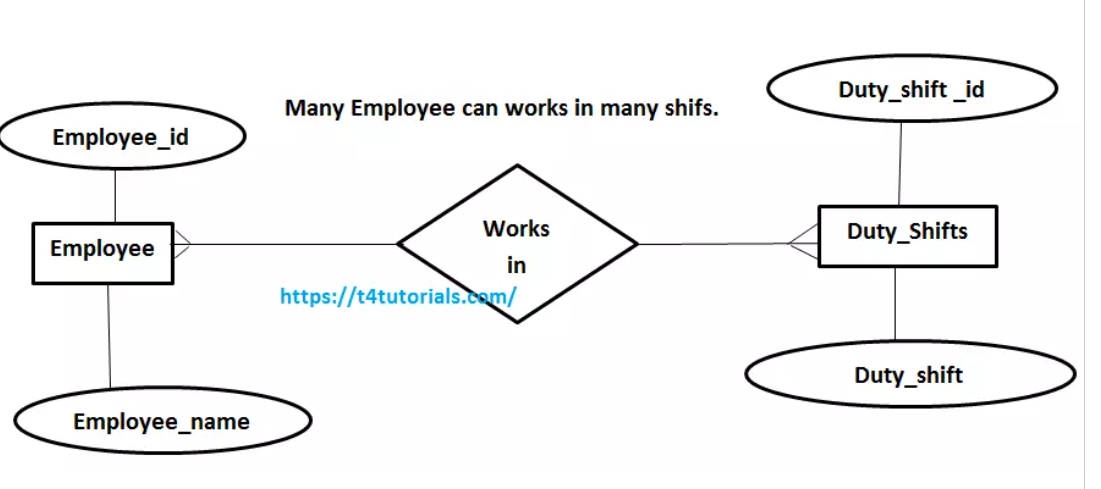
**Many-to-many.**

Every record in one table has a connection to at least one or more records in the other table, and vice versa. For instance, any consumer may place numerous orders, each including a variety of products.



A many-to-many relationship exists when several records from one table are linked to multiple records from another table. Reflecting such a relationship in the database can be difficult.

listed below is an illustration of this method found online. Employees and duty shifts make up the entities. The following are the employee's attributes: employee\_id and employee\_name. The properties for the second object, duty shift, also include duty\_shift\_id and duty\_shift.



After this stage where the diagram is created, a table can be made. For example, I will create one below.

|  |  |
| --- | --- |
| Employee\_id | Employee\_name |
| EID0 | callumn |
| EID1 | jack |
| EID2 | rick |
| EID3 | Marvin |

Then, this list splits again into smaller sub tables. Here is an example.

|  |  |
| --- | --- |
| Employee\_id | Duty\_Shift \_id |
| EID0 | DS0 |
| EID1 | DS1 |

(Source of research: [Relational Data Modeling - Cardinality | Modeling | Datacadamia - Data and Co](https://datacadamia.com/data/type/relation/modeling/cardinality))

## **Entity relationships models (overview)**

Entity relationship models might be used to illustrate how entities in a database are related to one another. The following are the top three categories of relationships between entities:

* One-to-one: In databases, one of the relationships is one to one. This is one of the most basic links, 2 tables are required. If one piece of data is also found on the other table, this is a one-to-one relationship.
* One-to-many: Each record in one table is related to one or more records in the other table. For example, one customer can have many orders. A one to many is also known as a primary key relationship.
* Many-to-many: Each record in one table is related to one or more records in the other table, and vice versa. This also appeals to when many instances of one entity are connected to multiple instances of the other thing, there is this kind of link between the two entities.

## **Relational algebra sets, symbols**

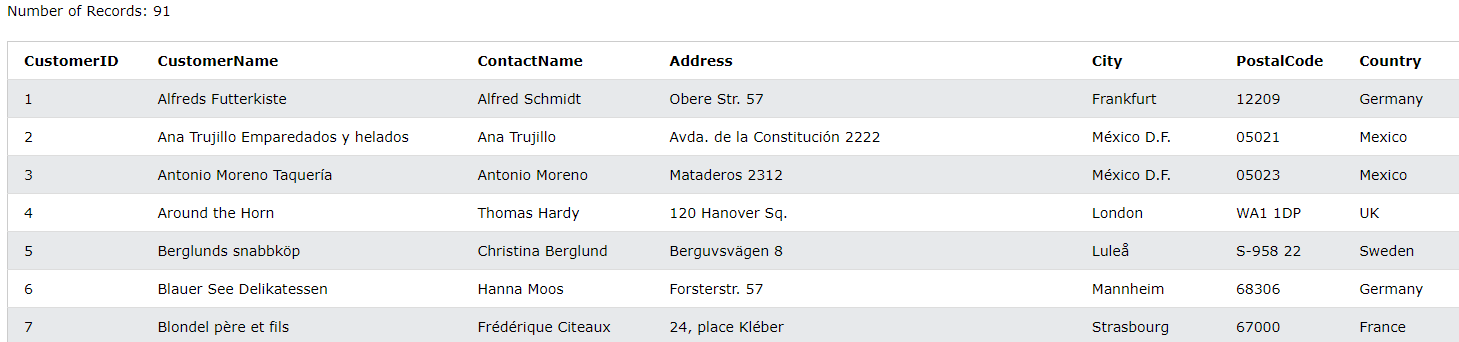
in relational algebra, we use both symbols and sets. I have listed below some examples:

* Sets: unique elements are referred to as a set. Users of relation databases use sets to show tables.
* Projection (π): Projection is represented by the symbol π its purpose is to remove existing columns from a table.
* Selection (σ): To show selection we use the symbol σ. The use of this symbol is for the user to highlight selected rows in the table.
* Cartesian Product (X): The X symbol presents cartesian product. The role of this symbol is to join two or more tables into a single table by connecting rows from one table to the other.
* Union (U): Union is represented by the symbol U and is used to combine two tables into a single table. This is like cartesian product however, only specific rows will be joined.
* Intersection (∩): Intersection is represented by the symbol ∩ and is used to retrieve the rows that exist in both of two tables.
* Difference (-): The – symbol is used to show difference. The usage of this is to subtract one table from another.
* Join (⋈): ⋈ is the symbol used to show join. This symbol is used to combine two or more tables based on a common column.

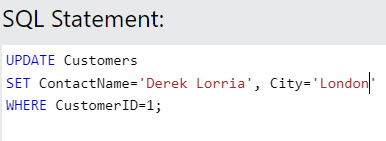
**Evaluate how data is manipulated in data structures and relational databases when:**

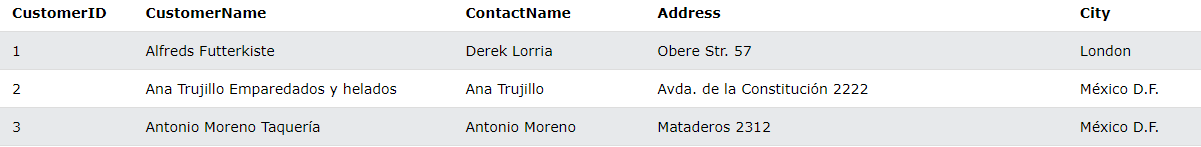
**Updating**

You will also need to modify the same value in the other tab if you chose to update the data in one table. This occurs when it is necessary to update the same data across numerous tables and the tables have not been appropriately normalised.



## **After**

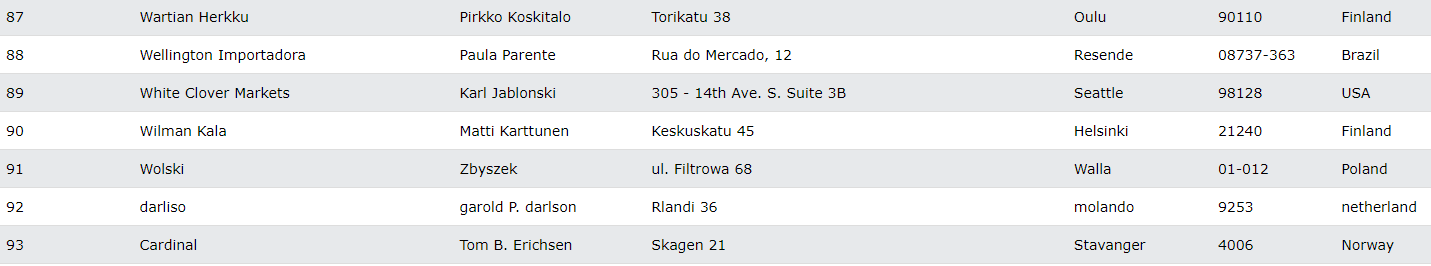




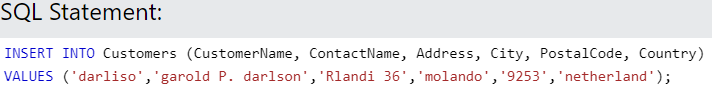
**Inserting**

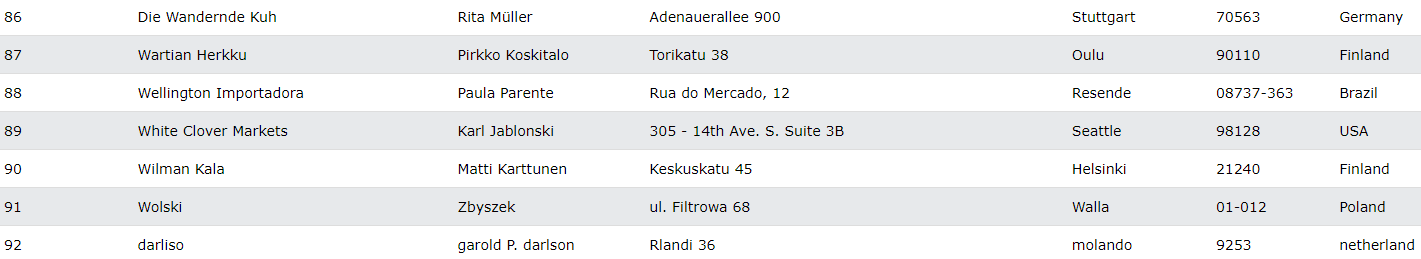
These happen when it is impossible to add data to one table without first adding data to another. This occurs when there is a dependency between two tables and the tables are not correctly normalised.

**Before**



**after**





**Modifying**

It is crucial to ensure sure the data is consistently accurate and consistent while changing it in a database. Relational databases use constraints and rules to maintain the integrity of the data. Examples of this kind of restriction include the database's primary key constraints, which prevent data from being removed or changed in ways that go against the logical relationship between tables.

The technique modification can be built up in 3 parts, these are:

* Inserting tuples
* Deleting tuples
* Updating existing tuples

Advantages

* Changes can be made
* Easy to fix errors
* Tabe does not need to be made again

Disadvantages

* May need editing rights
* May need to request permission

**Deletion**

Only when data is unintentionally lost because of record deletion does deletion take place. This occurs when a record that was deleted included only one instance of a specific piece of information that was connected to other information in the database.

Here is an example of how this code will look. By running this code, deletion will be made to the database.

DELETE FROM receipe  
WHERE type = ‘standard';

If you wanted to delete all the rows from your table, then you would type and run the following command.

DELETE FROM mytable;

Facts:

* Delete in SQL is a DML command. (Data manipulation language)
* Backroll command is useful to undo any deletion
* You can limit the amount you want to delete

## **Retrieval of data for queries and reports**

In databases, extracting and identifying data from a database is the expression for the term data retrieval. The use of this technique is to allow users to grab information from a database and use it in an application.

A database will need to run commands for executing and writing data for data retrieval to be held. The information which has been requested is the data which gets retrieved. Both software's and applications use queries to restore data from a database.

In the context of databases, the commonly used term queries simply mean to request for data. If an individual were to access, delete, retrieve, or manipulate from a database they would need to use a query in the database management.

## **Administration of users**

This heading can also be read as user management. This means that the administrator has access to manage how the rest of the team can access the database and what features within the database they can access. The administrator has access to adding users to the database Aswell as blocking and removing users from managing the database.

Below are some listed responsibilities which the administration has access to:

* Removing users
* Adding users
* Editing permission for users
* Changing user passwords
* Managing accounts of users

## **Security**

To keep a database, secure the following must be protected:

* The information which is stored inside the database
* The entire system which holds the database
* Hardware, software, and other applications which are linked to the table
* Network joint to the database

Before, database security was not taken as seriously as it is now. This is due to potential severity of a database compromise. There are many new ways for information to get stolen, whether it be from outsider or even your own people. By holding a database, essential information is stored, this should be backed up and only given access to few and reliable people to keep data safe.

Databases may contain sensitive information just as sensitive as credit card numbers, passwords, API keys, personally identifiable information and many other data which should not be shared. By reducing access to the tables will result in less chance of harm.

## **Integrity**

Mechanisms are in place to ensure that data in use, data in transit, and data at rest cannot be altered by an unauthorised person or programme when data has integrity. Making sure that recovered data can be trusted after an interruption is a key objective of maintaining data integrity.

Formatting mistakes, syntax mistakes, malware, ransomware assaults, and other harmful behaviours are frequent causes of data integrity breaches.

Entity integrity ensures that no data is redundant, and no fields are empty. This is related to the idea of primary keys. According to the requirement, every table needs to have a unique primary key that is not null.

Referential integrity guarantee that data is stored and used consistently, all policies and guidelines must be followed. This is how foreign keys work.

## **Recovery**

In database management systems, database recovery procedures are used to return a database to a consistent condition following a failure or error. The basic objectives of recovery strategies are to guarantee data consistency and integrity and stop data loss.

The rollback/undo recover methodology is based on the idea of reversing the results of a transaction that failed to complete owing to a bug or fault in the system. Using the log records saved in the transaction log, this method reverses the modifications performed by the transaction.

The idea behind the commit/redo recovery procedure is to apply the database changes made by an operation again after it has successfully completed. This method is carried out by re-executing the modifications made by the process that was active at the time of the interruption or error utilising the log records that are saved in the transaction log. The database is returned to its most recently modified stable state by the system using the log records for applying the transaction's changes.

**What is normalisation?**

Normalisation is a technique which we use in relational databases to assist in reducing duplicated data and redundancy. This technique also simplifies in data integrity improved. By using normalisation techniques in your database, there are a few benefits such as:

* Less data redundancy
* Data is simplified
* Tables are shorter
* Searching and sorting is quicker
* Less empty values (null)

**Evaluate the various stages of normalisation, including:**

## **Un-normalised form (UNF)**

During the normalisation stages, UNF which can also be defined as 0NF is the first stage. After this from is created, first normal form (1NF) can then take place. Users who are experienced in working with forms in normalisation may tend to skip out this stage and start at 1nf but for those users who are new to database normalisation, this will be the first stage for them.

Below is an example of un-normalised forms which I have created:

The example below shows 2 columns. 1 column is defined as company and the other is referred to as address of store. Below these columns are different values. The company is assigned to the address of the store beside it. For example, ASDA is assigned to 14 ark road. ASDA is the name of the company and 14 ark road is defined as the address of store.

|  |  |
| --- | --- |
| company | Address of store |
| ASDA | 14 ark road |
| LIDL | 1 gold road,  3 hull road |
| TESCO | 71 church road |
| SAINSBURY | 9 willow road,  4 can road |

As you can see in this stage, there are field with more than one value. In the next stage, this will be separated.

## **First normal form (1NF)**

During the first normal form stage, the main concept of this stage is to separate the database. By doing this, each value should be in one field. As the example above shows, some fields consist of more than one value. Below is a table I have designed to show this.

Before:

|  |  |
| --- | --- |
| company | Address of store |
| ASDA | 14 ark road |
| LIDL | 1 gold road,  3 hull road |
| TESCO | 71 church road |
| SAINSBURY | 9 willow road,  4 can road |

After:

|  |  |
| --- | --- |
| company | Address of store |
| ASDA | 14 ark road |
| LIDL | 1 gold road |
| LIDL | 3 hull road |
| TESCO | 71 church road |
| SAINSBURY | 9 willow road |
| SAINSBURY | 4 can road |

KEY:

VALUES HIGHLIGHTED YELLOW SHOWS VALUES MOVED AROUND

FIELDS HIGHLIGHTED GREEN SHOW NEW FIELDS ADDED

The updated table is now ready for the next stage.

## **Second normal form (2NF)**

After the 1nf, the 2nf begins implementing. As you can see the biggest change which we can see here is more columns.

Highlighted in green are the new columns and fields.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product  Id | Product name | Product type | Quantity | company | Address of store |
| faa1 | apple | fruit | 22 | ASDA | 14 ark road |
| fpl1 | pear | fruit | 15 | LIDL | 1 gold road,  3 hull road |
| fpt1 | apple | fruit | 13 | TESCO | 71 church road |
| fbs1 | banana | fruit | 9 | SAINSBURY | 9 willow road,  4 can road |

After the table has been modified, to 2nf, it can move over to the final stage or normalision which is the third normal form.

## **Third normal form (3NF)**

First stage:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product  Id | Quantity | Company | Address of store | ManId |
| faa1 | 22 | ASDA | 14 ark road | A14a |
| fpl1 | 15 | LIDL | 1 gold road,  3 hull road | L1g,  L3h |
| fpt1 | 13 | TESCO | 71 church road | T71c |
| fbs1 | 9 | SAINSBURY | 9 willow road,  4 can road | S9w,  S4c |

Final stage:

|  |  |  |
| --- | --- | --- |
| Product  Id | Quantity | ManId |
| faa1 | 22 | A14a |
| fpl1 | 15 | L1g,  L3h |
| fpt1 | 13 | T71c |
| fbs1 | 9 | S9w,  S4c |

As you can see, the database table becomes a lot shortened. In not every example will use quality therefore, you can create a database table with normalisation with as little as 2 columns.

## **Boyce-Codd normal form (BCNF).**

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

The table should be displayed in 3NF

A -> B, A should be a super key.

A -> B. this expression displays that A will not take the place of a non-prime attribute if the character B is a prime attribute

## **Anomalies (update, insertion, deletion)**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |
| tom | 24 |

Above I have created a table which contains two aspects of data, name, and age. There are 4 entities (john, tom, 21 and 24). Below I will give the term and importance for the 3 types of anomalies which I will be showing with examples.

Update: an update anomaly is carried out when we make an update to a row on our table. This is important because we can very easily make a spelling error and we may need to change data over time. If update anomaly did not exist, then we would have to create a new table each time we wanted to make a change which can be seen as very inconvenient.

**BEFORE**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |
| tom | 24 |

**AFTER**

|  |  |
| --- | --- |
| Name | Age |
| john | 22 |
| tom | 24 |

In this example, the field has changed value within the same data type. Because no new fields were inserted into the table nor where any deleted, this example is updating anomaly.

Insertion: When an individual inserts a new row to an existing table, this phrase is used.

**BEFORE**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |
| tom | 24 |

**AFTER**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |
| tom | 24 |
| Larry | 23 |

The example above shows a clear before and after of insertion, a new row has been added with values to the bottom of the table.

Deletion: in a database, when the user decides to remove a row from the corresponding table, deletion anomaly is carried out.

**BEFORE**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |
| tom | 24 |

**AFTER**

|  |  |
| --- | --- |
| Name | Age |
| john | 21 |

As you can see in the example above, one of the rows from the first table has been removed. This was done by using deletion technique.

Source:

[Anomalies in DBMS - javatpoint](https://www.javatpoint.com/anomalies-in-dbms)

[Anomalies - Introducing databases - Higher Computing Science Revision - BBC Bitesize](https://www.bbc.co.uk/bitesize/guides/ztsvb9q/revision/2)

## **Primary keys**

The primary key can be is used to show each record across a database table. A candidate key is a common name given for any individual key alone. However, throughout the entire table, there can only be one primary key.

* A table can only have one main key.
* One or more columns make up the primary key.
* The table's entity integrity is protected by the primary key.
* Every defined column must be defined as NOT NULL.
* A row is uniquely identified by its primary key.

Data consistency maintenance is a crucial responsibility. Due to its unique constraint, primary keys help prevent duplicate records, one of the most prevalent data consistency issues. A new record will be refused if we attempt to insert it using an existing value for the primary key.

The database builds an index for that table whenever a primary key is created for it. Similar columns to the main keys are used to create the index key. The construction of the main key includes an implied generation of this index. To prevent the duplication of primary key values, some databases create PK indexes as unique key indexes.

**foreign keys**

A foreign key is slightly less common than a primary key. A foreign key is when there is a reference in a linked table. the main purpose of this key is to create a link by getting data from another table and inserting into this one.

Assume we have a database containing information about countries included in a table titled country. The biggest cities of the world should also be represented in this database. At first, we consider building a single table called city to record the name of the city, a numerical city\_id, and the nation in which it is situated. We add a column named country\_id that points to a record in the country database for this final property. As a result, the name of the country in which the city is located, which makes up a portion of the city information, is saved in the country table. These tables have a connection, as we can see.

A database constraint that connects two tables is known as a foreign key constraint. A primary key or unique constraint on the nation table is required before we can construct a foreign key on the city table.

**composite keys**

The term composite key is best described as, a key in which composes multiple attributes to describe each record.

When two or more columns in a table are merged, a composite key is produced; when the columns are taken separately, however, a row's uniqueness is not ensured. It can also be viewed as a primary key that identifies each row in a database individually by fusing two or more properties.

* Composite keys cannot be left empty
* Combination of multiple keys

## **Indexing**

Databases include indexing. The definition behind indexing can be expressed as: using minimal disk access to optimise performance within a database. The overall structure of indexing is first represented as 2 parts. Search key and data reference.

The search key has a backup of the primary key Aswell as candidate of the database table. They are well arranged which benefits in finding them easily. The reason for this is because they are sorted in a list. This can be numerically or alphabetically. However, it is not necessary for the data to be stored nevertheless it can be more convenient.

The next part of indexing has 2 abbreviations, these are “pointer” and better know “data reference”. The function given to this is to carry the disk block address which leads to the key values.

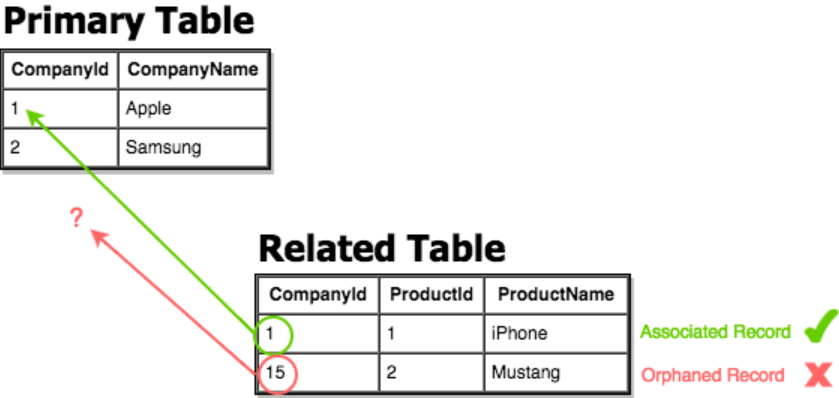
A few attributes given in index are:

* Access type - which describes the access type, including access time and range and value searches.
* Access time - the same as access type, such as value searches and range access.
* Insertion time - Insertion time indicates the amount of time needed to place data correctly.
* Deletion time – finding data and removing it
* Space overhead – overall extra space

Source: [Indexing in Databases | Set 1 - GeeksforGeeks](https://www.geeksforgeeks.org/indexing-in-databases-set-1/)

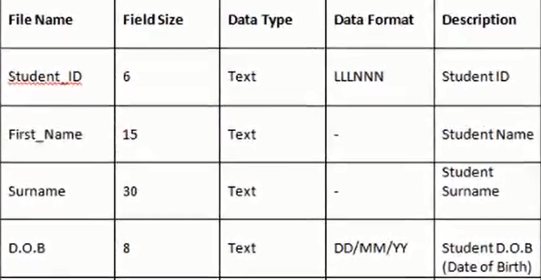
## **Referential integrity**

The simple meaning of this term is the relationship from one table to another. A primary key is required for any table in a relational database and can appear in various other tables because of its connection to the data in those other tables. Foreign keys are used when a primary key from a particular table appears in another.



## **Data dictionary**

In a computer system, database, or as a component of a research project, data elements are used or recorded. A data dictionary is an array of terms, definitions, and characteristics about those data elements.



Advantages of using a data dictionary:

* Represents clear information
* Helpful for working members or groups
* Allow a lot of data to be stored

Disadvantages of using a data dictionary:

* Hard to share between software's
* Can be hard to read
* May cause confusions

Below is an example which I have created.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **fields** | **Data type** | **Size / format** | **Validation** | **Input mask** | **note** |
| **Assessor details** | Assessor name | string | Easy to read text | No letters or symbols | MMIM028 |  |
| Assessor ID | Short string | integer | Is not null |  | PK |
| Assessor Address | Long string |  | Min 1 space |  |  |
| Post code | string | 7 | 6 – 8 characters | \*\*\*\*’ ‘\*\*\* |  |
| Assessor mobile number | Multiple Integer string | integer | 11 int values | 00000000000 |  |
| Assessor email address | Long String |  | Must include '@'  character | \*\*’\*@ |  |
| Course Details | Course name | string | Easy to read text | Is not null |  |  |
| Course ID | string | integer | Is not null | CS9032 | PK |
| Course cost | Int value | 2 d.p | Symbol and int values | £0,000 - £00,000 |  |
| Venue  details | Venue name | string | Easy to read text | No int values |  |  |
| Venue ID | string | integer | Is not null | 25 | PK |
| venue address | Long string |  | Must include 1 @ symbol | \*\*’\*@ |  |
| post code | string | 7 | Must include 1 space | \*\*\*\*’ ‘\*\*\* |  |
| venue mobile number | Multiple Integer string | 11 | No alphabet letters | 0000 000 0000 |  |
| manager name | string | Easy to read text | No int values |  |  |
| venue email | Long string |  | Must include 1 @ symbol | \*\*’\*@ |  |
| address | Long string |  | Must include 1 space |  |  |
| venue cost | Int | integer | Currency symbol and int values | £0,000-£000,000 |  |
| google map link | Long string | hyperlink |  |  |  |
| Course schedule details | course date | String with int | dd/mm/yy | Is not null | dd/mm/yy |  |
| course duration | Time / date | HH:MM dd/mm/yy  -  HH:MM dd/mm/yy | Is not null | HH:MM dd/mm/yy  -  HH:MM dd/mm/yy |  |
| achieved | Boolean | Y/N or YES/NO | Boolean output | Y/N or YES/NO |  |
| Student details | student name | string | Easy to read text | No int values |  |  |
| student id | string | integer | Is not null | 0000000 | FK |
| address | Long string |  | Must include 1 space |  |  |
| post code | Long string | 7 | Must include 1 space | \*\*\*\*’ ‘\*\*\* |  |
| email address | Long string |  | Must include 1 @ symbol | \*\*’\*@ |  |
| student mobile number | Multiple Integer string | 11 | No alphabet letters | 0000000000 |  |

## **Cascading update, deletion techniques**

The third alternative in the case of a referential integrity constraint violation is cascading. If the assertion that defies the restriction is:

* Cascading involves adding the foreign key into the primary key column during an insert or update.

* When you delete something cascading, you also delete any foreign keys that match the primary key in addition to the primary key itself.

When we use the delete cascade option to generate a foreign key, when the referenced row is destroyed in the parent table that contains a primary key, it also deletes the referencing rows in the child table.

update cascade: The reference rows in the child table are updated whenever the referred row in the parent table with the main key is updated when a foreign key is generated using update cascade.

**unions**

A single result that includes the rows provided by both queries is created by joining the results of two subqueries together using the UNION function.

To merge the result-set of two or more select statements, we can use the any of the 3 following union operators.

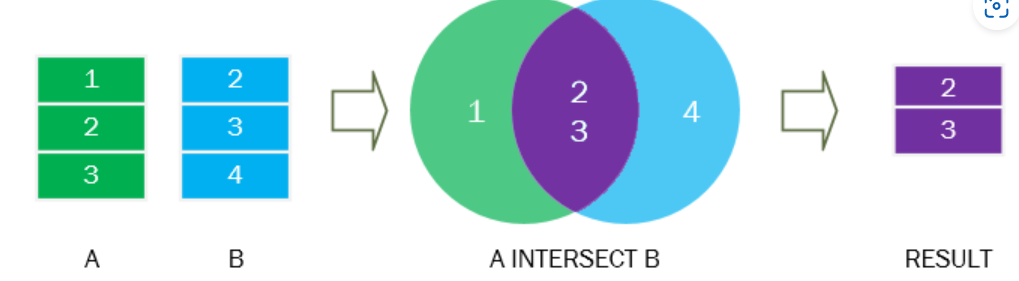
* Within UNION, all SELECT statements must have an identical number of columns.

* Similar data types must also be present in the columns.

* Every SELECT statement's columns must be in the same order.

**Intersects**

With the INTERSECT procedure, all the rows shared by both queries' results are combined into a single result. INTERSECT is a logical AND operation as opposed to UNION, which is a logical OR.



The junction of the green and blue result sets is represented by the purple area.

The INTERSECT operator eliminates the duplicate rows from the result set, just like the UNION operator does.

**Joins**

Queries could simultaneously access several tables or access the same table while simultaneously processing multiple rows of the table. A join query is one that simultaneously reads many rows from the same or other tables.

Inner join stipulations

When two tables are joined internally, the values in their shared columns are used to match rows from the two tables. The most typical kind of joins are inner joins.

Queries with a left outer join

All the rows from the left table are returned by a left-outer join query, not only the ones where the joined column matches.

queries that self-join

A table is joined against itself by a self-join query.