# Relational Databases

**Relational Database**

Is where data is stored in separate tables that are linked through the use of primary and foreign keys. Each table stores data about a single entity.

**Flat File Database**

Single table which stores all the data held within a database.

**Primary Key**

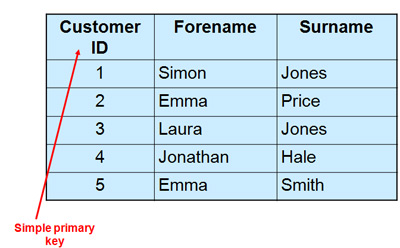
Is a field within a database which enables every record to be uniquely identified.

**Foreign Key**

A field in a table that is a primary key field in another table.

**Secondary Key**

Used to locate one (or more) records - it is different from the primary key because a primary key is guaranteed to return a single record whereas a secondary key may return more than one record.



An entity is a thing that can be distinctly different. Ie STUDENT, SUBJECT, TEACHER.

An ERD is a planning tool for designing the structure of a relational database.

There are 3 types of relationships:

- One-to-one.

- One-to-many.

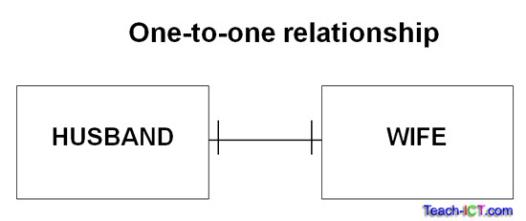
- Many-to-many.

One to one

- Uncommon in a relational databse.

- EMPLOYEE to NIN

- HUSBAND to WIFE

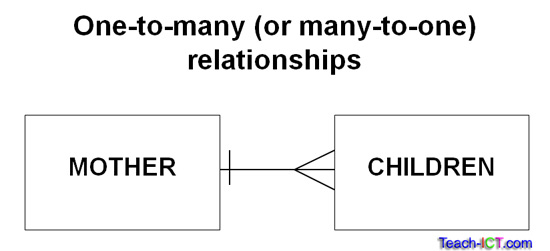


One to Many

- Most common type.

- LIBRARY MEMBER to LOAN.

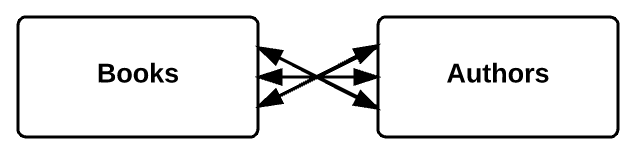
- CUSTOMER to ORDER.



Many to Many

- Many to many relationships should not exist in an efficient relational database.

- If you have many-to-many relationships a link must be added.



This shouldn’t exist.

**Referential Integrity**

- Making sure that you cannot delete records that are used by other tables.

**Normalisation**

- Process used to design relational databases in an efficient way is called **normalisation.**

- There are different forms of normalisation.

- First Normal Form (1NF)

- Second Normal Form (2NF)

- Third Normal Form (3NF)

- There are more but they’re not on the course.

- There are **specific rules** for each normal form I must know.

**First Normal Form (1NF)**

1. No columns with repeated or similar data.
2. Each data item is **atomic.**
3. Each row in every table has a primary key.
4. Each field has a unique name.

**Second Normal Form (2NF)**

1. Data must be in 1NF.
2. Non-key attributes must depend on every part of the primary key.

**Third Normal Form (3NF)**

1. Data must be in 2NF.
2. No non-key attributes that depend on another non-key attribute.

**Atomic** data is data that cannot be broken down any further. Full name is not atomic, first name and second name is.

**Benefits of Normalisation**

- Database does not have redundant data, it is smaller in size so less money needs to be spent on storage.

- Less data to search through so it is much faster to run a query on the data.

- No data duplication so better data integrity and less risk of mistakes.

- No data duplication so less chance of storing two or more copies of the data.

- One change can be made which can be cascaded across any related records.

**Problems With Normalisation**

- You need to be careful with making data atomic. Some data shouldn’t be split for example phone number: 01203 203032.

- You can end up with more tables than an unnormalised database/

- The more tables and the more complex a database, the slower queries can be to run.

- It is necessary to assign more relationships to interact with larger numbers of tables.

- With more tables, setting up queries can become more complex.

**Queries and Reports**

- A **query** is used to extract a **subset** of the data in a database.

- Queries can combine data from more than one table and present data in whatever order is required.

- Queries can be used to perform calculation.

- The results of a **query** can be presented in a **report.**

**- Reports** are used to present information in a user friendly format - they can be set up to summarise, group and select data.

**Queries**

- Parameter queries are used in databases to search for information that meets certain criteria.

- The parameter is the criteria used to select records.

- Static parameter queries have the parameter hard coded.

- Dynamic parameter queries ask the user for the value to search for by creating a dialogue box for them to enter a value.

- This makes the query more flexible and saves having create a large number of hard coded queries.

**DBMS (Database Management System)**

- Software that creates and maintains a database.

- Jobs include the creation and use of:

- database structure.

- queries.

- views.

- individual tables.

- interfaces.

- outputs.

**Database Views**

- Data held in a database can be envisaged at three levels or ‘views’.

**Physical View**

- How the data is actually recorded in the storage medium. This level of understanding needs to be achieved by software so that storage is correctly managed but not necessarily by the user.

**Logical View**

- Concerned with how the data is organised for processing.

- Looks at the construction of tables, queries, reports and software that will deliver functionality to the owners of the system.

- Constructing this level involves the creation of a **data dictionary.**

**User View**

- About appearance and functionality of the database. The user is not concerned about the structure of the database just needs to access whatever data they need for their job.

**Data Dictionary:** metadata; data about data. Will contain all the information about the structure of the database including the relationships, fields and tables.

**Transaction Processing**

- Type of processing that attempts to provide a response to a user within a quick time frame.

- Not as time critical as RTS and normally has a limited range of operations planned in advance such as bank balance or withdrawal.

**CRUD**

All databases must have certain basic functionality to be useful. This is remembered by the acronym CRUD.

**C -** Create

**R** - Read

**U -** Update

**D -** Delete

Which are mirrored by the SQL statements:

- INSERT/CREATE

- SELECT

- UPDATE

- DELETE

Three of these result in a transaction taking place.

Transactions must not allow a database to become damaged.

**Data Integrity:** The maintenance of a state of consistency in a data store. This means the data in the store accurately reflects the reality it is trying to represent. Data is as intended and fit for purpose.

**Data Corruption:** Opposite.

**Data Security:** Keeping data safe.

**Referential Integrity:** Part of Data Integrity refers to a state where inconsistent transactions are not possible.

**The ACID Rules**

Set of rules to protect the integrity of a database. Remembered by acronym ACID.

**Atomicity:** Change in a database is either completed fully or not at all. Software must prevent a half finished transaction being made.

**Consistency:** A transaction must take the database from one consistent state to another.

**Isolation:** Concurrent execution of transactions has the same result as if the who transactions were executed sequentially.

**Durability:** Once one transaction has been committed it will remain so, even after power loss, crashes or errors. Must be recorded in non-volatile memory.

**Record Locking**

- Record locking is the technique of preventing simultaneous access to data in a database, to prevent inconsistent results.

**Redundancy**

- Redundant data should be avoided in databases to save space.

**SQL**

- A database can have millions of records in it. There must be a way to interrogate a database to extract information from it and also an efficient way to add or change data.

- The answer is the development of a standard database language **structured query language (SQL).**

- Language is “engli9sh-like” and as such is easy to learn.

- A statement in SQL is called a **query.** Queries are formed from the built in commands in SQL.

CREATE TABLE – create an empty table.

DROP – delete a table or even an entire database.

INSERT – insert a row into a table.

UPDATE – alter data in a row.

SELECT – retrieve data from the database, more than one table can be accessed in the query.

WHERE – sets up one or more conditions to filter the query.

AND – allows more than one condition to be included.

FROM – identifies the table to be used.

Example:

CREATE TABLE STUDENTS (`ID` INT(11), `Surname` TEXT, `OtherNames` TEXT, `AGE` INT(3))

- ID is 11 digits.

- Age is 3 digits.

- Other fields are text.

UPDATE `student` SET `Age`=15 WHERE `ID`=103

changes id 103’s age to 15.