## ICT @ Brooke Weston Academy

Unit 23 (LO1) – Database Design

**Database Notes**

***Database***

A shared collection of logically related data (and a description of this data), designed to meet the information needs of an organisation

The database represents the entities, the attributes and the logical relationships between the entities. In other words, the database holds data that is logically related.

***Database Management System (DBMS)***

A software system that enables users to define, create and maintain the database and which provides controlled access to this database

**Roles in the Database Environment**

There are four distinct types of people:

1. ***Data and Database Administrators***

Roles of management and control of a DBMS and its data

The **Data Administrator (DA)** is responsible for the logical database design; planning, development and maintenance of standards, policies and procedures.

The **Database Administrator (DBA)** is responsible for the physical database design; implementation, security and integrity control, maintenance and performance.

1. ***Database Designers***

There are two distinguish types of engineers:

The **Logical Database Designer** is concerned with identifying the data (entities and attributes), the relationships between the data and the constraints on the data that is to be stored in the database.

The **Physical Database Designer** takes the logical data model and decides how it is to be physically realised, which involves:

1. Mapping the logical data model into a set of tables and integrity constraints
2. Selecting specific storage structures and access methods for the data to achieve good performances for the database activities
3. Designing any security measures required on the data
4. ***Application Programmers***

The application programs provide the required functionality for the end user can be implemented; allowing the DBMS to perform operations (retrieval, insertion, updating and deletion of data)

1. ***End Users***

*Clients* of the database - who the database is designed and implemented for

**Advantages of Database Systems**

1. Control of data redundancy *(minimising data duplication)*
2. Data consistency *(modification of a record, ensures that all associated data items are kept consistent)*
3. More information from the same piece of data
4. Sharing of data
5. Improved data integrity *(Constraints within the database)*
6. Improved security *(User authority)*
7. Enforcement of standards
8. Economy of scale *(Cost of centralising the organisations operational data is minimised, by one storage area of information for several departments. No extra costs to maintain several departments information, only one)*
9. Balanced conflicting requirements *(All users have different needs. By the control of DBA, they make decisions about the design and operational use of the database)*
10. Improved data accessibility and responsiveness
11. Increased productivity
12. Improved maintenance through data independence
13. Increased concurrency
14. Improved backup and recovery services

**Disadvantages of Database Systems**

1. Complexity *(all user of the database must have knowledge, to take full advantage, else failure to understand can lead to bad design decisions, and most of all be disastrous to an organisation)*
2. Size
3. Cost of DBMS
4. Additional hardware costs
5. Cost of conversion
6. Performance
7. Higher impact of a failure (*in the case of a break down, centralisation of resources, can lead to a halt for all concerned)*

**Three-Level ANSI-SPARC Architecture**

The fundamental point is the identification of three levels of abstraction; that is:

1. **EXTERNAL (highest level)**

Users view of the database

Customised view of data that is relevant to a particular user.

1. **CONCEPTUAL**

The logical structure of the entire database seen by the DBA. The conceptual level represent:

1. All entities their attributes and their relationship
2. The constraints on the data
3. Semantic information about the data
4. Security and integrity information.
5. **INTERNAL (lowest level)**

The physical representation of the database. This level describes how data is stored in the database. It interfaces with the operating system access methods (file management techniques for storing and retrieving data records) to place data on the storage devices, build indexes etc.

The internal level is concerned with such things as:

1. Storage space allocation for data and indexes;
2. Record descriptions for storage (with stored sizes for data items);
3. Record placement
4. Data compression and data encryption techniques.

Below this level there is physical level that is managed by the operating system under the directions of the DBMS

**Schemas**

1. The overall description of the database is called database schema.
2. There are three types of schemas, which are defined according to the level of abstractions (internal, conceptual and external)

**Database Language**

The DBMS is the software the interacts with the users' application programs and the database. A data sub language consists of two parts:

1. ***Data Definition Language (DDL)***

The DDL is used to specify the database schema.

1. ***Data Manipulation Language (DML)***

The DML is used to both read and update the database

DMLs are distinguished between two types:

1. **Procedural DML**

A language that allows the user to tell the system exactly ***how*** to manipulate the data

1. **Non-Procedural DML**

A language that allows the user to state ***what*** data is needed rather that how it is to be retrieved ***(SQL, QBE)***

**Functions of a DBMS**

Types of functions and services provided by a database management system are:

 Data storage, retrieval and update

 User-Accessible catalog

 Transaction support

 Con-currency control services

 Recovery

 Authorisation

 Data communication

 Integrity

 Data independence

 Utility

**Multi-User DBMS Architectures**

Common architectures of multi-user DBMS are:

 Teleprocessing (Dumb terminals that are incapable of functioning on their own are connected to a central computer)

 File-Server (Files are distributed about the network)

 Client-Server (Selected files are distributed about the network, dependent on the user)

**Relational Integrity**

Since every attribute has an associated domain, there are constraints, called domain constraints, in the form of restriction on the set of values. The two principal rules for the relational model are know as:

 *Entity Integrity* - No attribute of a primary key can be null, otherwise the definition of the primary key is contradicted, value must be present to uniquely identify tuples

 *Referential Integrity* - If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key must be wholly null

**Enhanced Entity-Relationship Modelling**

The ER model includes the components of:

 ***Entity Types***

 ***Attributes***

 ***Keys***

 ***Relationship Types***

The concept of ER modelling is:

 ***Relationships with attributes***

 ***Structural concepts***

 ***Connection Traps***

**Problems with ER Models**

Problems that may occur when designing a conceptual data model. These problems are referred to as **Connection Traps**, which normally occur due to misinterpretation of the meaning of certain relationships. The main type of connection traps are:

 ***Fan Traps***

When the relationship between entity types are ambiguous

 ***Chasm Traps***

When the relationship between entity types are suggested to be existing but in actual fact have no common connector to create the relation

**Enhanced Entity-Relationship Modelling**

The EER model includes all the components of the ER model together with the additional concepts of:

 ***Categorisation***

 ***Aggregation***

 ***Specialization/Generalization***

The concept of specialization/generalization is associated with the related concepts of entity types described as superclasses or subclasses, and process of attribute inheritance.

 **Specialization** is the process of defining a set of subclasses of an entity type.

 **Generalization** is the process of minimising the differences between entity types by identifying their common features.

**Constraints on Specialization and Generalization**

There are two types of constraints that apply to specialization or generalization:

 *Disjoint/Non-Disjoint Constraints*

 Disjoint Constraint

This constraint specifies that if the subclasses are disjointed, then an entity can be a member of only one of the subclasses. A **"d"** is placed in the circle that connects the subclasses to the superclass.

 Non-Disjoint

This constraint specifies that if the subclasses are not disjoint, then an entity may be a member of more than one subclass. A **"o"** is placed in the circle that connects the subclasses to the superclass

 *Participation Constraint*

It can be either total *(must)* or partial *(may be)*

**Normalization**

A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.

Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form.

**The Process of Normalization**

Normalization is a formal technique for analysing relations based on their primary key (or candidate keys in the case of BCNF) and functional dependencies. The technique involves a series of rules that can be tested against individual relations so that a database can be normalisaed to any degree. When a requirement is not met, the relation violating the requirement must be decomposed into relations that individually meet the requirements of normalization.

 ***Unnormalized Form (UNF)***

A table that contains one or more repeating groups.

 ***First Normal Form (1NF)***

A relation in which the intersection of each row and column contains one and only one value. Eliminate all repeating groups of attributes within a table that have multiple values.

 ***Second Normal Form (2NF)***

Every non-primary-key attribute is fully functionally dependent on the primary key. Removal of partial dependencies within the table.

 ***Third Normal Form (3NF)***

No non-primary-key attribute is transitively dependent on the primary. Removal of transitive dependencies within the table, by placing the attribute(s) in a new relation with a copy of the determinant(s).

A-->B B-->C then A-->C exists via B

 ***Fourth Normal Form (4NF)***

Multivalued dependency that is dependant on the primary key.

 ***Fifth Normal Form (5NF)***

Multivalued dependencies are independent of each other.

1. ***Boyce-Codd Normal Form (BCNF)***

Database relations are designed so that they have neither partial dependencies nor transitive dependencies, because these types of dependencies result in update consistency. A relation is in BCNF if and only if every determinant is a candidate key (uniquely identifies an entity).

**Transaction**

An action or series of actions (sequence of operations), carried out by a single user or application program, which assesses or changes the contents of the database (INSERT or UPDATE). Logical unit that takes database from one consistent state to another.

1. ***Transaction Commit***: all operations ahould be done and recorded
2. ***Transaction Abort***: none tall operations ahould be done and recorded

**Properties of Transactions**

There are properties that all transactions should possess, the four basic properties of transactions are:

1. ***Atomicity***

The 'all or nothing' property

A transaction is an indivisible unit that is either performed in its entirety or it is not performed at all

1. ***Consistency***
2. ***Isolation***

Transactions executed independently of one another, therefore should not be visible to other transactions

1. ***Durability***

Transaction are permanent and therefore cannot be lost because of a subsequent failure

**Serializability**

A schedule is an interleaving of a set of actions of different transactions, in a sequential order.

***Initial state + Schedule -> Final state***

This is preformed to prevent inconsistency from transactions.

* Non-conflict Serializaility:
* Conflict Serializaility:

**Concurrency Control**

The process of managing simultaneous operations on the database without having them interfere with one another.

* Concurrency control is needed when multiple users are allowed to access the database simultaneously. Without it, problems of lost update, uncommitted dependency and inconsistent analysis can arise
* Serial execution means executing one transaction at a time, with no interleaving of operations. A Schedule shows the sequence of the operations of transactions. A schedule is serializable if it produces the same results as some serial schedule

**Concurrency Control Techniques**

Serializability can be achieved in several ways. There are two basic concurrency control techniques that allow transactions to execute safely in parallel to certain constraints: locking and timestamping methods.

1. **Locking**

A procedure used control concurrent access to data. When one transaction is accessing the database, a lock may deny access to other transactions to prevent incorrect updates

1. **Timestamping**

A concurrency control protocol in which the fundamental goal is to order transactions globally in such a way that older transaction, transactions with smaller timestamps, get priority in the event of conflict.

**Recovery Control**

Causes of failure are system crashes, media failures, application software errors, carelessness, natural physical disasters sabotage and others. They can result in the loss of main memory and/or the disk copy of the database. Recovery techniques minimise these effects.

* To facilitate recovery, the system maintains a log file containing transaction records that identify the start/end of transactions and detail write operations. Using deferred updates, writes are done initially to the log only and the log records are used to perform actual updates to the database. If the system fails, it examines the log to determine which transactions it needs to redo, but there is no need to undo any writes. Using immediate updates, an update may be made to the database itself any time after a log record is written. The log can be used to undo and redo transactions in the event of a failure.
* Checkpoints are used to improve database recovery. At a checkpoint, all modified buffer blocks, all log records and a checkpoint record identifying all active transactions are written to disk. If a failure occurs, the checkpoint record identifies which transactions need to be redone

**Integrity and Security**

**Integrity**

The data is consistent with all the stated constraints that apply to it, and hence can be considered to be valid

Types of integrity:

1. ***Entity Integrity***

No attribute of the primary key can be null

1. ***Referential Integrity***

If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in another relation

1. ***Entity Participation in a Relationship***

Existence Dependency

1. ***Domain Constraints***

A set of allowable values for one or more attributes

1. ***Enterprise Constraints***

Additional rules specified by the users or database administrators of the database

One obvious problem is that of data input; an end-user could easily mistype input data, which is acceptable by the system constraints, but still incorrect. To overcome this problem, is to have someone else check the data entered against the source documents.

**Security**

Data integrity is closely associated with security

The protection of the database against threats using both technical and administrative controls.

Database security in relations is considered in the following situations:

1. Theft and Fraud
2. Loss of Confidentiality (Secrecy)
3. Loss of Privacy
4. Loss of Integrity
5. Loss of Availability

**Threats**

Any situation or event, whether intentional or unintentional, that will adversely affect a system and consequently an organisation

**Countermeasures - Computer-Based Controls**

The following are considered to deal with multi-user environments:

1. ***Authorisation***

Granting a right or privilege to use certain areas of the system

1. ***Views***
2. ***Back-Ups***
3. ***Journalising***

Records of maintaining of all changes made to the database to enable recovery to be undertaken effectively

1. ***Check Pointing***
2. ***Integrity***
3. ***Encryption***

Encoding of data, so the data is unreadable by any other program without the decryption code

**Countermeasures - Non-Computer-Based Controls**

The following are considered to support the computer-based controls:

1. ***Establishment of a Security Policy and Contingency Plan***

A security policy defines how an organisation is to maintain a secure system.

A contingency plan defines how an organisation is to continue functioning in any given emergency situation.

1. ***Personnel Controls***
2. ***Secure Positioning of Equipment***
3. ***Secure Data and Software***
4. ***Escrow Agreement***

Agreements concerning software made between developers and their clients, whereby a third party holds the source code for the client's applications. It is a form of insurance for the client who can acquire the source code if the developer goes out of business, and means they will not be left with non-maintainable systems.

1. ***Maintenance Agreements***
2. ***Physical Access Controls***
3. ***Build Controls***
4. ***Emergency Arrangements***

**Data Protection and Privacy Laws**

Privacy concerns the right of an individual not to have personal information collected, stored and disclosed either wilfully or indiscriminately.

Data protection is the protection of personal data from unlawful acquisition, storage and disclosure, and the provision of the necessary safeguards to avoid the destruction or corruption of the legitimate data held.

**Distributed Database Systems**

1. **Distributed Database**

A logically interrelated collection of shared data, physically distributed over a computer network.

1. **Distributed DBMS**

The software system that permits the management of the distributed database and makes the distribution transparent to others.

The advantages of a **DDBMS** are that it reflects the organisational structure, it makes remote data more sharable, it improves reliability, availability and performance. It may be more economical and provides for modular growth.

The major disadvantages are cost, complexity, lack of standards and experience.

**Distributed Relational Database Systems**

A relation maybe divided into a number of sub-relations called **fragments**, which may be horizontal, vertical, mixed or derived. Fragments are **allocated** to one or more sites. Fragments may be **replicated** to provide improved availability and performance.

**Date's 12 Rules for a DDBMS**

* + 1. **Local Autonomy**

Local data is locally owned and managed

Local operations remain purely local

All operations at a given site are controlled by that site

* + 1. **No reliance on a Central Site**
    2. **Continuous Operation**
    3. **Location Independence**

User should be able to access the database from any site

* + 1. **Fragmentation Independence**

User should be able to access the data, no matter how it is fragmented

* + 1. **Replication Independence**
    2. **Distributed Query Processing**

System should be able to process queries that reference the data form more than one site

* + 1. **Distributed Transaction Process**

The system should support the transactions as the unit of recovery. The system should ensure that both the global and local transactions conform to the ACID rules for transactions; *atomicity consistency, independence* and *durability*

* + 1. **Hardware Independence**
    2. **Operating System Independence**
    3. **Network Independence**
    4. **Database Independence**

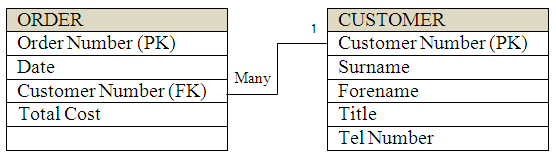
**Normalisation**

Before we begin with Normalisation we should discuss or recap some basic concepts.

* A **field** is a column in a table.
* A **record** is a row in a table, also known as a collection of fields.
* A **table** is a collection of related records.
* A **Database** is a collection of related tables.
* A **Relational** **Database** is a database that has more than one table that has been related to each other through 1-1, 1-many or many-1 relations.
* An **entity** is a table in a Relational Database.
* An **attribute** is a field, its data type and size in a table, e.g. name, Text 30.
* A **primary** **key** is a field that uniquely identifies the record in a table.

For example: name is not a primary key as many people have the same name. A National Insurance number or Student number are examples of good primary keys since they can uniquely identify one person’s record.

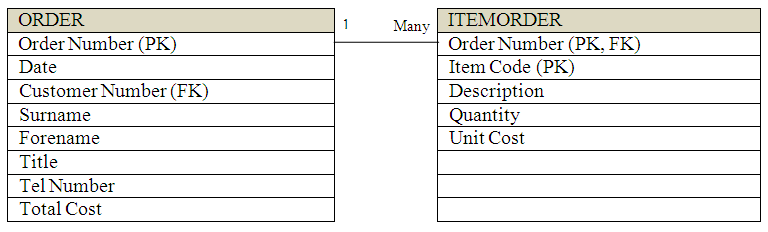
* A **foreign** **key** is a field that is used to link two tables together in a Relational Database. A foreign key is a primary key in another table and that is why it is known as foreign. For example:

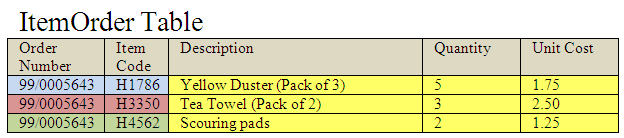
Why is there a Many-to-1 relationship between Order and Customer?

This is because in the Customer table the Customer Number is the primary key and therefore uniquely identifies each customer. There can only be one customer with a certain number. In the Order table the Customer Number is a foreign key, so it does not have to be unique (there can be more than one appearance of the same Customer Number).

**Think about it:** Customers can have more than one Order! Therefore there is a Many-to-1 relationship between Order and Customer.

* A **compound** **key** is a primary key that is made up of two fields. In other words, two fields are needed to uniquely identify a table. Look at the following example:

In the Item order table all the items ordered and there details are stored. Order Number cannot be the only primary key as there are many items per order. Item Code cannot be the primary key for the same reason. Therefore the only way to uniquely identify a record in the ItemOrder table is by using both Order Number and Item Code as a compound key.



Look at the table on the left to see an example of this. Look how the compound key uniquely identifies each record.

**NORMALISATION**

Part of database design – follow a number of stages to complete the normalisation process and end up with and the correct structure for the tables, the relationships created and minimise data redundancy (unnecessarily duplicating fields in tables).

Before normalising data, write out all the attributes (fields) required by the system. You can use the outputs that you have for the system in your Input, Processing and Output diagram to determine these attributes (fields).

For example: Using the order form report below you can list all attributes.

Leemingville Supplies Ltd

31 Waverley Road

Midchester

**Order Number**

99/0005643

**Date of order**

14/11/2008

**Customer Number**

654983

**Name**

Mrs Penelope Higgins **Telephone Number** 01962 785439

**Item Code Description Quantity Unit Cost**

H1786 Yellow duster (pack of 3) 5 1.75

H3350 Tea Towel (pack of 2) 3 2.50

H4562 Scouring pads 2 1.25

**Total** 18.75

*Payment should be received within a month of the date shown on the order. Thank you for your custom.*

Attributes:

Order number Date Customer Number

Customer Surname Customer Forename Customer Title

Tel Number \*Item Code \*Description

\*Quantity \*Unit Cost Total Cost

\* There can be a number of values in these fields in any orders – they are called repeated fields

These can now be written in **Zero Normal Form (0NF)** using standard notation as given below.

Standard notation for Listing Fields in a Table

* The table name is written in UPPER CASE letters
* The table name is a singular noun – VIDEO not VIDEOS
* Fields (attributes) are written in parentheses after the table name
* Primary key field(s) are underlined
* Repeated fields have a line over them

ORDER (Order Number, *Date, Customer Number,* Surname, Forename, Title, Tel Number, *Item Code*, Description, Quantity, Unit Cost, Total Cost)

Table name = ORDER

Primary key = Order Number as it is a unique number for each order.

*Repeated fields* = Three fields could be repeated many times in one order.

In a table this looks as follows:

|  |
| --- |
| ORDER |
| Order Number (PK) |
| Date |
| Customer Number |
| Surname |
| Forename |
| Title |
| Tel Number |
| Item Code |
| Description |
| Quantity |
| Unit Cost |
| Total Cost |

Order Table

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Order Number** | **Date** | **Customer Number** | **Surname** | **Forename** | **Title** | **Tel Number** | **Item Code** | **Description** | | **QTY** | **Unit Cost** | **Total Cost** |
| 99/0005643 | 14/11/2008 | 654983 | Higgins | Penelope | Mrs | 01962785439 | H1786 | Yellow Duster (Pack of 3) | 5 | | 1.75 | 18.75 |
| 99/0005643 | 14/11/2008 | 654983 | Higgins | Penelope | Mrs | 01962785439 | H3350 | Tea Towel (Pack of 2) | 3 | | 2.50 | 18.75 |
| 99/0005643 | 14/11/2008 | 654983 | Higgins | Penelope | Mrs | 01962785439 | H4562 | Scouring pads | 2 | | 1.25 | 18.75 |

Can you see the repetition of data above? Because Item Code, Description, Quantity and Unit Cost can contain many items per order it creates the unnecessary repetition of the Order Number, Date, Customer Number, Surname, Forename, Title and Total Cost fields.

**First Normal Form (1NF)**

This only occurs when there are repeated fields in the 0NF.

Item Code, Description, Quantity and Unit Cost are repeated. These fields are removed from ORDER and placed into a new table called ITEMORDER

ORDER (Order Number, Date, Customer Number, Surname, Forename, Title, Tel Number, Total Cost)

ITEMORDER(Order Number, Item Code, Description, Quantity, Unit Cost)

Note – key field Order Number has been copied to ITEMORDER table. If not, there would be no link between the tables. Order Number and Item Code comprise the compound primary key for ITEMORDER table.

A compound key is when two or more fields are used to uniquely identify a record in a table.

In the ITEMORDER table Order Number and Item Code is the compound primary key because Item Code alone could not uniquely identify the Description, Quantity and Unit Cost.

In tables this will look as follows:

|  |  |  |
| --- | --- | --- |
| ORDER |  | ITEMORDER |
| Order Number (PK) |  | Order Number (PK, FK) |
| Date |  | Item Code (PK) |
| Customer Number (FK) |  | Description |
| Surname |  | Quantity |
| Forename |  | Unit Cost |
| Title |  |  |
| Tel Number |  |  |
| Total Cost |  |  |

Order Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order Number | Date | Customer Number | Surname | Forename | Title | Tel Number | Total Cost |
| 99/0005643 | 14/11/2008 | 654983 | Higgins | Penelope | Mrs | 01962785439 | 18.75 |
| 99/0005644 | 16/11/2008 | 789222 | Callaghan | Dylan | Mr | 01951432567 | 18.75 |
| 99/0005645 | 01/12/2008 | 235213 | Hildyard | Jenny | Mrs | 01934563210 | 18.75 |

ItemOrder Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order Number | Item Code | Description | Quantity | Unit Cost |
| 99/0005643 | H1786 | Yellow Duster (Pack of 3) | 5 | 1.75 |
| 99/0005643 | H3350 | Tea Towel (Pack of 2) | 3 | 2.50 |
| 99/0005643 | H4562 | Scouring pads | 2 | 1.25 |

As discussed above, the repeated data is now in a separate table. This helps prevent the unnecessary repetition of data and therefore simplifies the Order table.

**Second Normal Form (2NF)**

2NF only exists if 1NF produces a table with a compound key. If so, all attributes must be examined to see if they are dependent upon just one part of the compound (composite) key.

Order Number part of the key

Item Code part of the key

Description always the same for a given Item Code regardless of Order Number, so it is dependent on Item Code.

Unit Cost is always the same for a given Item Code, regardless of Order Number, so it is dependent on Item Code.

Quantity relates to the Item Code but will vary for different orders – thus it is dependent on both Order Number and Item Code

We can create a new table ITEM that holds all data about a particular item. The attribute Item Code is also left in the ITEMORDER table to provide a link (foreign key).

ORDER (Order Number, Date, Customer Number, Surname, Forename, Title, Tel Number, Total Cost)

ITEMORDER (Order Number, (FK) Item Code, Quantity)

ITEM (Item Code, Description, Unit Cost)

In tables this will look as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ORDER |  | ITEMORDER |  | ITEM |
| Order Number (PK) |  | Order Number (PK, FK) |  | Item Code (PK) |
| Date |  | Item Code (PK, FK) |  | Description |
| Customer Number (FK) |  | Quantity |  | Unit Cost |
| Surname |  |  |  |  |
| Forename |  |  |  |  |
| Title |  |  |  |  |
| Tel Number |  |  |  |  |
| Total Cost |  |  |  |  |

Order Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order Number | Date | Customer Number | Surname | Forename | Title | Tel Number | Total Cost |
| 99/0005643 | 14/11/2008 | 654983 | Higgins | Penelope | Mrs | 01962785439 | 18.75 |
| 99/0005644 | 16/11/2008 | 789222 | Callaghan | Dylan | Mr | 01951432567 | 18.75 |
| 99/0005645 | 01/12/2008 | 235213 | Hildyard | Jenny | Mrs | 01934563210 | 18.75 |

ItemOrder Table Item Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order Number | Item Code | Quantity |  | Item Code | Description | Quantity | Unit Cost |
| 99/0005643 | H1786 | 5 |  | H1786 | Yellow Duster (Pack of 3) | 5 | 1.75 |
| 99/0005643 | H3350 | 3 |  | H3350 | Tea Towel (Pack of 2) | 3 | 2.50 |
| 99/0005643 | H4562 | 2 |  | H4562 | Scouring pads | 2 | 1.25 |

**Third Normal Form (3NF)**

Tidying up stage – where all tables are examined for dependencies (links between attributes). In 3NF no non-key field should be a fact about another non-key field (or dependent on another non-key field).

Links can be found in the ORDER table where Customer Number, Surname, Forename, Title and Tel Number can be said to be dependant on Customer Number.

These attributes can be taken out to form a new table – CUSTOMER. Customer number is left behind to provide the link (foreign key)

The final design of the tables is:

ORDER (Order Number, Date, Customer Number (FK), Total Cost)

CUSTOMER (Customer Number, Surname, Forename, Title, Tel Number)

ITEMORDER (Order Number (FK) Item Code (FK), Quantity)

ITEM (Item Code, Description, Unit Cost)

In tables this will look as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ORDER |  | CUSTOMER |  | ITEMORDER |  | ITEM |
| Order Number (PK) |  | Customer Number (PK) |  | Order Number (PK, FK) |  | Item Code (PK) |
| Date |  | Surname |  | Item Code (PK, FK) |  | Description |
| Customer Number (FK) |  | Forename |  | Quantity |  | Unit Cost |
| Total Cost |  | Title |  |  |  |  |
|  |  | Tel Number |  |  |  |  |

Order Table

|  |  |  |  |
| --- | --- | --- | --- |
| Order Number | Date | Customer Number | Total Cost |
| 99/0005643 | 14/11/2008 | 654983 | 18.75 |
| 99/0005644 | 16/11/2008 | 789222 | 18.75 |
| 99/0005645 | 01/12/2008 | 235213 | 18.75 |

Customer Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer Number | Surname | Forename | Title | Tel Number |
| 654983 | Higgins | Penelope | Mrs | 01962785439 |
| 789222 | Callaghan | Dylan | Mr | 01951432567 |
| 235213 | Hildyard | Jenny | Mrs | 01934563210 |

ItemOrder Table Item Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Order Number | Item Code | Quantity |  | Item Code | Description | Quantity | Unit Cost |
| 99/0005643 | H1786 | 5 |  | H1786 | Yellow Duster (Pack of 3) | 5 | 1.75 |
| 99/0005643 | H3350 | 3 |  | H3350 | Tea Towel (Pack of 2) | 3 | 2.50 |
| 99/0005643 | H4562 | 2 |  | H4562 | Scouring pads | 2 | 1.25 |

**Entity and Attributes**

The final result of the normalisation including all entities and attributes will look as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ORDER |  |  | CUSTOMER |  |
| Order Number (PK) | Text – 10 Characters |  | Customer Number (PK) | Number – Long Integer |
| Date | Date/Time – Short Date |  | Surname | Text – 30 Characters |
| Customer Number (FK) | Number – Long Integer |  | Forename | Text – 30 Characters |
| Total Cost | Currency – 2 Decimals |  | Title | Text – 4 Characters |
|  |  |  | Tel Number | Text – 12 Characters |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ITEMORDER |  |  | ITEM |  |
| Order Number (PK, FK) | Text – 10 Characters |  | Item Code (PK) | Text – 5 Characters |
| Item Code (PK, FK) | Text – 5 Characters |  | Description | Text – 50 Characters |
| Quantity | Number – Integer |  | Unit Cost | Currency – 2 Decimals |

**Entity Relationship Diagram**

The following ERD can be created for the tables above. Here you can see the 1-to-many relationships between the entities (tables).

Generally there will always be a “many” relationship where there is a foreign key and a “1” relationship where there is a primary key.

ITEM

CUSTOMER

ITEMORDER

ORDER

Note – some data duplication remains but this is necessary redundancy as the fields are used to create the links (relationships) in the relational database.

How are the 1-to-1, 1-to-many or many-to-1 relationships determined? Well it is really a question of you looking carefully at the tables and determining where the Primary and Foreign key relationships exist.

Generally there will always be:

* A 1-to-1 relationship when tables are related by Primary Keys. This is rare.
* A 1-to-many relationship when tables are related by a Primary to Foreign Key.
* A many-to-1 relationship when tables are related by a Foreign to Primary Key.