Topic: 1.8.2 Relational database modelling

Relational Database Management System (RDBMS)

The Relational Model is an attempt to simplify database structures. It represents all data in the database as simple row-column tables of data values. An RDBMS is a software program that helps to create, maintain, and manipulate a relational database. A relational database is a database divided into logical units called tables, where tables are related to one another within the database.

Tables are related in a relational database, allowing adequate data to be retrieved in a single query (although the desired data may exist in more than one table). By having common keys, or fields, among relational database tables, data from multiple tables can be joined to form one large result set.

Figure 1.5 shows two tables related to one another through a common key (data value) in a relational database.

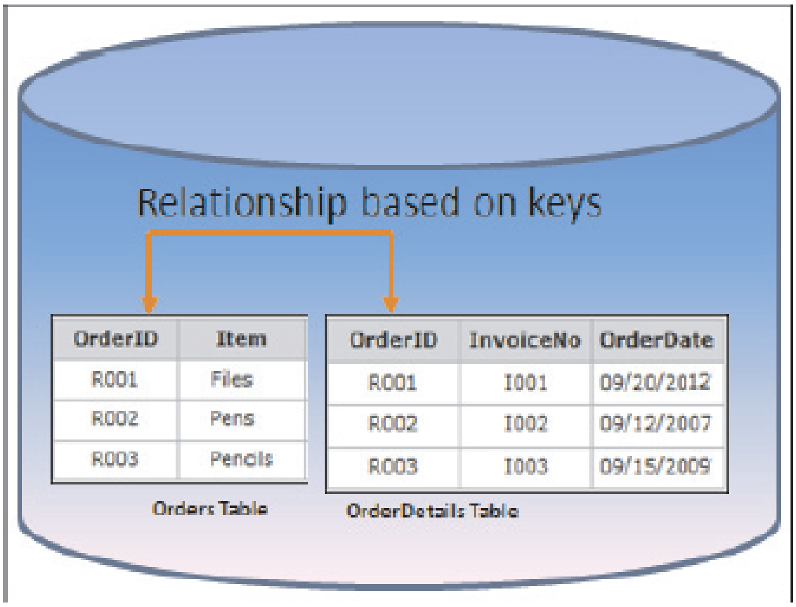


Figure 1.5: Relationship between Tables

Thus, a relational database is a database structured on the relational model. The basic characteristic of a relational model is that in a relational model, data is stored in relations.

Topic: 1.8.2 Relational database modelling

To understand relations, consider the following example. The **Capitals** table shown in table 1.6 displays a list of countries and their capitals, and the **Currency** table shown in table 1.7 displays the countries and the local currencies used by them.

Country	Capital
Greece	Athens
Italy	Rome
USA	Washington
China	Beijing
Japan	Tokyo
Australia	Sydney
France	Paris

Table 1.6: Capitals

Country	Currency
Greece	Drachma
Italy	Lira
USA	Dollar
China	Renminbi (Yuan)
Japan	Yen
Australia	Australian Dollar
France	Francs

Table 1.7: Currency

Both the tables have a common column, that is, the **Country** column. Now, if the user wants to display the information about the currency used in Rome, first find the name of the country to which Rome belongs. This information can be retrieved from table 1.6. Next, that country should be looked up in table 1.7 to find out the currency.

It is possible to get this information because it is possible to establish a relation between the two tables through a common column called **Country**.

Terms related to RDBMS

There are certain terms that are mostly used in an RDBMS. These are described as follows:

- Data is presented as a collection of relations.
- Each relation is depicted as a table.
- Columns are attributes.
- Rows ('tuples') represent entities.
- Every table has a set of attributes that are taken together as a 'key' (technically, a 'superkey'), which uniquely identifies each entity.

For example, a company might have an **Employee** table with a row for each employee; typically called the entity. What attributes might be interesting for such a table? This will depend on the application and the type of use the data will be put to, and is determined at database design time.

An **entity** is anything living or nonliving with certain attributes to which some values can be assigned. These are separate entities for which different tables are designed in a schema.

Topic: 1.8.2 Relational database modelling

Consider the scenario of a company maintaining customer and order information for products being sold and customer-order details for a specific month, such as, August.

The tables 1.8, 1.9, 1.10, and 1.11 are used to illustrate this scenario. These tables depict tuples and attributes in the form of rows and columns. Various terms related to these tables are given in table 1.12.

Cust_No	Cust_Name	Phone No
002	David Gordon	0231-5466356
003	Prince Fernandes	0221-5762382
003	Charles Yale	0321-8734723
002	Ryan Ford	0241-2343444
005	Bruce Smith	0241-8472198

Table 1.8: Customer

Item_No	Description	Price
HW1	Power Supply	4000
HW2	Keyboard	2000
HW3	Mouse	800
SW1	Office Suite	15000
SW2	Payroll Software	8000

Table 1.9: Items

Ord_No	Item_No	Qty
101	HW3	50
101	SW1	150
102	HW2	10
103	HW3	50
104	HW2	25
104	HW3	100
105	SW1	100

Table 1.10 Order_Details

Ord_No	Ord_Date	Cust_No
101	02-08-12	002
102	11-08-12	003
103	21-08-12	003
104	28-08-12	002
105	30-08-12	005

Table 1.11 Order_August

Topic: 1.8.2 Relational database modelling

Term	Meaning	Example from the Scenario
Relation	A table	Order_August, Order_Details, Customer and Items
Tuple	A row or a record in a relation	A row from Customer relation is a Customer tuple
Attribute	A field or a column in a relation	Ord_Date, Item_No, Cust_Name, and so on
Cardinality of a relation	The number of tuples in a relation	Cardinality of Order_Details relation is 7
Degree of a relation	The number of attributes in a relation	Degree of Customer relation is 3
Domain of an attribute	The set of all values that can be taken by the attribute	Domain of Qty in Order_Details is the set of all values which can represent quantity of an ordered item
Primary Key of a relation	An attribute or a combination of attributes that uniquely defines each tuple in a relation	Primary Key of Customer relation is Cust_No Ord_No and Item_No combination forms the primary key of Order_Details
Foreign Key	An attribute or a combination of attributes in one relation R1 that indicates the relationship of R1 with another relation R2 The foreign key attributes in R1 must contain values matching with those of the values in R2	Cust_No in Order_August relation is a foreign key creating reference from Order_August to Customer. This is required to indicate the relationship between orders in Order_August and Customer

Table 1.12: Terms Related to Tables

Entities and Tables

The components of an RDBMS are entities and tables, which will be explained in this section.

Entity

An entity is a person, place, thing, object, event, or even a concept, which can be distinctly identified. For example, the entities in a university are students, faculty members, and courses.

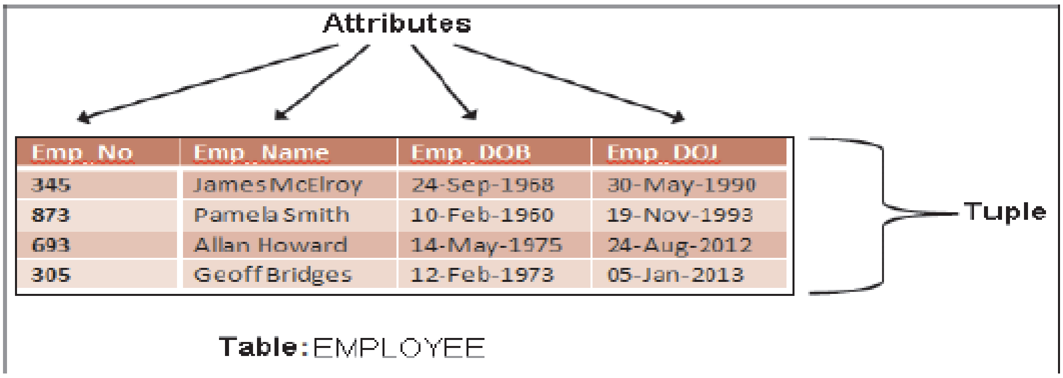
Each entity has certain characteristics known as attributes. For example, the student entity might include attributes such as student number, name, and grade. Each attribute should be named appropriately.

A grouping of related entities becomes an entity set. Each entity set is given a name. The name of the entity set reflects the contents. Thus, the attributes of all the students of the university will be stored in an entity set called **Student**.

Topic: 1.8.2 Relational database modelling

Tables and their Characteristics

The access and manipulation of data is facilitated by the creation of data relationships based on a construct known as a table. A table contains a group of related entities that is an entity set. The terms entity set and table are often used interchangeably. A table is also called a relation. The rows are known as tuples. The columns are known as attributes. Figure 1.6 highlights the characteristics of a table.



Emp_No	Emp_Name	Emp_DOB	Emp_DOJ
345	James McElroy	24-Sep-1968	30-May-1990
873	Pamela Smith	10-Feb-1960	19-Nov-1993
693	Allan Howard	14-May-1975	24-Aug-2012
305	Geoff Bridges	12-Feb-1973	05-Jan-2013

Figure 1.6: Characteristics of a Table

The characteristics of a table are as follows:

- A two-dimensional structure composed of rows and columns is perceived as a table.
- Each tuple represents a single entity within the entity set.
- Each column has a distinct name.
- Each row/column intersection represents a single data value.
- Each table must have a key known as primary key that uniquely identifies each row.
- All values in a column must conform to the same data format. For example, if the attribute is assigned a decimal data format, all values in the column representing that attribute must be in decimals.
- Each column has a specific range of values known as the attribute domain.
- Each row carries information describing one entity occurrence.
- The order of the rows and columns is immaterial in a DBMS.

Topic: 1.8.2 Relational database modelling

Differences between a DBMS and an RDBMS

The differences between a DBMS and an RDBMS are listed in table 1.13.

DBMS	RDBMS
It does not need to have data in tabular structure nor does it enforce tabular relationships between data items.	In an RDBMS, tabular structure is a must and table relationships are enforced by the system. These relationships enable the user to apply and manage business rules with minimal coding.
Small amount of data can be stored and retrieved.	An RDBMS can store and retrieve large amount of data.
A DBMS is less secure than an RDBMS.	An RDBMS is more secure than a DBMS.
It is a single user system.	It is a multi-user system.
Most DBMSs do not support client/server architecture.	It supports client/server architecture.

Table 1.13: Difference between DBMS and RDBMS

In an RDBMS, a relation is given more importance. Thus, the tables in an RDBMS are dependent and the user can establish various integrity constraints on these tables so that the ultimate data used by the user remains correct. In case of a DBMS, entities are given more importance and there is no relation established among these entities.

Topic: 1.8.2 Relational database modelling

Relational Databases and Normalisation

Consider the following delivery note from Easy Fasteners Ltd.

Easy Fasteners Ltd

Old Park, The Square, Berrington, Midshire BN2 5RG

To: Bill Jones No.: 005
London Date: 14/08/11
England

Product No.	**Description**
1	Table
2	Desk
3	Chair

Fig. 3.6. (b)1

In this example, the delivery note has more than one part on it. This is called a repeating group. In the relational database model, each record must be of a fixed length and each field must contain only one item of data. Also, each record must be of a fixed length so a variable number of fields is not allowed. In this example, we cannot say 'let there be three fields for the products as some customers may order more products than this and other fewer products. So, repeating groups are not allowed.

At this stage we should start to use the correct vocabulary for relational databases. Instead of fields we call the columns *attributes* and the rows are called *tuples*. The files are called *relations* (or tables).

Topic: 1.8.2 Relational database modelling

We write the details of our delivery note as

DELNOTE(Num, CustName, City, Country, (ProdID, Description))

where DELNOTE is the name of the relation (or table) and Num, CustName, City, Country, ProdID and Description are the attributes. ProdID and Description are put inside parentheses because they form a repeating group. In tabular form the data may be represented by Fig. 3.6 (b)2.

Num	CustName	City	Country	ProdID	Description
005	Bill Jones	London	England	1	Table
				2	Desk
				3	Chair

Fig. 3.6 (b)2

This again shows the repeating group. We say that this is in un-normalised form (UNF). To put it into 1st normal form (1NF) we complete the table and identify a key that will make each tuple unique. This is shown in Fig. Fig. 3.6 (b)3.

Num	CustName	City	Country	ProdID	Description
005	Bill Jones	London	England	1	Table
005	Bill Jones	London	England	2	Desk
005	Bill Jones	London	England	3	Chair

Fig 3.6 (b)3

To make each row unique we need to choose Num together with ProdID as the key. Remember, another delivery note may have the same products on it, so we need to use the combination of Num and ProdID to form the key. We can write this as

DELNOTE(<u>Num</u>, CustName, City, Country, <u>ProdID</u>, Description)

To indicate the key, we simply underline the attributes that make up the key.

Because we have identified a key that uniquely identifies each tuple, we have removed the repeating group.

Topic: 1.8.2 Relational database modelling

Definition of 1NF

A relation with repeating groups removed is said to be in First Normal Form (1NF). That is, a relation in which the intersection of each tuple and attribute (row and column) contains one and only one value.

However, the relation DELNOTE still contains redundancy. Do we really need to record the details of the customer for each item on the delivery note? Clearly, the answer is no. Normalisation theory recognises this and allows relations to be converted to Third Normal Form (3NF). This form solves most problems. (Note: Occasionally we need to use Boyce-Codd Normal Form, 4NF and 5NF. This is rare and beyond the scope of this syllabus.)

Let us now see how to move from 1NF to 2NF and on to 3NF.

Definition of 2NF

A relation that is in 1NF and every non-primary key attribute is fully dependent on the primary key is in Second Normal Form (2NF). That is, all the incomplete dependencies have been removed.

In our example, using the data supplied, CustName, City and Country depend only on Num and not on ProdID. Description only depends on ProdID, it does not depend on Num. We say that

Num*determines*CustName, City, Country

ProdID*determines* Description

and write

Num →CustName, City, Country

ProdID → Description

Topic: 1.8.2 Relational database modelling

If we do this, we lose the connection that tells us which parts have been delivered to which customer. To maintain this connection we add the dependency

Num, ProdID → 0 (Dummy functional dependency)

We now have three relations.

DELNOTE(<u>Num</u>, CustName, City, Country)

PRODUCT(<u>ProdID</u>, Description)

DEL_PROD(<u>Num</u>, <u>ProdID</u>)

Note the keys (underlined) for each relation. DEL_PROD needs a compound key because a delivery note may contain several parts and similar parts may be on several delivery notes. We now have the relations in 2NF.

Can you see any more data repetitions? The following table of data may help.

Num	CustName	City	Country	ProdID	Description
005	Bill Jones	London	England	1	Table
005	Bill Jones	London	England	2	Desk
005	Bill Jones	London	England	3	Chair
008	Mary Hill	Paris	France	2	Desk
008	Mary Hill	Paris	France	7	Cupboard
014	Anne Smith	New York	USA	5	Cabinet
002	Tom Allen	London	England	7	Cupboard
002	Tom Allen	London	England	1	Table
002	Tom Allen	London	England	2	Desk

Country depends on City not directly on Num. We need to move on to 3NF.

Topic: 1.8.2 Relational database modelling

Definition of 3NF

A relation that is in 1NF and 2NF, and in which no non-primary key attribute is transitively dependent on the primary key is in 3NF. That is, all non-key elements are fully dependent on the primary key.

In our example we are saying

Num →CustName, City, Country

but it is City that determines Country, that is

City → Country

and we can write

Num → City → Country

Num → CustName

We say that Num transitively functionally determines Country.

Removing this transitive functional determinacy, we have

DELNOTE(<u>Num</u>, CustName, City)

CITY_COUNTRY(<u>City</u>, Country)

PRODUCT(<u>ProdID</u>, Description)

DEL_PROD(<u>Num</u>, <u>ProdID</u>)

Topic: 1.8.2 Relational database modelling

Let us now use the data above and see what happens to it as the relations are normalised.

1NF

DELNOTE

Num	CustName	City	Country	ProdID	Description
005	Bill Jones	London	England	1	Table
005	Bill Jones	London	England	2	Desk
005	Bill Jones	London	England	3	Chair
008	Mary Hill	Paris	France	2	Desk
008	Mary Hill	Paris	France	7	Cupboard
014	Anne Smith	New York	USA	5	Cabinet
002	Tom Allen	London	England	7	Cupboard
002	Tom Allen	London	England	1	Table
002	Tom Allen	London	England	2	Desk

Topic: 1.8.2 Relational database modelling

2 NF

DELNOTE

Num	CustName	City	Country
005	Bill Jones	London	England
008	Mary Hill	Paris	France
014	Anne Smith	New York	USA
002	Tom Allen	London	England

PRODUCT

ProdID	Description
1	Table
2	Desk
3	Chair
7	Cupboard
5	Cabinet

DEL_PROD

Num	ProdID
005	1
005	2
005	3
008	2
008	7
014	5
002	7
002	1
002	2

Topic: 1.8.2 Relational database modelling

3 NF

DELNOTE

Num	**CustName**	**City**
005	Bill Jones	London
008	Mary Hill	Paris
014	Anne Smith	New York
002	Tom Allen	London

DEL_PROD

Num	**ProdID**
005	1
005	2
005	3
008	2
008	7
014	5
002	7
002	1
002	2

PRODUCT

ProdID	**Description**
1	Table
2	Desk
3	Chair
7	Cupboard
5	Cabinet

CITY_COUNTRY

City	**Country**
London	England
Paris	France
New York	USA

Topic: 1.8.2 Relational database modelling

Now we can see that redundancy of data has been removed.

In tabular form we have

UNF

DELNOTE(Num, CustName, City, Country, (ProdID, Description))

1NF

DELNOTE(<u>Num</u>, CustName, City, Country, <u>ProdID</u>, Description)

2NF

DELNOTE(<u>Num</u>, CustName, City, Country)

PRODUCT(<u>ProdID</u>, Description)

DEL_PROD(<u>Num</u>, <u>ProdID</u>)

3NF

DELNOTE(<u>Num</u>, CustName, City)

CITY_COUNTRY(<u>City</u>, Country)

PRODUCT(<u>ProdID</u>, Description)

DEL_PROD(<u>Num</u>, <u>ProdID</u>)

In this Section we have seen the data presented as tables. These tables give us a *view* of the data. The tables do NOT tell us how the data is stored in the computer, whether it be in memory or on backing store. Tables are used simply because this is how users view the data. We can create new tables from the ones that hold the data in 3NF. Remember, these tables simply define relations.

Users often require different views of data. For example, a user may wish to find out the countries to which they have sent desks. This is a simple view consisting of one column. We can create this table by using the following relations (tables).

PRODUCT	to find ProdID for Desk
DEL_PROD	to find Num for this ProdID
DELNOTE	to find City corresponding to Num
CITY_COUNTRY	to find Country from City

Topic: 1.8.2 Relational database modelling

Here is another example of normalization.

Films are shown at many cinemas, each of which has a manager. A manager may manage more than one cinema. The takings for each film are recorded for each cinema at which the film was shown.

The following table is in UNF and uses the attribute names

FID	Unique number identifying a film
Title	Film title
CID	Unique string identifying a cinema
Cname	Name of cinema
Loc	Location of cinema
MID	Unique 2-digit string identifying a manager
MName	Manager's name
Takings	Takings for a film

Topic: 1.8.2 Relational database modelling

FID	Title	CID	Cname	Loc	MID	MName	Takings
15	Jaws	TF	Odeon	Croyden	01	Smith	£350
		GH	Embassy	Osney	01	Smith	£180
		JK	Palace	Lye	02	Jones	£220
23	Tomb Raider	TF	Odeon	Croyden	01	Smith	£430
		GH	Embassy	Osney	01	Smith	£200
		JK	Palace	Lye	02	Jones	£250
		FB	Classic	Sutton	03	Allen	£300
		NM	Roxy	Longden	03	Allen	£290
45	Cats & Dogs	TF	Odeon	Croyden	01	Smith	£390
		LM	Odeon	Sutton	03	Allen	£310
56	Colditz	TF	Odeon	Croyden	01	Smith	£310
		NM	Roxy	Longden	03	Allen	£250

Topic: 1.8.2 Relational database modelling

Converting this to 1NF can be achieved by 'filling in the blanks' to give the relation

FID	Title	CID	Cname	Loc	MID	MName	Takings
15	Jaws	TF	Odeon	Croyden	01	Smith	£350
15	Jaws	GH	Embassy	Osney	01	Smith	£180
15	Jaws	JK	Palace	Lye	02	Jones	£220
23	Tomb Raider	TF	Odeon	Croyden	01	Smith	£430
23	Tomb Raider	GH	Embassy	Osney	01	Smith	£200
23	Tomb Raider	JK	Palace	Lye	02	Jones	£250
23	Tomb Raider	FB	Classic	Sutton	03	Allen	£300
23	Tomb Raider	NM	Roxy	Longden	03	Allen	£290
45	Cats & Dogs	TF	Odeon	Croyden	01	Smith	£390
45	Cats & Dogs	LM	Odeon	Sutton	03	Allen	£310
56	Colditz	TF	Odeon	Croyden	01	Smith	£310
56	Colditz	NM	Roxy	Longden	03	Allen	£250

Topic: 1.8.2 Relational database modelling

This is the relation

R(<u>FID</u>, Title, <u>CID</u>, Cname, Loc, MID, MName, Takings)

Title is only dependent on FID

Cname, Loc, MID, MName are only dependent on CID

Takings is dependent on both FID and CID

Therefore 2NF is

FILM(<u>FID</u>, Title)

CINEMA(<u>CID</u>, Cname, Loc, MID, MName)

TAKINGS(<u>FID</u>, <u>CID</u>, Takings)

In Cinema, the non-key attribute MName is dependent on MID. This means that it is transitively dependent on the primary key. So we must move this out to get the 3NF relations

FILM(<u>FID</u>, Title)

CINEMA(<u>CID</u>, Cname, Loc, MID)

TAKINGS(<u>FID</u>, <u>CID</u>, Takings)

MANAGER(<u>MID</u>, MName)

Topic: 1.8.2 Relational database modelling

Entity-Relationship (E-R) Diagrams

Entity-Relationship (E-R) diagrams can be used to illustrate the relationships between entities. In the earlier example we had the four relations

1. DELNOTE(Num, CustName, City)
2. CITY_COUNTRY(City, Country)
3. PRODUCT(ProdID, Description)
4. DEL_PROD(Num, ProdID)

In an E-R diagram DELNOTE, CITY_COUNTRY, PRODUCT and DEL_PROD are called *entities*. Entities have the same names as relations but we do not usually show the attributes in E-R diagrams.

We now consider the *relationships* between the entities.

Each DELNOTE can be for only one CITY_COUNTRY
because a City only occurs once on DELNOTE

Each CITY_COUNTRY may have many DELNOTE
because a City may occur on more than one DELNOTE

Each DELNOTE will have many DEL_PROD
because Num in DELNOTE could occur more than once in DEL_PROD

Each DEL_PROD will be for only one DELNOTE
because each Num in DEL_PROD can only occur once in DELNOTE

Each PRODUCT will be on many DEL_PROD
because PRODUCT can occur more than once in DEL_PROD

Each DEL_PROD will have only one PRODUCT
because each ProdID in DEL_PROD can only occur once in PRODUCT

03-111-222-ZAK | OlevelComputer AlevelComputer | @zakonweb | zak@zakonweb.com | www.zakonweb.com

Topic: 1.8.2 Relational database modelling

The statements show two types of relationship. There are in fact four altogether. These are

one-to-one	represented by	
one-to-many	represented by	
many-to-one	represented by	
many-to-many	represented by	

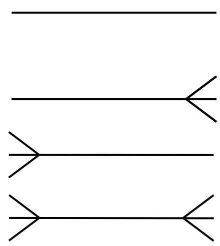


Fig. 3.6 (c)1 is the E-R diagram showing the relationships between DELNOTE, CITY_COUNTRY, PRODUCT and DEL_PROD.

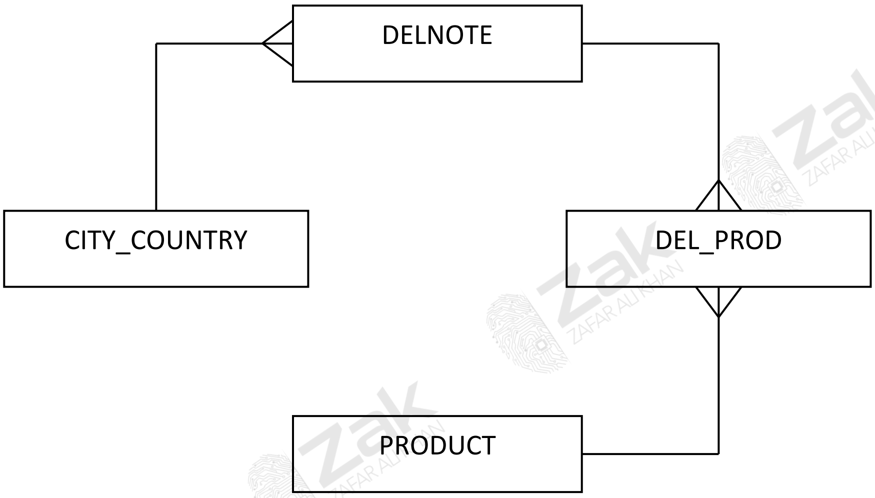


Fig. 3.6 (c)1

If the relations are in 3NF, the E-R diagram will not contain any many-to-many relationships. If there are any one-to-one relationships, one of the entities can be removed and its attributes added to the entity that is left.

Topic: 1.8.2 Relational database modelling

Let us now look at our solution to the cinema problem which contained the relations in 3NF.

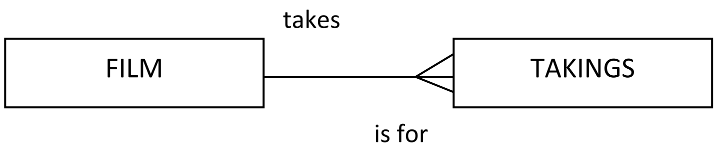
FILM(FID, Title)

CINEMA(CID, Cname, Loc, MID)

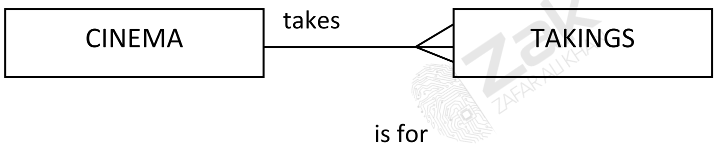
TAKINGS(FID, CID, Takings)

MANAGER(MID, MName)

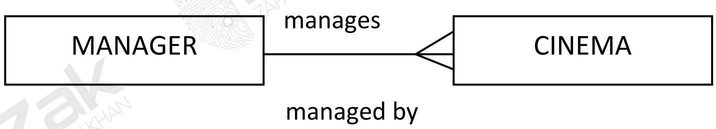
We have the following relationships.



Connected by FID



Connected by CID



Connected by MID

Topic: 1.8.2 Relational database modelling

These produce the ERD shown in Fig. 3.6 (c)2.

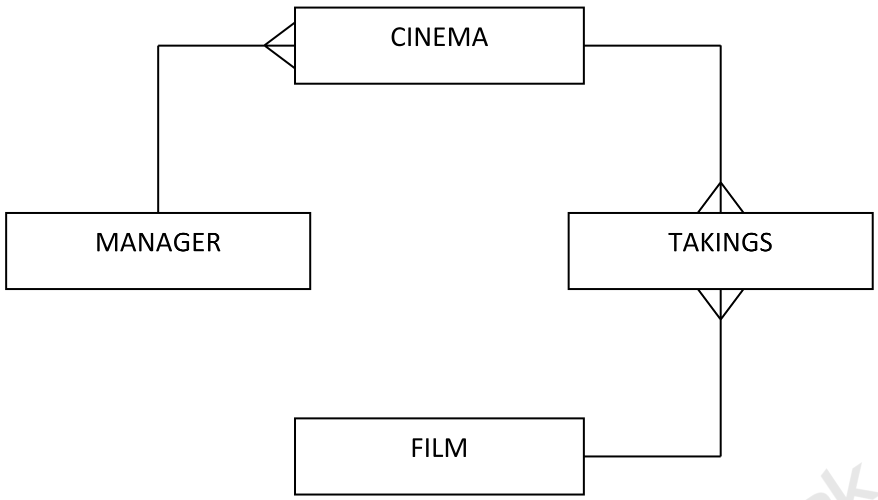


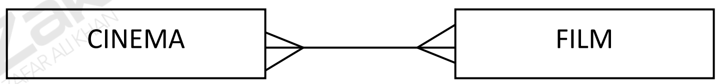
Fig. 3.6 (c)2

In this problem we actually have the relationship

CINEMA shows many FILMs

FILM is shown at many CINEMAs

That is



But this cannot be normalized to 3NF because it is a many-to-many relationship. Many-to-many relationships are removed by using a *link entity* as shown here.

If you now look at Fig. 3.6.c.2, you will see that the link entity is TAKINGS.