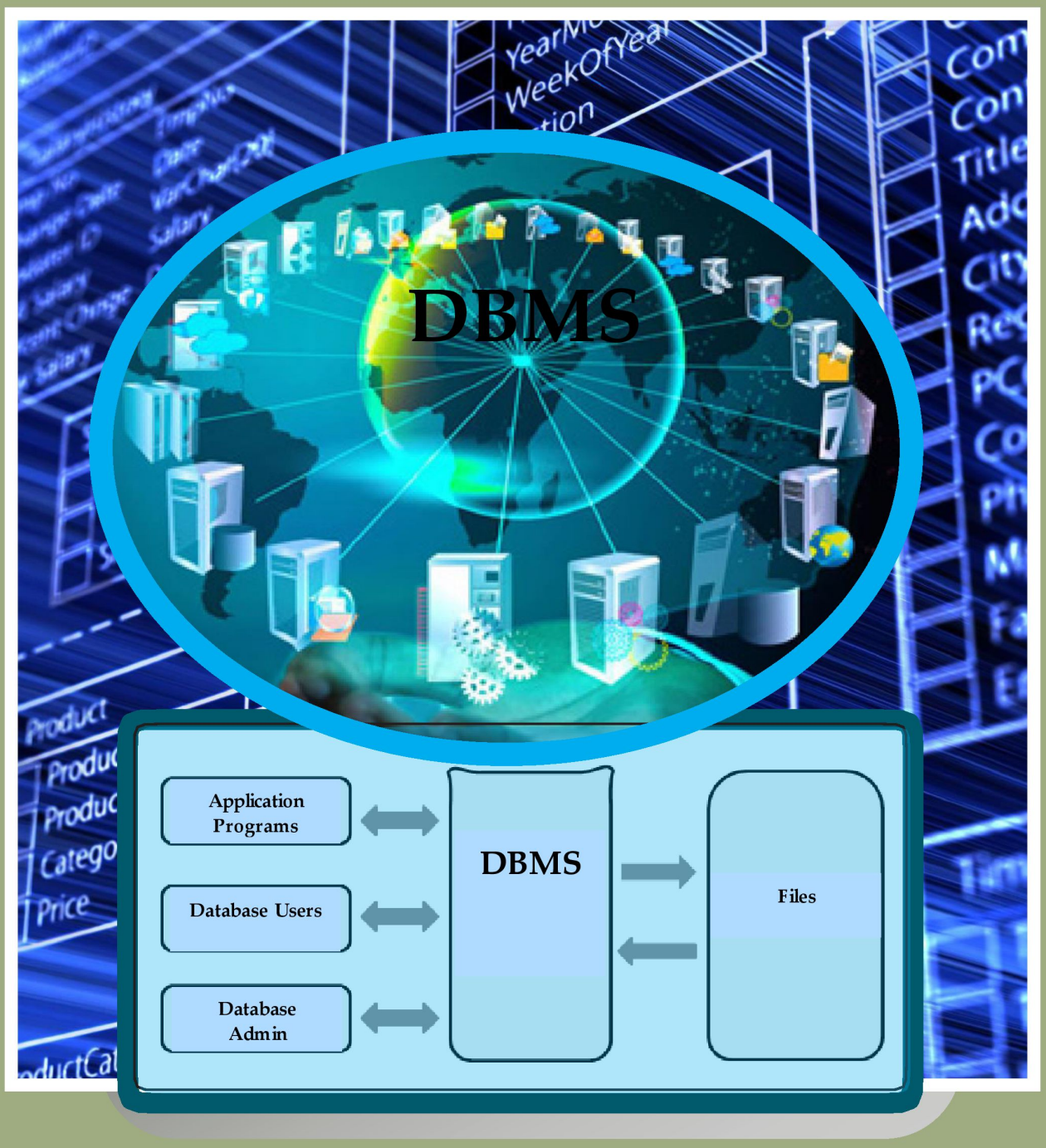
Indira Gandhi National Open University
School of Computer and Information Sciences (SOCIS)

BCS-092 Introduction of Databases



Database Design

2

Indira Gandhi
National Open University
School of Computer and
Information Sciences

BCS-092
Introduction to Databases

Block

2

DATABASE DESIGN

Course Coordinator (for the year 2021)

Mr. Akshay Kumar, Associate Professor
SOCIS, IGNOU, New Delhi-110068

In all the Units of this Block, Check Your Progress and Answers to Check Your Progress Questions are written and added in 2021 BY

Mr. Akshay Kumar, Associate Professor
SOCIS, IGNOU, New Delhi-110068

PRODUCTION

Mr. Tilak Raj
Asst. Registrar (Pub.)
MPDD, IGNOU, New Delhi

Mr. Yashpal
Section Officer (Pub.)
MPDD, IGNOU, New Delhi

January, 2021

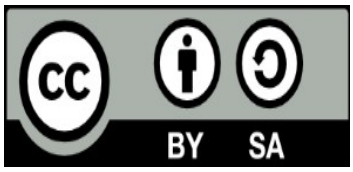
Laser Typesetting : Akashdeep Printers, 20-Ansari Road, Daryaganj, New Delhi-110002

IGNOU is one of the participants for the development of courses of this Programme

Copyright

This course has been developed as part of the collaborative advanced ICT course development project of the Commonwealth of Learning (COL). COL is an intergovernmental organization created by Commonwealth Heads of Government to promote the development and sharing of open learning and distance education knowledge, resources and technologies.

The Open University of Tanzania (OUT) is a fully fledged, autonomous and accredited public University. It offers its certificate, diploma, degree and postgraduate courses through the open and distance learning system which includes various means of communication such as face-to-face, broadcasting, telecasting, correspondence, seminars, e-learning as well as a blended mode. The OUT's academic programmes are quality-assured and as centrally regulated by the Tanzania Commission for Universities (TCU).



The Open University of Tanzania
P. O. Box 23409,
Kinondoni,
Dar Es Salaam,
Tanzania
Phone: +255 22 2668992
Fax: +255 22 2668756
Email: hod.ict@out.ac.tz
Website: www.out.ac.tz

Commonwealth of Learning
4710 Kingsway, Suite 2500,
Burnaby V5H 4M2,
British Columbia,
Canada
Phone: +1 604 775 8200
Fax: +1 604 775 8210
Email: info@col.org
Website: www.col.org

ACKNOWLEDGEMENTS By AUTHORS

The Open University of Tanzania (OUT), Faculty of Science, Technology and Environmental Studies, (FSTES) and the ICT Department wish to thank those below for their contribution to the production of this course material and video lectures:

Authors:
Ms. Grace W. Mbwete (Assistant Lecturer, ICT Department, OUT)
Mr. Elia A. Lukwaro (Tutorial Assistant, ICT Department, OUT)

Copy Editor:
Mr. Justin Kimaro (Principal Editor, IEMT, OUT)

Reviewers:
Ms. Lilian C. Mutalemwa (Assistant Lecturer, ICT Department, OUT)
Ms. Regina Monyemangene (Instructional Designer, IEMT, OUT)

Video Production:
Mr. Othman Mwinchoum (Multimedia studio, IEMT, OUT)
Ms. Grace W. Mbwete (Assistant Lecturer, ICT Department, OUT)
Ms. Lilian C. Mutalemwa (Assistant Lecturer, ICT Department, OUT)

This course has been developed with the support of the ***Commonwealth of Learning, Canada.***

BLOCK INTRODUCTION

In the first block of this course, we have discussed about the basic concepts relating to database management systems. The concept included the database approach, data models and relational database models. This Block will focus on the design aspects of relational database.

Unit 6 of this block explains the concepts related to entity relationship diagram, which helps to create a conceptual design of the relational database system.

Unit 7 focuses on the correctness of data, which is entered in the database system. It may be noted that a good database is the one which does not allow incorrect data to be entered in a database system.

Unit 8 covers the concepts related to data redundancy in a relational database system. Data redundancy must be minimized or eliminated in a good database system. This unit elaborates the concepts of data anomalies.

Unit 9 presents an important aspect called functional dependence, which indentifies basic relationships between a set of attributes in a relation.

Unit 10 introduces the process of normalization of a database which uses functional dependence, to minimize the data redundancy.

You must keep watching the supporting video lectures as indicated in the units.

ignou
THE PEOPLE'S
UNIVERSITY

UNIT 6 ENTITY RELATIONSHIP MODEL

Structure

6.1 INTRODUCTION

For a correct design and implementation of database, there is a need to understand which entities should hold data and identify the connections that may exist between entities. In this unit, you are going to learn about the Entity-Relationship Model which provides a technique in creating a graphical view of the different elements of a database *as well as the relationships between them. In addition, you will also learn the drawing conventions of the E-R model, beginning with those conventions used to represent a single entity, and concluding with conventions used to represent all relations in a database.*

6.2 OBJECTIVES

Upon completion of this unit you will be able to:

- *Describe* the significance ER Model in database design.
- *Explain* the entity-relationship model and its components.
- *Convert* user requirements into an ER Model.

6.3 TERMINOLOGIES

Entity : A thing with distinct and independent existence.

Attribute : Characteristic of an entity.

Relationship : Connection between entities/tables in a database.

ERD : Entity Relationship Diagram - also called an ER schema, are represented by ER diagrams. These are well suited to data modeling for use with databases.

Primary Key : A special relational database table column (or combination of columns) designated to uniquely identify all table records.

Foreign Key : An attribute in a table that references the primary key in another table OR it can be null.

6.4 INTRODUCTION TO ER MODEL

The *entity relationship (ER) data model* was developed for database design by Peter Chen and published in a 1976 paper. It is well suited to data modeling for use with databases because it is fairly abstract and is easy to discuss and explain. ER models are readily translated to relations. ER models, also called an ER schema, are represented by ER diagrams. ER modeling is based on two concepts:

- Entities, defined as tables that hold specific information (data).
- *Relationships*, defined as the associations or interactions between entities

For the rest of this unit, we will use a sample database called the UNIVERSITY database to illustrate the concepts of the ER model. This database contains information about lecturers, students and courses. Important points to note include:

- There must be lecturers in the university. Each lecturer has a unique identification (PF Number), a name, location of the office, salary, birth date, start date and contacts. In addition, data of lecturer spouses, such as name and birth date, may also be stored.
- A department controls a number of academic programme, each of which has a unique name, a unique number, lecturers and a budget.
- Each academic programme has several courses which have course identification (course code), lecturers, units and subjects.
- Students have to enroll in an academic programme and register for its respective programme in the university. The student has student ID, name and contacts.

6.5 ENTITIES

In this section the term entity is explained with the help of an example.

6.5.1 Definition of an Entity

An entity is an object in the real world with an independent existence and can be differentiated from other objects. An entity might be

- An object with physical existence, e.g. a lecturer or a student.
- An object with conceptual existence, e.g. a course and a job.

An Entity Type defines a collection of similar entities.

An Entity Set is a collection of entities of an entity type at a point of time. In ER diagrams, an entity type is represented by a name in a box.

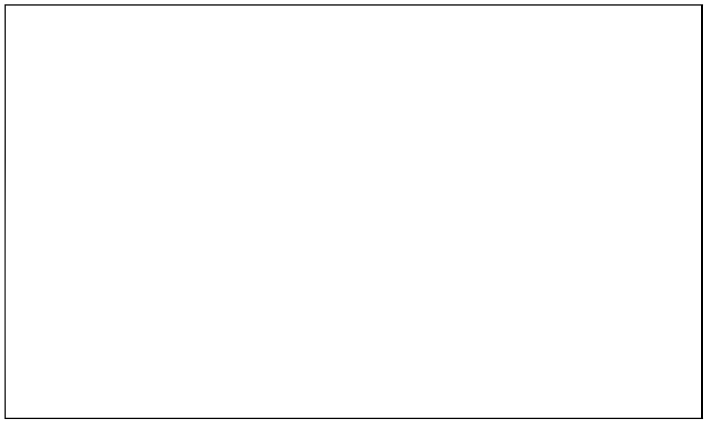


Figure 6.1: A Student entity, by G. Mbwete

6.5.2 Classification of Entities

Entities are classified as strong entities and weak entities or existence dependencies as detailed in sub-sections below.

i) Strong Entities

An entity is considered a 'strong' entity, if it has the following properties:

- If it can exist apart from all of its related entities.
- Independent entities are strong entity.
- A table without a foreign key is or a table that contains a foreign key which can contain NULLS is a strong entity.

ii) Weak Entities

An entity is considered a 'weak' entity, if it has the following properties:

- These tables are existence dependent.
- They cannot exist without entity with which it has a relationship.
- Primary key (PK) is derived from the primary key of the parent entity.

For example; considering the lecturer's spouse table in the UNIVERSITY database referred above. This is a weak entity because its PK is dependent on the lecturer's table whereby without a corresponding lecturer's record, the spouse record could not exist.

6.5.3 Types of Entities

i) Independent Entities

Independent entities, also referred to as Kernels, are the backbone of the database. It is what other tables are based on. Kernels have the following characteristics:

- They are the 'building blocks' of a database
- The primary key may be simple or composite
- The primary key is not a foreign key

They do not depend on another entity for their existence. For example, Lecturer's table and Course table in a UNIVERSITY database referred above.

ii) Dependent Entities

Dependent entities are used to connect two kernels together. Dependent entities have the following characteristics:

- They are said to be existent dependent on two or more tables.
- Many to many relationships become associative tables with at least two foreign keys.
- They may contain other attributes.
- The foreign key identifies each associated table.

There are three options for the primary key for dependent entities:

- Use a composite of foreign keys of associated tables if unique
- Use a composite of foreign keys and qualifying column
- Create a new simple primary key

iii) Characteristic Entities

Characteristic entities provide more information about another table. These entities have the following characteristics:

- They represent multi-valued attributes.
- They describe other entities.
- They typically have a one to many relationship.
- The foreign key is used to further identify the characterized table.

Options for primary key for this entity are either of the following:

- Foreign key plus a qualifying column
- Create a new simple primary key

  For example, for the independent entity Lecturer, a characteristics entity may be LecturerPhone as shown below:

  - Lecturer (PFNo, Name, Address, Age, Salary)
  - LecturerPhone(PFNo, Phone)

Check Your Progress 1

Q1: What are the entities in the following statement:

An employee of a specific bank can also open a saving account.

.....

.....

.....

.....

Q2: Identify the strong and weak entities in the following statement:

The information is required to be stored about the employees and their dependents.

.....

.....

.....

.....

Q3: What is a charactristic entity?

.....

.....

.....

.....

6.5.4 Attributes

Each entity is described by a set of attributes, e.g. Lecturer = (PFNo, Name, Address, Age, Salary).

Each attribute has a name, associated with an entity and is associated with a domain of legal values. However the information about attribute domain is not presented on the ER diagram.

In the diagram, each attribute is represented by an oval with a name inside.

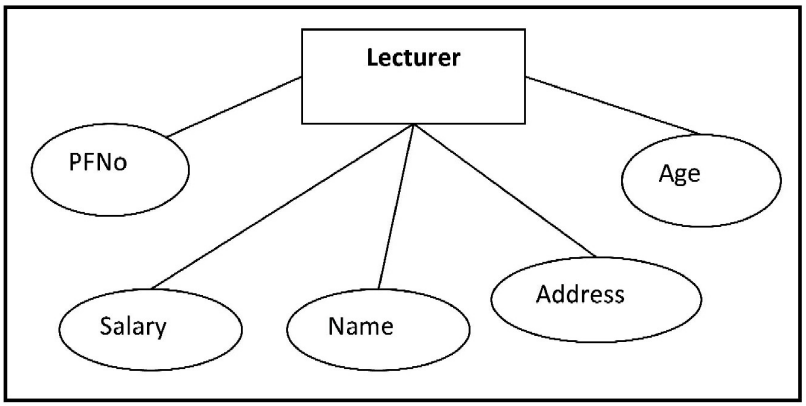


Figure 6.2: Illustration of attributes of 'Lecturer'entity, by G. Mbwete

i) Types of Attributes

There are a few types of attributes you need to be familiar with. Some of these are to be left as is, but some need to be adjusted to facilitate representation in the relational

model. This first section will discuss the types of attributes. Later on we will discuss fixing the attributes to fit correctly into the relational model.

- **Simple Attributes**

Simple attributes are those drawn from the atomic value domains; they are also called single-valued attributes. In the UNIVERSITY database, an example of this would be: PFNo = {908}; Name = {Grace}, Age = {37}

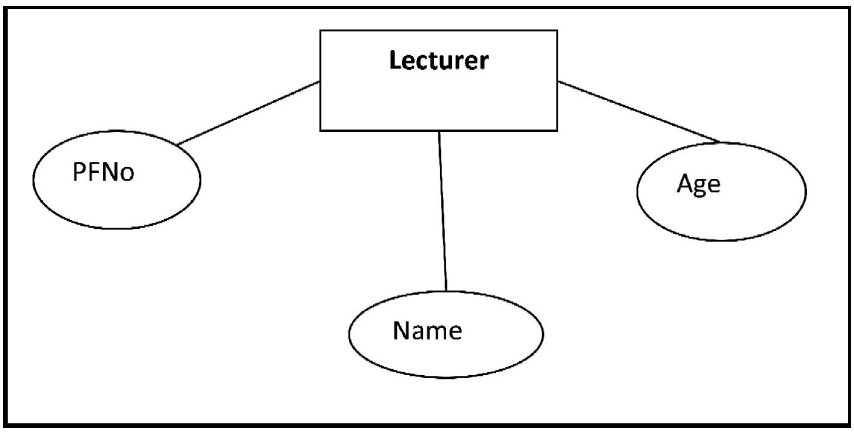


Figure 6.3: Illustration of simple attributes of 'Lecturer' entity, by G. Mbwete

- **Composite Attribute**

Composite attributes are those that consist of a hierarchy of attributes. Using our database example, and shown in Figure 6.4, Address may consist of Number, Street and Suburb. So this would be written as follows:-

Address = {30 + 'Tumaini' + 'Mikocheni'}

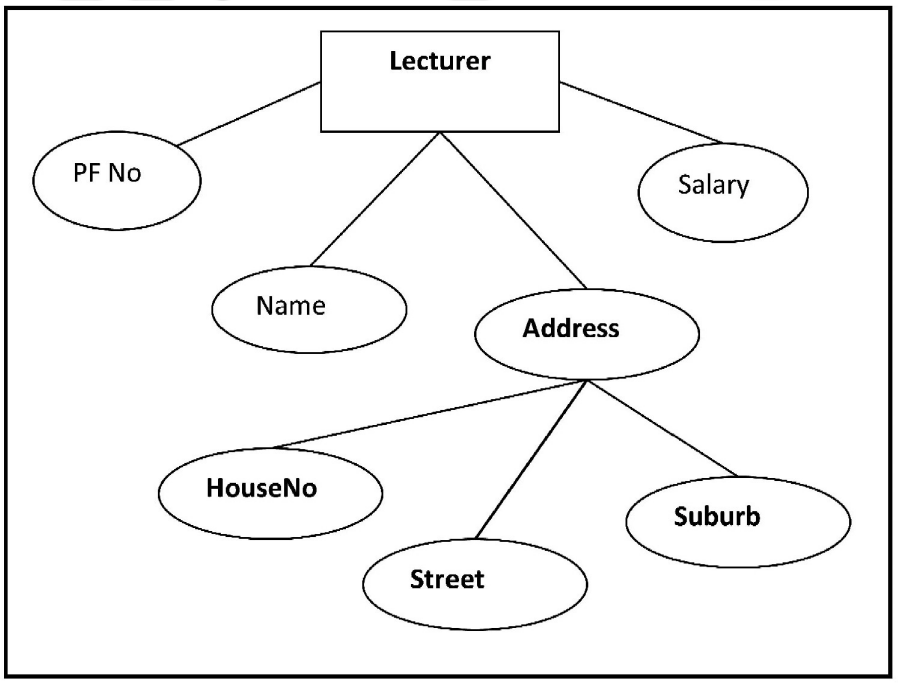


Figure 6.4: Illustration of composite attribute 'address' of 'Lecturer' entity, by G. Mbwete.

- **Multivalued attributes**

Multivalued attributes are attributes that have a set of values for each entity. Examples

of a multivalued attribute from the UNIVERSITY database, as seen in Figure 6.5 below, are the degrees of a lecturer: BSc, MIT and PhD.

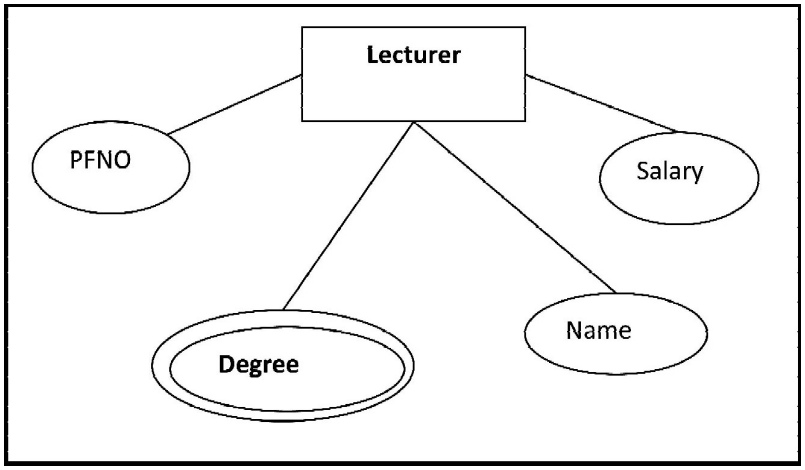


Figure 6.5: Illustration of multi-valued attribute 'degree' of 'Lecturer',by G. Mbwete

- **Derived attributes**

Derived attributes are attributes that contain values calculated from other attributes. An example of this can be seen in Figure 6.6. Age can be derived from the attribute Birthdate. In this situation, Birthdate is called a stored attribute, which is physically saved to the database.

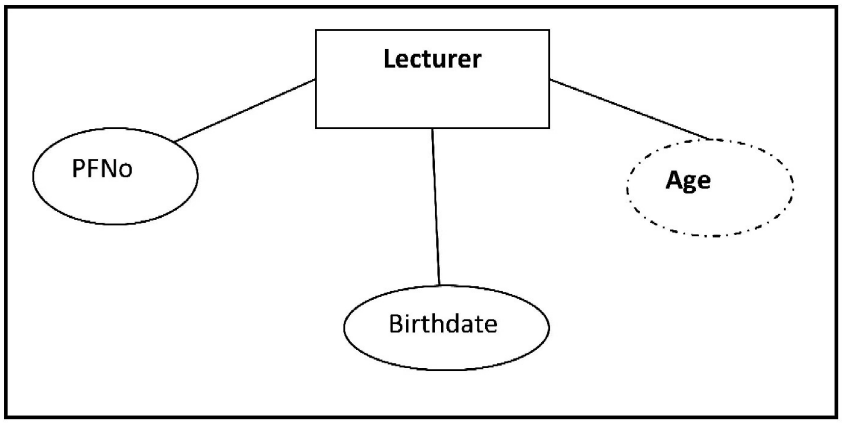


Figure 6.6: Illustration of derived attribute 'Age' of 'Lecturer' entity,byG.Mbwete

6.5.5 Keys

An important constraint on an entity is the key. The key is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set. The following are types of keys:

i) Candidate Key

A candidate key is a simple or composite key that is unique and minimal. It is unique because no two rows in a table may have the same value at any time. It is minimal because every column is necessary in order to attain uniqueness.

From our UNIVERSITY database example, if the entity is **Lecturer** (LID, FirstName, LastName, Address, Phone, BirthDate, Salary, DepartmentID), possible candidate keys are:

- LID
- First Name and Last Name - assuming there is no one else in the company with the same name
- Last Name and DepartmentID - assuming two people with the same last name don't work in the same department

ii) Composite key

A *composite key* is composed of two or more attributes, but it must be minimal. Using the example from the candidate key section, possible composite keys are:

- First Name and Last Name - assuming there is no one else in the company with the same name.
- Last Name and Department ID - assuming two people with the same last name don't work in the same department.

iii) Primary Key

The primary key is a candidate key that is selected by the database designer to be used as an identifying mechanism for the whole entity set. It must uniquely identify tuples in a table and not be null. The primary key is indicated in the ER model by underlining the attribute.

- A candidate key is selected by the designer to uniquely identify tuples in a table. It must not be null.
- A key is chosen by the database designer to be used as an identifying mechanism for the whole entity set. This is referred to as the primary key. This key is indicated by underlining the attribute in the ER model.

In the following example, LID is the primary key:

Lecturer (LID, First Name, Last Name, Address, Phone, BirthDate, Salary, DepartmentID)

iv) Secondary key

A *secondary key* is an attribute used strictly for retrieval purposes (can be composite), for example: Phone and Last Name.

v) Alternate key

Alternate keys are all candidate keys not chosen as the primary key.

vi) Foreign key

A *foreign key (FK)* is an attribute in a table that references the primary key in another table OR it can be null. Both foreign and primary keys must be of the same data type. In the UNIVERSITY database example below, DepartmentID is the foreign key in the relation Lecturer:

Lecturer (LID, First Name, Last Name, Address, Phone, BirthDate, Salary, DepartmentID)

Department (DepartmentID,DepartmentName,DepartmentSpecialities)

Check Your Progress 2

Q1: Identify which of them is simple or composite attributes:

Name, Salary

.....

.....

.....

.....

Q2: Which of the following is multi-valued attribute or a derived attribute?

DurationOf Work in present post, Experience.

.....

.....

.....

.....

Q3: Given the following relation, identify primary key, composite key, and foreign keys.

Student (Id, name, programmecode, aadharno, phone, dateofbirth)

.....

.....

.....

.....

6.6 RELATIONSHIPS

The relationship in ERP connects two or more entities. They are discussed in details in this section.

6.6.1 Introduction

Relationships are the glue that holds the tables together. They are used to connect related information between tables.

Relationship strength is based on how the primary key of a related entity is defined. A weak, or non-identifying, relationship exists if the primary key of the related entity does not contain a primary key component of the parent entity. For example:

- Customer(**CustID**, CustName)
- Order(**OrderID**, CustID, Date)

A strong, or identifying, relationship exists when the primary key of the related entity contains the primary key component of the parent entity. Examples include:

- Course(**CourseCode**, DepartmentID, Description)
- Class(**CrsCode, Section,** ClassTime…)

6.6.2 Types of Relationships

Below are descriptions of the various types of relationships.

i) One to many (1:M) relationship

A one to many (1:M) relationship should be the norm in any relational database design and is found in all relational database environments.

For example, in a UNIVERSITY database; one department has many lecturers. Figure 6.7 shows the relationship (one to many) of one of these lecturers to the department.

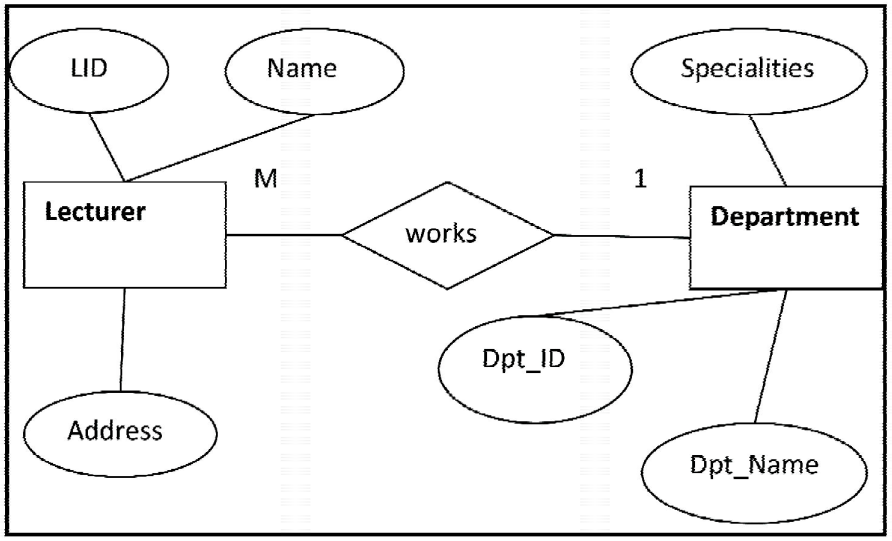


Figure 6.7: Example of a one to many relationship, by G.Mbwete

ii) One-to-One Relationship

Aone to one (1:1) relationship is the relationship of one entity to only one other entity, and vice versa. It should be rare in any relational database design. In fact, it could indicate that two entities actually belong in the same table.

An example from the UNIVERSITY database is one lecturer is associated with one spouse, and one spouse is associated with one lecturer.

iii) Many-to-many relationship

For a many to many relationship, consider the following points:

- It cannot be implemented as such in the relational model.
- It can be changed into two 1:M relationships.
- It can be implemented by breaking up to produce a set of 1:M relationships.
- It involves the implementation of a composite entity.
- Creates two or more 1:M relationships.

- The composite entity table must contain at least the primary keys of the original tables.
- The linking table contains multiple occurrences of the foreign key values.
- Additional attributes may be assigned as needed.
- It can avoid problems inherent in an M:N relationship by creating a composite entity or bridge entity. For example, an employee can work on many projects OR a project can have many employees working on it, depending on the business rules. Or, a student can have many classes and a class can hold many students.

Figure 6.8 shows another aspect of the M:N relationship where a lecturer has different start dates for different academic programmes. Therefore, you need a JOIN table that contains the LID, PrgrmCode and StartDate.

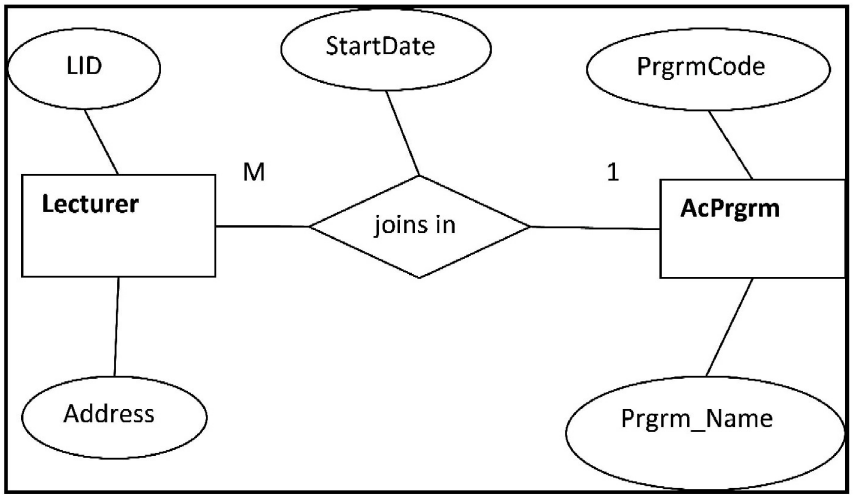


Figure 6.8: Illustration of where lecturer has different start dates for different Programmes, by G.Mbwete

Example of mapping an M:N binary relationship type

- For each M:N binary relationship, identify two relations.
- A and B represent two entity types participating in R.
- Create a new relation S to represent R.
- S needs to contain the PKs of A and B. These together can be the PK in the S table OR these together with another simple attribute in the new table R can be the PK.
- The combination of the primary keys (A and B) will make the primary key of S.

iv) Unary relationship (recursive)

A *unary relationship*, also called *recursive*, is one in which a relationship exists between occurrences of the same entity set. In this relationship, the primary and foreign keys are the same, but they represent two entities with different roles. See Figure 6.8 for an example. For some entities in a unary relationship, a separate column can be created that refers to the primary key of the same entity set.

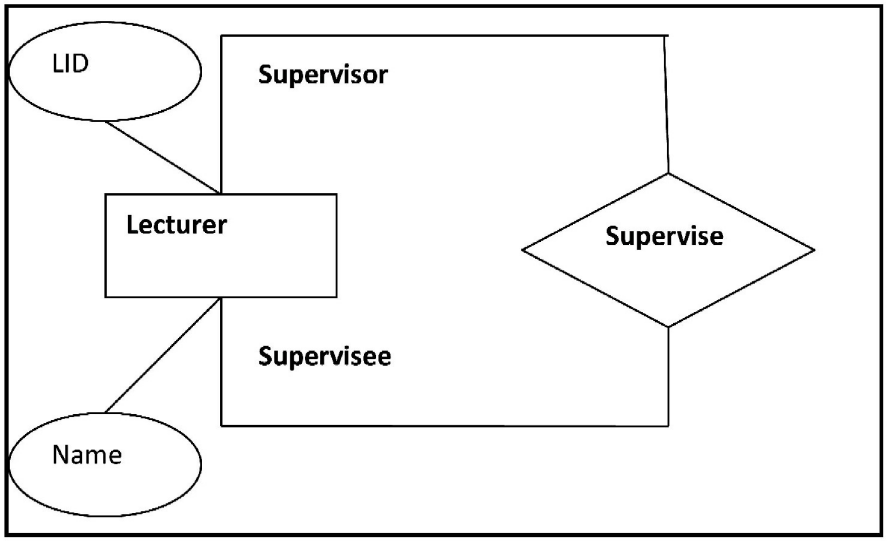


Figure 6.8: Illustration of where lecturer has different two roles as a supervisor for junior staff and supervisee for his/her immediate supervisor, by G. Mbwete

v) Ternary Relationship

A *ternary relationship* is a relationship type that involves many to many relationships between three tables. The primary key of the new relation is a combination of the primary keys of the participating entities that hold the N (many) side. In most cases of an n-ary relationship, all the participating entities hold a many side.

Check Your Progress 3

Q1: Give an example of 1:M relationship.

.....

.....

.....

.....

Q2: Give an example of 1:1 relationship.

.....

.....

.....

.....

Q3: Give an example of many to many relationship

.....

.....

.....

.....

6.7 VIDEO LECTURE

https://tinyurl.com/h22xxdl

https://tinyurl.com/htfkunh

https://tinyurl.com/jbbrq9g

6.8 ACTIVITY


Group Activity

Activity 6.0

ER Modelling Exercises

Motivation: To become conversant with basic ER Modelling techniques.

Resources: Unit5 and 6 learning materials and ER Diagram Tool.

What to do:

Read the following scenario carefully.

A manufacturing company produces products. Product information stored is product name, id, and quantity on hand. These products are made up of many components. Each component can be supplied by one or more suppliers. Component information kept is component id, name, description, suppliers who supply them, and which products they are used in. Create an ERD to show how you would track this information. Show entity names,

primary keys, attributes for each entity, relationships between the entities and cardinality.

How to do it:

For the above scenario, draw an entity-relationship model diagram that captures the given requirements. Once done, **validate the model against the requirements** to make sure nothing was missed. Remember that the four entity-relationship elements we want to pull out of these requirements are: entities, attributes, identifiers, and relationships.

Duration: Expect to spend about 2 hour on this activity.

Feedback: The learner should submit this activity for assessment to the course instructor and get a feedback on this activity.

Group Activity

Activity 6 .1

ER Modelling Exercises

Motivation: To become conversant with basic ER Modelling techniques.

Resources: Unit 5 and 6 learning materials and ER Diagram Tool.

What to do:

Read the following scenario carefully;

The SSR marketing department needs a database to keep track of its product line. The department creates new and unique razor models all of the time (e.g. The Ripper, Slashmatic, Yankie 3000, Lady Yankie, ElectrothrobYankie Deluxe). Each model features a matched pair of a cartridge and handle. A model may have alternate handles (e.g. battery powered vibration, metallic blue finish, Hello Kitty™ decal) but only uses one cartridge type. To inflate costs, neither handles nor cartridges are reused for new models. Part numbers for the various handles and cartridges are recorded. The company packages together sets of two cartridges and a handle under the model name. The company also packages sets of eight replacement cartridges under the model name. The retail price of each package is recorded.

How to do it:

For the above scenario, draw an entity-relationship model diagram that captures the given requirements. Once done, validate the model against the requirements to make sure nothing was missed. Remember that the four entity-relationship elements we want to pull out of these requirements are: entities, attributes, identifiers, and relationships.

Duration: Expect to spend about 2 hour on this activity.

Feedback: The learner should submit this activity for assessment to the course instructor and get a feedback on this activity.

6.9 SUMMARY

In this unit you learned about Entity relationship Model which helps in the logical design stage of a relational database. The unit content has given details on entities and its types, attributes and its characteristics and finally; relationships which are the basis/ connectivity between entities/tables in the database.

6.10 ANSWERS

Check Your Progress 1

Ans 1: Employee, Bank, Saving Account.

Ans 2: Strong entity- Employee

Weak entitity-Dependents of an employee

Ans 3: Which provides more information about other relations. e.g. multivalued attribute based entities.

Check Your Progress 2

Ans 1: Name, if stored as first name, middle name and last name separately, then it is composite attribute.

Salary is a simple attribute.

Ans 2: Experience is multi-valued attribute, as a person may have multiple work experiences

Durationofwork in present post can be compute from date of joining, therefore, is a derived attribute.

Ans 3: Two candidate keys may be

- Id
- AadharNo

The id may be selected as the primary key.

One of the composite key may be phone + dateofbirth assuming no twins have joined.

Programme code is the foreign key to a programme relation.

Check Your Progress 3

Ans 1: A staff member is assigned to one department only, and one department may have many staff members.

Ans 2: At a specific time, only one employee can be the CEO of the company.

Ans 3: A student may join for many subjects and one subject has many students.

6.11 CASE STUDY

Aim: This section contains several case studies where they are aimed at motivating the

learner to develop techniques of converting user requirements into an ER Diagram for each case.

Duration: Expect to spend about 1.30 hour on this activity.

Feedback: These case studies should be submitted to the course instructor for assessment and feedback.

Case Study I

UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number (unique), weight, dimensions, insurance amount, destination, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, uniqueID, and address. Shipped items make their way to their destination via one or more standard UPS transportation events (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute. Please create an Entity Relationship diagram that captures this information about the UPS system. Be certain to indicate identifiers and cardinality constraints.

Case Study II

Acar dealership sells both new and used cars, and it operates a service facility. Base your design on the following business rules:

- A salesperson may sell many cars, but each car is sold by only one salesperson.
- A customer may buy many cars, but each car is sold to only one customer.
- A salesperson writes a single invoice for each car he or she buys.
- A customer gets an invoice for each car he or she buys.
- A customer may come in just to have his or her car serviced, that is, one need not buy a car to be classified as a customer.
- When a customer takes one or more cars in for repair or service, one service ticket is written for each car.
- The car dealership maintains a service history for each of the cars serviced. The service records are referenced by the car's serial number.
- A car brought in for service can be worked on by many mechanics, and each mechanic may work on many cars.
- A car that is serviced may or may not need parts. (For example, adjusting a carburetor or cleaning a fuel injector nozzle does not require the use of parts).

Case Study III

Secretive Sci-Fi Star Maps

- This company creates and sells maps that indicate the address of select science fiction film stars and also movie shoot locations. Several maps are needed to cover the large Los Angeles region. The map borders don't overlap one other. Each sci-fi star's address is shown on the appropriate map at the proper

coordinates. The film or films that each star has appeared in are recorded, along with the role played.

- Customers of SSFSM often want to find all of the stars that appeared in a certain movie (e.g. *Star Trek II, Star Wars IV, The Adventures of Buckaroo Banzai Across the 8th Dimension*).
- Special shoot locations for some sci-fi films are also displayed on the maps.

Case Study IV

Raining Cats 'n Dogs Animal Shelter

The shelter is pretty selective in which animals it takes in, and also what it chooses to record for the two species.

- The animal shelter accepts just cats and dogs.
- For both cats and dogs, the name and gender of the animal is recorded, as well as whether it has been neutered or not.
- For cats, the mother of the cat is noted, if the mother is known to the shelter. For any given mother, the shelter wants to be able to find all descendants through potentially many generations.
- Dogs are classified by breed. Dogs can be of one breed (purebred), two breeds (crossbreed), or just be a mutt. The specific breeds, except in the case of a mutt, are tracked.
- The names, addresses, and telephone numbers of people adopting a dog or a cat (adopters) is recorded.
- The names, addresses, and telephone numbers of people surrendering a dog or a cat (donators) is recorded.
- For both adopters and donators, a reference to the dog or cat they adopted or donated, respectively, is kept.
- Adopters can adopt many dogs or cats, or may just be interested in adoption

Case Study V

Draw an EER diagram of the conceptual schema for another part of a University database, described as follows:

- Academic staff, general staff and students are the only persons at the university.
- Each person is either an academic staff, or a general staff, or a student.
- A person is uniquely identified by a PerId (person's ID), and has a Name, and an Address. An Address is composed of HouseNo, Street , and City.
- A characteristic property of a student is that she/he has at least one Major and one NoOfPts (number of points) for each major.
- An academic staff has a Position and an AcQual (academic qualification).
- A general staff has a GenPos (general position).

- An academic staff teaches at most one course, whereas a student takes at least one course.
- A course is uniquely identified by a CourId (course ID), and has a CourName (course name).
- Each course is taught by at least one academic staff, and can be taken by many students, but there may be courses that are not taken by any students.
- Each course can use more than one textbook, but there may be courses with no textbook.
- A textbook is uniquely identified by the course which uses the book, and by an OrdNo. The attribute OrdNo is the ordinal number of the book in the list of the textbooks of a particular course. Abook also has a Title.

6.12 FURTHER READINGS

1. Entity relationship model. (2015, October 19). Retrieved May 23, 2016, from Open TextBooks for Hongkong, http://www.opentextbooks.org.hk/ditatopic/30738
2. Eng, N., & Watt, A. (2013). Database design - 2nd edition. Retrieved May 27, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/chapter-10-er-modelling/

Download this book for free at http://open.bccampus.ca

3. 1 .264 Lecture 5 - Data Modelling. Retrieved September 19, 2016, from MIT OCW, http://dspace.mit.edu/bitstream/handle/1721.1/39819/1-264JFall-2004/NR/rdonlyres/Civil-and-Environmental-Engineering/1-264JFall-2004/E2D7822C-FF88-423C-BC30-3A6694480274/0/le5_f2004solns_h.pdf

Rogers, D. (2002). COMP210: Week 1 exercises: Entity-relationship Modelling. Retrieved May 27, 2016, from Yukon College, http://college.yukondude.com/2006_09_comp210/html/note.php?note=01^Exercises^Entity-Relationship_Modelling.tpl

6.13 ATTRUBUTION

The ER Case Studies are authored by David Rogers from Yukon College.

The following material was written by Grace Mbwete:

1. Introduction
2. Summary

UNIT 7 INTEGRITY RULES AND CONSTRAINTS

Structure

7.1 INTRODUCTION

One of the important functionality of a DBMS is to enable the specification of integrity constraints and to enforce them. Constraints are useful because they allow a designer to specify the semantics of data in the database. Constraints are the rules that force DBMSs to check that data satisfies the semantics. The concepts which will be covered in this unit, will assist the student in understanding and defining the database constraints during the practical sessions relating to this course.

7.2 OBJECTIVES

Upon completion of this unit you will be able to:

- *Describe* the basic concepts of data integrity rules.
- *Specify* integrity constraints and how to enforce them.
- *Identify* business rules when gathering user requirements.

7.3 TERMINOLOGIES

Constraints : The rules that force DBMSs to check that data satisfies the semantics.

Entity Integrity	:	Requires that every table have a primary key; neither the primary key, nor any part of it, can contain null values.
Integrity constraints	:	Logical statements that state what data values are or are not allowed and which format is suitable for an attribute.
Business rules	:	Obtained from users when gathering requirements and are used to determine cardinality.
Cardinality	:	Expresses the minimum and maximum number of entity occurrences associated with one occurrence of a related entity.
Identifying relationship	:	Is a condition where the primary key contains the foreign key; indicated in an ERD by a solid line.

7.4 INTEGRITY CONSTRAINTS

Constraints are a very important feature in a relational model. In fact, the relational model supports the well-defined theory of constraints on attributes or tables. Constraints are useful because they allow a designer to specify the semantics of data in the database. *Constraints* are the rules that force DBMSs to check that data satisfies the semantics.

Domain restricts the values of attributes in the relation and is a constraint of the relational model. However, there are real-world semantics for data that cannot be specified if used only with domain constraints. We need more specific ways to state what data values are or are not allowed and which format is suitable for an attribute. For example, the Employee ID (EID) must be unique or the employee Birthdate is in the range [Jan 1, 1950, Jan 1, 2000]. Such information is provided in logical statements called integrity constraints. There are several kinds of integrity constraints, as detailed below:

7.4.1 Entity Integrity

To ensure *entity integrity*, it is required that every table have a primary key. Neither the Primary Key (PK) nor any part of it can contain null values. This is because null values for the primary key mean we cannot identify some rows. For example, in the EMPLOYEE table, Phone cannot be a primary key since some people may not have a telephone.

7.4.2 Referential Integrity

Referential integrity requires that a foreign key must have a matching primary key or it must be null. This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.

Examples of referential integrity constraint in the Customer/Order database of the Company:

- Customer(**CustID**, CustName)
- Order(**OrderID**, CustID, OrderDate)

To ensure that there are no orphan records, we need to enforce referential integrity. An orphan record is one whose foreign key (FK) value is not found in the corresponding

entity - the entity where the PK is located. Recall that a typical join is between a PK and FK.

The referential integrity constraint states that the customer ID (CustID) in the Order table must match a valid CustID in the Customer table. Most relational databases have declarative referential integrity. In other words, when the tables are created the referential integrity constraints are set up.

Here is another example from a Course/Class database:

- Course(**CrsCode**, DeptCode, Description)
- Class(**CrsCode**, Section, ClassTime)

The referential integrity constraint states that CrsCode in the Class table must match a valid CrsCode in the Course table. In this situation, it's not enough that the CrsCode and Section in the Class table make up the PK, we must also enforce referential integrity.

When setting up referential integrity it is important that the PK and FK have the same data types and come from the same domain, otherwise the relational database management system (RDBMS) will not allow the join. RDBMS is a popular database system that is based on the relational model introduced by E. F. Codd of IBM's San Jose Research Laboratory. Relational database systems are easier to use and understand than other database systems.

i) Referential integrity in Microsoft Access

In Microsoft (MS) Access, referential integrity is set up by joining the PK in the Customer table to the CustID in the Order table. See Figure 9.1 for a view of how this is done on the Edit Relationships screen in MS Access.

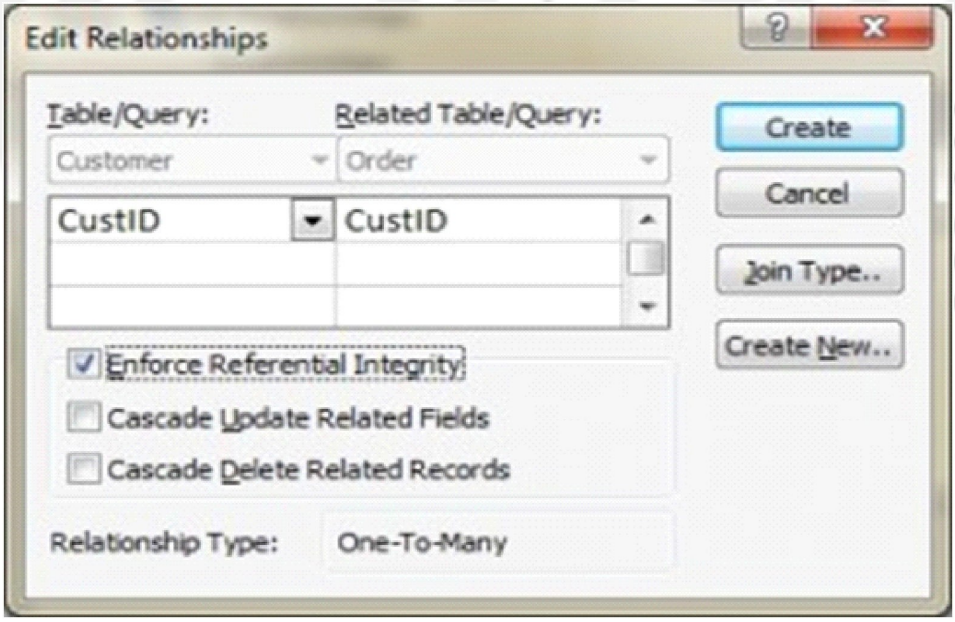


Figure 7.1: Referential access in MS Access, by A. Watt.

Referential integrity using Transact-SQL (MS SQL Server)

When using Transact-SQL, the referential integrity is set when creating the Order table with the FK. Listed below are the statements showing the FK in the Order table referencing the PK in the Customer table.

```
CREATE TABLE Customer
( CustID INTEGER PRIMARY KEY,
CustName CHAR(35) )
CREATE TABLE Orders
( OrderID INTEGER PRIMARY KEY,
CustID INTEGER REFERENCES Customer(CustID),
OrderDate DATETIME )
```

ii) Foreign key rules

Additional foreign key rules may be added when setting referential integrity, such as what to do with the child rows (in the Orders table) when the record with the PK, part of the parent (Customer), is deleted or changed (updated). For example, the Edit Relationships window in MS Access (see Figure 9.1) shows two additional options for FK rules: Cascade Update and Cascade Delete. If these are not selected, the system will prevent the deletion or update of PK values in the parent table (Customer table) if a child record exists. The child record is any record with a matching PK.

In some databases, an additional option exists when selecting the Delete option called Set to Null. In this is chosen, the PK row is deleted, but the FK in the child table is set to NULL. Though this creates an orphan row, it is acceptable.

7.4.3 Enterprise Constraints

Enterprise constraints - sometimes referred to as semantic constraints - are additional rules specified by users or database administrators and can be based on multiple tables. Here are some examples.

- A class can have a maximum of 30 students.
- A teacher can teach a maximum of four classes per semester.
- An employee cannot take part in more than five projects.
- The salary of an employee cannot exceed the salary of the employee's manager.

Check Your Progress 1

Consider the following tables

STUDENT

St-ID	Name	Programme-code

PROGRAMME

Programme-code	Name	Duration

Q1: What may be the Entity integrity constraints?

.....

.....

.....

.....

Q2: Identify foreign key and referential constraints.

.....

.....

.....

.....

Q3: Is there any enterprise constraints in these two tables?

.....

.....

.....

.....

7.5 BUSINESS RULES

Business rules are obtained from users when gathering requirements. The requirements-gathering process is very important, and its results should be verified by the user before the database design is built. If the business rules are incorrect, the design will be incorrect, and ultimately the application built will not function as expected by the users. Some examples of business rules are:

- A teacher can teach many students.
- A class can have a maximum of 35 students.
- A course can be taught many times, but by only one instructor.
- Not all teachers teach classes.

Cardinality and connectivity

Business rules are used to determine cardinality and connectivity. *Cardinality* describes the relationship between two data tables by expressing the minimum and maximum number of entity occurrences associated with one occurrence of a related entity. In Figure 9.2, you can see that cardinality is represented by the innermost markings on the relationship symbol. In this figure, the cardinality is 0 (zero) on the right and 1 (one) on the left.

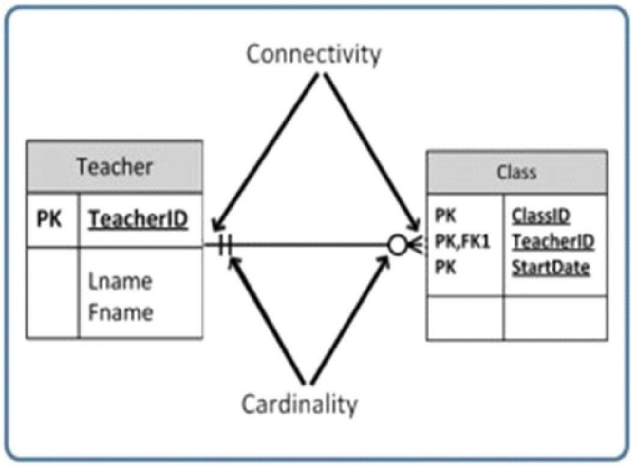


Figure 7.2: Position of connectivity and cardinality on a relationship symbol, by A. Watt

The outermost symbol of the relationship symbol, on the other hand, represents the connectivity between the two tables. Connectivity is the relationship between two tables, e.g., one to one or one to many. The only time it is zero is when the FK can be null. When it comes to participation, there are three options to the relationship between these entities: either 0 (zero), 1 (one) or many. In Figure 9.2, for example, the connectivity is 1 (one) on the outer, left-hand side of this line and many on the outer, right-hand side.

Figure 7.3: One to many relationship

In Figure 7.4 both inner (representing cardinality) and outer (representing connectivity) markers are shown. The left side of this symbol is read as minimum 1 and maximum 1. On the right side, it is read as: minimum 1 and maximum many.

Figure 7.4: Illustration of cardinality

7.6 RELATIONSHIP TYPES

The line that connects two tables, in an ERD, indicates the relationship type between the tables: either identifying or non-identifying. An identifying relationship will have a solid line (where the PK contains the FK). A non-identifying relationship is indicated by a broken line and does not contain the FK in the PK. Please refer to section in Unit 6 that discusses weak and strong relationships for more explanation.

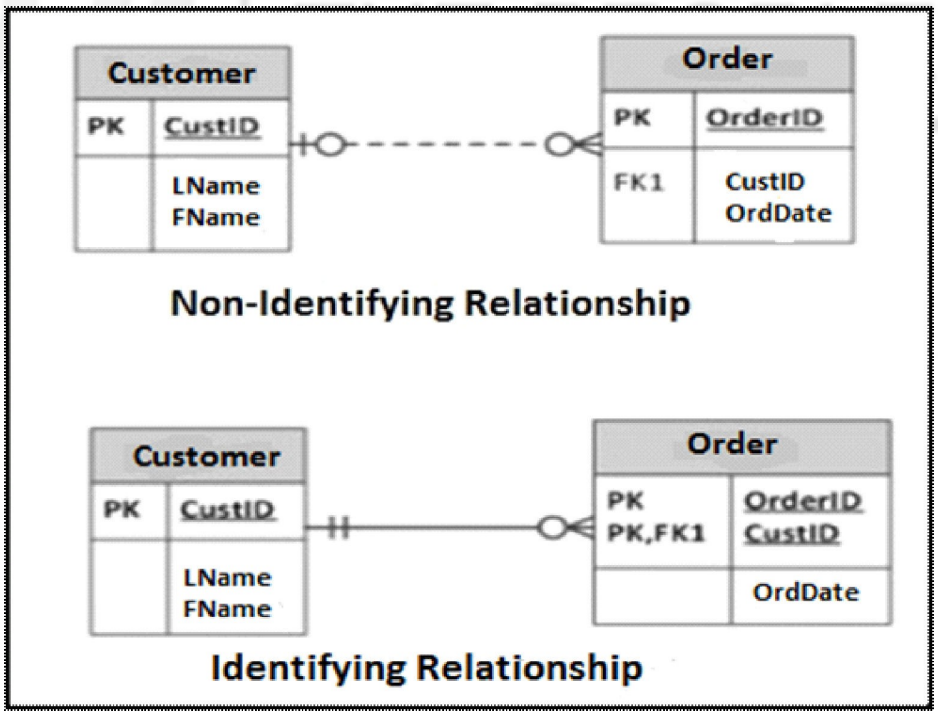


Figure 7.5: Identifying and non-identifying relationship, by A. Watt.

7.6.1 Optional relationships

In an optional relationship, the FK can be null or the parent table does not need to have a corresponding child table occurrence. The symbol, shown in Figure 7.6 below, illustrates one type with a zero and three prongs (indicating many) which is interpreted as zero OR many.

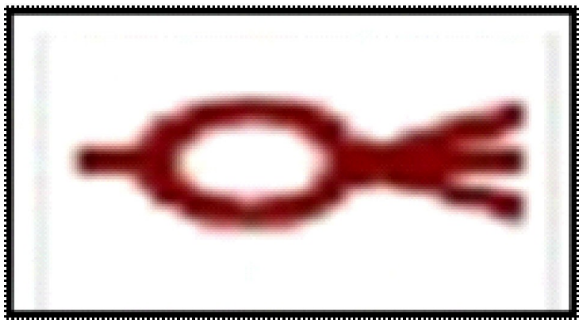


Figure 7.6: Optional relationship, by A. Watt.

For example, if you look at the Order table on the right-hand side of Figure 7.5, you'll notice that a customer doesn't need to place an order to be a customer. In other words, the **many side** is optional.

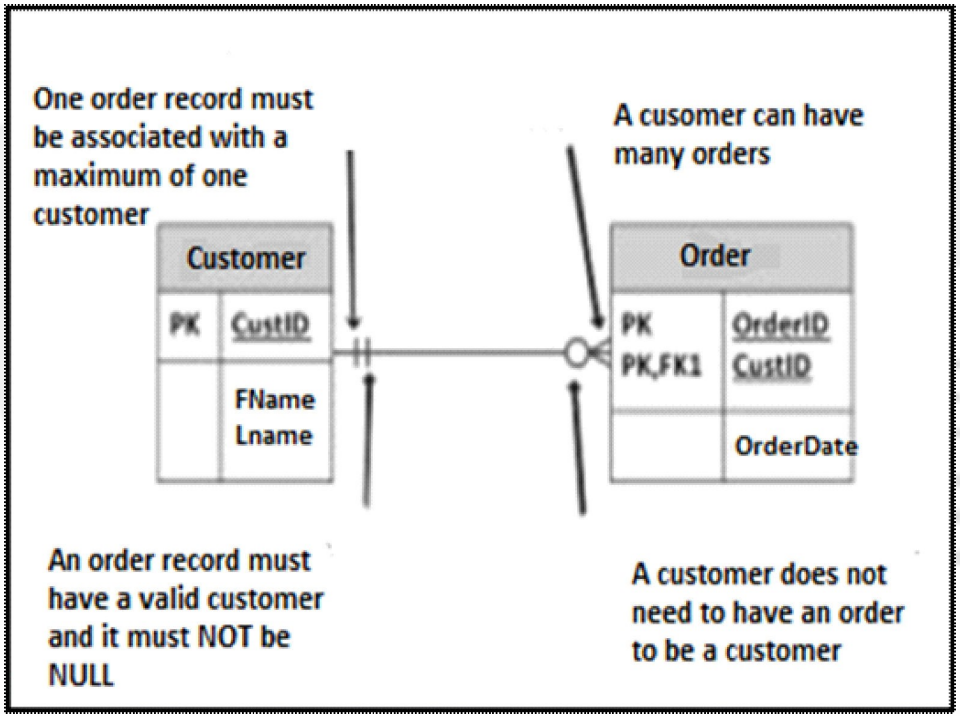


Figure 7.7: Example usage of a zero to many optional relationship symbol, by A. Watt.

The relationship symbol in Figure 7.7 above, can also be read as follows:

i. Left side: The order entity must contain a minimum of one related entity in the Customer table and a maximum of one related entity.

ii. Right side: A customer can place a minimum of zero orders or a maximum of many orders.

Figure 7.8 , shows another type of optional relationship symbol with a zero and one, meaning zero OR one. The **one side** is optional.

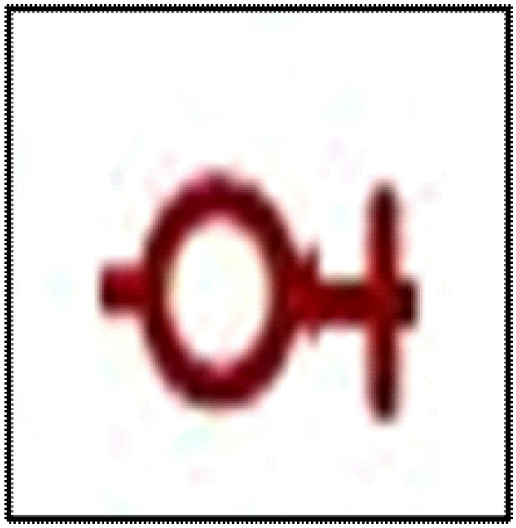


Figure 7.8: Optional relationship with 'optional' one side, by A. Watt

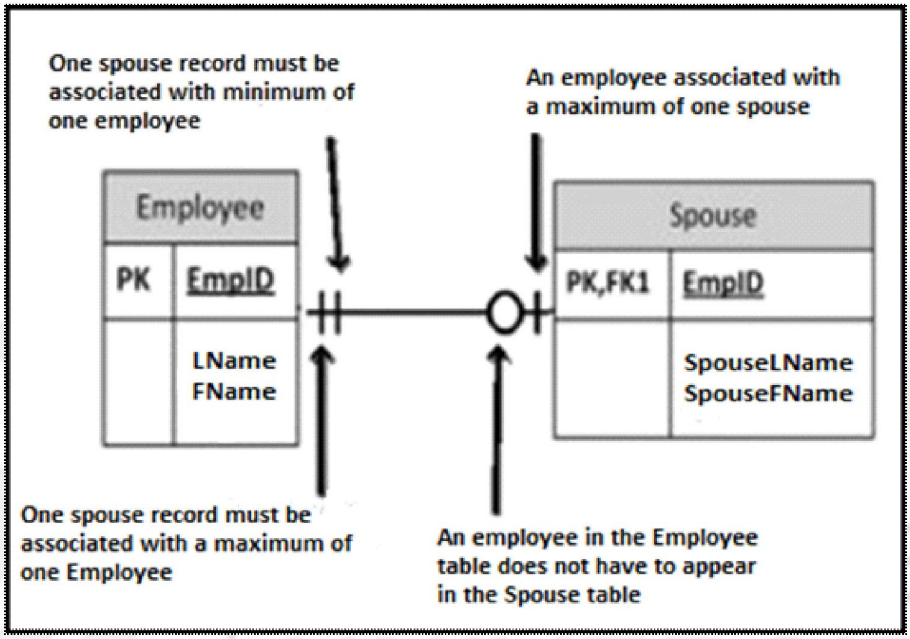


Figure 7.9: Example usage of a zero to one optional relationship symbol, by A. Watt.

7.6.2 Mandatory relationships

In a *mandatory relationship*, one entity occurrence requires a corresponding entity occurrence. The symbol for this relationship shows *one and only one* as shown in Figure 7.10. The one side is mandatory.

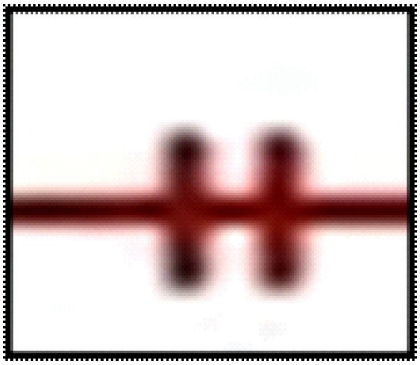


Figure 7.10: illustration of 'mandatory' relationship, by A. Watt

See Figure 7.11 as an example of how the one and only one mandatory symbol is used.

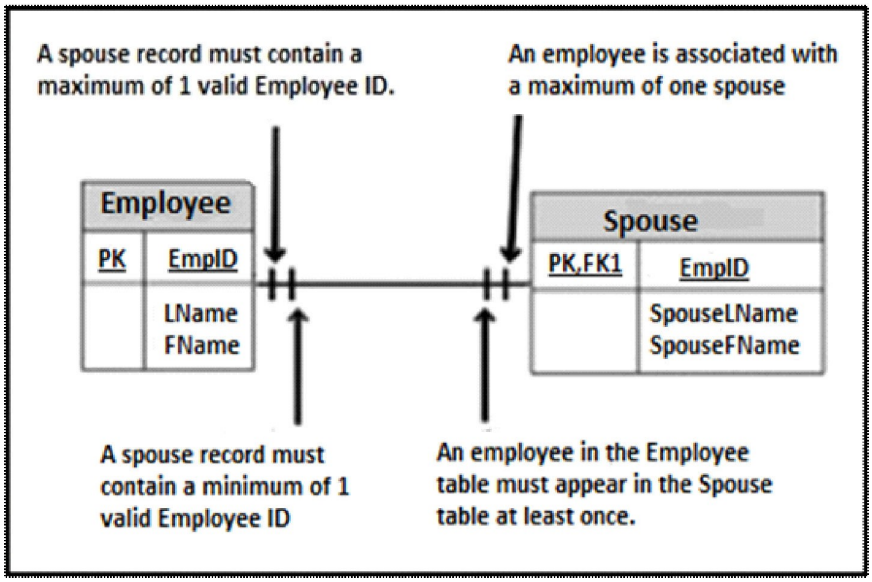


Figure 7.11: Example of a one and only one mandatory relationship symbol, by A. Watt.

Figure 7.12 illustrates what a one to many relationship symbol looks like where the many side is mandatory.

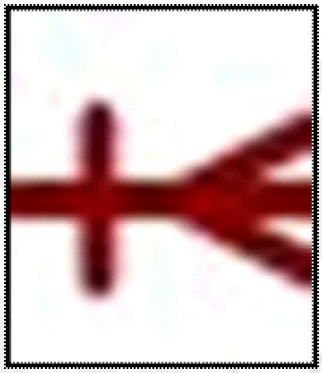


Figure 7.12: One to many with 'many' as mandatory, by A. Watt

Refer to Figure 7.13 for an example of how the one to many symbol may be used.

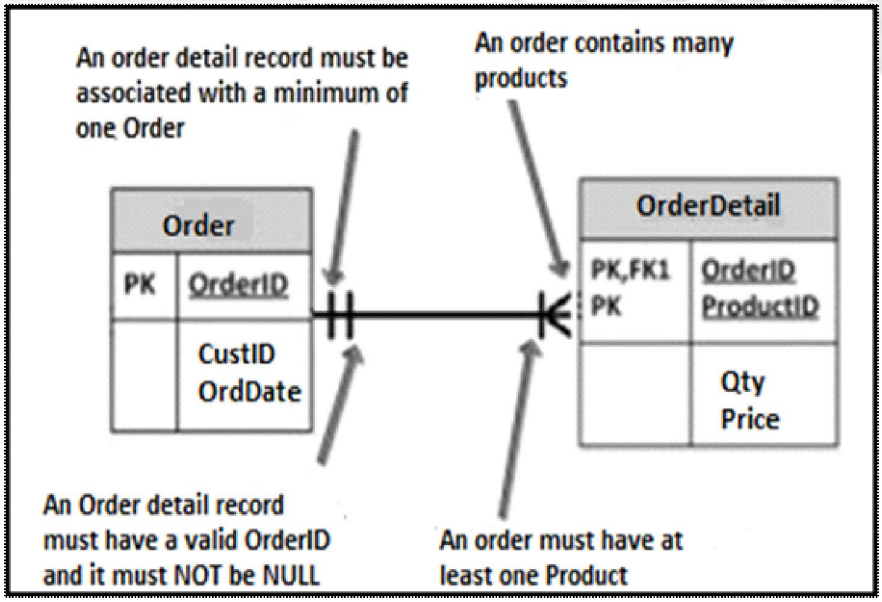


Figure 7.13: Example of a one to many mandatory relationship symbol, by A. Watt.

So far you have seen that the innermost side of a relationship symbol (on the left-side of the symbol in Figure 7.14) can have a 0 (zero) cardinality and a connectivity of many (shown on the right-side of the symbol in Figure 7.14), or one (not shown).

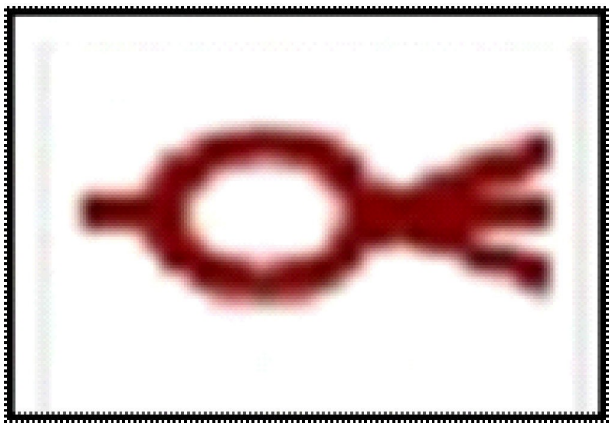


Figure 7.14: Illustration of connectivity and cardinality, by A. Watt

However, it cannot have a connectivity of 0 (zero), as displayed in Figure 7.15. The connectivity can only be 1.

Figure 7.15: Example of connectivity, by A. watt

The connectivity symbols show maximums. So if you think about it logically, if the connectivity symbol on the left side shows 0 (zero), then there would be no connection between the tables. The way to read a relationship symbol, such as the one in Figure 7.16, is as follows.

- The CustID in the Order table must also be found in the Customer table a minimum of 0 and a maximum of 1 time.
- The 0 means that the CustID in the Order table may be null.
- The left-most 1 (right before the 0 representing connectivity) says that if there is a CustID in the Order table, it can only be in the Customer table once.

When you see the 0 symbol for cardinality, you can assume two things:

i. The FK in the Order table allows nulls, and

ii. The FK is not part of the PK since PKs must not contain null values.

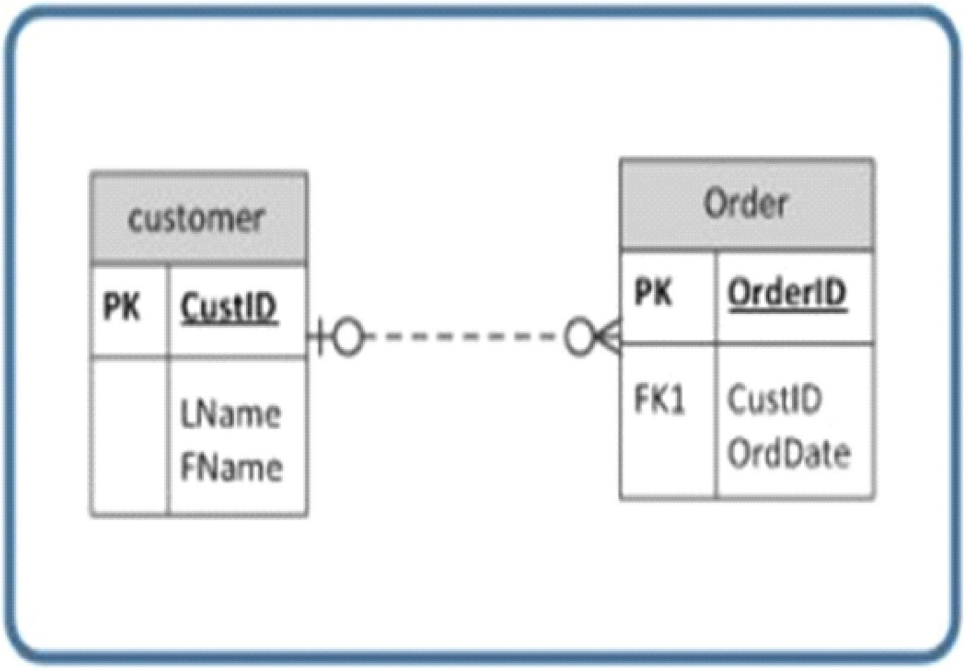


Figure 7.16: The relationship between a Customer table and an Order table, by A. Watt.

Check Your Progress 2

Q1: How will you represent the following using an ERD?

- A student must enrol in exactly one programme.
- A programme may have 0 or many students.

.....

.....

.....

.....

Q2: How can you represent that a student data must have exactly one parent name?

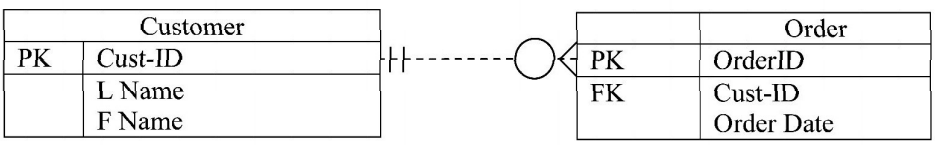
.....

.....

.....

.....

Q3: What is the meaning of following ERD?



.....

.....

.....

.....

7.8 VIDEO LECTURE

https://tinyurl.com/j7a3jo5

7.8 ACTIVITY

Activity 7.0

Integrity Rules and Constraints

Motivation: Demonstrate the basic concepts on data integrity and constraints.

Resources: Internet access and Unit 7 learning materials.

What to do:

Read the following description and then answer questions 1-5 at the end.

The swim club database in Figure 7.17 has been designed to hold information about students who are enrolled in swim classes. The following information is stored: students, enrollment, swim classes, pools where classes are held, instructors for the classes, and various levels of swim classes. Use Figure 7.17 to answer questions 1 to 5.

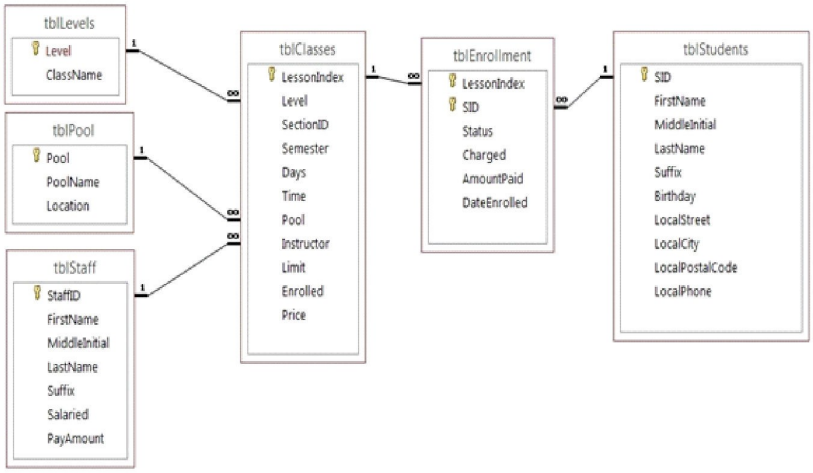


Figure 7.17. ERD for questions 1-5. (Diagram by A. Watt.)

The primary keys are identified below. The following data types should be defined in MySQL.

tblLevels

Level - Identity PK

ClassName - text 20 - nulls are not allowed

tblPool

Pool - Identity PK

PoolName - text 20 - nulls are not allowed

Location - text 30

tblStaff

StaffID - Identity PK

FirstName - text 20

MiddleInitial - text 3

LastName - text 30

Suffix - text 3

Salaried - Bit

PayAmount - money

tblClasses

LessonIndex - Identity PK

Level - Integer FK

SectionID - Integer

Semester - TinyInt

Days - text 20

Time - datetime (formatted for time)

Pool - Integer FK

Instructor - Integer FK

Limit - TinyInt

Enrolled - TinyInt

Price - money

tblEnrollment

LessonIndex - Integer FK

SID - Integer FK (LessonIndex and SID) Primary Key

Status - text 30

Charged - bit

AmountPaid - money

DateEnrolled - datetime

tblStudents

SID - Identity PK

FirstName - text 20

MiddleInitial - text 3

LastName - text 30

Suffix - text 3

Birthday - datetime

LocalStreet - text 30

LocalCity - text 20

LocalPostalCode - text 6

LocalPhone - text 10

Implement this schema in My SQL or MS-Access (you will need to pick comparable data types). Submit a screenshot of your ERD in the database.

1. Explain the relationship rules for each relationship (e.g., tblEnrollment and tblStudents: Astudent can enroll in many classes).
2. Identify cardinality for each relationship, assuming the following rules:
   i. A pool may or may not ever have a class.
   ii. The levels table must always be associated with at least one class.
   iii. The staff table may not have ever taught a class.
   iv. All students must be enrolled in at least one class.
   v. The class must have students enrolled in it.
   vi. The class must have a valid pool.
   vii. The class may not have an instructor assigned.

The class must always be associated with an existing level.

i) Which tables are weak and which tables are strong (covered in an earlier chapter)?

ii) Which of the tables are non-identifying and which are identifying

Duration: Expect to spend about 2 hour on this activity.

Feedback: This activity should be submitted to the course instructor for assessment and feedback.

7.9 SUMMARY

In this unit you learned about Data Integrity and Constraints. You have learnt about entity integrity and referential integrity. In addition, the unit content has also covered different types of relationships. Data Integrity and Constraints concepts assist the database designer especially in relational databases to ensure that the data is accurate and consistent.

7.10 ANSWERS

Check Your Progress 1

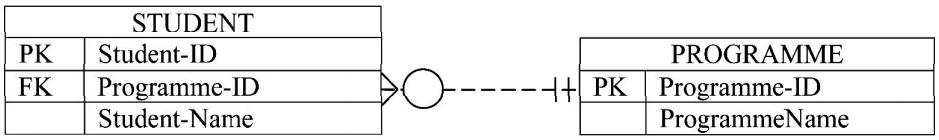
Ans 1: Expected primary keys are St-ID in STUDENT table and Programme-code in PROGRAMME table. The entity integrity states that St-ID in STUDENT table and Programme-code in PROGRAMME table should not left NULL.

Ans 2: Programme-code field in STUDENT table is a foreign key. It references programme-code of PROGRAMME table. The Programme-code of STUDENT table can take only those values which are contained in Programme-code of PROGRAMME table.

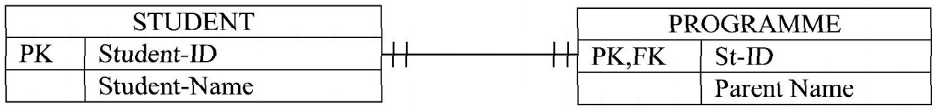
Ans 3: One constraint may be that a student can join only one programme at a time. But it will require more information to be recoded.

Check Your Progress 2

Ans 1.



Ans 2:



Ans 3: The Cust-ID in order table must be found in customer table exactly in one record.

– Customer can give 0 or more orders.

7.11 FURTHER READING

1. Eng, N., & Watt, A. (2013). Database design - 2nd edition. Retrieved May 19, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/chapter-9-integrity-rules-and-constraints/

Download this book for free at http://open.bccampus.ca

2. Integrity rules and constraints. (2015, October 19). Retrieved April 15, 2016, from Open TextBooks for Hongkong, http://www.opentextbooks.org.hk/ditatopic/30764

7.12 ATTRUBUTION

The content of this unit of the Learning Material - Introduction to Databases (including its images, unless otherwise noted) is a derivative copy of materials from the book Database Design by Adrienne Watt and Nelson Eng licensed under **Creative Commons Attribution 4.0 International License**.

Download this book for free at http://open.bccampus.ca

The following material was written by Grace Mbwete:

1. Introduction
2. Summary?

UNIT 8 RELATIONAL DESIGN AND REDUNDANCY

Structure

8.1 INTRODUCTION

A good relational database design must capture all of the necessary attributes and associations. The design should do this with a minimal amount of stored information and no redundant data. In database design, redundancy is generally undesirable because it causes problems maintaining consistency after updates. In this unit, you will be introduced to the basic concepts of data redundancy in relation to database design and data anomalies.

8.2 OBJECTIVES

Upon completion of this unit you should be able to:

- *Explain* the basic concepts of data redundancy in database design.
- *Compare* and contrast different types of data anomalies.
- *Apply* various techniques in removing data anomalies during database design.

8.3 TERMINOLOGIES

Deletion anomaly : Occurs when you delete a record that may contain attributes that shouldn't be deleted.

Functional Dependency (FD) : Describes how individual attributes are related

Insertion anomaly : Occurs when you are inserting inconsistent information into a table.

Join : Used when you need to obtain information based on two related tables.

8.4 DATAREDUNDANCY

In database design, redundancy is generally undesirable because it causes problems maintaining consistency after updates. However, redundancy can sometimes lead to performance improvements; for example, when redundancy can be used in place of a join to connect data. A join is used when you need to obtain information based on two related tables.

Consider table 8.1 (Example of 'Customer' bank account details in a certain bank) where 'customer 1313131' is displayed twice, once for account no. A-101 and again for account A-102. In this case, the customer number is not redundant, although there are deletion anomalies with the table. Having a separate customer table would solve this problem. However, if a branch address were to change, it would have to be updated in multiple places. On the other hand, if the customer number was left in the table as is, then you wouldn't need a branch table and no join would be required, and performance is improved. Table 8.1 below illustrates redundancy used with bank accounts and branches.

Table 8.1: Bank Accounts

AccountNo	Bal	Customer	Branch	Address	Assets
A-305	350	1234567	RoundHill	Horseneck	8000000
A-101	*500*	*1313131*	*Downtown*	*Brooklyn*	*9000000*
A-102	*400*	*1313131*	*Perryridge*	*Horseneck*	*1700000*
A-113	600	9876543	Roundhill	Horseneck	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Ato	2100000

8.5 DATAANOMALIES

Data anomalies are problems that can occur in poorly planned databases where all data and/or information are stored in one file. An example is a file-based systems. There are three types of data anomalies; namely, insertion anomaly, deletion anomaly and update anomaly.

8.5.1 Insertion Anomaly

An insertion anomaly occurs when you are inserting inconsistent information into a

table. When you insert a new record, such as account no. A-306 in table 8.2, you need to check that the branch data is consistent with existing rows.

Table 8.2: Insertion anomaly - 'Insert account A-306 at Roundhill'

AccountNo	Bal	Customer	Branch	Address	Assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Roundhill	Horseneck	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Ato	2100000
A-305	350	1234567	RoundHill	Horseneck	8000000
A-306	800	1111111	Roundhill	Horseneck	8000000

8.5.2 Update Anomaly

If a branch changes address, such as the Round Hill branch in table 8.3, we need to update all rows referring to that branch. Changing existing information incorrectly is called an update anomaly.

Table 8.3: Update anomaly - 'Update address for account A-113 at Roundhill'

AccountNo	Bal	Customer	Branch	Address	Assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Roundhill	Palo Ato	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Ato	2100000
A-305	350	1234567	RoundHill	Horseneck	8000000

8.5.3 Deletion Anomaly

A deletion anomaly occurs when you delete a record that may contain attributes that shouldn't be deleted. For instance, if we remove information about the last account at a branch, such as account A-101 at the Downtown branch in table 8.4, all of the information of Downtown branch disappears. Compare and contrast the two tables below which illustrate the concept of deletion anomaly. Table 8.4 (before deletion of account A-101) and table 8.5 (after deletion of account A-101).

Table 8.4: Before deletion of A-101

AccountNo	Bal	Customer	Branch	Address	Assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Roundhill	Horseneck	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Ato	2100000
A-305	350	1234567	RoundHill	Horseneck	8000000

Table 8.5: Illustration of deletion anomaly

AccountNo	Bal	Customer	Branch	Address	Assets
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Roundhill	Horseneck	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Manus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Ato	2100000
A-305	350	1234567	RoundHill	Horseneck	8000000

The problem with deleting the A-101 row is we don't know where the Downtown branch is located and we lose all information regarding customer 1313131. To avoid these kinds of update or deletion problems, you need to decompose the original table into several smaller tables where each table has minimal overlap with other tables. The idea of the decomposition is to ensure that each table contains information of a single entity.

Each bank account table must contain information about one entity only, such as the Branch or Customer, as displayed in Figure 8.5.

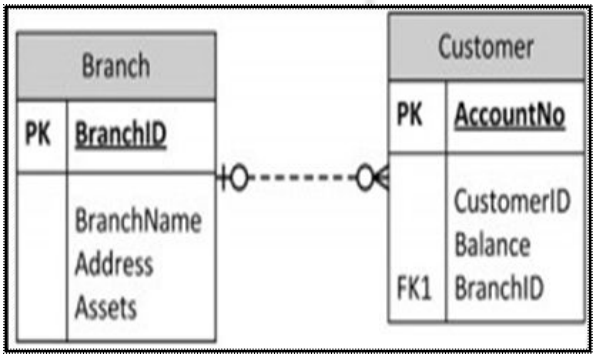


Figure 8.5. Examples of bank account tables that contain one entity each, by A. Watt

Following this practice will ensure that when branch information is added or updated it will only affect one record. So, when customer information is added or deleted, the branch information will not be accidentally modified or incorrectly recorded.

Example: employee project table and anomalies

Table 8.6 shows an example of an employee project table. From this table, we can assume that:

i. EmpID and ProjectID are a composite PK.

ii. Project ID determines Budget (i.e., Project P1 has a budget of 32 hours).

Table 8.6: Example of an employee-project table

EmpID	Budget	ProjectID	Hours
S75	32	P1	7
S75	40	P2	3
S79	32	P1	4
S79	27	P3	1
S80	40	P2	5
	17	P4	

Next, let's look at some possible anomalies that might occur with this table during the following steps:

i. Action: Add row {S85,35,P1,9}

ii. Problem: There are two tuples with conflicting budgets

iii. Action: Delete tuple {S79, 27, P3, 1}

iv. Problem: Step #3 deletes the budget for project P3

v. Action: Update tuple {S75, 32, P1, 7} to {S75, 35, P1, 7}

vi. Problem: Step #5 creates two tuples with different values for project P1's budget

vii. Solution: Create a separate table, each, for Projects and Employees, as shown in Figure 8.2

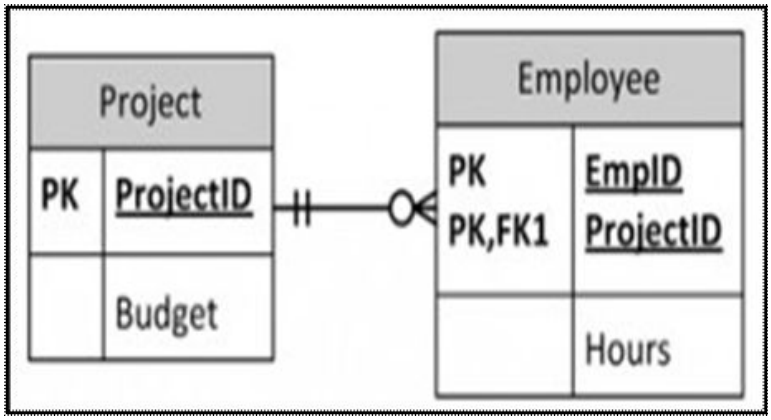


Figure 8.2: Solution: separate tables for Project and Employee, by A.

Check Your Progress 1

Consider the following table.

Student-ID	Student-Name	Programme-ID	Programme-Name
S01	XYZ	P1	MCA
S02	ABC	P2	MBA
S03	DEF	P1	MCA

Q1: Is there any insertion anomaly in the table as above?

.....

.....

.....

.....

Q2: Is there any deletion anomaly in the table?

.....

.....

.....

.....

Q3: Is there an update anomaly possible?

.....

.....

.....

.....

8.6 HOW TO AVOID ANOMALIES

The best approach to creating tables without anomalies is to ensure that the tables are normalized, and that's accomplished by understanding functional dependencies. FD ensures that all attributes in a table belong to that table. In other words, it will eliminate redundancies and anomalies.

Example: separate Project and Employee tables.

Table 8.7: Project details table

ProjectID	Budget
P1	32
P2	40
P3	27
P4	17

Table 8.8: Employee details table

EmpID	ProjectID	Hours
S75	P1	7
S75	P2	3
S79	P1	4
S79	P3	1
S80	P2	5

Tables 8.7 and 8.8 respectively, separate Project and Employee tables with data. By keeping data separate using individual Project and Employee tables:

i. No anomalies will be created if a budget is changed.

ii. No dummy values are needed for projects that have no employees assigned.

iii. If an employee's contribution is deleted, no important data is lost.

iv. No anomalies are created if an employee's contribution is added.

Check Your Progress 2

Q1: What should be done to minimize anomaly for the table shown in check your progress 1.

.....

.....

.....

.....

Q2: What is the logic for decomposition of tables?

.....

.....

.....

.....

8.7 VIDEO LECTURE

https://tinyurl.com/zpgydth

8.8 ACTIVITY

Activity 8.0

Data Anomalies

Aim: Demonstrate basic techniques of how to remove data anomalies.

Resources: Unit 8 learning materials.

What to do:

By using the given data below, please remove all the data anomalies involved.

Attribute Name	Sample Value	Sample Value	Sample Value
StudentID	1	2	3
StudentName	John Smith	Sandy Law	Sue Rogers
CourseID	2	2	3
CourseName	Programming Level 1	Programming Level 1	Business
Grade	75%	61%	81%
CourseDate	Jan 5th, 2014	Jan 5th, 2014	Jan 7th, 2014

Duration: Expect to spend about 1 hour on this activity.

Feedback: This activity should be submitted to the course instructor for assessment and feedback.

8.9 SUMMARY

In this unit you learned about data redundancy in relation with database design; specifically covering the three major types of data anomalies. The unit concluded by illustrating how to remove them. In the next unit the process of such decomposition called normalization is covered in details.

8.10 ANSWERS

Check Your Progress 1

Ans 1. There are several other Programmes of the University for example BCA, CMAD etc., which cannot be entered in the table till a student data is to be stored for those programmes.

Ans 2: The deletion anomaly in this table is that if you delete the data of student S02, the information of Programme-ID P2 will be deleted.

Ans 3: If you update first record to

S01 XYZ P1 BCA

Then the information that P1 is code for BCA or MCA is not clear. This may be an update anomaly.

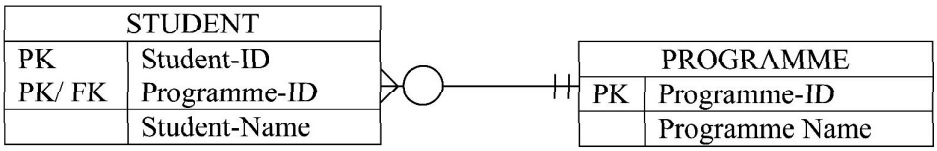
Check Your Progress 2

Ans 1: Decompose the table on the following points:

- Astudent can enroll for several programmes.

- AProgramme has a Programme_ID as primary key.

So the two tables would be:



Ans 2: Logic is simple that one entity should be included in one table.

8.11 REVIEW QUESTIONS BY AUTHORS

Aim: These review questions are aimed at assisting the students in understanding the concept of data redundancy and data anomalies. Therefore, it is advised that they should be posted in a discussion forum where each student can contribute.

1. What is meaning of the term 'data redundancy'? Explain in details why this concept is a problem in relational database?
2. Why is it important to remove data redundancy before continuing with database design and implementation?
3. Compare and contrast between update and insertion anomalies.
4. Explain in details the concept of deletion anomaly.

Describe the mechanism of removing data anomalies before database implementation.

8.12 FURTHER READING

1. ER Modelling. (2015). Retrieved May 23, 2016, from Open TextBooks for Hongkong, http://www.opentextbooks.org.hk/ditatopic/30776
2. Eng, N., & Watt, A. (2013). Database Design - 2nd Edition. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/chapter-10-er-modelling/

Download this book for free at http://open.bccampus.ca

8.13 ATTRIBUTION

The following material was written by Grace Mbwete:

1. Introduction
2. Unit Summary
3. Review questions

UNIT 9 FUNCTIONAL DEPENDENCIES

Structure

9.1 INTRODUCTION

In this unit, we will be introduced to the concept of Functional Dependency (FD) which is an important part of relational database design and a pre-requisite for the next unit (normalization). FD in relational databases defines relationship between two sets of attributes. Therefore, we will learn more on functional dependencies categories.

9.2 OBJECTIVES

Upon completion of this unit you should be able to:

- *Describe* the concept of functional dependency.
- *Compare and contrast different types of functional dependencies*.
- *dentify* various types of inference rules in relation to FD.
- *Use* dependency diagram tool to define FD.

9.3 TERMINOLOGIES

Armstrong's axioms : A set of inference rules used to infer all the functional dependencies on a relational database.

Decomposition : A rule that suggests if you have a table that appears to contain two entities that are determined by the same PK, consider breaking them up into two tables.

Dependent	:	The right side of the functional dependency diagram.
Determinant	:	It is the left side of the functional dependency diagram.
Functional Dependency (FD)	:	A relationship between two attributes, typically between the PK and other non-key attributes within a table.
Non-normalized table	:	A table that has data redundancy in it.
Union	:	A rule that suggests that if two tables are separate, and the PK is the same, consider putting them together.

9.4 FUNCTIONAL DEPENDENCY

A functional dependency (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table. For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y. This relationship is indicated by the representation below:

X → Y

The left side of the above FD diagram is called the determinant, and the right side is the dependent. Here are a few examples.

In the first example, below, AadharNo determines Name, Address and Birthdate. Given AadharNo, you can determine any of the other attributes within the table.

AadharNo → Name, Address, Birthdate

For the second example, AadharNo and Course determine the date completed (DateCompleted). This must also work for a composite PK.

Aadhar No, Course → DateCompleted

The third example indicates that ISBN number of a Book can determine its Title.

ISBN → Title

9.5 RULES OF FUNCTIONAL DEPENDENCIES

Consider the following table of data r(R) of the relation schema R(ABCDE) shown in Table 9.1 below:-

Table 9.1: Functional dependency

A	B	C	D	E
a 1	b1	c1	d1	e1
a 2	b1	c2	d2	e1
a 3	b2	c1	d1	e1
a 4	b2	c2	d2	e1
a 5	b3	c3	d1	e1

As you look at this table, ask yourself: What kind of dependencies can we observe among the attributes in Table R? Since the values of A are unique (a1, a2, a3, etc.), it follows from the FD definition that:

$A \rightarrow B, \quad A \rightarrow C, \quad A \rightarrow D, \quad A \rightarrow E$

1. It also follows that $A \rightarrow BC$ (or any other subset of ABCDE).
2. This can be summarized as $A \rightarrow BCDE$.
3. From our understanding of primary keys, A is a primary key.

Since the values of E are always the same (all e1) (it means for any value of A or B or C or D, E is always e1), it follows that:

$A \rightarrow E, \quad B \rightarrow E, \quad C \rightarrow E, \quad D \rightarrow E$

However, we cannot generally summarize the above with $ABCD ? E$ because, in general, $A \rightarrow E, \quad B \rightarrow E$ does not mean, $AB \rightarrow E$.

Other observations:

1. Combinations of BC are unique, therefore $BC \rightarrow ADE$.
2. Combinations of BD are unique, therefore $BD \rightarrow ACE$.
3. If C values match, so do D values.
   i. Therefore, $C \rightarrow D$
   ii. However, D values don't determine C values
   iii. So C does not determine D, and D does not determine C.

Looking at actual data can help clarify which attributes are dependent and which are determinants.

9.5.1 Inference Rules

Armstrong's axioms are a set of inference rules used to infer all the functional dependencies on a relational database. They were developed by William W. Armstrong. The following describes what will be used, in terms of notation, to explain these axioms.

Let R(U) be a relation scheme over the set of attributes U. We will use the letters X, Y, Z to represent any subset of and, for short, the union of two sets of attributes, instead of the usual $X \rightarrow Y$.

i) Axiom of reflexivity

This axiom says, if Y is a subset of X, then X determines Y (see Figure 9.1).

$$\text{If } Y \subseteq X, \text{ then } X \rightarrow Y$$

Figure 9.1: Equation for axiom of reflexivity, by A. Watt.

For example, **PartNo → NT123** where X (PartNo) is composed of more than one piece of information; i.e., Y (NT) and partID (123).

ii) Axiom of augmentation

The axiom of augmentation, also known as a partial dependency, says if X determines Y, then XZ determines YZ for any Z (see Figure 11.2).

$$\text{If } X \rightarrow Y, \text{ then } XZ \rightarrow YZ \text{ for any } Z$$

Figure 9.2: Equation for axiom of augmentation, by A.Watt

The axiom of augmentation says that every non-key attribute must be fully dependent on the PK. In the example shown below, StudentName, Address, City, Prov, and PC (postal code) are only dependent on the StudentNo, not on the StudentNo and Course.

StudentNo, Course ->StudentName, Address, City, Prov, PC, Grade, DateCompleted

This situation is not desirable because every non-key attribute has to be fully dependent on the PK. In this situation, student information is only partially dependent on the PK (StudentNo).

To fix this problem, we need to break the original table down into two as follows:

- Table 1: StudentNo, Course, Grade, DateCompleted
- Table 2: StudentNo, StudentName, Address, City, Prov, PC

iii) Axiom of transitivity

The axiom of transitivity says if X determines Y, and Y determines Z, then X must also determine Z (see Figure 9.3).

$$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z, \text{ then } X \rightarrow Z$$

Figure 9.3: Equation for axiom of transitivity, by A. Watt

The table below has information not directly related to the student; for instance, ProgramID and ProgramName should have a table of its own. ProgramName is not dependent on StudentNo; it's dependent on ProgramID.

StudentNo StudentName, Address, City, Prov, PC, ProgramID, ProgramName

This situation is not desirable because a non-key attribute (ProgramName) depends on another non-key attribute (ProgramID).

To fix this problem, we need to break this table into two: one to hold information about the student and the other to hold information about the program.

- Table 1: StudentNo → StudentName, Address, City, Prov, PC, ProgramID
- Table 2: ProgramID → ProgramName

However we still need to leave an FK in the student table so that we can identify which program the student is enrolled in.

Check Your Progress 1

Q1: Identify the FDs in the following table:

A	B	C
a1	b1	c1
a2	b1	c2
a3	b2	c3

.....

.....

.....

.....

Q2: Enrolment number of a student includes the year of admission. Does any FD exist between the two?

.....

.....

.....

.....

Q3: Consider the following table and identity functional dependencies.

CUSTOMER(Cust-id, Cust-name, Account -No, Balance of Account)

Please note that an account holder can have multiple accounts.

.....

.....

.....

.....

Q4: Consider the following table and identify functional dependencies.

CUST(Cust-id, costumer-name, customer type, customer-credit-rating)

.....

.....

.....

.....

iv) Union

This rule suggests that if two tables are separate, and the PK is the same, you may want to consider putting them together.

It states that if X determines Y and X determines Z then X must also determine Y and Z (see Figure 9.4).

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

Figure 9.4: Equation for the Union rule, by A. Watt

For example, if:

- EmployeeID → EmpName
- EmployeeID → SpouseName

You may want to join these two tables into one as follows:

EmployeeID → EmpName, SpouseName

Some database administrators (DBA) might choose to keep these tables separated for a couple of reasons. One, each table describes a different entity so the entities should be kept apart. Two, if SpouseName is to be left NULL most of the time, there is no need to include it in the same table as EmpName.

v) Decomposition

Decomposition is the reverse of the Union rule. If you have a table that appears to contain two entities that are determined by the same PK, consider breaking them up into two tables. This rule states that if X determines Y and Z, then X determines Y and X determines Z separately (see Figure 9.5).

$$\text{If } X \rightarrow YZ \text{ then } X \rightarrow Y \text{ and } X \rightarrow Z$$

Figure 9.5: Equation for decomposition rule, by A. Watt

9.6 DEPENDENCY DIAGRAM

A dependency diagram, shown in Figure 9.6, illustrates the various dependencies that might exist in a *non-normalized table*. A non-normalized table is one that has data redundancy in it.

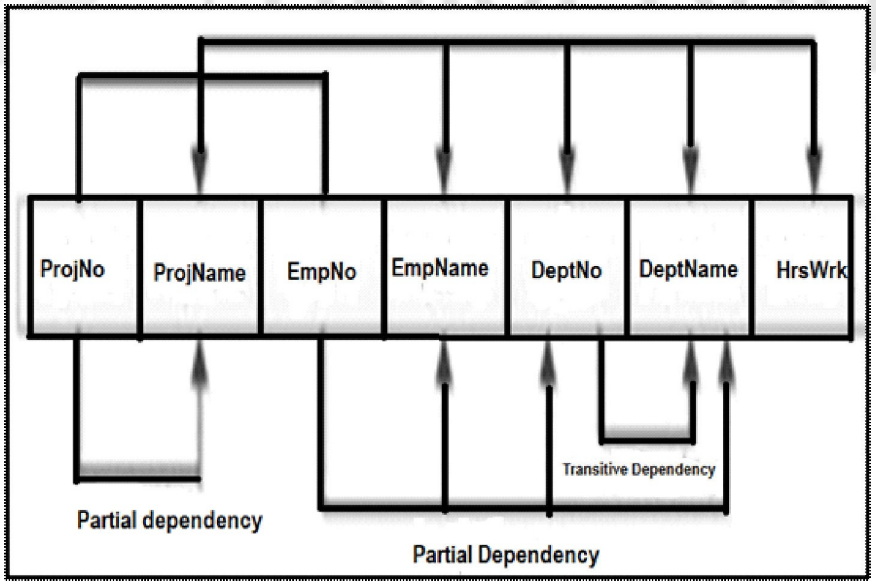


Figure 9.6: Dependency diagram, by A. Watt

The following dependencies are identified in this table:

i. ProjNo and EmpNo, combined, are the PK.

ii. Partial Dependencies:

- ProjNo → ProjName
- EmpNo → EmpName, DeptNo,
- ProjNo, EmpNo → HrsWork
- Transitive Dependency:
- DeptNo → DeptName

Check Your Progress 2

Q1: Given the two FDs of two different tables:

Table 1: Student-ID → Name

Table 2: Student-ID → Programme-code

Can these two tables combined?

.....

.....

.....

.....

Q2: Given the FDs in a table Student-ID → Name, Phone would you like to decompose them in two tables?

.....

.....

.....

.....

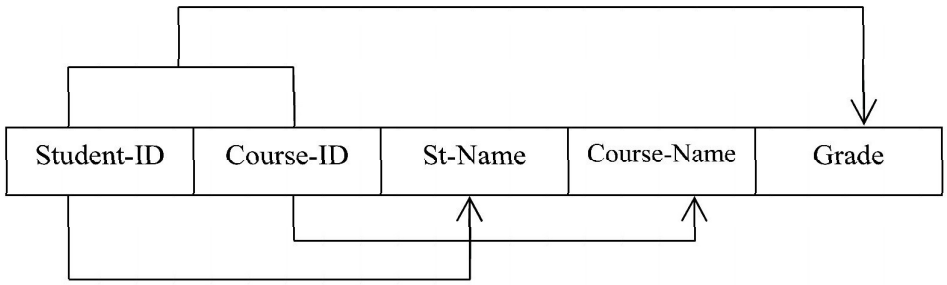
Q3: List the FDs for the following dependency diagram

.....

.....

.....

.....



.....

.....

.....

.....

9.7 VIDEO LECTURE

https://tinyurl.com/gohmf7z

9.8 ACTIVITY

Activity 9.0

Functional Dependency Exercises

Aim: Demonstrate basic knowledge of Functional Dependencies Concepts.

Resources: Internet access and Unit 8 learning materials.

What to do:

1. By using the definition of functional dependency to confirm that each of Armstrong's axioms (reflexivity, augmentation and transitivity) is correct.
2. In the following table, identify two functional dependencies (A, B and C are attributes)

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

3. Express the following real world facts using functional dependencies: A lecturer, identified by the value of the attribute

	LecturerId, has a name (Name),an office (Office), and a phone extension number (ExtensionNo). The number of students (NoOfStud) enrolled in a course, which is identified by the value of the attribute CourseId , depends on the term when the course is offered (Term) and the year (Year). Each office (Office) has only one phone extension number (ExtensionNo) and each phone extension number belongs to at most one office. **Duration:** Expect to spend about 1 hour on this activity. **Feedback:** This is an activity should be submitted to the course instructor for assessment and feedback.

9.9 SUMMARY

In this unit you learned about Functional Dependencies (FD). The unit has also covered different inference rules and dependency diagram. The concepts covered in this unit are a pre-requisite for the next unit of data normalization. The normalization has been discussed in the next unit in details.

9.10 ANSWERS

Check Your Progress 1

Ans 1: A → B, C ; B → C and C → B

AB → C, AC → B

Ans 2: As Enrolment No → year of admission

Therefore, the FD: Enrolment No → year of admission holds.

Ans 3: Cust-id, Account-no → Cust-name, balance of account

However, cust-name can be determined only by Cust-id. The FDs can be represented as:

Cust-id → Cust-name

Cust-id, Account-no → Balance of account.

Ans 4: The FDs are

Cust-id → Customer-name, Customer-type

Customer-type → customer-credit-rating

Check Your Progress 2

Ans 1: Yes, the FDs would be Student-ID → Name, Programme-code

Ans 2: As both the Name & Phone are important fields, so not advisable to decompose.

Ans 3: Student-Id → St-Name

Course-Id → Course-Name

Student-ID, Course-ID → Grade.

9.11 REVIEW QUESTION BY AUTHORS

Motivation: These review questions are aimed at assisting the students in understanding of functional dependencies concepts. Therefore, it is advised that they should be posted in a discussion forum where each student can contribute.

Part I

Consider a relation R with five attributes ABCDE. You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

i. List all keys for R.

ii. Is R in 3NF?

iii. Is R in BCNF?

Part II

i. Define the term functional dependency and give an example to illustrate.

ii. Why are some functional dependencies called trivial?

Part III

Bank G wants to have a database for a bank that contains accounts (C), branches (H) and customers (D). Detailed below are the requirements given the following constraints:

i. An account cannot be shared by multiple customers.

ii. Two different branches do not have the same account.

iii. Each customer can have at most one account in a branch (but different accounts in different branches).

Write the functional dependencies implied by the above mentioned requirements.

Part IV

Identify the functional dependencies in the following relations and represent the dependencies using the proper notation.

i. Course (CourseCode, CourseName,CourseUnits)

ii. Instructor(InstrnID), InstrName, InstrAge, CourseCode)

9.12 FURTHER READING

1. Functional Dependencies. (2015, October 20). Retrieved May 23, 2016, from Open TextBooks for Hongkong, http://www.opentextbooks.org.hk/ditatopic/31109

2. Eng, N., & Watt, A. (2013). Database design - 2nd edition. Retrieved September 19, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/chapter-11-functional-dependencies/

9.13 ATTRIBUTION

The following material was written by Grace Mbwete:

1. Introduction
2. Unit Summary
3. Review questions

UNIT 10 INTRODUCTION TO DATA NORMALIZATION

Structure

10.1 INTRODUCTION

In the previous units, you have learnt theories on data modeling, relational data modeling and entity relationship model, which create logical design for relational databases. In this unit, we will learn that for any data in database table must be stored in a normalized way. This unit will explain the properties of a normalized table, process of normalization and its importance to the structure of a database.

10.2 OBJECTIVES

Upon completion of this unit you will be able to:

- *Describe* the fundamental concepts of normalization.
- *Compare and Contrast different* types of normal forms.
- *Normalize* a database relation to a third normal form.

10.3 TERMINOLOGIES

Normalization : The process of determining how much redundancy exists in a database table.

1NF : First Normal Forms - only single values are permitted at the intersection of each row and column so there are no repeating groups.

2NF : Second Normal Form - the relation must be in 1NF and either the PK comprises a single attribute or the non-key attributes are not functionally dependent on any part of the composite PK.

3NF : Third Normal Form - the relation must be in 2NF and all transitive dependencies must be removed; a non-key attribute may not be functionally dependent on another non-key attribute.

Normalization : The process of determining how much redundancy exists in a database table.

10.4 OVERVIEW

Normalization is the branch of relational theory that provides design insights. It is the process of determining how much redundancy exists in a table. The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema.
- Provide mechanisms for transforming schemas in order to remove redundancy.

Normalization theory draws heavily on the theory of functional dependencies. Normalization theory defines six normal forms (NF). Each normal form involves a set of dependency properties that a schema must satisfy and each normal form gives guarantees about the presence and/or absence of update anomalies. This means that higher normal forms have less redundancy, and as a result, fewer update problems.

Normalization should be part of the database design process. However, it is difficult to separate the normalization process from the ER modeling process so the two techniques should be used concurrently as follows:

- Use an entity relation diagram (ERD) to provide the big picture, or macro view, of an organization's data requirements and operations. This is created through an iterative process that involves identifying relevant entities, their attributes and their relationships.
- Normalization procedure focuses on characteristics of specific entities and represents the micro view of entities within the ERD.

10.5 NORMAL FORMS

All the tables in any database can be in one of the normal forms we will discuss next. Ideally we only want minimal redundancy for PK to FK. Everything else should be derived from other tables. There are six normal forms, but we will only look at the first four, which are:

- First normal form (1NF)
- Second normal form (2NF)

- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)

BCNF is rarely used.

10.5.1 First Normal Form (1NF)

In the *first normal form*, only single values are permitted at the intersection of each row and column; hence, there are no repeating groups.

To normalize a relation that contains a repeating group, remove the repeating group and form two new relations. The PK of the new relation is a combination of the PK of the original relation plus an attribute from the newly created relation for unique identification.

Process for 1NF

We will use the Student_Grade_Report table below, from a School database, as our example to explain the process for 1NF.

Student_Grade_Report (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- In the Student Grade Report table, the repeating group is the course information. A student can take many courses.
- Remove the repeating group. In this case, it's the course information for each student.
- Identify the PK for your new table.
- The PK must uniquely identify the attribute value (Student No and Course No).
- After removing all the attributes related to the course and student, you are left with the student course table (StudentCourse).
- The Student table (Student) is now in first normal form with the repeating group removed.

The two new tables are shown below.

- Student (StudentNo, StudentName, Major)
- Student Course (Student No, CourseNo, CourseName, Instructor No, Instructor Name, Instructor Location, Grade)

i) How to update 1NF anomalies

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- To add a new course, we need a student.
- When course information needs to be updated, we may have inconsistencies.
- To delete a *student*, we might also delete critical information about a course.

10.5.2 Second normal form (2NF)

For the *second normal form*, the relation must first be in 1NF. The relation is automatically in 2NF if, and only if, the PK comprises a single attribute.

If the relation has a composite PK, then each non-key attribute must be fully dependent on the entire PK and not on a subset of the PK (i.e., there must be no partial dependency or augmentation).

ii) Process for 2NF

To move to 2NF, a table must first be in 1NF.

- The Student table is already in 2NF because it has a single-column PK.
- When examining the Student Course table, we see that not all the attributes are fully dependent on the PK; specifically, all course information. The only attribute that is fully dependent is grade.
- Identify the new table that contains the course information.
- Identify the PK for the new table.

The three new tables are shown below.

- **Student** (StudentNo, StudentName, Major)
- **CourseGrade** (StudentNo, CourseNo, Grade)
- **CourseInstructor**(CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)

iii) How to update 2NF anomalies

- When adding a new instructor, we need a course.
- Updating course information could lead to inconsistencies for instructor information.
- Deleting a course may also delete instructor information.

Check Your Progress 1

Q1: What is the need of Normalisation?

.....

.....

.....

.....

Q2: Consider the following relation:

Student (Id, name, phone)

It is assumed that a student can have multiple phone numbers. Is the relation in 1NF?

.....

.....

.....

.....

Q3: Consider the following relation

Bank (Cust-id, name, Account-number, balance) is the relation in 2NF?

.....

.....

.....

.....

10.5.3 Third normal form (3NF)

To be in *third normal form*, the relation must be in second normal form. Also all transitive dependencies must be removed; a non-key attribute may not be functionally dependent on another non-key attribute.

i) Process for 3NF

The following are the stages involved in normalizing data to 3NF:-

- Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have a transitive relationship.
- Create new table(s) with removed dependency.
- Check new table(s) as well as table(s) modified to make sure that each table has a determinant and that no table contains inappropriate dependencies.

See the four new tables below. How have they been created? Please refer to figure 10.1 for transitive dependencies. The transitive dependency- InstructorNo ? Instructor-Name, InstructorLocation is used to perform the normalization of CourseInstructor relation.

- **Student** (StudentNo, StudentName, Major)
- **CourseGrade** (StudentNo, CourseNo, Grade)
- **Course** (CourseNo, CourseName, InstructorNo)
- **Instructor** (InstructorNo, InstructorName, InstructorLocation)

At this stage, there should be no anomalies in third normal form. Let's look at the dependency diagram (Figure 10.1) for this example. The first step is to remove repeating groups, as discussed above.

- **Student** (StudentNo, StudentName, Major)
- **StudentCourse** (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

To recap the normalization process for the School database, review the dependencies shown in Figure 10.1.

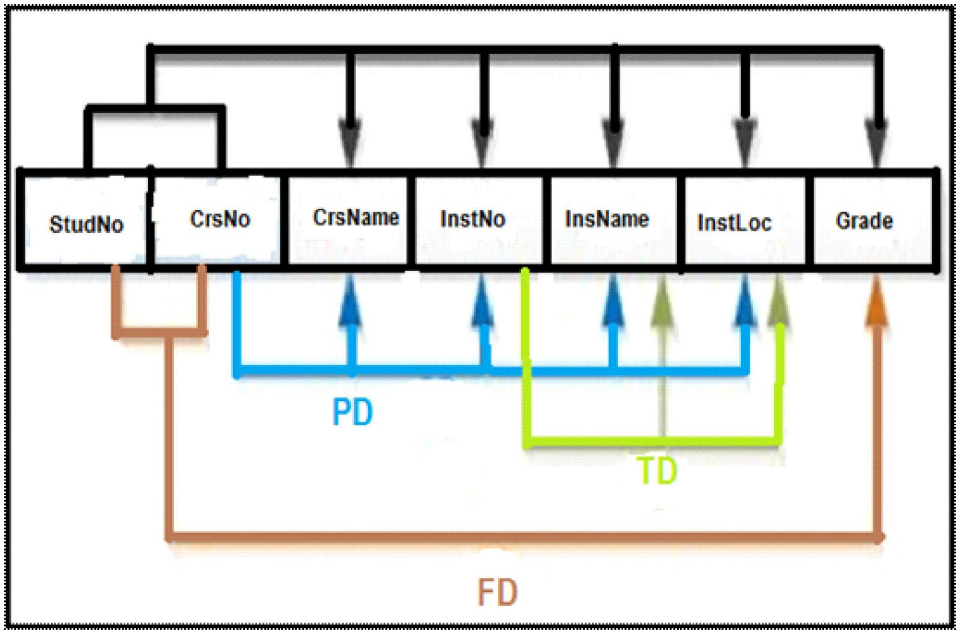


Figure 10.1 Dependency diagram, by A. Watt.

The abbreviations used in Figure 10.1 are as follows:

i. PD: partial dependency

ii. TD: transitive dependency

iii. FD: full dependency (Note: FD typically stands for functional dependency. Using FD as an abbreviation for full dependency is only used in Figure 10.1.)

10.5.4 Boyce-Codd Normal Form (BCNF)

When a table has more than one candidate key, anomalies may result even though the relation is in 3NF. *Boyce-Codd normal form* is a special case of 3NF. A relation is in BCNF if, and only if, every determinant is a candidate key.

BCNF Example 1- Consider the following table 10.1 below;

Table 10.1: St_maj_Adv

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

The semantic rules (business rules applied to the database) for this table are:

i. Each Student may major in several subjects.

ii. For each Major, a given Student has only one Advisor.

iii. Each Major has several Advisors.

iv. Each Advisor advises only one Major.

v. Each Advisor advises several Students in one Major.

The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.

i. Student_id, Major → Advisor

ii. Advisor → Major

Anomalies for this table include:

i. Delete - student deletes advisor info

ii. Insert - a new advisor needs a student

iii. Update - inconsistencies

Note: No single attribute is a candidate key.

PK can be Student_id, Major or Student_id, Advisor.

To reduce the **St_Maj_Adv** relation to BCNF, you create two new tables as illustrated below:

i. **St_Adv** (Student_id, Advisor)

Table 10.2: St_Adv

Student_id	Advisor
111	Smith
111	Chan
320	Dobbs
671	White
803	Smith

ii. Adv_Maj (Advisor, Major)

Table 10.3:Adv_Maj

Advisor	Major
Smith	Physics
Chan	Music
Dobbs	Math
White	Physics

BCNF Example 2

Consider the following table 10.4 below;

Table 10.4: Client_Interview

ClientNo	InterviewDate	InterviewTime	StaffNo	RoomNo
CR76	13-May-02	10.30	SG5	G101
CR56	13-May-02	12.00	SG5	G101
CR74	13-May-02	12.00	SG37	G102
CR56	1-July-02	10.30	SG5	G102

i. FD1 - ClientNo, InterviewDate → InterviewTime, StaffNo, RoomNo (PK)

ii. FD2 - staffNo, interviewDate, interviewTime → clientNO (candidate key: CK)

iii. FD3 - roomNo, interviewDate, interviewTime → staffNo, clientNo (CK)

iv. FD4 - staffNo, interviewDate → roomNo

A relation is in BCNF if, and only if, every determinant is a candidate key. We need to create a table that incorporates the first three FDs table 10.5 **(Client_Interview2)** and table 10.6 **(StaffRoom)** for the fourth FD.

Table 10.5: Client_Interview2

ClientNo	InterviewDate	InterViewTime	StaffNo
CR76	13-May-02	10.30	SG5
CR56	13-May-02	12.00	SG5
CR74	13-May-02	12.00	SG37
CR56	1-July-02	10.30	SG5

Table 10.6:StaffRoom

StaffNo	InterviewDate	RoomNo
SG5	13-May-02	G101
SG37	13-May-02	G102
SG5	1-July-02	G102

10.6 NORMALIZATION AND DATABASE DESIGN

During the normalization process of database design, make sure that proposed entities meet required normal form before table structures are created. Many real-world databases have been improperly designed or burdened with anomalies if improperly modified during the course of time. You may be asked to redesign and modify existing databases. This can be a large undertaking if the tables are not properly normalized.

Check Your Progress 2

Q1: Consider the following relation:\
Student (Student-id, Course-code, taught by, teacher-qualification)
Is the relation in 3NF?

.....

.....

.....

.....

Q2: Is the following relation in BCNF

(Staffid, room-no, work-type)

The FDs are staff-id, room-no ?work type

work-type ?room no

(a) What are the alternate keys of the relation?

(b) Is the relation in BCNF?

.....

.....

.....

.....

10.7 VIDEO LECTURE

https://tinyurl.com/jc6dbyp

https://tinyurl.com/gu49pgj

10.8 ACTIVITY

Group Activity

Activity 10.1

Introduction to Normalization Scenarios

Motivation: To become conversant with basic techniques of normalizing data.

Resources: Unit 10 learning materials.

What to do:

Read the following scenarios carefully;

1. BRANCH (Branch#, Branch_Addr, (ISBN, Title, Author, Publisher, Num_copies))
2. CLIENT (Client#, Name, Location, Manager#, Manager_name, Manager_location, (Contract#, Estimated_cost, Completion_date, (Staff#, Staff_name, Staff_location)))
3. PATIENT (Patient#, Name, DOB, Address, (Prescription#, Drug, Date, Dosage, Doctor#, Doctor, Secretary))
4. DOCTOR (Doctor#, DoctorName, Secretary, (Patient#, PatientName, PatientDOB, PatientAddress, (Prescription#, Drug, Date, Dosage)))

How to do it:

You are provided with different scenarios; please normalize the data to 3NF.

Duration: Expect to spend about 2 hours on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

10.9 SUMMARY

In this unit you learned data normalization process which is used parallel with ER modelling in the logical design of database implementation. This unit also covered insert, update and delete anomalies which can be removed by applying normalization to that particular database. The last part covers the first three stages of data normalization, namely, 1NF, 2NF and 3NF.

10.10 ANSWERS

Check Your Progress 1

Ans 1: It reduces the redundancy, thus is useful in minimizing the three anomalies, viz. Insert, update & Delete anomalies.

Ans 2: No, the relation may be decomposed as

Student (Id, name)

Student phone (Id, phone)

The scheme as above is in 1NF.

Ans 3: No, FDs are Cust-id → Name and Cust-id, Account-number ? balance Therefore, the 2NF relation would be:

(Cust-id, name)

(Cust-id, Account-number, balance)

Check Your Progress 2

Ans 1: The FDs are

student-id, course-code → taught by, teacher-qualification taught by → teacher-qualification.

Therefore, the relation is not in 3NF.

The 3NF relation would be

(student-id, course-code, taught-by)

(taught-by, teacher-qualification)

Ans 2: (a) The alternate keys of the relations are (staff-if, room-no) and (staff-if, work-type) any one of these may be selected as primary key.

(b) The BCNF of this table would be:
(staff-id, room-no) and (work-type, room-no)

10.11 CASE STUDY

Motivation: The aim of these case studies is to assist you in understanding different case studies in database design and use the correct techniques in normalizing the data. The following are the case studies where students will read and then normalize the data given to 3NF.

Case Study I

A new client comes and asks for you to build them a database from the spreadsheet they've been using to track the company employees. Here's an excerpt:

1	2	3	4	5	6	7	8	9
Sally	Sales	Manager	Sally	$25	35	$875	N/A	Company Morale
Sally	Sales	Manager	Sally	$25	35	$875	N/A	Recycling Program
Joe	Sales	Sales person	Sally	$10	35	$350	6%	N/A
Sean	Shipping	Clerk	Bob	$8	20	$160	N/A	Recycling
Sean	Security	Guard	Kirk	$12	16	$192	N/A	United Way

Key for field name:

1 = Name, 2= Department, 3=Position, 4=Manager, 5=Rate, 6=Work_Hrs, 7 = Week_Pay, 8= Commission%, 9= Task_force

Case Study II

The ABC Manufacturing company has a completely automated application system. The system, however, resides on index files and does not allow for decision support at all. In order to move to ad hoc queries, and "what if" queries, the company has decided to convert the existing system to a database. Initially, the only criterion for the application is to replace the existing system with a database system. No ad hoc screen or reports have been anticipated. You will see the reports and screens that exist currently.

Customer Order and Product Application Considerations

1. Each customer must be on file before an order can be placed. The name, address(s), phone number(s), and credit limit must be recorded. All other data items are optional. If there is no shipping address, then the mailing address is used instead. Since customers can have identical names, a customer id has been assigned to each customer.

2. Each order will have a computer generated id number. The order can have up to 10 line items. Discounts can be given to preferred customers and this discount amount will be recorded on the customer's record. Customers without a discount amount will not be given a discount.

3. Each product listed on the order will show the standard price for that product. Discounts will be shown at the bottom of the order form.

4. Orders that can be filled or partially filled are shipped immediately, and the product data is updated accordingly. Orders, or partial orders that cannot be filled will be backordered.

5. As products are manufactured the product data is updated accordingly along with the part inventory data.

6. A customer can place numerous orders. Products can be ordered by many different customers. The same part can be used in numerous products. (eg. a screw can be used in a chair, bar stool etc.)

Case Study III

A college keeps details of its students, staff and courses in a file. Part of this file is shown below.

1	2	3	4	5	6	7	8
0567	J Evans	11/4/89	M	COMP7	Computing A Level	186	H Smith
8453	R Begum	18/3/88	F	BIOL9	Biology A Level	78	D Jones
0567	J Evans	11/4/89	M	MATH5	Maths A Level	186	H Smith

Key for field name:

1 = StudentNo, 2= StudentName, 3=DateOfBirth, 4= Gender, 5= Course No, 6= Course Name, 7 = LecturerNo, 8= Lecturer Name

The data in this file is not normalised.

i. Using data from the above file to illustrate your answer, describe two different problems associated with data not being normalised.

ii. The above data can be re-organised into a normalised relational database with tables linked using primary and foreign keys.

Re-organise this data into a normalised relational database using two tables. You should clearly indicate the table names and any primary or foreign keys that you use.

Feedback: These case studies should be submitted to the course Instructor for assessment and feedback.

10.11 FURTHER READING

1. Eng, N., & Watt, A. (2013). Database Design. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign/chapter/chapter-12-normalization/

Download this book for free at http://open.bccampus.ca

2. CCNC/CCNC module 5/the database application/database concepts/ normalization. Retrieved September 19, 2016, from Wiki Educator, http://wikieducator.org/CCNC/CCNC_Module_5/The_database_application/Database_Concepts/Normalization

3. 1.264 Lecture 10 - Data Normalization. (2013). Retrieved April 15, 2016, from MIT OCW, http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-264j-database-internet-and-systems-integration-technologies-fall-2013/lecture-notes-exercises/MIT1_264JF13_lect_10.pdf

4. Normalization. (2015). Retrieved May 23, 2016, from Open TextBooks for Hongkong, http://www.opentextbooks.org.hk/ditatopic/31119

10.12 ATTRIBUTION