DEPARTMENT OF INFORMATION TECHNOLOGY

BACHELOR OF BUSINESS INFORMATION TECNNOLOGY (BBIT)

COURSE TITLE: DATABASE SYSTEMS

Course outline

BIT 2102 DATABASE SYSTEMS

Purpose: To introduce the concept of rational database and the techniques and tools for developing and utilizing databases in business.

Objectives By the end of the course units the learner should be able to;

- Explain the principles underlying relational database
- Design and develop a practical database system
- Use structured query language (SQL) to access and manipulate data.

Course Content

- Introduction to database concepts – tuples, attributes
- Database models – Flat, Hierarchy, Network, Relational and object oriented.
- Database management systems, Integrity and security in database systems. Concurrency.
- Normalization 1st, 2nd and 3rd normal form.
- SQL – Data manipulation, Definition and control
- Database modeling, Conceptual , logical and physical models
- Networked and distributed database systems. (partial and full)
- Data warehousing and mining.

Assessment: Examination - 70%: Coursework - 30%

Required text books

Elmasri R & Navathe, Fundamentals of database systems in business, Prentice Hall Date C.J, An Introduction to database systems

Text books for further reading

Connolly T. & Begg C., Database systems: A practical approach to design, implementation and management.

CHAPTER ONE ERROR! BOOKMARK NOT DEFINED.

INTRODUCTION TO DATABASE AND ITS ENVIRONMENT 4
Database system 6
Advantages of the Database Systems 6
Disadvantages of Database Systems 7
File System Environment 8
Database System Environment 8
Basic Structure of SQL Statement 12
Select clause 12
Ordering Display of Tuples 14
Aggregate Functions 15
Null Values 15
Tuple Variables 15
String Operations 16
SQL and Set 17
VIEWS 18
Modification Of The Database 18
Schema Definition in SQL 20
Data Independence 25

CHAPTER TWO 28

CONCEPTUAL DATA MODELING 28
2.1 The E- R Model (Entity Relationship) 28
Entity-Relationship Diagram 30
Types of Relationships 31
Characteristics of Attributes 35
Enhanced E-R Model 39
The E-R Model, as an E-R Diagram 42

CHAPTER THREE 45

CONCEPTUAL DATA MODELING 45
3.1 Properties of Relations 45
3.2 Relational Keys 45
4.1 ER-to-Relational Mapping 47

CHAPTER FIVE 52

FUNCTIONAL DEPENDENCIES AND NORMALIZATION 52
Definition of partial dependencies and full dependencies 52
Example: An invoice 57
Advantages of Normalization Approach 58
Disadvantages 58
Review Questions 59
Sample exam questions 62

CHAPTER ONE

INTRODUCTION TO DATABASE AND ITS ENVIRONMENT

Learning objectives:

☺
By the end of the chapter a student shall be able to:

i. **Understand the meaning of database system**
ii. **Compare the database system and traditional file system**
iii. **Evaluate the Database Management System**
iv. **Differentiate types of database systems**
v. **Understand advantages and disadvantages of database system**
vi. **Understand the database system environment**

1.1 Definition of terms

Data management: focuses on data collection, storage and retrieval, constitutes a core activity for any organization. To generate relevant information efficiently you need quick access to data (raw facts) from which the required information is produced. Efficient data management requires the use of a computer database. A database is a shared, integrated computer structure that houses a collection of: **End -user data**: raw facts of interest to the user.

Meta data: The Meta data provides a description of the data characteristics and the set of relationships that link the data found within the database.

The database: resembles a very well organized electronic filing cabinet in which powerful software referred to as DBMS helps manage the cabinet"s contents.

DBMS: Database Management system that enables the creation of and management of the database

1.2 Database vs. file based system

File based system

Consider a saving bank enterprise that keeps information about all customers and savings accounts in permanent system files at the bank. The bank will need a number of applications e.g.

i. Program to debit or credit an account
ii. A program to add a new account
iii. A program to find the balance of an account
iv. A program to generate monthly statements
v. Any new program would be added as per the banks requirements

Such a typical filing /processing system has the limitation of more and more files and application programs being added to the system at any time. Such a scheme has a number of major disadvantages:

1. **Data redundancy and inconsistency** - Since the files and application programs are created by different programmers over a long period of type, the files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several files. This redundancy leads to higher storage and access costs. It may also lead to inconsistency i.e. the various copies of the same data may no longer agree.
2. **Difficulty in accessing** - Suppose that one of the bank officers needs to find out the names of all customers who live within the city's 78-phone code. The officer would ask the data processing department to generate such a list. Such a request may not have been anticipated while designing the system originally and the only options available are:-
   - Extract the data manually
   - Write the necessary application; therefore do not allow the data to be accessed conveniently and efficiently

3. **Data isolation** - Since data is scattered in various files and files may be in different formats, it may be difficult to write new applications programs to retrieve the appropriate data.
4. **Concurrent access anomalies** - Interaction of concurrent updates may result in inconsistent data e.g. if 2 customers withdraw funds say 50/= and 100/= from an account at about the same time the result of the concurrent execution may leave the account in an incorrect state.
5. **Security problems** - Not every user of the database system should be able to access all the data. Since application programs are added to the system in an ad-hoc manner, it is difficult to enforce security constraints.
6. **Integrity** - The data value stored in the database must satisfy certain types of consistency constraints e.g. a balance of a bank account may never fall below a prescribed value e.g. 5,000/=. These constraints are enforced in a system by adding appropriate code in the various application programs. However, when new constraints are added there is need to change the other programs to enforce.

Conclusion.

These difficulties among others have prompted the development of DBMS.

Database system

Unlike the file system with many separate and unrelated files, the Database consists of logically related data store in a single data repository. The problems inherent in file systems make using the database system very desirable and therefore, the database represents a change in the way the end user data are stored accessed and arranged.

Advantages of the Database Systems

1. **Centralized Control -** Via the DBA it is possible to enforce centralized management and control of data. This means that necessary modifications, which do not affect other application changes, meet the data independence DBMS requirement.

2. **Reduction of redundancies -** Unnecessary duplication of data is avoided effectively reducing total amount of data required, consequently the reduction of storage space. It also eliminates extra processing necessary to trace the required data in a large mass of data. It also eliminates inconsistencies. Any redundancies that exist in the DBMS are controlled and the system ensures that his multiple copies are consistent.

3. **Shared data -** In a DBMS, sharing of data under its control by a number of application programs and user is possible e.g. backups.
4.

5. **Integrity -** Centralized control can also ensure that adequate checks are incorporated to the DBMS provide data integrity. Data integrity means that the data contained in the database is both accurate and consistent e.g. employee age must be between 28-25 years.

6. **Security -** Only authorized people must access confidential data. The DBA ensures that proper access procedures are followed including proper authentication schemes process that the DBMS and additional checks before permitting access to sensitive data. Different levels of security can be implemented for various types of data or operations.

7. **Conflict Resolution -** The DBA is in a position to resolve conflicting resolve conflicting requirements of various users and applications. It is by choosing the best file structure and access method to get optimum performance for the response. This could be by classifying applications into critical and less critical applications.

8. **Data Independence** - It involves both logical and physical independence logical data independence indicates that the conceptual schemes can be changed without affecting the existing external schemes. Physical data independence indicates that the physical storage structures/devices used for storing the data would be changed without necessitating a change in the conceptual view or any of the external use.

Disadvantages of Database Systems

1. Cost - in terms of:

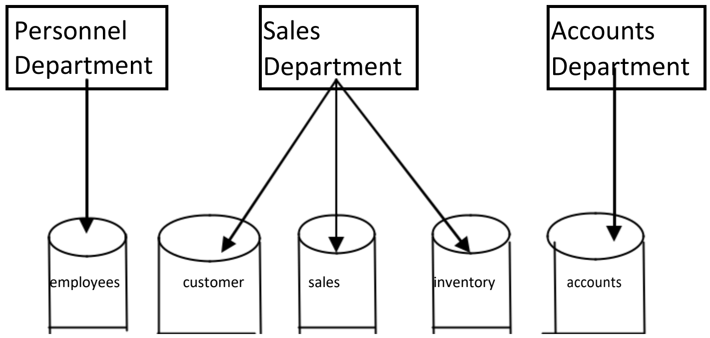
    - The DBMS - software
    - Purchasing or developing S/W
    - H/W
    - Workspace (disks for storage)
    - Migration (movement from tradition separate systems to an integrated one)

2. Centralization Problems

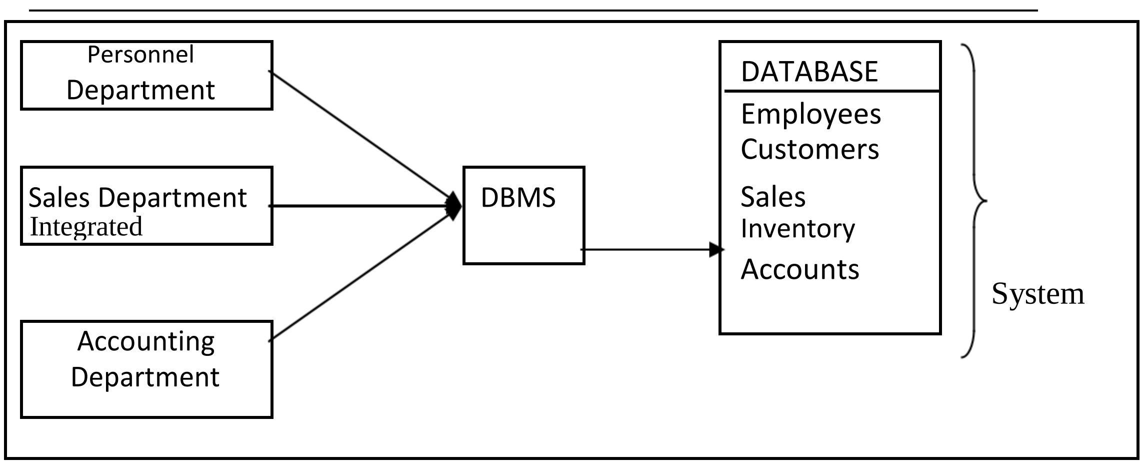
You would require adequate backup incase of failure
You would require increased severity of security breaches and disruption of operation of the organization because of downtimes and failures.

3. Complexity of Backup and recovery

File System Environment



Database System Environment



The database eliminates most of the file systems' data inconsistencies, anomalies and structural dependency problems. The current generation of DBMS software stores not only the data structures in a central location but also stores the relationships between the database components. The DBMS also takes care of defining all the required access paths of the required component.

The term database system refers to an organization of components that define and regulate the collection storage, management and use of data within a database environment. The database system is composed of 5 major parts i.e.

a. Hardware
b. Software
c. People
d. Procedures
e. Data

Hardware
This identifies all the systems physical devices e.g. the composition peripherals, storage devices etc.

Software
These are a collection of programs used by the computers within the database system.

i. O.S - manages all hardware components and makes it possible for all other and software to run on the composition.
ii. The DBMS - manages the database within the database system e.g. Oracle, DB2, Ms Access etc.
iii. Applications programs and utilities to access and manipulate data in the DBMS.

People
These are all database systems users:-

1. **Systems administrator** - Oversees the database systems general operations.
2. **Database administrator (DBA)** - Manages the DBMS use and ensures that the database is functioning properly. His functions include:

    i. Scheme definition - The original database scheme is created by writing a set of definitions, which are translated by DDL compiler to a set of tables that are permanently stored in the data dictionary.
    ii. Storage structure and Access Methods Definitions - By writing a set of definitions for appropriate storage structures and access methods, which are translated by the data storage and definition language compiler.
    iii. Scheme and physical organisation modifications - Modification to either the database schema or description of the physical storage organisation are accompanied by writing a set of definitions which are used by either the DDL compiler or the data storage and definition language compiler to generate modification to appropriate internal systems tables e.g. data dictionary.
    iv. Granting authorization to data access - This is so as to regulate which parts of the database users can access.
    v. The database manager keeps integrity Constrains in a special system structure whenever an update takes place in the system.

3. **Database designers** - These are the database architects who design the database structure.

4. **Systems Analysts & Programmers (application programmers)** - They design and implement the application programs they design & create the data entry scheme, reports & procedures through which users access and manipulate the databases data.

5. **End users** - These are the people who use the application programs to run the organizations daily operations. They fall in the following classes:

    i. Sophisticated users - These interact with the system without writing programs. They form their requests in a database query language.
    ii. Specialized database applications that do not fit in the traditional data processing framework e.g. CAD Systems, knowledge based & expect systems.
    iii. Application programmers: These interact with the system through the DML & applications.
    iv. Naive – Unsophisticated user who interact with the systems by invoking one of the permanent application programs that have been written previously.

Procedures

- These are instructions and rules that govern the design and use of the database system.
- They enforce standards by which business is conducted within the organisation an with customers.
- They also ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of such data.

Data

This covers the collection for facts stored in the database and since data is the raw material from which information is generated the determination of what data is to be stored into the database and how the data is to be organized is a vital part of the database

1.3 Database languages

A DBMS is software used to build, maintain and control database systems. It allo s a systematic approach to the storage and retrieval of data in a computer.
Most DBMS(s) have several major components, which include the following:

1. Data Definition Language (DDL) - These are commands used for creating and altering the structure of the database.
   The structures comprise of Field Names, Field sizes, Type of data for each field, File organizational technique. The DDL commands are used to create new objects, alter the structure of existing ones or completely remove objects from the system.

2. Data Manipulation language (DML) - This is the user language interface and is used for executing and modifying the contents of the database. These commands allow access and manipulation of data for output. They include commands for adding, inserting, deleting, sorting, displaying, painting etc. These are the most frequently used commands once the database has been created.
   Interactive Data Manipulation Language (DML) - DML includes a query language based on both relational calculus. It includes commands to insert tuples into, delete tuples from and modify tuples in the database.
   Embedded DML - This is designed for use within general purpose programming languages such as PL/1. Cobol, Pascal, Fortran and C.

3. Data Control Language (DCL) - These are commands used to control access to the database in response to DML commands. It acts as an interface between the DML and the OS. It provides security and control to the data.

4. Query Languages - A query language is a formalized method of constructing queries in database system. It provides the ways in which the user interrogates the database for data without using conventional programs. For relation database, structures query languages (SQL) has emerged as the standard language. Almost all the DBMS(s) use SQL running on machines ranging from microcomputers to large main frames.

i. View Definition - The SQL DDL includes commands for specifying access rights to relations and view.
ii. Integrity - The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints as disallowed.
iii. Transaction Control - SQL includes commands for specifying the beginning and ending of transactions. Several implementations also allow explicit locking of data for concurrency control.

Basic Structure of SQL Statement

Basic structure of an SQL expression consists of 3 clauses;

i. SELECT
ii. FROM
iii. WHERE

SELECT
This corresponds o a projection operation of the relational algebra. Its used to list the attributed desired in the result of a query.
FROM
This corresponds to a Cartesian product operation of the relational algebra. It lists the relations to be scanned in the evaluation of the expression
WHERE
Corresponds to the predicate of the relational algebra. It consist of a predicate involving attributes of the relations that appear in the **FROM** clause.

A typical SQL query will be of the form:
SELECT
A_1,A_2, A_3, ...A_n

FROM
R_1, R_2, R_3, ...R_n
WHERE
P

A_i represents an attribute; each r a relation and P is a predicate.

Select clause

Examples (i) **SELECT** Branch name
FROM Loan

(ii) **SELECT DISTINCT** Branch-name
FROM Loan

The symbol * can be used to denote all attributes of a given relation

(iii) **SELECT** *
FROM Loan

STUDENT

Code	***Stud.id***	***Name***
IMIS	001	Charles
BIT	002	Mary
BIT	003	Maina
CIT	004	Judy

COURSE

Code	***Title***
IMIS	Info. Systems
BIT	Bachelor of IT
CIT	Cert in IT
DIT	Dip in IT

Select Stud-Id, Name, Code, Title
From Student, Course
Where Student.Code = Course.Code

The select clause can also contain arithmetical expressions involving operations +, -, *, and operating on constants or attributes of tables e.g.

SELECT Branch_name, Loan_number, Amount*100
FROM loan

Where Clause

Specifies a condition that has to be met. SQL uses the logical connectives AND, OR and NOT in the where clause. It also uses operands of logical connectives <, < =, >, >=, = and < >. It also includes a BETWEEN operations e.g.

(i) **Select** loan_number
From loan

(ii) **Select** loan_number
From loan
Where branch_name = "River Road" and Amount **Between** 10,000 **And** 15,000.

From Clause

This specifies the source (relations), which is a Cartesian product. The SQL uses the notion relation-name. Attribute-name to avoid ambiguity in case where an attribute appears in the schemer of more that one relation e.g.

Example

Select Customer_name, borrower. loan number
From borrower, loan
Where borrower.loan_number = loan.loan_number
AND branch_name= "Moi Avenue"

This will return the name of the customer the loan-number is the customer loan no. appears in Moi Avenue.

SQL provides a mechanism for renaming both relations and attributes by use of the As clause it is of the form
Old_name **AS** New_name. e.g.

Select distinct Customer_name, Borrower. Loan_number **AS**
loan_Id **From** Borrower, loan
Where Borrower. Loan_number = loan.loan_number
AND Branch_name = "Koinange Street"

Ordering Display of Tuples

The "order by" clause case the tuples in the result for a query to appear in sorted order e.g.

Select distinct Customer - name
From borrower, loan
Where borrower.loan_number = loan.loan_number
And Branch name = "University way"
Order by customer_name

By default the order by clause lists items in ascending order. To specify the sort order use '**desc**' for descending order or „**asc**" for ascending e.g.

Select *
From loan
Order by amount desc, loan-number desc

Aggregate Functions

These are functions that take a collection (set or multi-set) of values as input and return a single value. These are

Average: **Avg**
Minimum: **Min**
Maximum: **Max**
Total: **Sum**
Count: **Count**

The input to sum and average must be a collection of numbers but the other operators can operate on collection of non-numeric data-types e.g. strings

Example

(i) **SELECT**Branch name, Avg(balance)
FROM Account **GROUP**
BY Branch -name

(ii) **SELECT**Branch_name, count (**distinct** customer_name)
FROMDepositor, account
WHERE Depositor, account-number = account - number
GROUP BY Branch name

(ii) **SELECT**Branch_name, Avg(balance)
FROM Account **GROUP**
BY Branch_name
HAVINGAverage (balance) > 1200

Null Values

Null values indicate absence of information about the value of an attribute. e.g.

SELECT loan-number
FROM loan
WHERE Amount is **Null**

Assignment: look into Inner Join and Outer Join

Tuple Variables

- A tuple variable in SQL must be associated with a particular relation

They are defined in the FROM clause via the use of the AS clause. e.g.

SELECT DISTINCT Customer_name, T**.**loan_number
FROM Borrower **AS** T, loan **AS** S **WHERE**
T**.**loan_number = S**.**Loan_number

Query to find the names of all branches that have assets greater than at least one branch located in Brooklyn would be.

SELECT Distinct T**.**Branch_name
FROM Branch **AS** T, Branch **AS** S
WHERE T**.**assets > S**.**assets **AND** S**.**Branch_city = "BROOKLYN"

When expressions of the form relation_name**.**Attribute_name are written, the relation name is an implicitly defined tuple variable.

String Operations

- Most commonly used operation on strings is pattern matching using
- "LIKE". Two characters are used
  - Percent (%) - matches any sub-string
  - Underscore (-) - matches any character
- Patterns are case sensitive i.e. uppercase do not match lower case characters.

Examples
(i) "Mary %" matches any string beginning with "Mary"
(ii) "%ry" Matches any string containing "ry" as a sub-string e.g. very, mary, ary etc.
(iii) "- - -" Matches any string of exactly three characters.
(iv) "- - -%" Matches any string of at least 3 characters.

The query to find customer names for all customers whose addresses include the sub-string "main" would be:-

SELECT Customer-name
FROM Customer
WHERE Customer -street LIKE "%main %"

For patterns to include special pattern characters (i.e. % and _) SQL allows the specification of an escape character. The escape character is placed immediately before a special pattern character to indicate the special pattern. Character is to be treated like a normal character. The key work **ESCAPE** is used.

Examples.
- **LIKE** "ab\%cd%"**ESCAPE** "\" - matches all strings beginning with "ab%cd"
- **LIKE** "ab\\cd%" **ESCAPE**"\" - matches all strings beginning with "ab\cd"

Mismatches.
SQL allows the search for mismatches using the **NOT LIKE** comparison operator Set Operations.

SQL and Set

SQL operations **Union**, **Intersect** and **Except** operate on relations and correspond to the relational operations , and **-**,

(i) Union

To find all customers having a loan, an account or both at the

bank (**SELECT** Customer_name **FROM** depositor)

UNION
(**SELECT** Customer_name
FROM Borrower)

To indicate duplicates

(**SELECT** Customer_name **FROM** Depositor)

UNION ALL
(**SELECT** Customer_name
FROM Borrower)

(ii) The Intersection

To find customers who have both a loan and an account at the bank

To include duplicates we use "intersect all"

(iii) The Exception

To find customers who have an account but no loan at the bank we

write (**SELECT Distinct** Customer_name **FROM** Depositor)

EXCEPT

(**SELECT** Customer_name
FROM Borrower)

To include duplicate we use "Except all"
Null Values

- The keyword is used in the predicate test.

Example

SELECT Loan_number
FROM Loan
WHERE Amount is **NULL**

- To test for the absence of a null value we use the predicate "IS NOT NULL"

VIEWS

Use **CREATE VIEW** command
Syntax

CREATE VIEW V **AS** <*query expression*> Where
query expression is a legal query expression.

Example
CREATE VIEW Customer **AS**
(**SELECT** Branch_name, Customer_name
FROM Depositor**.**account)
WHERE Depositor**.**Account_number, Account**.**account_number

The names of the attribute of a view can be specified as

CREATE VIEW Branch_total_loan(branch-name, total(loan)
AS
SELECT Branch_name, SUM (amount)
FROM loan
GROUP BY Branch_name

NB: A create view clause creates a view definition in the database which stays there until a command **DROP** View (view name) is executed.

Modification Of The Database

Involves **Add**, **REMOVE** or **CHANGE** of information in the database.

(i) Deletion

DELETE FROM r
WHERE P

- P represents the predicate, r represent the relation.
- The statement first finds all tuples t in r which P(t) is true & then deletes them from r
- Where clause can be omitted in which case all tuples in P are deleted.

Example
DELETE FROM Loan
 - Deletes all tuples from the loan relation.

To delete all loans with loan amounts between 1300 &1500

DELETE FROM loan
WHERE amount **BETWEEN** 1300 **AND** 1500

To delete all accounts at city square branch

DELETE FROM account
WHERE Branch-name = ”City Square”

(ii) Insertion
To insert data into a relation:-

- Specify a tuple to be inserted or
- Write a query whose result is a set of tuples to be inserted

Tuples to be inserted must be in the correct arity.

Example

INSERT INTO Account
VALUES (“City Square”, “ Account”, 6000)
or
INSERT INTO Account (branch-name, account-number, balance)
VALUES (“City Square”, ”Account”, 6000)

(iii) Updates

To change a value in a tuple without changing all values the **UPDATE** statement can be used.
Examples
(i) **UPDATE** Account
SET Balance = Balance * 1.05
(ii) **UPDATE** Account

SET Balance = Balance *1.06
WHERE balance >10,000

Update Of A View
A modification is permitted through a view only if the view in question is defined in terms of one relation of the actual relational database i.e. of a logical level db

Example
CREATE VIEW Branch_loan **AS**
SELECT Branch_name, loan_number
FROM loan
INSERT INTO Branch_loan
VALUES ("Moi Avenue", "Accoo8")

Schema Definition in SQL

Syntax
CREATE TABLE r(A1D1, A2D2, -----, A_nD_n,
[Integrity Constraints],
…………
………...
………...
[Integrity - constraints]

Examples
(i) **CREATE TABLE** Customer
(Customer_name **CHAR**(20) **NOT NULL,**
Customer_street **CHAR**(30),
Customer_city **CHAR**(30),
PRIMARY KEY (customer_name))

(ii) **CREATE TABLE** Branch
(Branch_name **CHAR** (15) **NOT NULL,**
Branch_city **CHAR** (30),
Assets Integer,
PRIMARY KEY (Branch_name)
Check (assets> = 0))

(iii) **CREATE TABLE** Depositor
(customer_name, **CHAR**(20) **NOT NULL**,
Account_name **CHAR**(20) **NOT NULL**,
PRIMARY KEY (Customer_name, Account_number))

The create table commands includes other integrity constraints.

- Primary key - includes a list of the attributes that constitute the primary key

- Unique - includes a list of the attributes that constitute a candidate key
- Foreign key - includes both a list of the attributes that constitute the foreign key & the name of the relation referenced by the foreign key.

1.4 The three-level architecture and its purpose

Abstraction and Data Integration

Abstraction is the simplification mechanism to hide superfluous (extra/surplus/unnecessary) details of a set of objects.
It allows one to concentrate on the properties that are of interest to the application e.g. a car is an abstraction of personal transportation vehicle but does not reveal details about model, year, colour etc.

Vehicle itself is an abstraction that includes the types; car, truck, bus and lorry.
Consider a non- database environment of a number of application programs as shown below:

Application 1 will contain values for the attributes employee Name and Employee. Address and this record can be described in pseudo-code as

```
Type   Employee = record
               Employee.name:string
               Employee.address:string
       End
```

Application 2 will have:

```
Type   Employee = record
               Employee.name: String
               Employee.soc_sec_No: Integer
               Employee.Adress: String
               Employee. Annual_salary:integer
       End
```

In a non-database environment each application is responsible for maintaining the currency of data and a change in data item.

In a database environment, data can be stored in this application and their requirement be integrated by whoever is responsible for centralized control (DBA).
The integrated version would appear as recorded containing attributes required by both applications.
The record will appear as:

```
Type   Employee = record
               Employee.Name:string
```

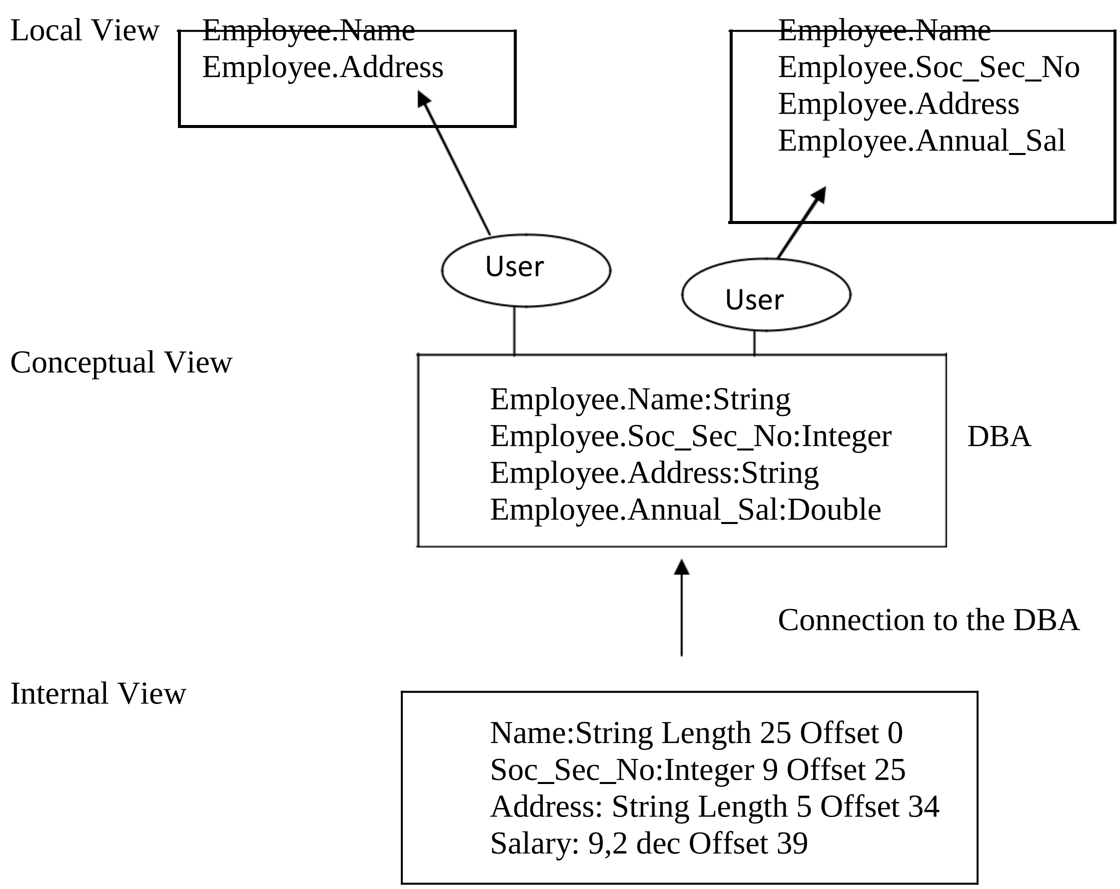
Employee.soc-sec.no: Integer
Employee.Address:string
Employee.Annual_Salary: double
End

The views supported are derived from the conceptual record by using appropriate mapping.

The application programs no longer require information about the storage structure; storage device types or access methods. These are absorbed by the DBMS.

There are 3 level abstractions corresponding to 3 views:

i. The highest level which is seen by the application programs or user called “external or user view"
ii. A sum total of users view called global view a conceptual view.
iii. Lower level which is the description of the actual method of storing the data. It is also referred to as the internal view.



The 3 level scheme architecture is called the ANSI/SPARC model (American National Standard Institute/Standards Planning and Requirements Committee.) It is divided into 3 levels:

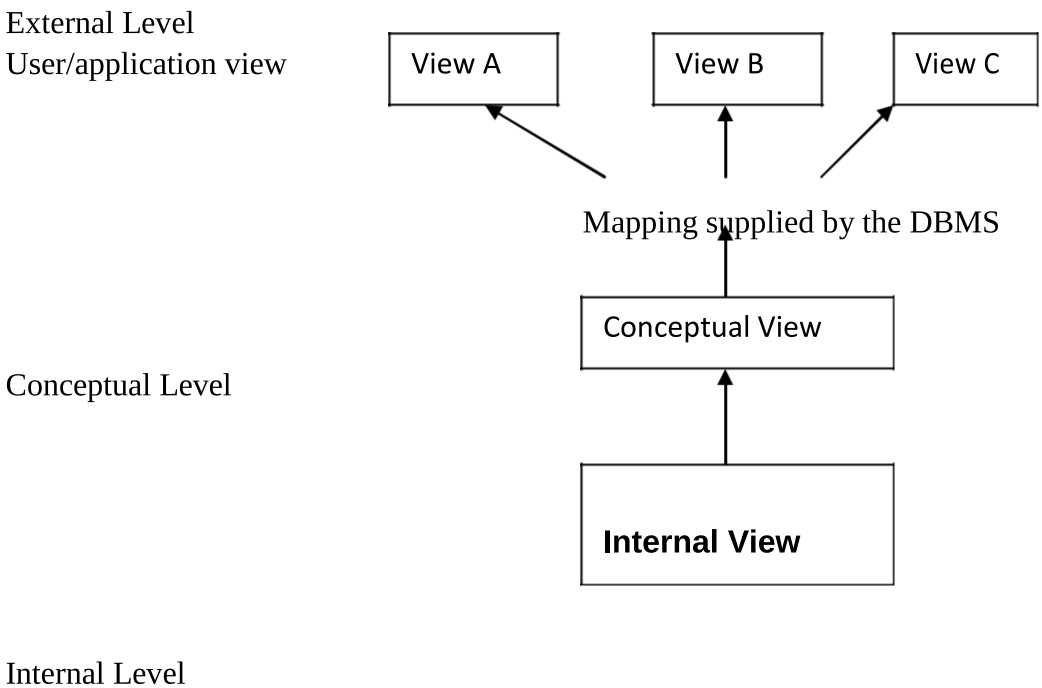
- External
- Conceptual
- Internal

The view of each level is described as a scheme, which is an outline or a plan that describes the records and relations existing in the view. It also describes the way in which entities at one level of abstraction can be mapped onto the next level.
External Level (External or User view)
This is at the highest level of database abstraction where only those portions of the database of concern to the user or application programs are included.

Any number of user views may be possible, some of which may be identical.
Each external view is described by means of a scheme called external scheme, which consists of a definition of the logical records and the relationships in the external view.
It also contains the method of devising the objects in the external view from the objects in the conceptual view (entities, attributes and relationships).

Conceptual or Global View
Contains all database entities and the relationships among them are included and one conceptual view represents the entire database.
It is defined by the conceptual scheme. Also contains the methods of deriving the objects in the conceptual view from the object s in the internal view. It is independent of the physical presentation.

Internal View
This is the lowest level of abstraction closest to the physical storage method used.
It indicated how data would be stored and describe the data structures and access methods to be used by the database. The internal schema implements it.



The 3 levels of architecture of a DBMS

Mapping between views
Two mappings are required, one between external and conceptual views and another between the conceptual records to internal ones.

Data Independence
This is the immunity of users/application programs from changes in storage structure and access mechanism.
The 3 levels of abstractions along with the mappings from internal to conceptual and from conceptual to external provide 2 distinct levels of data independence i.e.:

- Logical Data Independence
- Physical Data Independence

(i) Logical Data Independence

This indicates that the conceptual schema can be changed without affecting the existing external schema.
The mapping between the external and conceptual levels would absorb the change.

It also insulates application programs from operations such as combining two records into one or splitting an existing record into 2 or more records. The LDI is achieved by providing the external level or user view database.

The application programs or users see the database as described by the respective external view.
DBMS provided a mapping from this view to the conceptual view.
NB: The view at conceptual level of the database is the sum total of the current and anticipated views of the database.

(ii) Physical Data Independence

This indicates that the physical storage structures or devices used for storing the data can be changed without necessitating a change in the conceptual view or any of the external view. Any change is absorbed by the mapping between the conceptual and internal views.

1.5 Classification of DBMS

i. Single User database systems

This is a database system that supports one user at a time such that if user A is using the database, users B & C must wait until user A complete his or her database work. If a single user database runs on a personal computer it‟s called a desktop database.

ii. Multi-user database

This is a database that supports multiple users at the same time for relatively small number e.g. 50 users in a department the database is referred to as a workgroup database. While one, which supports many departments is called an enterprise database.

iii. Centralized Database system

This is a database system that supports a database located at a single site.

iv. Distributed database system

This is a database system that supports a database distributed across several different sites.

v. Transactional DBMS/Production DBMS

This is a database system that supports immediate response transaction e.g. sale of a product.

vi. Decision Support DBMS

It focuses primarily on the production of information required to make a tactical or strategic decision at middle and high management levels.

Chapter Review Questions

1. List and briefly describe advantages and disadvantages of database systems.
2. State the components of database system environment
3. State and explain four types of database languages
4. Identify and explain the classification of DBMS

Cornolly T. $Begg C., *Database systems: a practical approach to design, implementation and management*

CHAPTER TWO

CONCEPTUAL DATA MODELING

☺ **Learning objectives:**

By the end of the chapter a student shall be able to:

i. **Types Of Data Models**
ii. **The E- R Model (Entity Relationship)**
iii. **E-R Model Basic Concepts**
iv. **Characteristics Of Attributes**
v. **Types Of Relationships**
vi. **Entity-Relationship Diagram**
vii. **Entity modeling (Diagrammatic representation) relationships**

2.1 The E- R Model (Entity Relationship)

It is based on a perception over a real world, which consists of a collection of basic objects called entities and relationships among this objects. An entity is an object that is distinguished from other objects via a specific set of attributes.

E-R Model Basic Concepts
The model employs the following components:

- Entity sets
- Relationship sets
- Attributes

1. Entity sets
An entity is a thing or object in the real world that is distinguishable from all other objects. It may be concrete e.g. a person or a book or it may be abstract e.g. a loan, holiday a concept etc. An entity set is a set of entities of the same type that share the same properties or attitudes e.g. a set of all persons who are customers of a bank.

Weak Entity Set

This is an entity set that does not have sufficient attributes to form a primary e.g. an entity set payments comprising of the attributes payment number, payment date and payment amount. Although each payment entity is distinct, payment for different loan e.g. may share the same payment number thus this entity set does not have a primary key.

Strong Entity Set

This is an entity set that has a primary key. For weak entity set to be meaningful it must be part of a one to many relationships.

2. Relationship sets

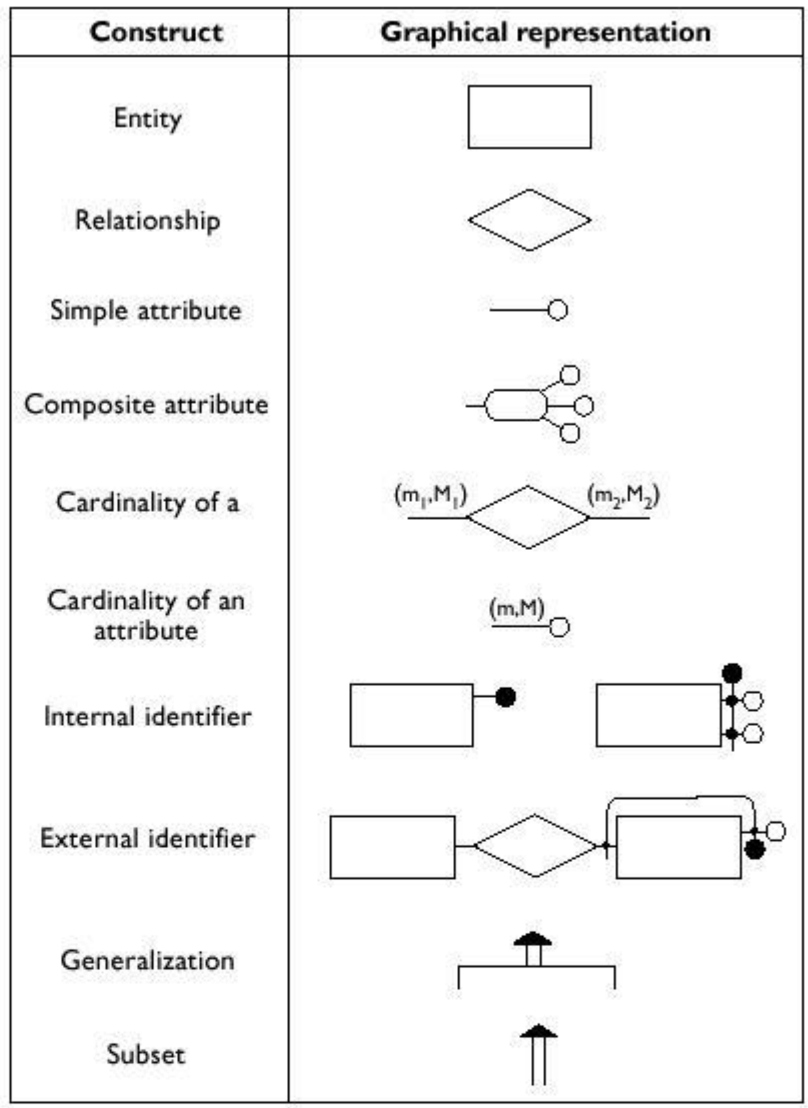
An association between two or more entities is called a relationship.

A relationship is an association amongst several entities while a relationship set is a set of relationships of the same tuple. It is a mathematical relation on n>2 possible non-distinct entity sets e.g. consider 2 entity sets, loan and branch. A relationship set loan, branch can be defined to denote association between a bank loan and the branch in which that loan is obtained.

They represent logical links between two or more entities.

Entity-Relationship Diagram

Components of E-R diagram

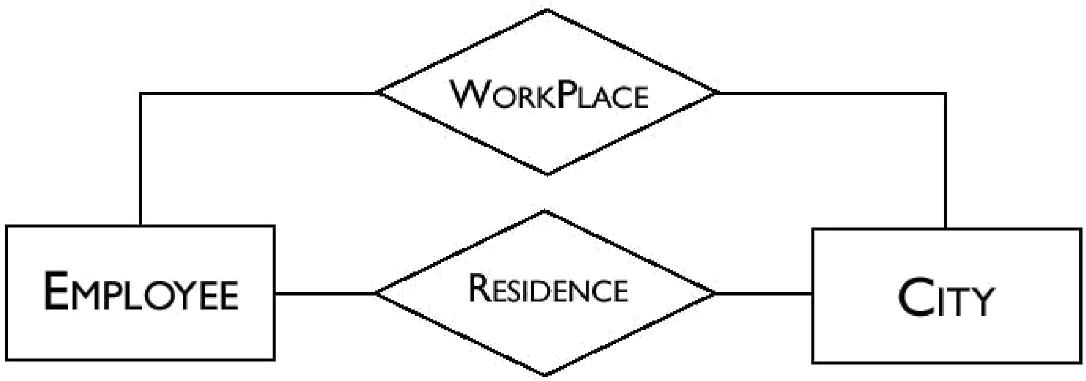


Example of relationship

Residence is an example of a relationship that can exist between the entities City and Employee; Exam is an example of a relationship that can exist between the entities Student and Course.

An instance of a relationship is an n-tuple made up of instances of entities, one for each of the entities involved.

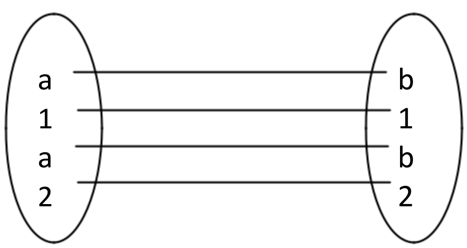
The pair (Johanssen,Stockholm), or the pair (Peterson,Oslo), are examples of instances in the relationship Residence.



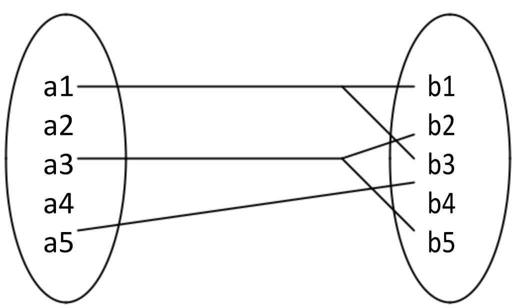
Types of Relationships

i. One to one relationship (1:1) - An entity in A is associated with utmost one entity in

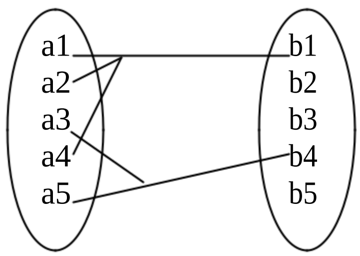
ii. B is associated with at utmost one entity in A.



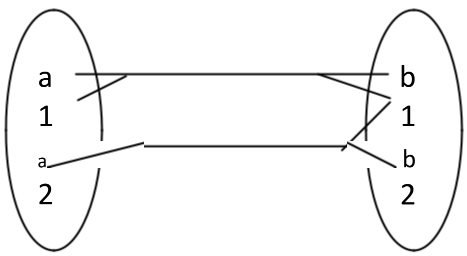
iii. One to Many relationship (1:M) - An entity in A is associated with any number of entities in B while an entity in B can be associated with at most one entity in A .



iv. Many to one relationship (M:1) - An entity in A is associated with at most one entity in B and an entity in B can be associated with a number of entities in A.



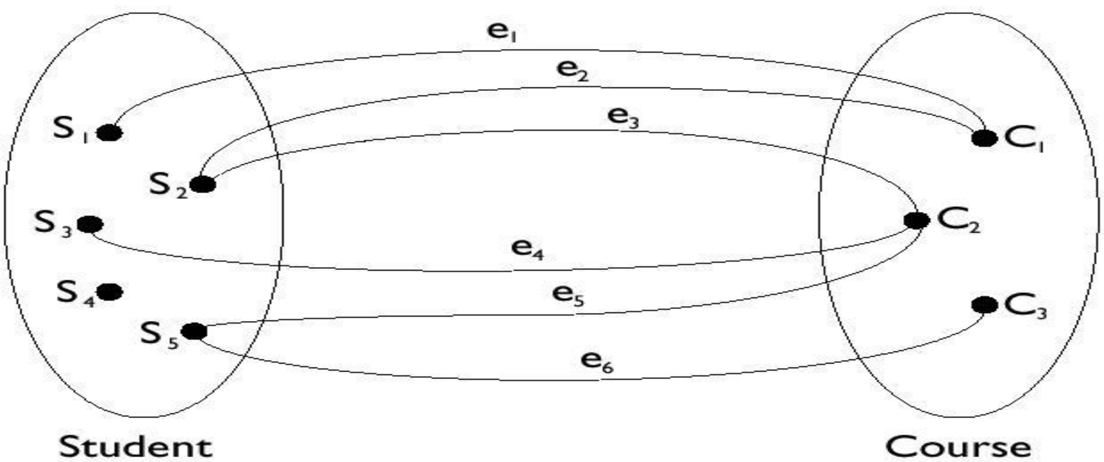
v. Many to many (M:N) - An entity in A is associated with at least one entity in B and an entity in B can be associated with a number of entities in A.



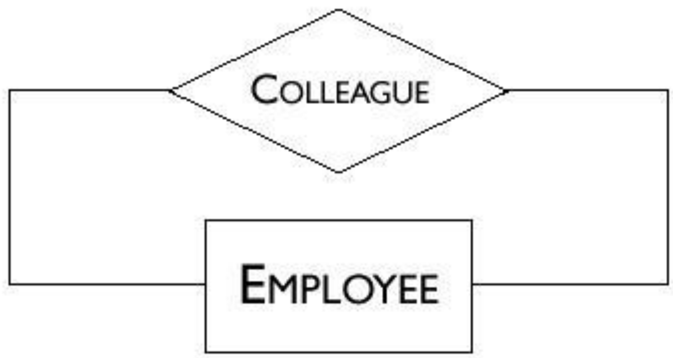
Existence Dependencies

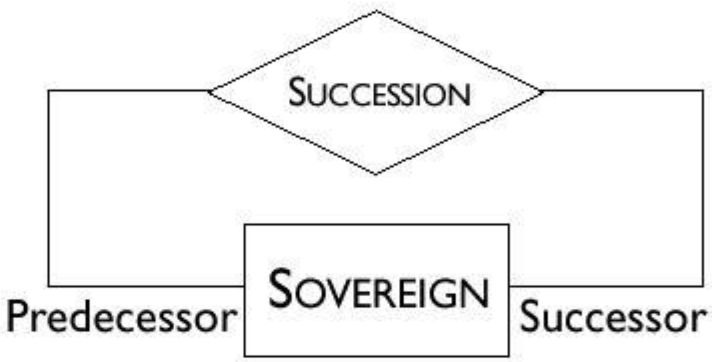
If the existence of an entity X depends on the existence of entity Y, then X is said to be existence dependent on Y. If Y is deleted, so is X. Y is said to be the dominant entity and X is said to be subordinate entity.

Example of Instances for Exam

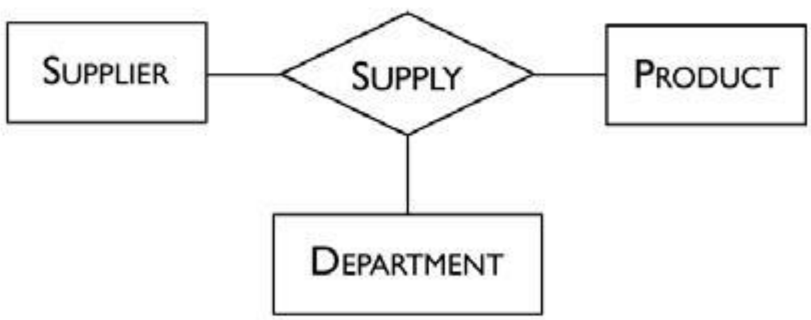


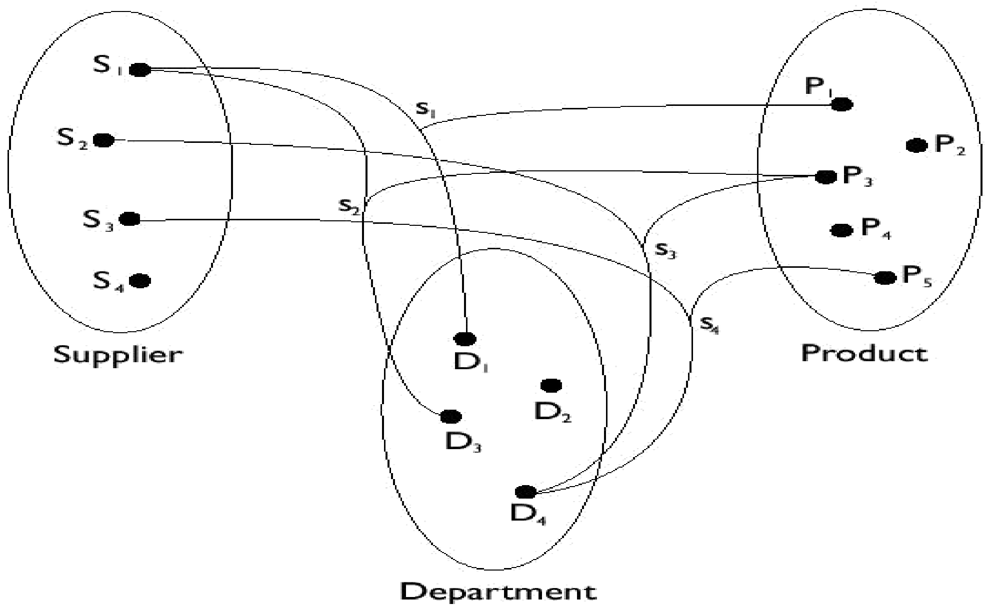
Recursive Relationships
Recursive relationships are also possible, that is relationships between an entity and itself. Note in the second example that the relationship is not symmetric. In this case it is necessary to indicate the two roles that the entity involved plays in the relationship.





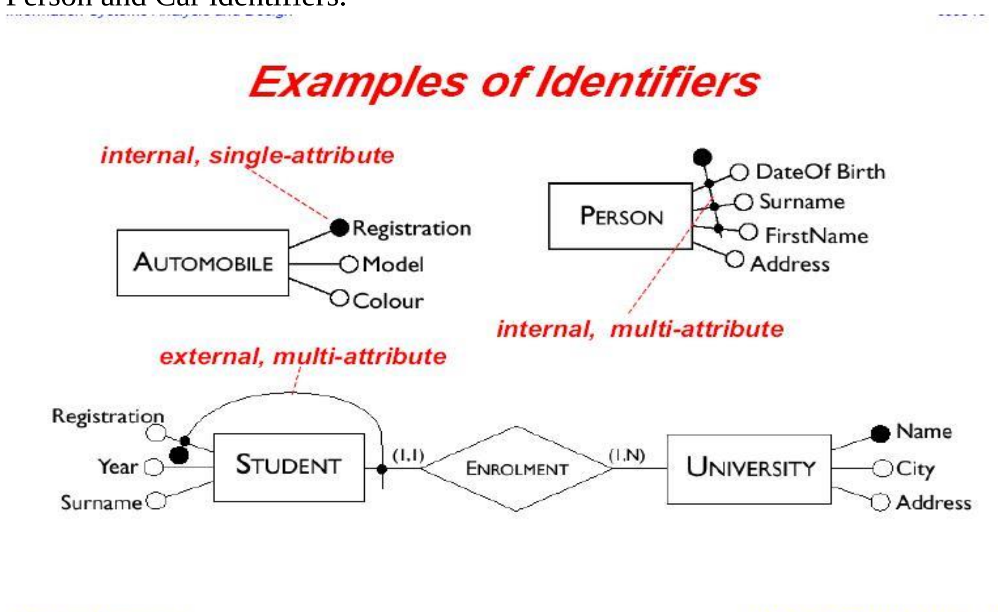
Ternary Relationships



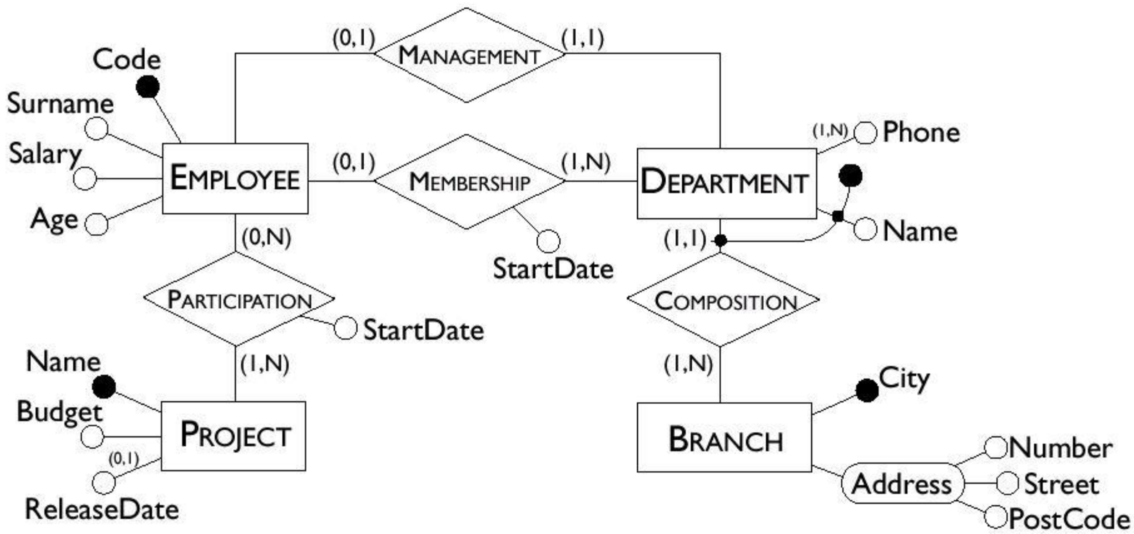


Identifiers

Identifiers (or keys) consist of one or more attributes which identify uniquely instances of an entity. In many cases, an identifier is formed by one or more attributes of the entity itself: in this case we talk about an internal identifier. Sometimes, however, the attributes of an entity are not sufficient to identify its instances unambiguously and other entities are involved in the identification. Identifiers of this type are called external identifiers. An identifier for a relationship consists of identifiers for all the entities it relates. For example, the identifier for the relationship (Person-) Owns(-Car) is a combination of the Person and Car identifiers.



Schema with Identifiers



3. Attributes

These describe the elementary properties of entities or relationships. For example, Surname, Salary and Age are possible attributes of the Employee entity, while Date and Mark are possible attributes for the relationship Exam between Student and Course. An attribute associates with each instance of an entity (or relationship) a value belonging to a set known as the domain of the attribute. The domain contains the admissible values for the attribute.

They are descriptive properties or characteristics possessed by each member of an entity set.

Characteristics of Attributes

1. Simple and Composite attributes - e.g. a customer name or first name, middle name, last name. Composite attributes are necessary if a user wishes to refer to entire attribute on some occasions and to only a component of the attributes on other occasions.

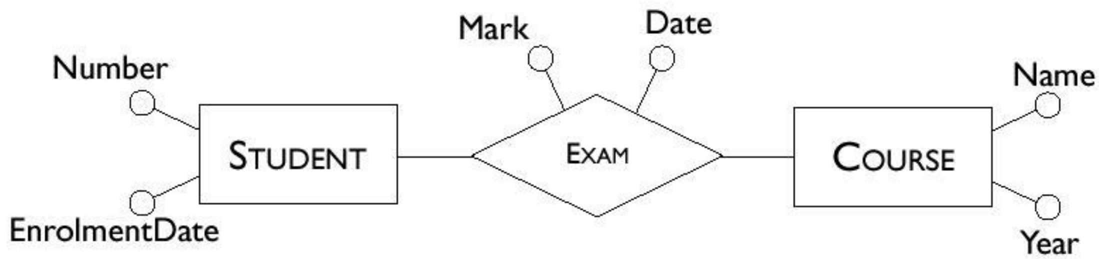
2. Single valued and Multi valued Attribute - The social security number or ID number can only have a single value at any instance and therefore its said to be single valued. An attribute like dependant name can take several values ranging from o-n thus it is said to be multi valued.

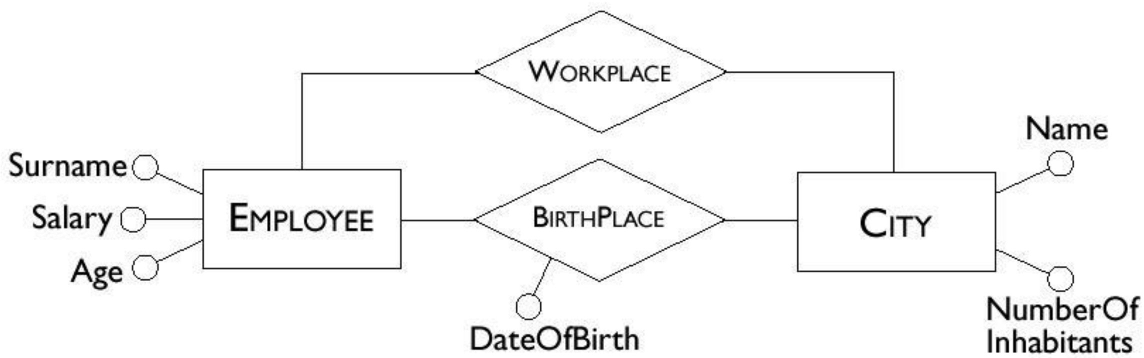
3. Null Attributes - A null value is used when an entity does not have a value for an attribute e.g. dependent name.

4. Calculated attribute - The value for this type of attribute can be derived from the values of other related attributes or entities e.g.

   i. Employment length value can be derived from the value for the start date and the current date.
   ii. Loans held can be a count of the number of loans a customer has.

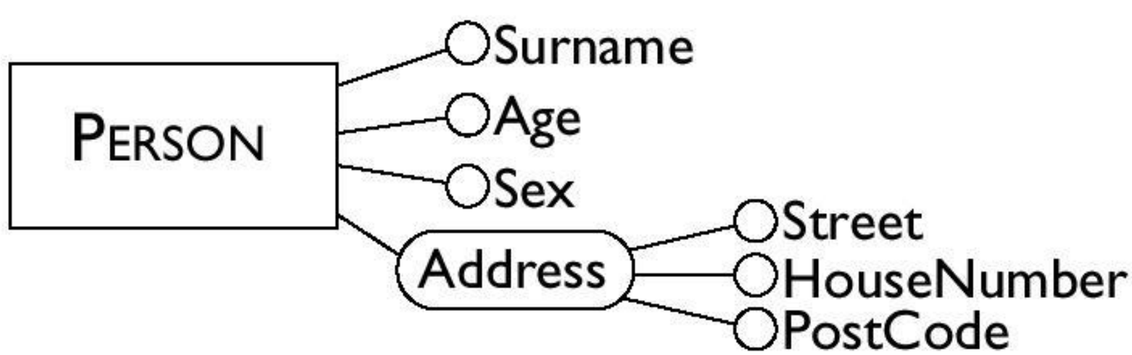
Attribute example



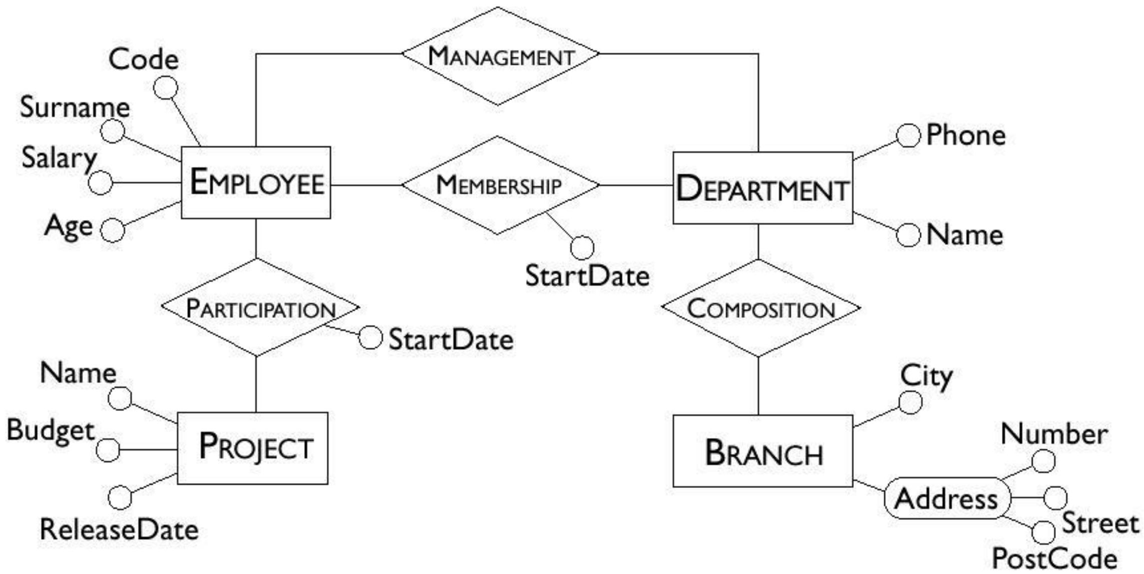


Composite Attributes

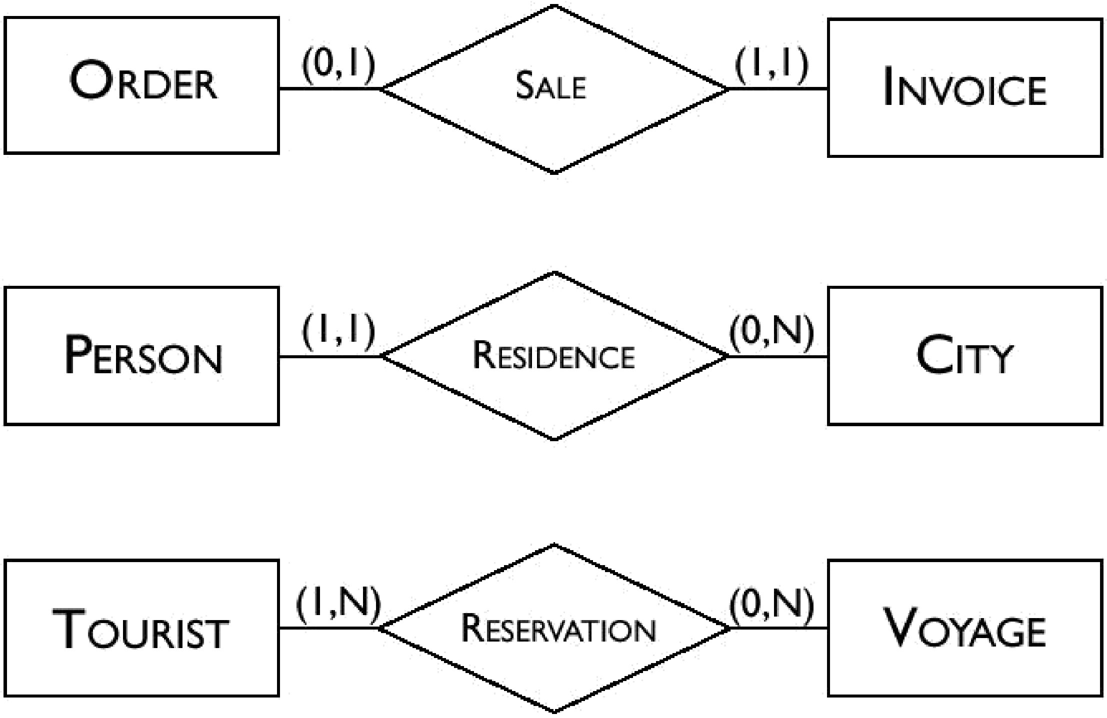
It is sometimes convenient to group attributes of the same entity or relationship that have closely connected meanings or uses. Such groupings are called composite attributes.

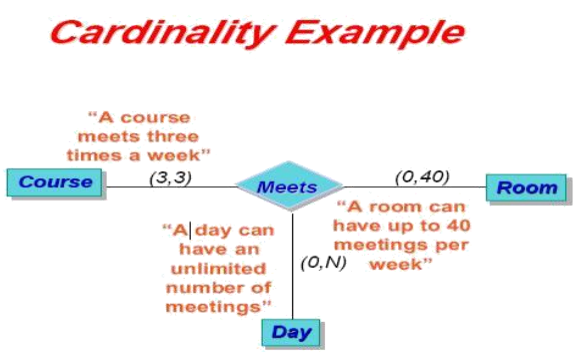


Schema with Attributes



Structured constrains (Cardinalities)
These are specified for each entity participating in a relationship and describe the maximum and minimum number of relationship occurrences in which an entity occurrence can participate. Cardinalities state how many times can an entity instance participate in instances of a given relationship. In principle, a cardinality is any pair of non-negative integers (n,m) such that n≤m. or a pair of the form (n,N) where N means "any number". If minimum cardinality is 0, we say that entity participation in a relationship is optional. If minimum cardinality is 1, we say that entity participation in a relationship is mandatory. If maximum cardinality is 1, each instance of the entity is associated at most with a single instance of the relationship; if maximum cardinality is N, then each instance of the entity is associated with an arbitrary number of instances of the relationship.


ORDER
(0,1)
SALE
(1,1)
INVOICE
PERSON
(1,1)
RESIDENCE
(0,N)
CITY
TOURIST
(1,N)
RESERVATION
(0,N)
VOYAGE


Cardinality Example
"A course meets three times a week"
Course
(3,3)
Meets
(0,40)
Room
"A room can have up to 40 meetings per week"
"A day can have an unlimited number of meetings"
(0,N)
Day

Enhanced E-R Model

Specialization
An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set. This is called specialization of the entity set e.g. the entity bank account could have different types e.g.

Credit account
Checking account
Savings account - interest rate
Checking account - overdraft amount

Under checking account you could have type:

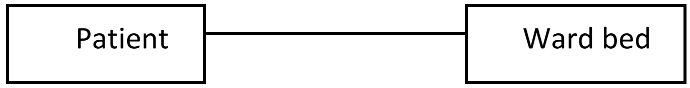
i. Standard check account
ii. Gold checking account
iii. Senior checking account

For the standard if may be divided by number count of checks gold minimum balance and an interest payment.
Senior checking account - age limit
A specialised entity set may be specialised by one or more distinguishing features.

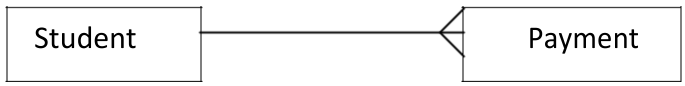
Aggregation
This is abstraction through which relationship are heated as higher-level entities e.g. the relationship set borrower and the entity sets customer and loan can be treated as a higher set called borrower as a whole.

Entity modeling (Diagrammatic representation) relationships

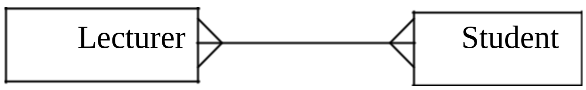
i. One to one relationship



ii. One to many relationship



iii. Many to many relationship

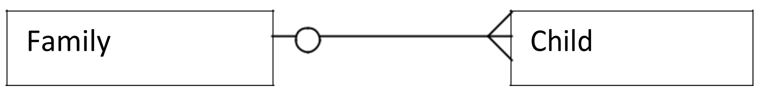


NB: Whenever the degree of a relationship is many to many we must decompose the relationship to one-to -one or one-to-many. The decomposition process will create a new entity.

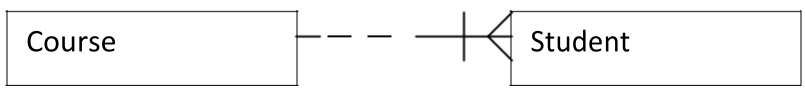
Mandatory and Optional

Optional relationships are shown by either, use of a small circle drawn along the line or a dotted line while mandatory relationships are shown by use of either a bar drawn across the line or a continuous line.

Optional

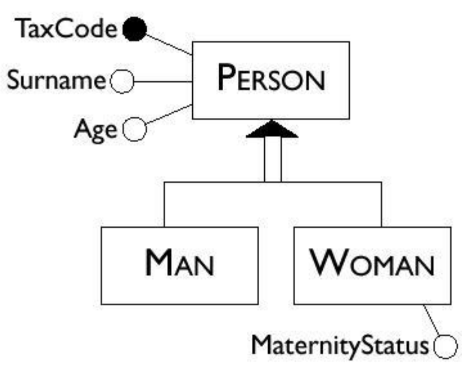


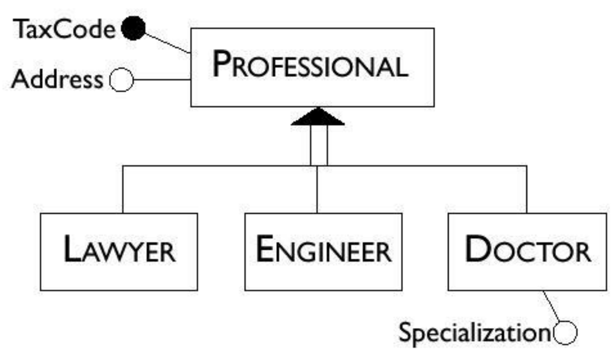
Mandatory



Generalizations

These represent logical links between an entity E, known as parent entity, and one or more entities E1,...,En called child entities, of which E is more general, in the sense that they are a particular case.
In this situation we say that E is a generalization of E1,...,En and that the entities E1,...,En are **specializations** of E.





Properties of Generalization

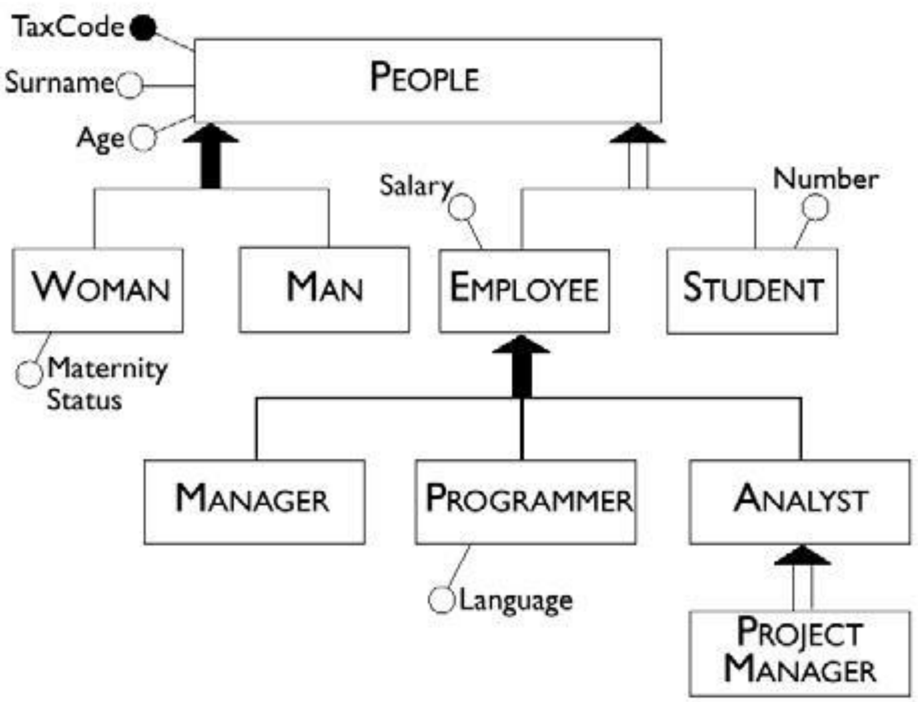
Every instance of a child entity is also an instance of the parent entity. Every property of the parent entity (attribute, identifier, relationship or other generalization) is also a property of a child entity. This property of generalizations is known as **inheritance.**

Types of Generalizations

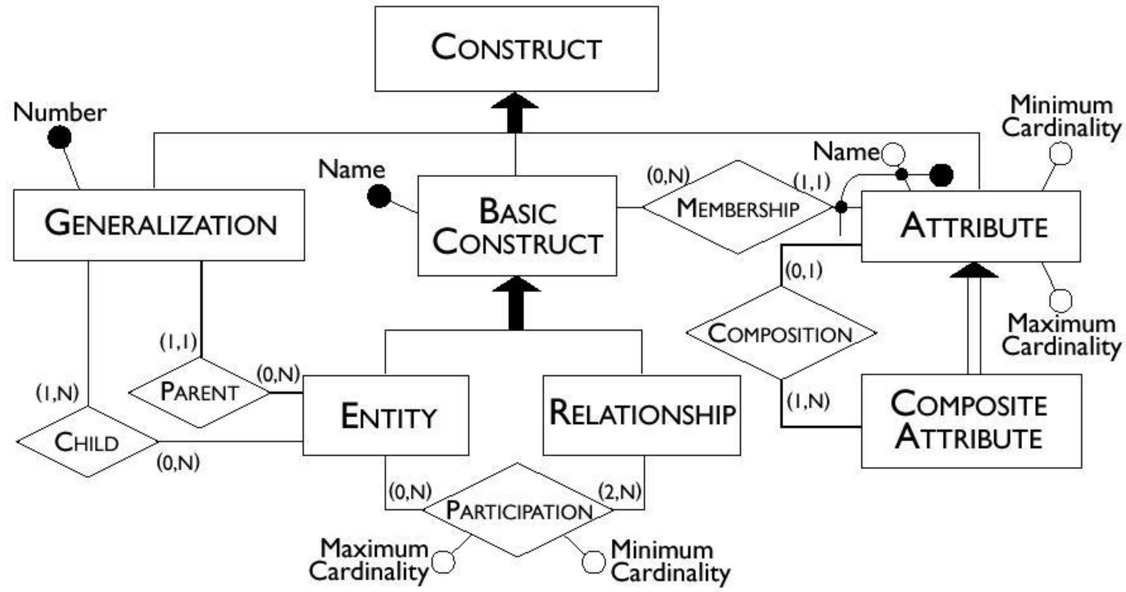
A generalization is **total** if every instance of the parent entity is also an instance of one of its children, otherwise it is **partial**. A generalization is **exclusive** if every instance of the parent entity is at most an instance of one of the children, otherwise it is **overlapping.** The generalization Person, of Man and Woman is total (the sets of men and the women constitute „all" the people) and exclusive (a person is either a man or a woman).
The generalization Vehicle of Automobile and Bicycle is partial and exclusive, because there are other types of vehicle (for example, motor bike) that are neither cars nor bicycle.
The generalization Person of Student and Employee is partial and overlapping, because there are students who are also employed.

Generalization Hierarchies

Total generalization is represented by a solid arrow. In most applications, modeling the domain involves a hierarchy of generalizations that includes several

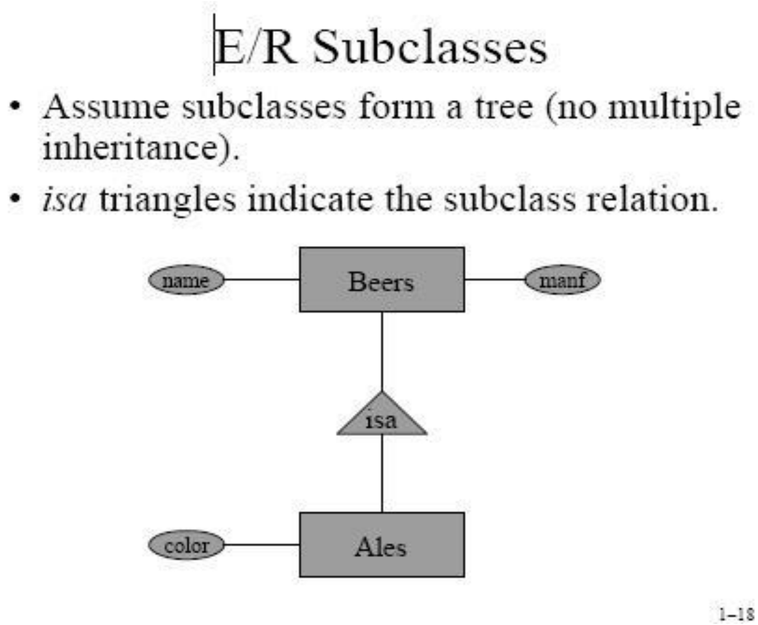


The E-R Model, as an E-R Diagram



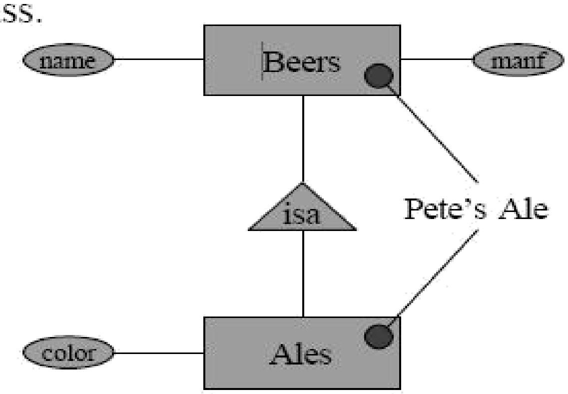
Super classes and Subclasses

Subclass = special case = fewer entities = more properties.
Example: Ales are a kind of beer. In addition to the properties (= attributes and relationships) of beers, there is a "color" attribute for tusker.



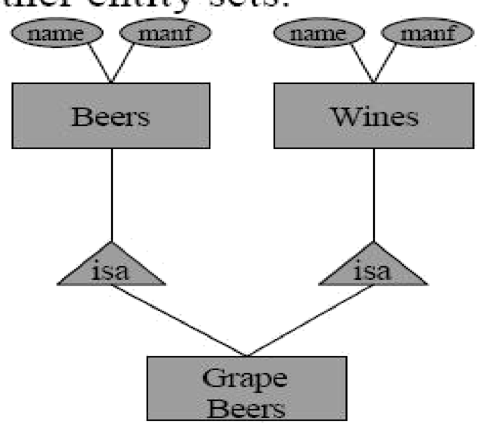
Different Subclass Viewpoints

1. *E/R viewpoint*: An entity has a *component* in each entity set to which it logically belongs.
   - Its properties are the union of the properties of these E.S.
2. Contrasts with *object-oriented viewpoint*: An object (entity) belongs to exactly one class.
   - It *inherits* properties of its superclasses.



Multiple Inheritance

Theoretically, an E.S. could be a subclass of several other entity sets.



Chapter Review Questions

1. Differentiate between an attribute and entity
2. What is the difference between specialization and generalization
3. Why and when do we use E-R diagram?

Cornolly T. $Begg C., *Database systems: a practical approach to design, implementation and management*

CHAPTER THREE

CONCEPTUAL DATA MODELING

Learning objectives:

☺

By the end of the chapter a student shall be able to:

i. **Understand properties of database relations**
ii. **Identify candidate, primary and foreign keys**
iii. **Entity integrity**
iv. **Referential integrity**

3.1 Properties of Relations

- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.

3.2 Relational Keys

Super key

An attribute or a set of attributes, that uniquely identifies a tuple within a relation.

Candidate Key

- ✓ Super key (K) such that no proper subset is a super key within the relation.
- ✓ In each tuple of R, values of K uniquely identify that tuple (uniqueness).
- ✓ No proper subset of K has the uniqueness property (irreducibility

Primary Key

- ✓ Candidate key selected to identify tuples uniquely within relation.

Alternate Keys

- ✓ Candidate keys that are not selected to be primary key.

Foreign Key

- ✓ Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Null

- ✓ Represents value for an attribute that is currently unknown or not applicable for tuple.
- ✓ Deals with incomplete or exceptional data.
- ✓ Represents the absence of a value and is not the same as zero or spaces, which are values.

Entity Integrity

- ✓ In a base relation, no attribute of a primary key can be null.

Referential Integrity

- ✓ If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.

Enterprise Constraints

- ✓ Additional rules specified by users or database administrators.

Chapter Review Questions

1. State the three properties of database relations
2. Differentiate between Entity integrity and Referential integrity

Cornolly T. $Begg C., *Database systems: a practical approach to design, implementation and management*

CHAPTER FOUR

ER-TO- RELATIONAL MAPPING

Learning objectives:

☺

By the end of the chapter a student shall be able to:

- ✓ **Understand and apply Direct mapping from conceptual model in to relational understand**

4.1 ER-to-Relational Mapping

To convert an Entity-Relationship design to a relational database schema, a procedure which includes the following steps may be followed.

Types of Entity Conversion

· Regular Entity Conversion
· Weak Entity Conversion

Relationship Conversion

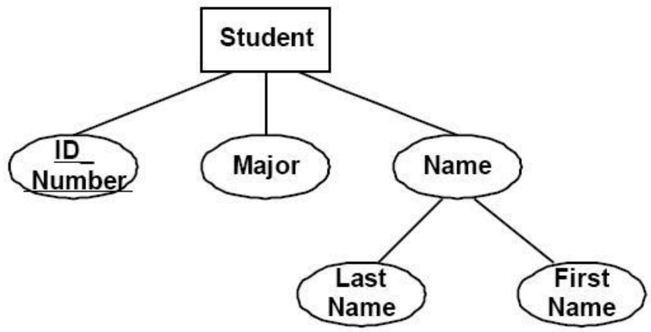
- ✓ **Binary Relationship Conversion** 1:1
  Relationship Conversion
  1:Many Relationship Conversion Many:Many
  Binary Relationship Conversion
- ✓ **Non-Binary Relationship Conversion**
  Attribute Conversion
    - Multivalued Attribute Conversion

Regular Entity Conversion

- ✓ A regular entity is implemented as a relation which contains as its attributes all of the attributes of the entity.
- ✓ Any key attribute (or set of attributes) may be used as a candidate key in the relation.
- ✓ All non-atomic attributes are lost. If it is truly desired to retain non-atomic attributes, then alter the ER model so that the compound attribute is an entity.

Consider the student example:

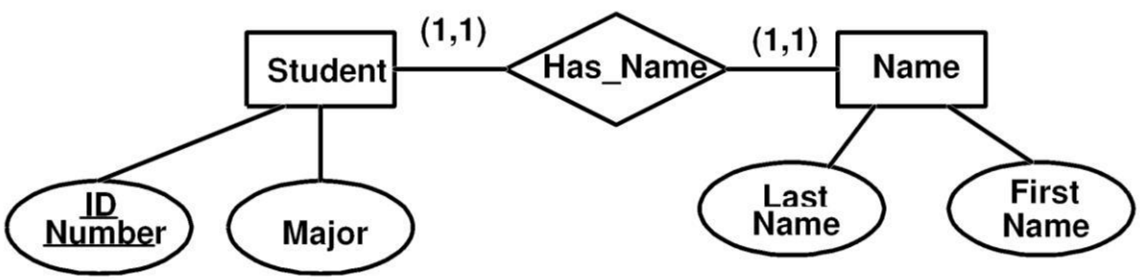
The corresponding relational model would be:



The corresponding relational model would be:

Student			
ID_number	Major	Last Name	First Name

✓ To maintain the existence of the compound attribute Name, re-design the ER schema as follows:



✓ Then apply the techniques described for binary relationships.

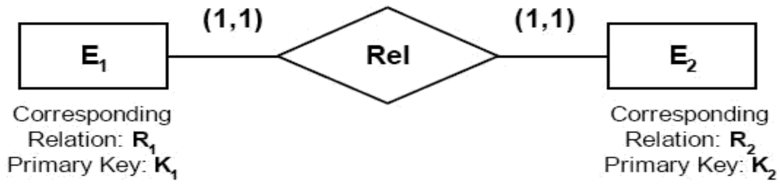
Weak Entity Conversion

The technique is similar to that for a regular entity conversion, except that the primary key of the owner entity must be included, as a foreign key, in the relation for the weak entity.

Relationship Conversion

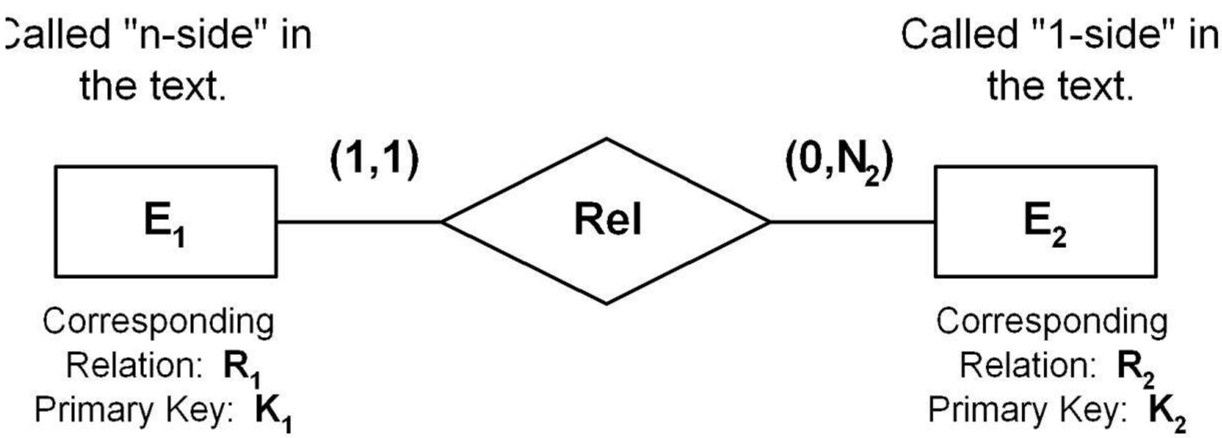
For relationship conversion, it is assumed that the entities participating in the relationship have already been converted according to the rules identified above.

One-to-One Binary Relationship Conversion:



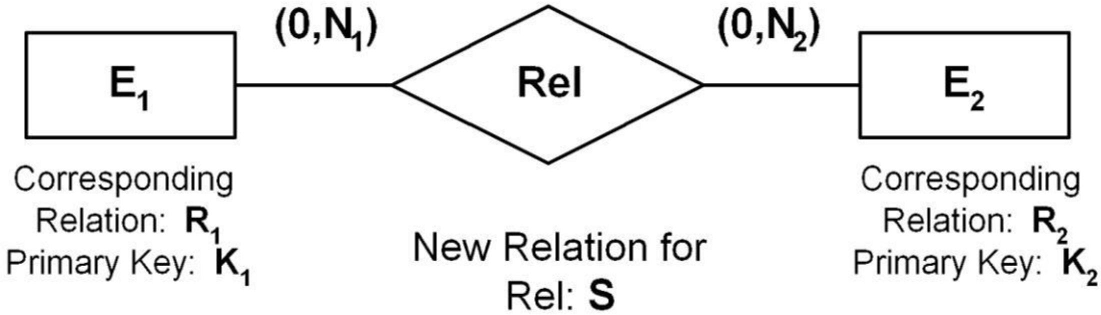
· Choose one of the relations (say R1) as the controller.
·Add the primary key of the controller to the other relation (R2 in this case) as a foreign key.
· Also add any attributes associated with Rel to R1.
·Alternative: Merge the two relations into one.(Only appropriate if the individual relations are not needed in other constructions.)
·If one side is (0,1) and the other (1,1), it is preferable to insert the key from the (0,1) side into the relation of the (1,1) side. This avoids unnecesary use of null values.

One-to-Many Binary Relationship Conversion:



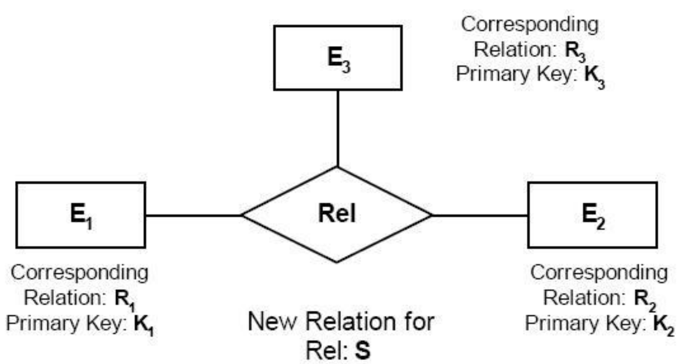
· Insert the primary key K2 of the “1-side” entity as a foreign key into the relation R1 of the “n-side” entity.
· Also add any attributes associated with Rel to R1.

Many-to-Many Binary Relationship Conversion:



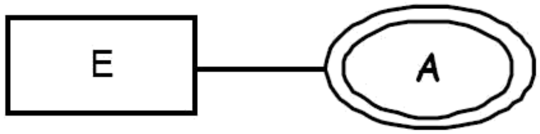
· The new relation S has, as foreign keys, the primary key of each of E1 and E2.
· Include primary key pairs which "match" in the relationship Relationship
· Also include any attributes directly associated with Relationship

Non-Binary Relationship Conversion:



· Create a new relation S which contains as attributes each of the primary keys of the participating entities. These will be foreign keys.
· Also include any attributes directly associated with Relationship

Multivalue Attribute Conversion:



· Create a new relation RA which contains the attribute A, in addition to the primary key of R.
· There is one tuple for each (Primary Key, Attribute value) pair.

Chapter Review Questions

1. Explain the difference between regular and weak entity conversion
2. Explain the steps of many-to-many relationship conversion

Cornolly T. $Begg C., *Database systems: a practical approach to design, implementation and management*

CHAPTER FIVE

FUNCTIONAL DEPENDENCIES AND NORMALIZATION

☺ **Learning objectives:**

By the end of the chapter a student shall be able to:

- ✓ **Define partial dependencies and full dependencies**
- ✓ **Understand transitive dependencies**
- ✓ **Understand dependency diagram**
- ✓ **Rules about functional dependency**
Normalization process

5.1 Definition of partial dependencies and full dependencies

What is Normalization?
Normalization is a data analysis technique for producing a set of relations with desirable properties
There are three basic levels of normalization: First (FNF), Second (SNF), and Third (TNF) Normal Forms.
It is the TNF that is usually used as the basis for the design of the data model and for mapping onto a database
Advantages of Normalization

- It is a formal technique with each stage of normalization process eliminating a particular type of undesirable dependency
- It highlights constraints and dependencies in the data and hence aids in understanding the nature of data
- The TNF produces well-designed databases which provide a higher degree of independence

Un-normalized Form
A relation that contains one or more repeating groups i.e. repeated values for particular attributes with a single record

First Normal Form
A relation is said to be in FNF if and only it contains no repeating groups
Remove repeating groups and propagate higher level primary keys by partitioning the un-normalized relation

Second Normal Form
A relation is in SNF if and only if it is in FNF and every non-key attribute is **fully functionally dependent** on the key attribute
Remove any partial dependencies by partitioning the relation
A relation that is in FNF and has no composite key is necessarily in SNF!

Third Normal Form
A relation is in TNF if and only if it is in SNF and every non-key attribute is independent of all other non-key attributes
Remove any non-key attributes that depend on other non-key dependencies (**transitive dependencies)**

Functional Dependency
Given a relation R, attribute Y of R is functionally dependent on attribute X of R, if and only if each X-value in R has associated with it precisely one Y-value in R (at any one time) R.X ---------------→ R.Y
(R.X functionally determines R.Y)

Normalization Steps
Represent the un-normalized relation
- List Attributes
- Identify Repeating Groups
- Identify Key Attributes

Convert to FNF
- By removing Repeating Groups
- Understand the Dependencies
- Functional Dependency Diagrams may be used

Convert to SNF
- by removing **partial dependencies**

Convert to TNF
- by removing **transitive dependencies**

Rationalize the Results
- Consider whether to combine any resulting relations that have identical keys
- Discard any relations that is redundant i.e. its attributes are contained within another relation
- Identify foreign keys
- Review the names of the relations to ensure they reflect the information content

Examples on how to remove dependencies

- Un-normalized Relation

INVOICE
<u>InvoiceNo</u>
CustomerNo
CustomerName
Address
City
Phone
Date
OrderNo
Rep
FOB
    Code
    Description
    Qty
    UnitPrice
    Total
Subtotal
Shipping
Vat
Grandtotal
Paymentmode
Ccno
Ccname
Ccexpiry
Authorizationcode

Normalization Example: FNF

INVOICE
<u>InvoiceNo</u>
CustomerNo
CustomerName
Address
City
Phone
Date
OrderNo
Rep
FOB
Subtotal

Shipping
Vat
Grandtotal
Paymentmode
Ccno
Ccname
Ccexpiry
Authorizationcode

PRODUCT
<u>InvoiceNo</u>
<u>Code</u>
Description
Qty
UnitPrice
Total

- Normalization Example: SNF

INVOICE
<u>InvoiceNo</u>
CustomerNo
CustomerName
Address
City
Phone
Date
OrderNo
Rep
FOB Subtotal

Shipping Vat

Grandtotal

Paymentmode

Ccno Ccname

Ccexpiry

Authorizationcode

INVOICE-
PRODUCT <u>InvoiceNo</u>
<u>Code</u>
Qty
Total
PRODUCT
<u>Code</u>
Description
UnitPrice

- Normalization Example: TNF

INVOICE
<u>InvoiceNo</u>
CustomerNo
* Date
OrderNo *
Rep
FOB Subtotal

Shipping Vat

Grandtotal

Paymentmode

Ccno *

Authorizationcode

CUSTOMER
<u>CustomerNo</u>
CustomerName
Address
City
Phone
CREDIT
<u>Ccno</u>
Ccname
Ccexpiry
INVOICE-PRODUCT **<u>InvoiceNo</u>**
<u>Code</u>
Qty
Total
PRODUCT
<u>Code</u>
Description
UnitPrice

Example: An invoice

Invoice No. ________________________ Date ________________________

Customer ____________________________ Delivery to ____________________

Address ____________________________

____

Product Code	Description	Quantity	Price	Amount

Thank you. Amount____________

Un-normalised data.

Invoice (Invoice no., Date, Customer, Cust_address, Deliv_To,Product code, Quantity, Unit Price, amount, Invoice amount)

INF (Identify and separate repeating groups to form a new entity)

INVOICE (Invoice number, date, customer address, Deliv_ address, Invoice_Amount) **PRODUCT** (Product code, invoice number, product description, Quantity,Unit price, amount)

2NF (identity and separate non-key attributes not fully dependent on key attribute)

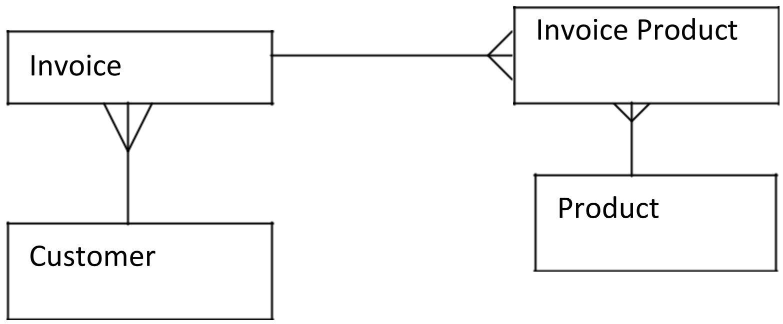
INVOICE (Invoice-no, date, customer address, del.address, invoice total)
PRODUCT (prod-code, prod-description, unit price)
INVOICE PRODUCT (Prod_Code,Invoice_No,Quantity,Amount)

3NF (Identify non-key attributes dependent on other non-key attributes)

INVOICE (Invoice-no, Customer_Number, Date, invoice total)
PRODUCT (Prod-code, prod-description, unit price)
INVOICE PRODUCT(Prod_Code,Invoice_No,Quantity,Amount)
CUSTOMER (Customer_Number,Customer_Name, customer_Address, del.address)

NB: Whenever there is no composite key the table is in 3NF

Corresponding ERD



Advantages of Normalization Approach

1. It is a formal technique with each stage of normalisation process elimination a particular type of undesirable dependency as well as each stage of normalisation eliminating a certain type of redundancy.
2. It highlights constraints and dependencies in the data and helps in understanding the nature of the data.
3. The 3NF produces well-designed databases, which provide a high degree of independence.

Disadvantages

1. It depends a thorough understanding of the entities and their relationships.
2. It"s a complex process particularly if the entities are many.

Review Questions

Suggested further reading

1 . A customer account details in a bank are stored in a table that has the

1. **Cornolly**following**T.** structure,**$BeggC.,**normalise*Database* this*systems:*data to*apractical*3NF. *approach to design, implementation and management*

Customer (branch -no, account no, address, postcode, tel)

2. What is transitive dependency?

1. **Cornolly T. $Begg C.,** *Database systems: a practical approach to design, implementation and management*

Sample Exam Questions

Mt Kenya University
UNIVERSITY EXAMINATION 2010/2011

SCHOOL OF APPLIED AND SOCIAL SCIENCES
DEPARTMENT OF INFORMATION TECHNOLOGY

BIT 2102 : DATABASE SYSTEMS

TIME: 2HRS

INSTRUCTIONS: SECTION A IS COMPULSORY.ANSWER ANY OTHER TWO QUESTIONS IN SECTION B

SECTION A
QUESTION ONE

a) Define the following terms
   i. Database
   ii. DBMS
   iii. Foreign key
   iv. Attributes

   v. Relationship 10marks b) Explain the three levels of ANSI-SPARC three level architecture 6marks

c) You are instructed to develop a database system to store the information of patients visiting the given hospital.
   i. Name six(6) attributes/fields needed for the system 6marks
   ii. Identify the primary key which can be used to differentiate between the patients. 1mark

d) Define the term normalization and explain the three normal forms 7marks

SECTION B
QUESTION ONE

a) Based on the following table

Hotel (hotelNo, hotelName, city)
Room (roomNo,hotelNo, type, price)
Booking (HotelNo,GuestNo, DateFrom, DateTo, RoomNo)
Guest(GuestNo, GuestName, GuestAddress)
Where hotel contains hotel details hotelNo is the primary key
Room contains room details for each hotel and (roomNo, HotelNo) forms the primary key.
and Guest contains Guest details and GuestNo is the primary key.

Write an SQL statement for the following:

a) List full details of all hotels in London. 5marks
b) List the names and addresses of all guests in London, alphabetically ordered by name 5marks
c) List the price and type of all rooms at the Grosvenor Hotel. 5marks
d) List the number of rooms in each hotel in London 5marks

QUESTION TWO

a) Describe five advantages of database management system 10marks
b) Explain the three database languages 6marks
c) "The File-based Approach had several challenges" explain the statement (4marks)

QUESTION THREE

a) ABC Ltd plans to computerize its sales ordering and stock control system. A feasibility study has strongly suggested that a relational database system be installed. The details of ABC's sales and stock control are as follows:

Customers send in orders for goods. Each order may contain requests for variable quantities of one or more products from ABC's range. ABC keeps a stock file showing for each product the product details and the preferred supplier, the quantity in stock, the reorder level and other details. ABC delivers those goods that it has in stock in response to the customer order and an invoice is produced for the despatched items. Any items that were not in stock are placed on a back order list and these items are usually re-ordered from the preferred supplier. Occasionally items are ordered from alternative sources. In response to the invoices that are sent out to ABC's customers, the customers send in payments. Sometimes a payment will be for one invoice, sometimes for part of an invoice and sometimes for several invoices and part-invoices.

Draw an entity-relationship model, stating any assumptions made.
10marks

b) Differentiate between the following
   i. Super-classes & Subclasses
   ii. Primary key and foreign key
   iii. Recursive Relationship and Inheritance
   iv. Weak Entity and entity
   10marks

QUESTION FOUR

a) Consider the following relation:

OrderItem (OrderNo, ItemCode, Quantity, OrderDate, Description, CustomerNo, CreditLimit, DeliveryAddress)

The following Functional Dependencies (FDs) apply to OrderItem: FD1 OrderNo, ItemCode -> Quantity

FD2 OrderNo, ItemCode -> OrderDate
FD3 ItemCode -> Description

FD4 OrderNo -> CustomerNo FD5
CustomerNo -> CreditLimit FD6
CustomerNo -> DeliveryAddress

(i) Define the two Transitive Dependencies that can be derived from the FDs and give an explanation of the derivation.

(4 marks)

(ii) (ii) Given the FDs above, normalise the OrderItem to 3NF, carefully showing your intermediate steps.

(8 marks)

b) Employees table

	Title	FirstName	LastName
1	Inside Sales Coordinator	Laura	Callahan
2	Sales Manager	Steven	Buchanan
3	Sales Representative	Nancy	Davolio
4	Sales Representative	Anne	Dodsworth
5	Sales Representative	Robert	King
6	Sales Representative	Janet	Leverling
7	Sales Representative	Margaret	Peacock
8	Sales Representative	Michael	Suyama
9	Vice President, Sales	Andrew	Fuller

Write SQL statements do the following

i. Retrieve all columns in the employee table 2marks

ii. List all the sales representatives 3marks

iii. List the firstname and last name of employees 3marks

Mt Kenya University

SCHOOL OF PURE & APPLIED SCIENCES
IT DEPARTMENT

BIT 2102: DATABASE SYSTEMS

2 HOURS

INSTRUCTIONS

- Answer all questions in section **A** and other **TWO** in section **B**
- **No** reference material allowed
- Time **2** Hours

SECTION A (COMPULSORY – 30 MARKS)

QUESTION ONE

a) Define the following terms.
   a) database
   b) database management system
   c) data integrity
   d) schema
   e) instance

[10mks]

ii) Discuss the business conditions that enable a business database to be effective. [10mks]

iii) a) distinguish between a logical and physical data independence. [4mks]
   b) Explain the purposes of the three DBMS languages [6mks]

SECTION B

QUESTION ONE

2 a) Consider the table structure below.

TABLE STUDENT:

STUDNO	LNAME	FNAME	INITIAL	DOB	UNIT CODE

101	Mwangi	John	N	11/8/80	2100
102	Kimaiyo	Peter	M	12/12/84	2200
103	Chebet	Martha	K	2/4/83	2305
104	Oduor	Louis	M	11/06/80	2200
105	Njuguna	Frank	G	15/9/85	2100

Using SQL,

i. Create the table STUDENT [3 Mks]

ii. Having created the table structure in (i) above, enter the first two records into the table. [2 Mks]

iii. Return the names of students taking unit code 9945 [2 Mks]

iv. Remove the table STUDENT from the database [2 Mks]

v. Order the table by unit code in ascending [2 Mks]

b) Hence explain the usage of these clauses: JOIN and CALCULATE [4 Mks]

c) Describe the following terms. [5mks]

i) entity
j) attribute
k) relationship

QUESTION TWO

a) Identify and briefly discuss any SIX of date's twelve rules of distribution [12 Mks]

b) What do you understand by E-R Modeling? [2 Mks]

Hence explain briefly terms: [6 Mks]

i. Cardinality
ii. Composite attribute
iii. Weak attribute

QUESTION THREE

a) with the aid of well labeled diagrams describe the three types of relationships. [12mks]

b) Distinguish between generalization and specialization. [4mks]

c) Explain the roles of a database administrator [5mks]

QUESTION FOUR

a) Explain the term normalization. and hence Outline the nature of the following normal forms

i) 1st normal form

ii) 2^{nd} normal form

iii) 3^{rd} normal form [3mks]

b) Explain five roles of a data base administrator. [10mks]

c) Discuss the properties of a transaction. [7mks]