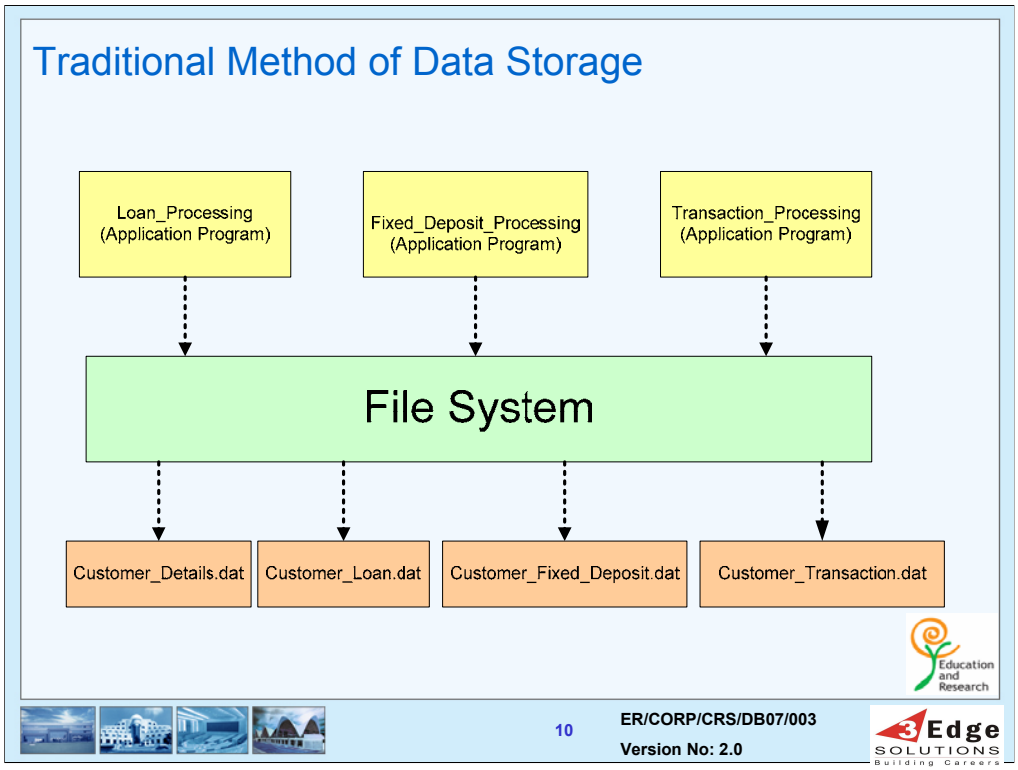
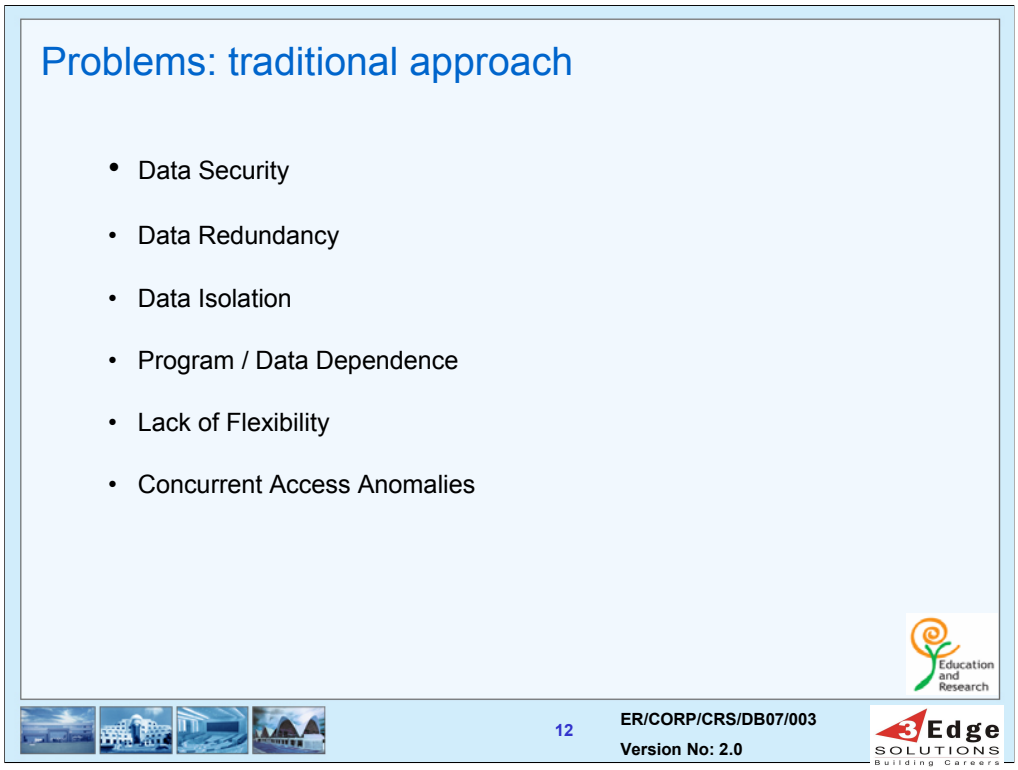


•In the traditional approach, we used to store information in flat files which are maintained by the file system under the operating system's control.

•Application programs go through the file system to access these flat files



Disadvantages of the traditional approach

Data Security: The data as maintained in the flat file(s) is easily accessible and therefore not secure

Example: *Consider the Banking System. The Customer_Transaction file has details about the total available balance of all customers. A Customer wants information about his account balance. In a file system it is difficult to give the Customer access to only his data in the file. Thus enforcing security constraints for the entire file or for certain data items are difficult.*

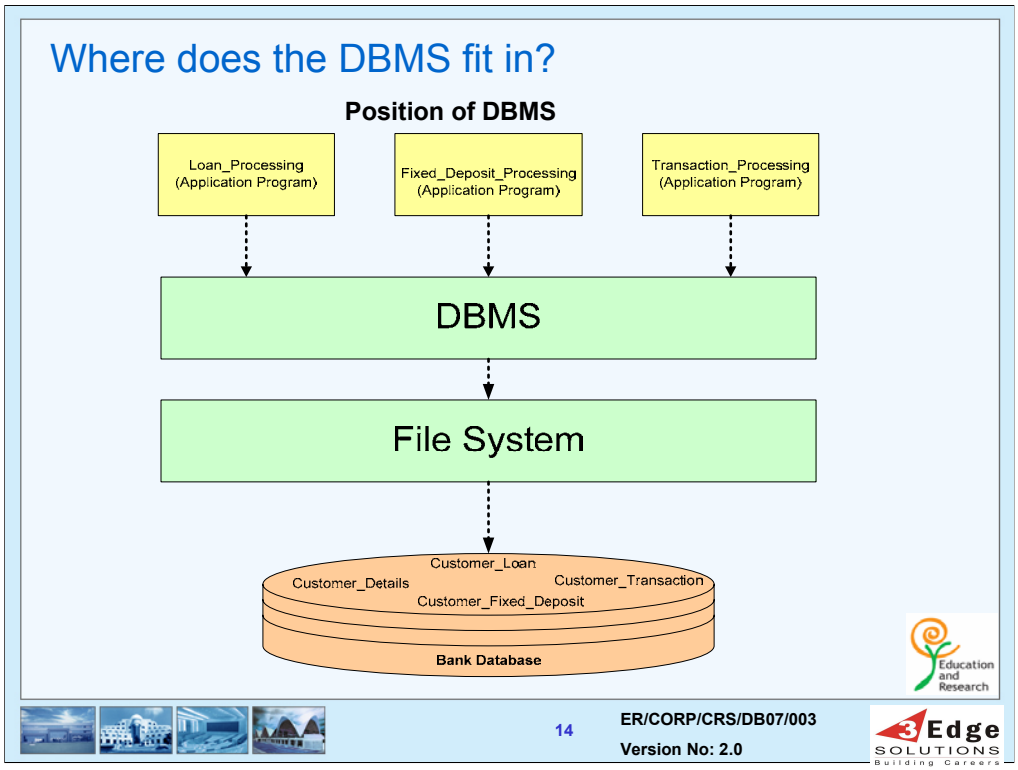
Data Redundancy: Often the same information is duplicated in two or more files.

This duplication of data (redundancy) leads to higher storage and access cost. In addition it may lead to data *inconsistency*[2]. For Example, assume the same data is repeated in two or more files. If change is made to data in one file, it is required that the change be made to the data in the other file as well. If this is not done, it will lead to error during access to the data.

Example: *Assume Customer's details such as Cust_Last_Name, Cust_Mid_Name, Cust_First_Name, Cust_Email is stored both in the Customer_Details file and the Customer_Fixed_Deposit file. If the Email ID of one Customer, for example, Langer S. Justin changes from Langer_Justin@yahoo.com to Langer_Justin@rediffmail.com, the Cust_Email has to be updated in both the files; otherwise it will lead to inconsistent data.*

However, one can design file systems with minimal redundancy. Data redundancy is sometimes preferred. Example: Assume the Customer's details such as Cust_Last_Name, Cust_Mid_Name, Cust_First_Name and Cust_Email are not stored in the Customer_Fixed_Deposit file. If it is required to get this information about the customer along with his fixed deposit details, it would mean that the details be retrieved from two files. This would mean an increased overhead. It is thus preferred to store the information in the Customer_Fixed Deposit file itself.

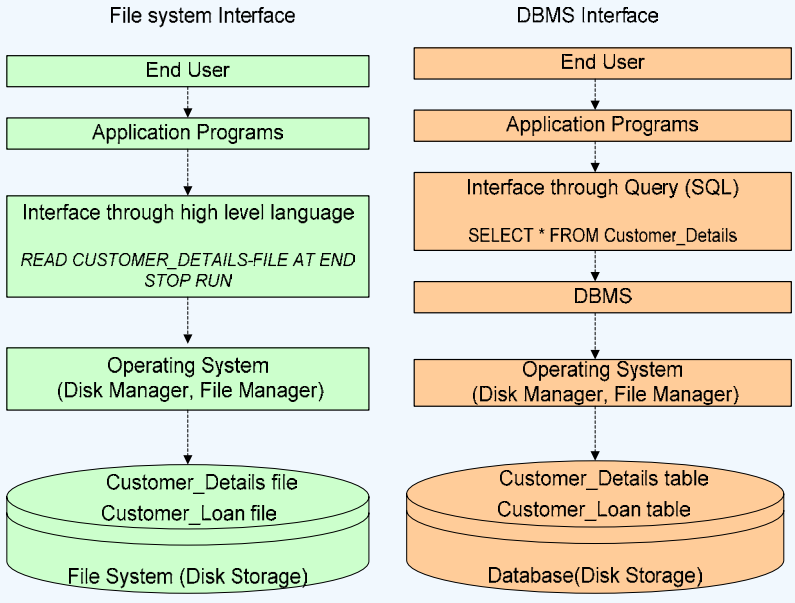
Data Isolation: Data Isolation means that all the related data is not available in one file. Generally, the data is scattered in various files, and the files may be in different formats, therefore writing new application programs to retrieve the appropriate data is difficult.



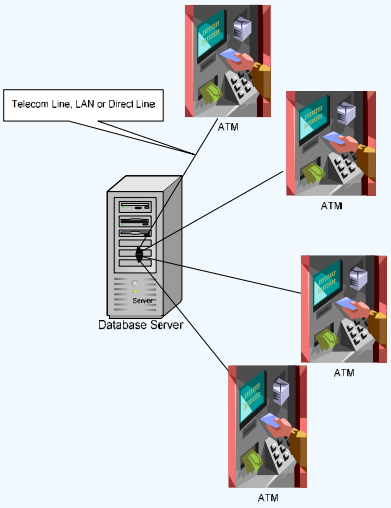
- Now, the DBMS acts as a layer of abstraction on top of the File system.

- You might have observed that, for interacting with the file system, we were using high level language functions for example, the ‘c’ file handling functions. For interacting with the DBMS we would be using a Query language called SQL

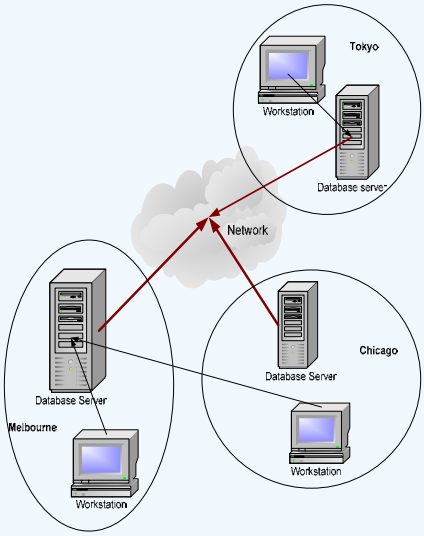
Difference Between File and DBMS Operations



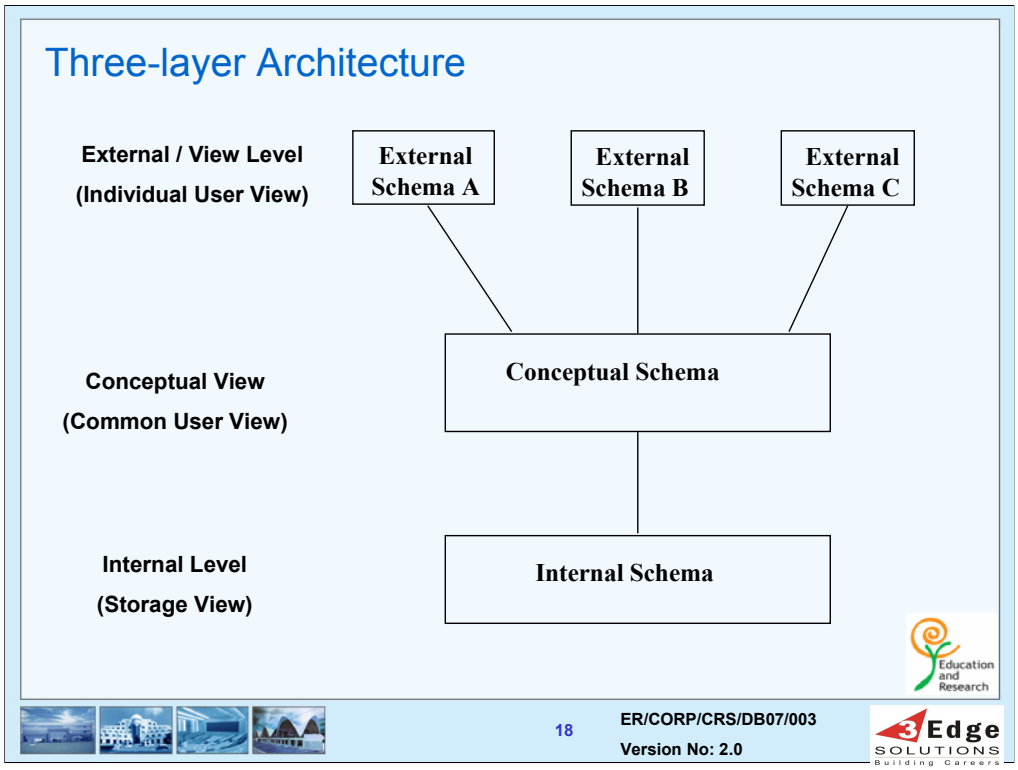
Types of Databases



Centralized Database

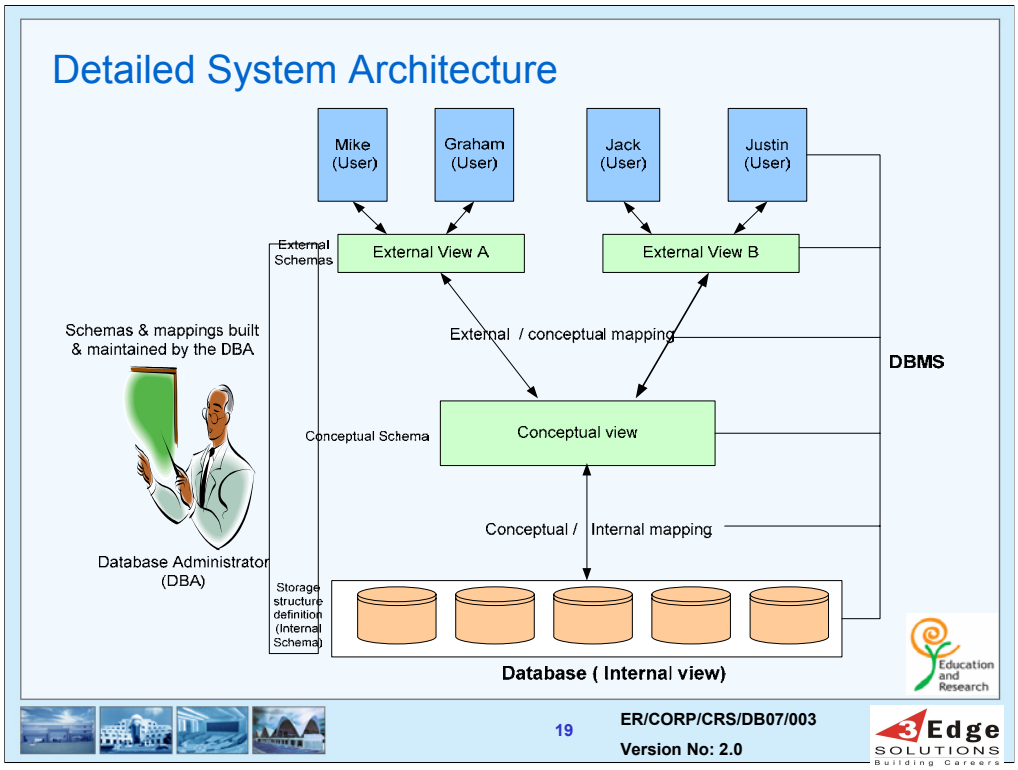


Distributed Database



Services provided by a DBMS

•Data management
•Data definition
•Transaction support
•Concurrency control
•Recovery
•Security & integrity
•Utilities- facilities like data import & export, user management, backup, performance analysis, logging & audit, physical storage control



In the above figure, the three level of DBMS architecture is depicted. The External view is how the Customer, Smith A. Mike views it. The Conceptual view is how the DBA views it. The Internal view is how the data is actually stored.

Data Models

Definition of data model :

A conceptual tool used to describe

- Data
- Data relationships
- Data semantics
- Consistency constraints

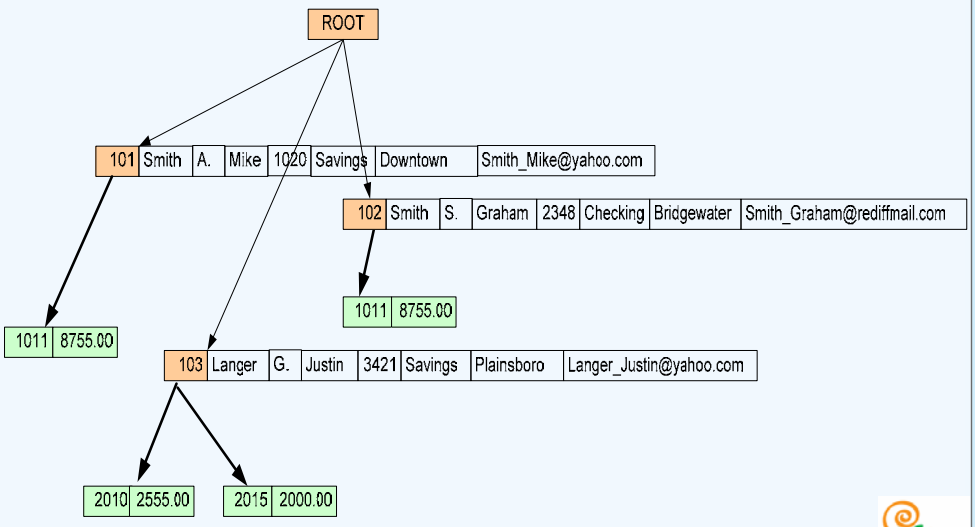
Commercial Packages

•Hierarchical Model - IMS
•Network Model - IDMS
•Relational Model - Oracle, DB2

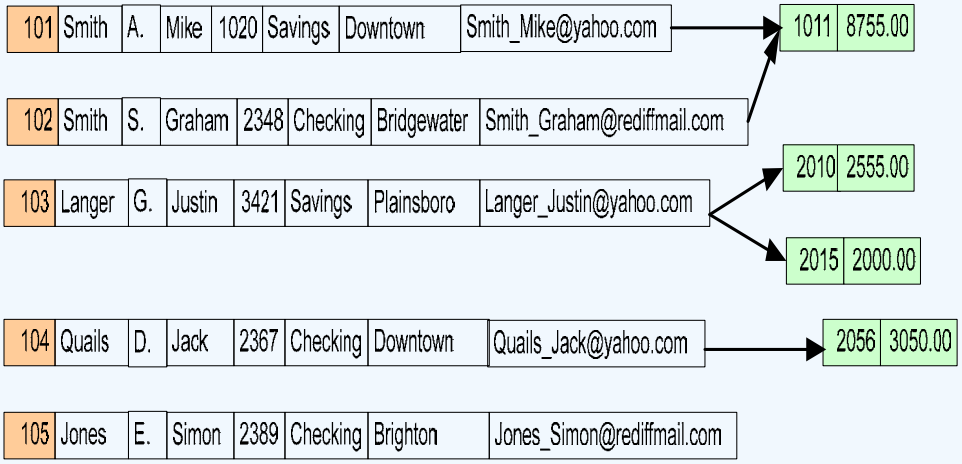
Types of data models

- Object based logical model
  - Entity relationship model
- Record based logical model
  - Hierarchical data model
  - Network data model
  - Relational data model

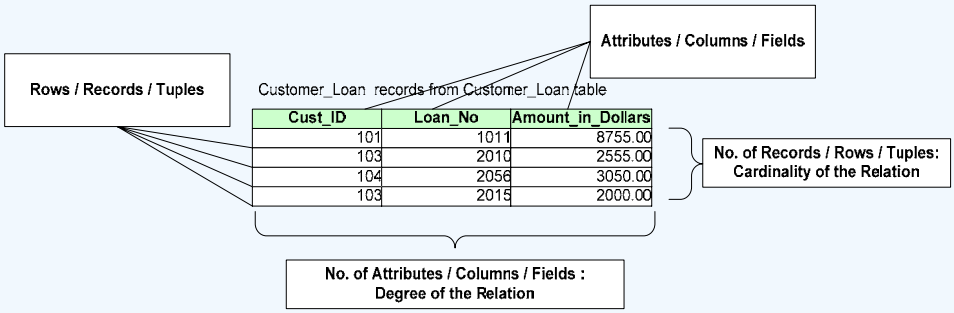
Record based data model – Hierarchical data model



Record based data model – Network data model



Record based data model – Relational data model



Cust_ID	Cust_Last_Name	Cust_Mid_Name	Cust_First_Name	Account_No	Account_Type	Bank_Branch	Cust_Email
101	Smith	A.	Mike	1020	Savings	Downtown	Smith_Mike@yahoo.com
102	Smith	S.	Graham	2348	Checking	Bridgewater	Smith_Graham@rediffmail.com
103	Langer	G.	Justin	3421	Savings	Plainsboro	Langer_Justin@yahoo.com
104	Quails	D.	Jack	2367	Checking	Downtown	Quails_Jack@yahoo.com
105	Jones	E.	Simon	2389	Checking	Brighton	Jones_Simon@rediffmail.com

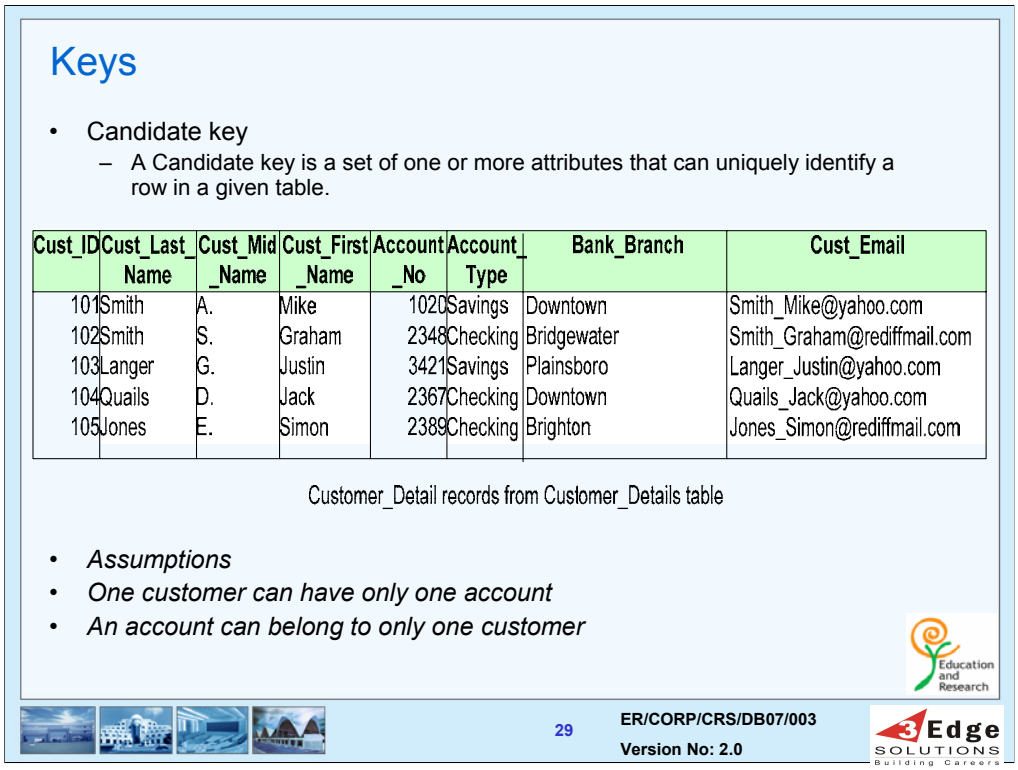
Customer_Detail records from Customer_Details table

Relational model basics

- Data is viewed as existing in two dimensional tables known as relations
- A relation (table) consists of unique attributes (columns) and tuples (rows)
- Tuples are unique
- Sometimes the value to be inserted into a particular cell may be unknown, or it may have no value. This is represented by a **null**
- Null is not the same as zero, blank or an empty string
- Relational Database: Any database whose logical organization is based on relational data model.
- RDBMS: A DBMS that manages the relational database.

•Though logically data is viewed as existing in the form of two dimensional tables, actually, the data is stored under the file system only.

•The RDBMS provides an abstraction on top of the file system and gives an illusion that data resides in the form of tables.



Superkey

An attribute, or group of attributes, that is sufficient to distinguish every tuple in the relation from every other one.
Each super key is called a *candidate key*

A candidate key is all those set of attributes which can uniquely identify a row. However, any subset of these set of attributes would not identify a row uniquely

For example, in a shipment table, "S#, P#" is a candidate key. But, S# alone or P# alone would not uniquely identify a row of the shipment table.

Primary key

The candidate key that is chosen to perform the identification task is called the ***primary key*** and any others are ***alternate keys***
Every tuple must have, by definition, **a unique value for its primary key.** A primary key which is a combination of more than one attribute is called a ***composite primary key***

Foreign key

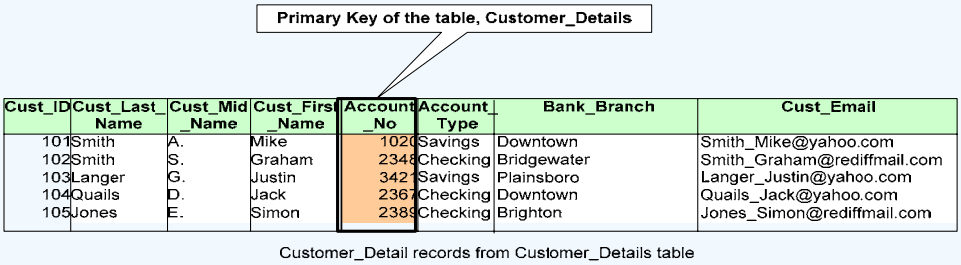
•A *foreign* key is a "copy" of a primary key that has been exported from one relation into another to represent the existence of a relationship between them. A foreign key is a copy of the *whole* of its parent primary key i.e if the primary key is composite, then so is the foreign key
•Foreign key values do not (usually) have to be unique
•Foreign keys can also be *null*
•A composite foreign key **cannot** have some attribute(s) null and others non-null

Overlapping candidate keys: Two candidate keys overlap if they involve any attribute in common. For e.g, in an Employee table, E#, Ename and Emailid, Ename are two overlapping candidate keys. (they have Ename in common)

Attribute that does not participate in any candidate key is called a **Non-key attribute**

Keys

- Primary key
  - During the creation of the table, the Database Designer chooses one of the Candidate Key from amongst the several available, to uniquely identify row in the given table.



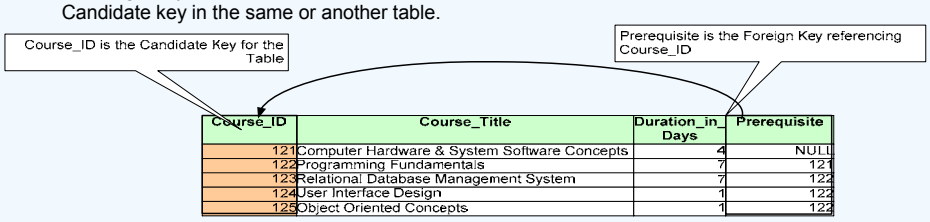
Cust_ID	Cust_Last_Name	Cust_Mid_Name	Cust_First_Name	Account_No	Account_Type	Bank_Branch	Cust_Email
101	Smith	A.	Mike	1020	Savings	Downtown	Smith_Mike@yahoo.com
102	Smith	S.	Graham	2348	Checking	Bridgewater	Smith_Graham@rediffmail.com
103	Langer	G.	Justin	3421	Savings	Plainsboro	Langer_Justin@yahoo.com
104	Quails	D.	Jack	2367	Checking	Downtown	Quails_Jack@yahoo.com
105	Jones	E.	Simon	2389	Checking	Brighton	Jones_Simon@rediffmail.com

Customer_Detail records from Customer_Details table

  - Give preference to numeric column(s)
  - Give preference to single attribute
  - Give preference to minimal composite key

Keys

- Foreign key
  - A Foreign Key is a set of attribute (s) whose values are required to match values of a Candidate key in the same or another table.



Course_ID	Course_Title	Duration_in_Days	Prerequisite
121	Computer Hardware & System Software Concepts	4	NULL
122	Programming Fundamentals	7	121
123	Relational Database Management System	7	122
124	User Interface Design	1	122
125	Object Oriented Concepts	1	122

- Point to remember
  - A Foreign Key is a set of attributes of a table, whose values are required to match values of some Candidate Key in the same or another table
  - The constraint that values of a given Foreign Key must match the values of the corresponding Candidate Key is known as Referential constraint
  - A table which has a Foreign Key referring to its own Candidate Key is known as Self-Referencing table
  - Any superset of a Candidate Key is a Super Key
  - The attributes other than the Primary Key attributes in a table/relation are called Non-Key attributes

Conceptual design

Entity Relationship modeling

Education and Research

Database Design Techniques

- Top down Approach
  - E R Modeling
- Bottom Up approach
  - Normalization

ER modeling

- **ER modeling:** A graphical technique for *understanding* and organizing the data independent of the actual database implementation
- **Entity:** Any thing that may have an independent existence and about which we intend to collect data.
  Also known as **Entity type**.
- **Entity instance:** a particular member of the entity type e.g. a particular student
- **Attributes:** Properties/characteristics that describe entities
- **Relationships:** Associations between entities

Attributes

- The set of possible values for an attribute is called the **domain** of the attribute
  Example:
  - The domain of attribute ***marital status*** is just the four values: single, married, divorced, widowed
  - The domain of the attribute month is the twelve values ranging from January to December
- **Key attribute***:* The attribute (or combination of attributes) that is unique for every entity instance
  - E.g the account number of an account, the employee id of an employee etc.
- If the key consists of two or more attributes in combination, it is called a **composite key**

Simple Vs composite attribute

- **Simple attribute**: cannot be divided into simpler components
  E.g age of an employee

- **Composite attribute**: can be split into components
  E.g Date of joining of the employee.
  - Can be split into day, month and year

Single Vs Multi-valued Attributes

- **Single valued** : can take on only a single value for each entity instance
  E.g. ***age*** of employee. There can be only one value for this

- **Multi-valued**: can take many values
  E.g. ***skill set*** of employee

Stored Vs Derived attribute

- **Stored Attribute**: Attribute that need to be stored permanently.
    - E.g. ***name*** of an employee
- **Derived Attribute**: Attribute that can be calculated based on other attributes
    - E.g. : ***years of service*** of employee can be calculated from date of joining and current date

Regular Vs. Weak entity type

- **Regular Entity**: Entity that has its own key attribute.

  E.g.: Employee, student ,customer, policy holder etc.

- **Weak entity**: Entity that depends on other entity for its existence and doesn't have key attribute of its own

  E.g. : spouse of employee

The spouse data is identified with the help of the employee id to which it is related

Relationships

- A **relationship type** between two entity types defines the set of all associations between these entity types
- Each instance of the relationship between members of these entity types is called a **relationship instance**

E.g if **Works-for** is the relationship between the Employee entity and the department entity, then Ram works for Comp.sc department, shyam works –for electrical department ..etc are relationship instances of the relationship, works-for

Degree of a Relationship

- **Degree**: the number of entity types involved
  - One *Unary*
  - Two *Binary*
  - Three *Ternary*

*E.g.: employee **manager-of** employee is unary*
*employee **works-for** department is binary*
*customer **purchase** item, shop keeper is a ternary relationship*

Cardinality

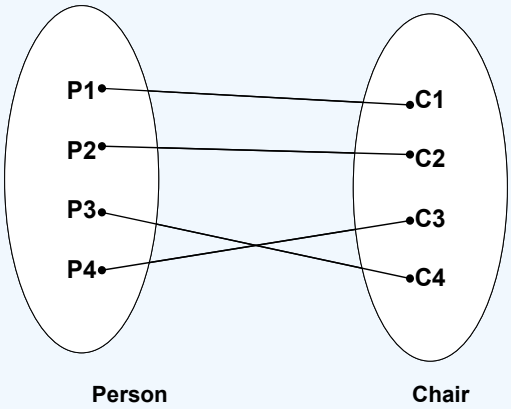
- Relationships can have different *connectivity*
  - **one-to-one** (1:1)
  - **one-to-many** (1:N)
  - **many-to- One** (M:1)
  - **many-to-many** (M:N)

E.g.:

Employee **head-of** department (1:1)
Lecturer **offers** course (1:n) assuming a course is taught by a single lecturer
Student **enrolls** course (m:n)

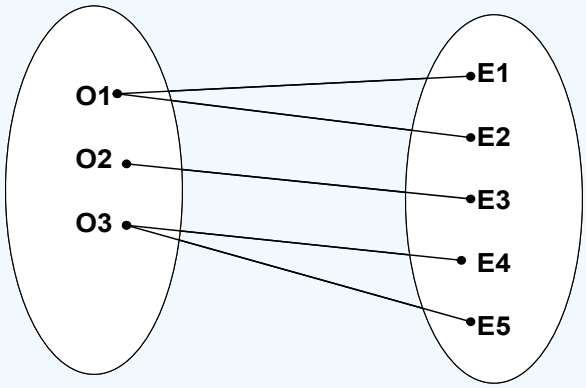
The minimum and maximum values of this connectivity is called the **cardinality of the relationship**

Cardinality – One - To - One



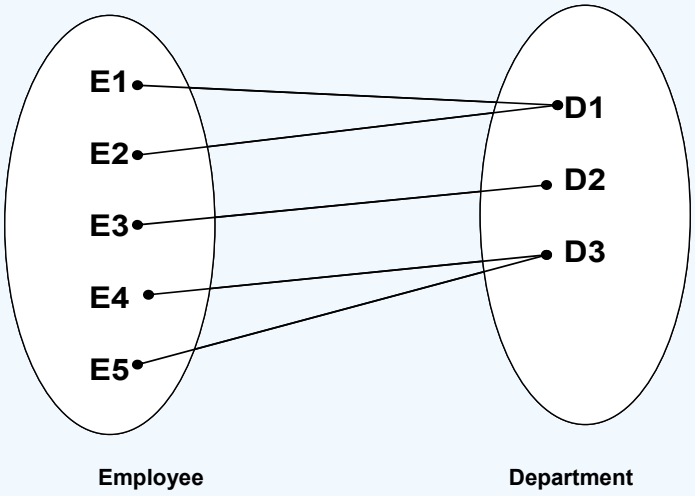
One instance of entity type Person is related to one instance of the entity type Chair.

Cardinality – One -to- Many



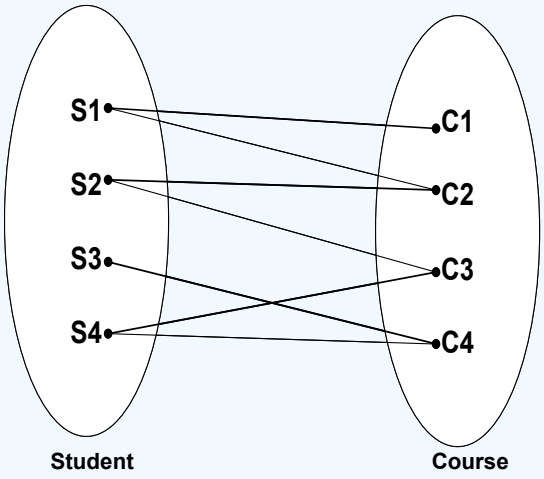
One instance of entity type Organization is related to multiple instances of entity type Employee

Cardinality – Many-to-One



Reverse of the One to Many relationship.

Cardinality – Many-to-Many



Multiple instances of one Entity are related to multiple instances of another Entity.

Relationship Participation

- **Total** : Every entity instance must be connected through the relationship to another instance of the other participating entity types
- **Partial***:* All instances need not participate

E.g.: Employee **Head-of** Department
Employee: partial
Department: total

All employees will not be head-of some department. So only few instances of employee entity participate in the above relationship. But each department will be headed by some employee. So department entity's participation is total and employee entity's participation is partial in the above relationship

ER Modeling -Notations

Education and Research

ER Modeling -Notations

An Entity is an object or concept about which business user wants to store information.

A weak Entity is dependent on another Entity to exist. Example Order Item depends upon Order Number for its existence. Without Order Number it is impossible to identify Order Item uniquely.

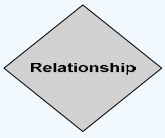
Attributes are the properties or characteristics of an Entity

A key attribute is the unique, distinguishing characteristic of the Entity

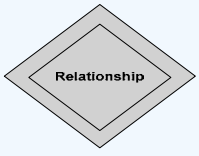
A multivalued attribute can have more than one value. For example, an employee Entity can have multiple skill values.

ER Modeling -Notations

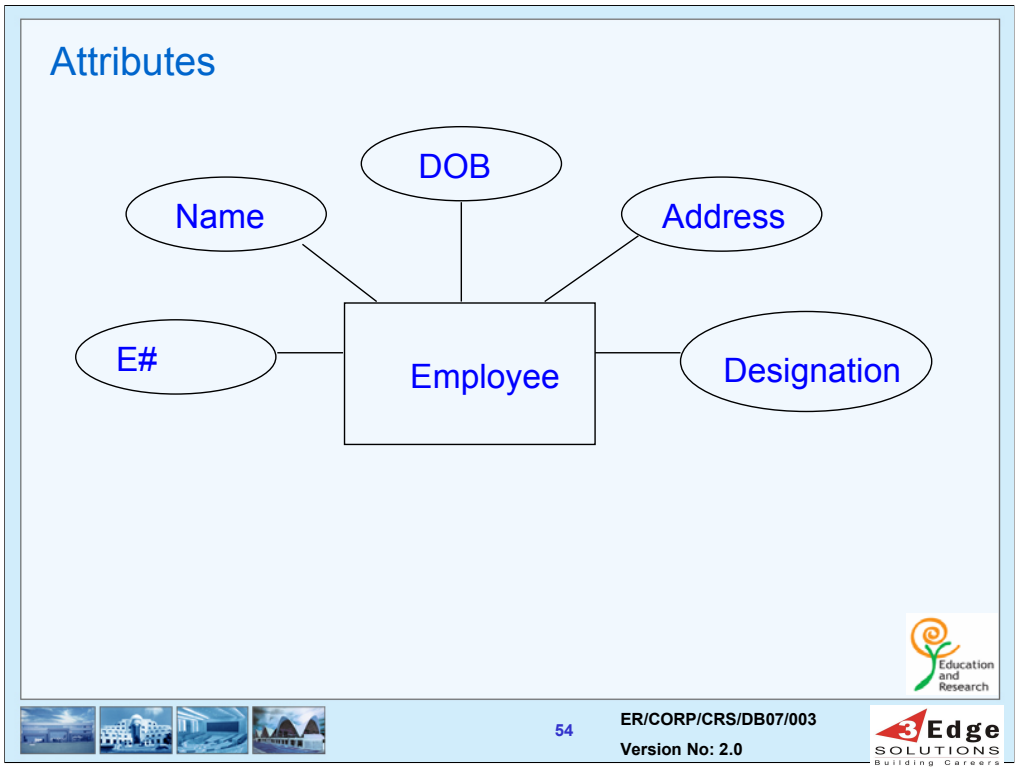
A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's basic salary and House rent allowance.



Relationships illustrate how two entities share information in the database structure.

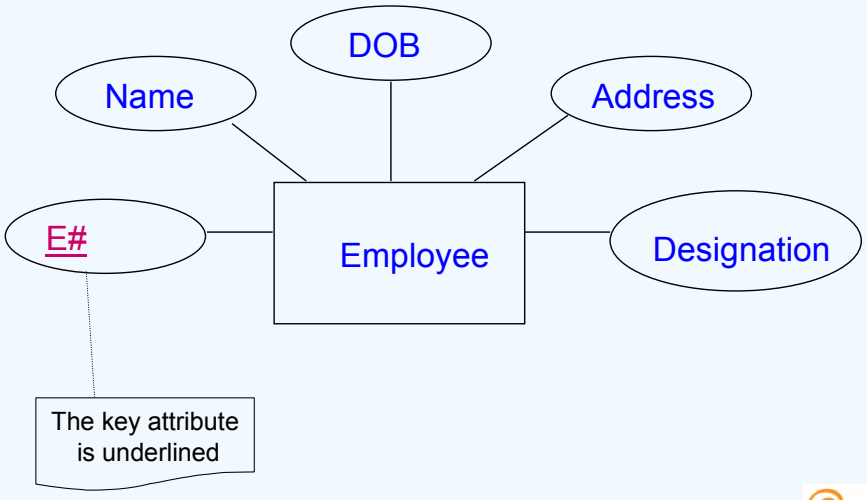


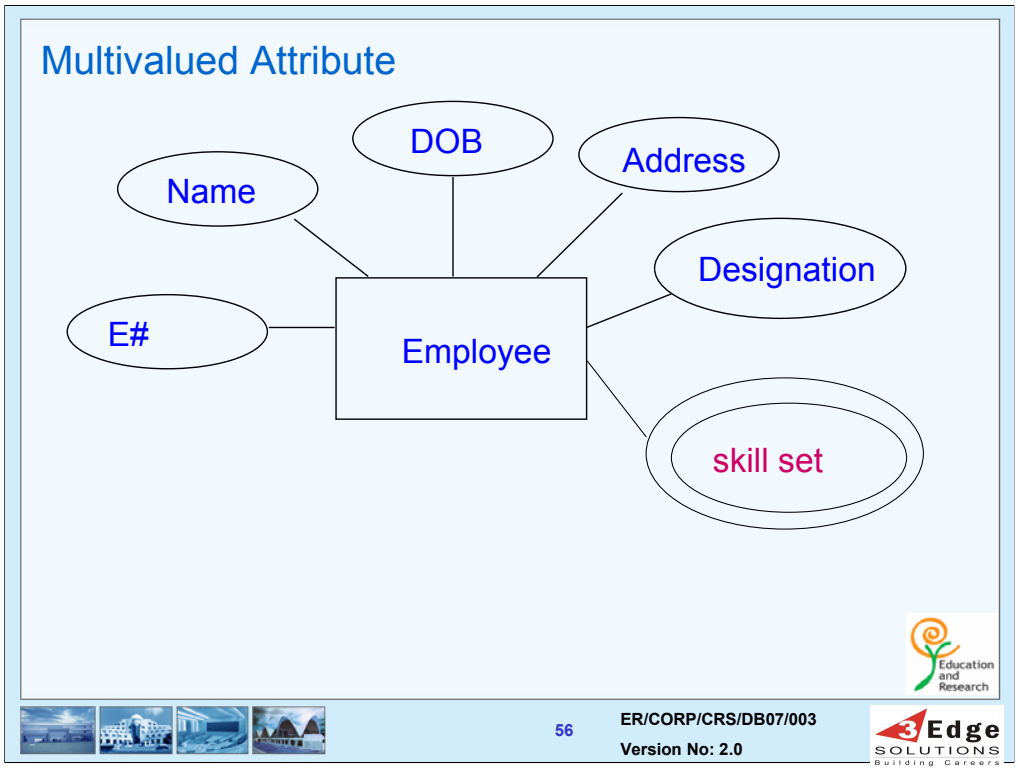
To connect a weak Entity with others, you should use a weak relationship notation.



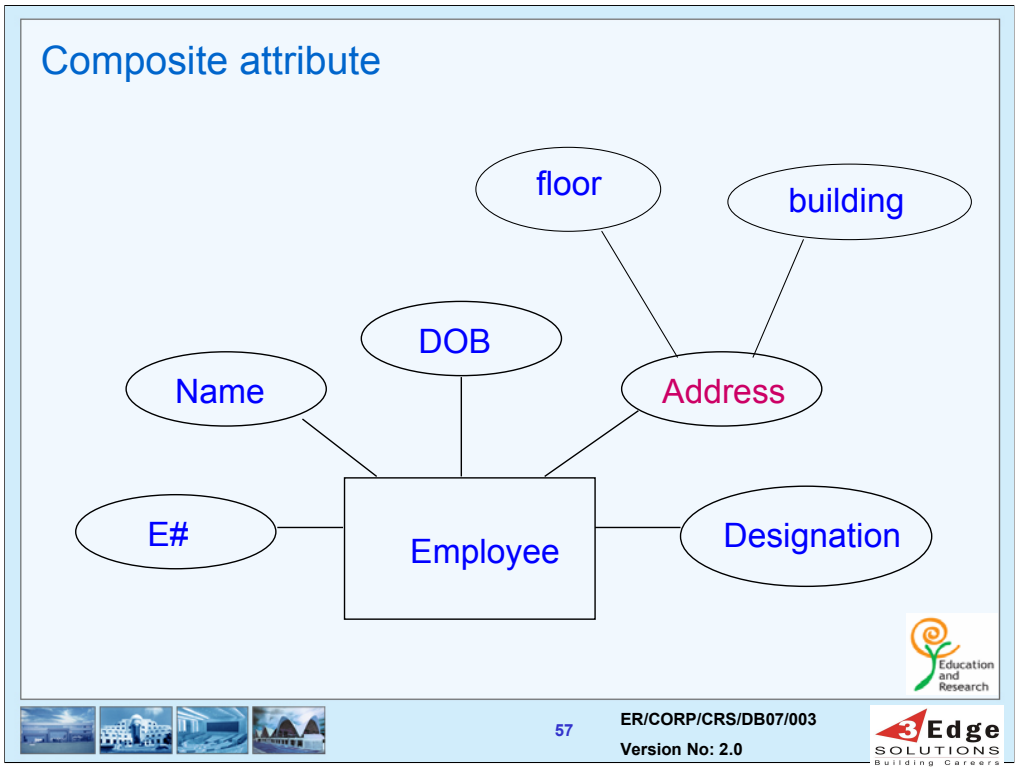
Represented by ellipses connected to the entity type by straight lines

Key attribute

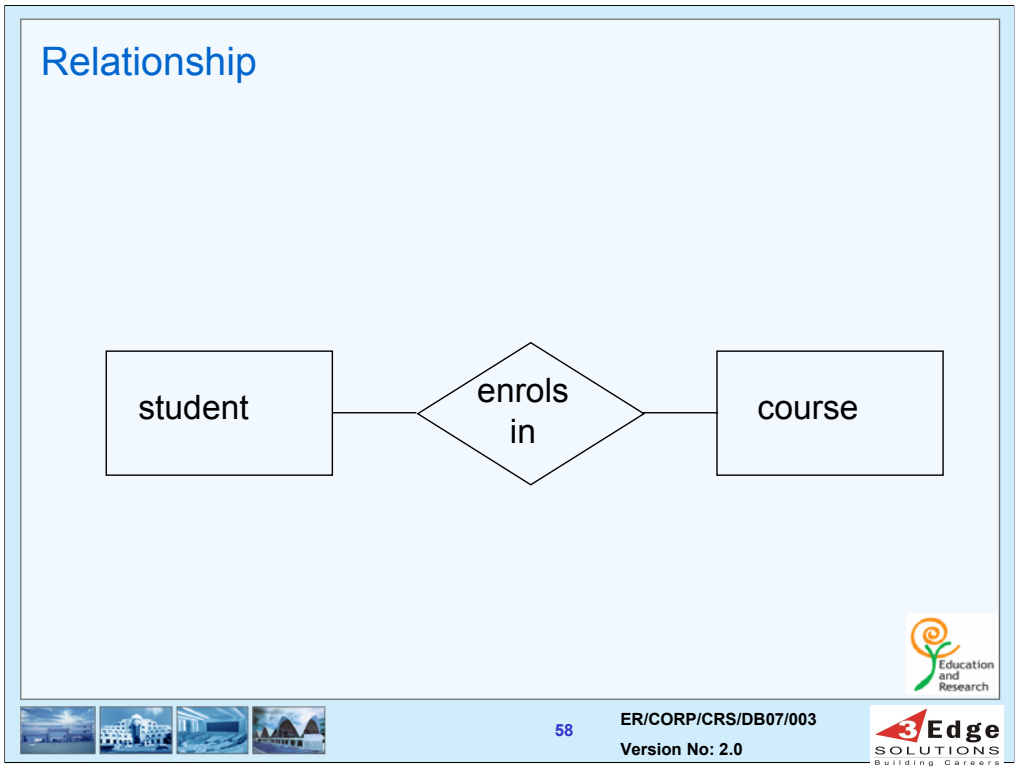




Indicated by a double lined ellipse as shown in the figure



Represented by an ellipse from which other ellipses emanate and represent the component attributes. E.g Address



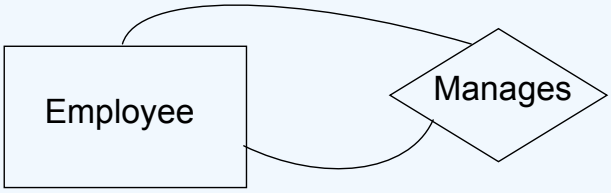
•A relationship is represented as a diamond between two entity types.

•It has a label that explains the relationship. Usually the convention is to read the ER diagram from top to bottom and from left to right.

•So, the relationship name is so chosen as to make sense when read from left to right.

•The relationship above is read as student enrolls-in course

Unary Relationship

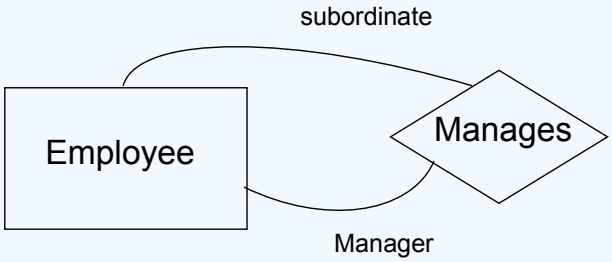


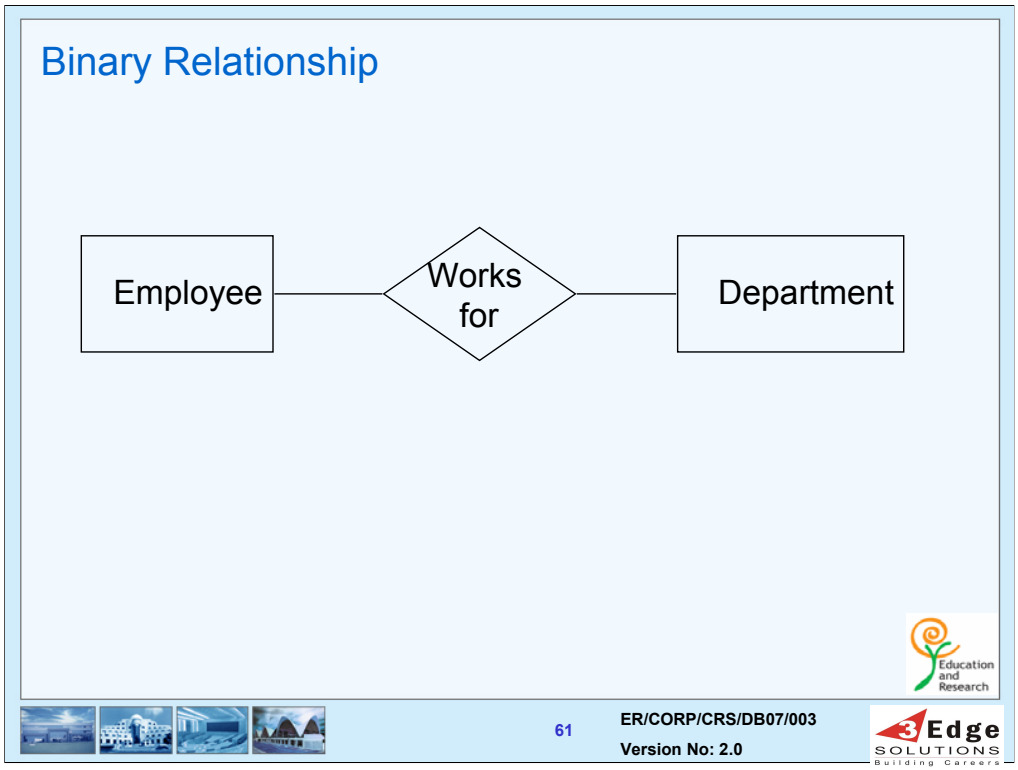
•A unary relationship is represented as a diamond which connects one entity to itself as a loop.

•The relationship above means, some instances of employee manage other instances of Employee.

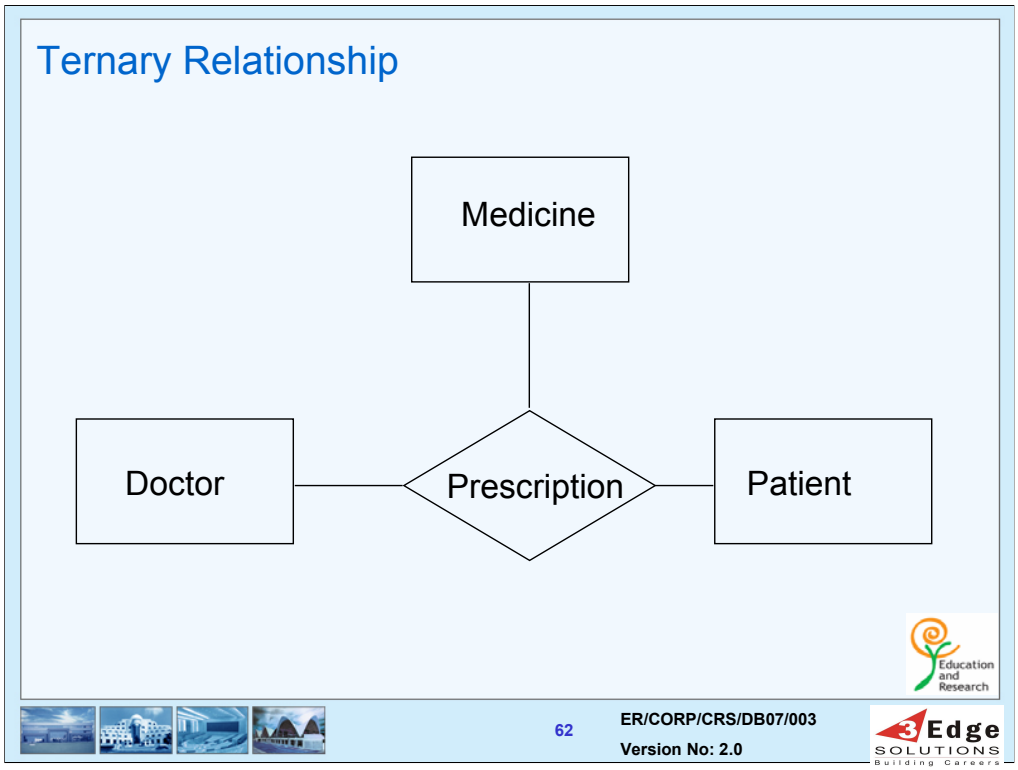
Role names

- Role names may be added to make the meaning more explicit

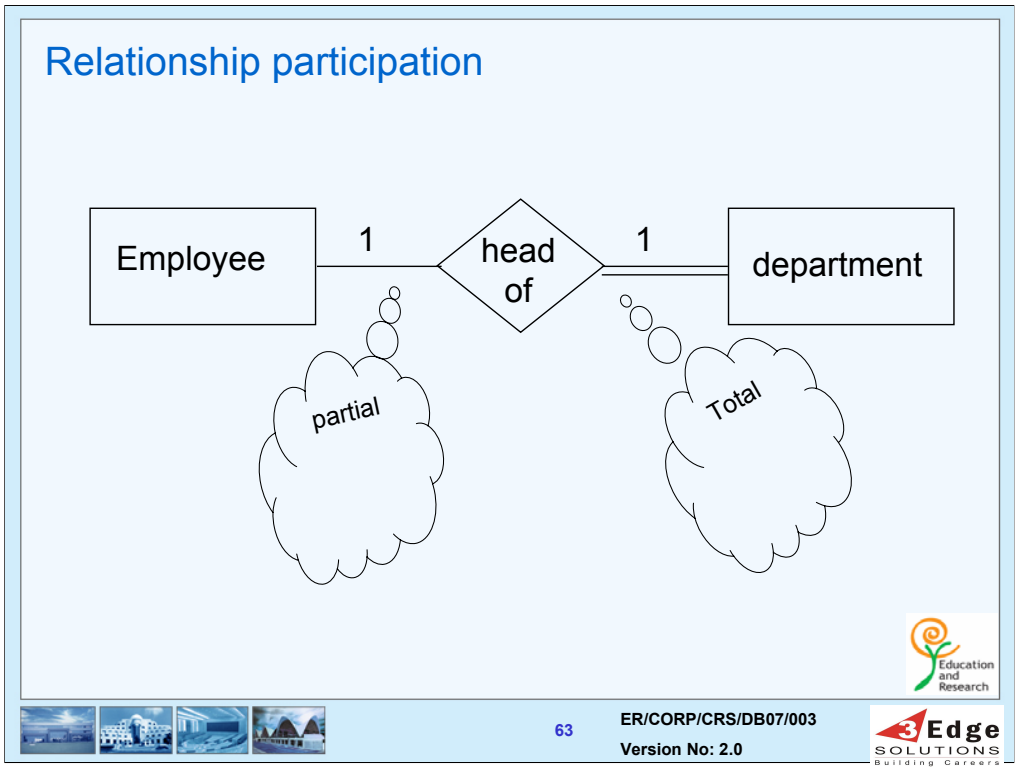




A relationship between two entity types



A relationship connecting three entity types.

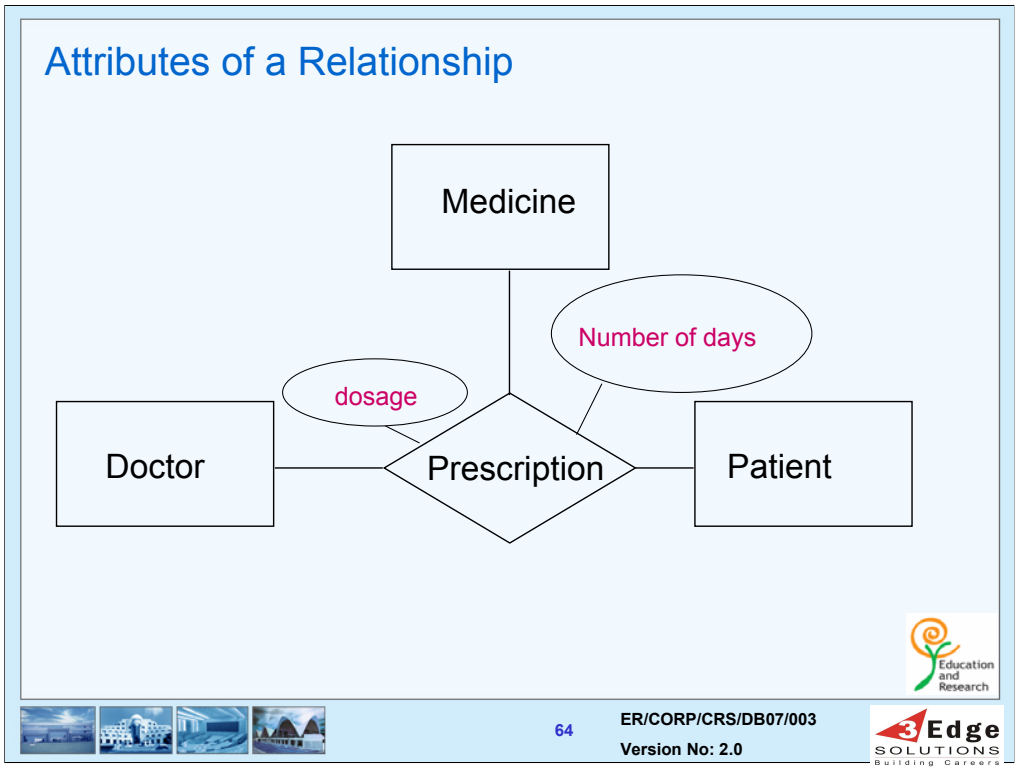


•All instances of the entity type Employee don't participate in the relationship, Head-of.

•Every employee doesn't head a department. So, employee entity type is said to partially participate in the relationship.

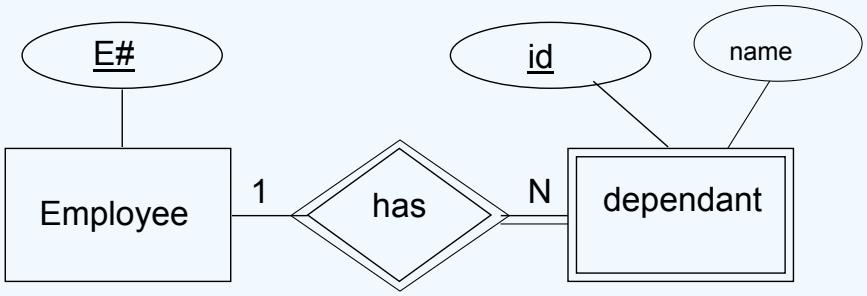
•But, every department would be headed by some employee.

•So, all instances of the entity type Department participate in this relationship. So, we say that it is total participation from the department side.



These attributes best describe the relationship rather than any individual entity

Weak entity



The dependant entity is represented by a double lined rectangle and the identifying relationship by a double lined diamond

The identifying relationship is the one which relates the weak entity with the strong entity on which it depends

Case Study – ER Model For a college DB

Assumptions :

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- A course can be taken by only one instructor
- A student can enroll for any number of courses
- Each course can have any number of students

Steps in ER Modeling

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key
- Review your results with your Business users

Step 1: Identify the Entities

- DEPARTMENT
- STUDENT
- COURSE
- INSTRUCTOR

Step 2: Find the relationships

- One course is enrolled by multiple students and one student enrolls for multiple courses, hence the cardinality between course and student is Many to Many.
- The department offers many courses and each course belongs to only one department, hence the cardinality between department and course is One to Many.
- One department has multiple instructors and one instructor belongs to one and only one department , hence the cardinality between department and instructor is one to Many.
- Each department there is a “Head of department” and one instructor is “Head of department “,hence the cardinality is one to one .
- One course is taught by only one instructor, but the instructor teaches many courses, hence the cardinality between course and instructor is many to one.

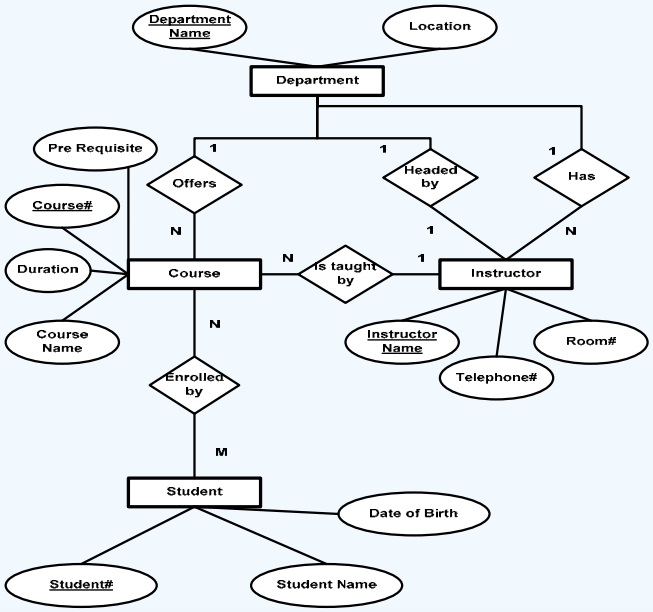
Step 3: Identify the key attributes

- Deptname is the key attribute for the Entity “Department”, as it identifies the Department uniquely.
- Course# (CourseId) is the key attribute for “Course” Entity.
- Student# (Student Number) is the key attribute for “Student” Entity.
- Instructor Name is the key attribute for “Instructor” Entity.

Step 4: Identify other relevant attributes

- For the department entity, the relevant attribute is location
- For course entity, course name,duration,prerequisite
- For instructor entity, room#, telephone#
- For student entity, student name, date of birth

Step 5: Draw complete E-R diagram with all attributes including Primary Key



Thank You!