ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management Systems**

**Assignment # 4**

**SUBMITTED BY:**

Apil Neupane

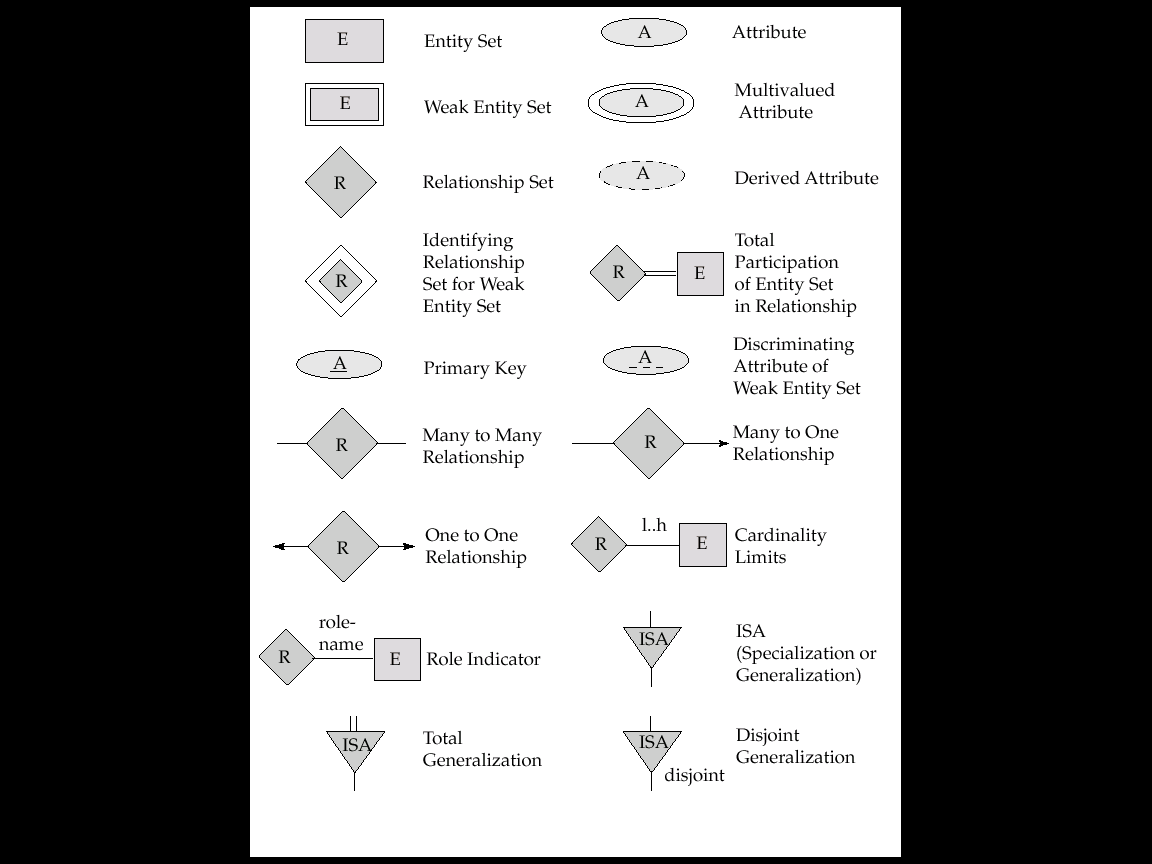
013BSCCSIT008

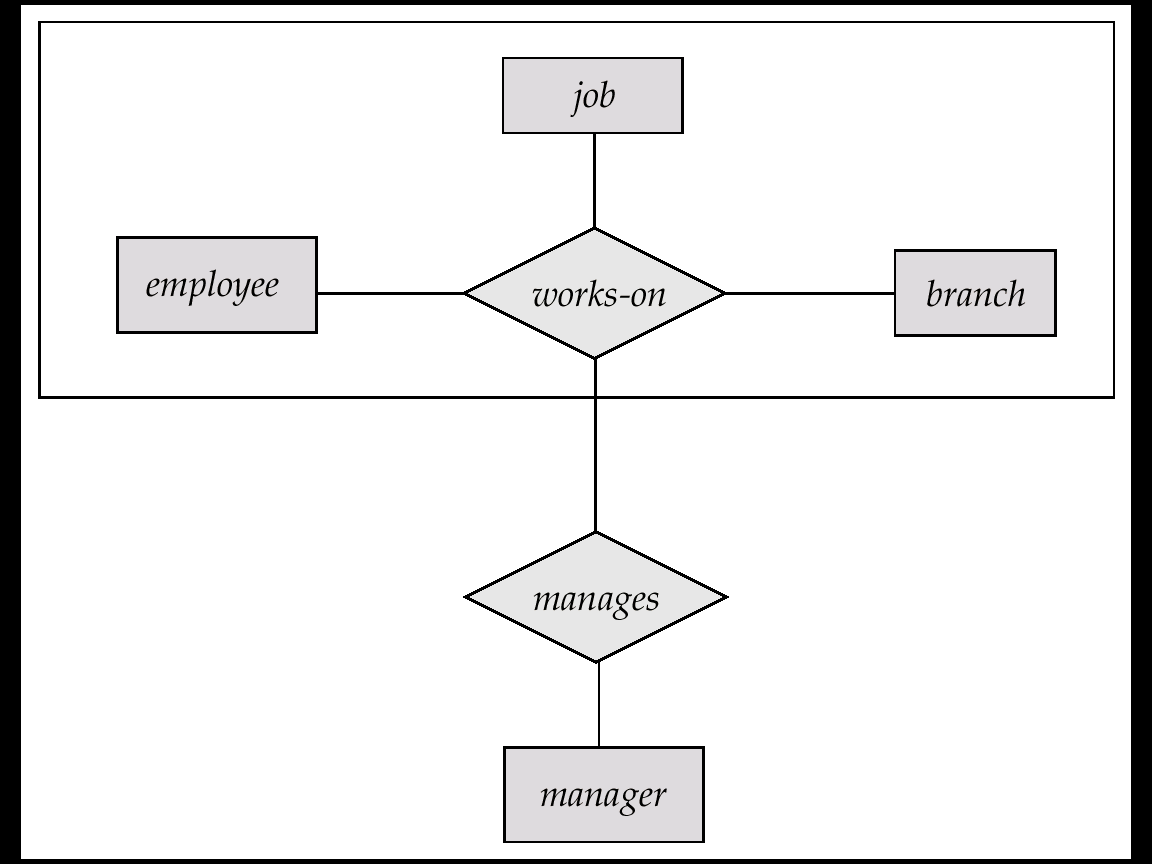
**SUBMITTED TO**

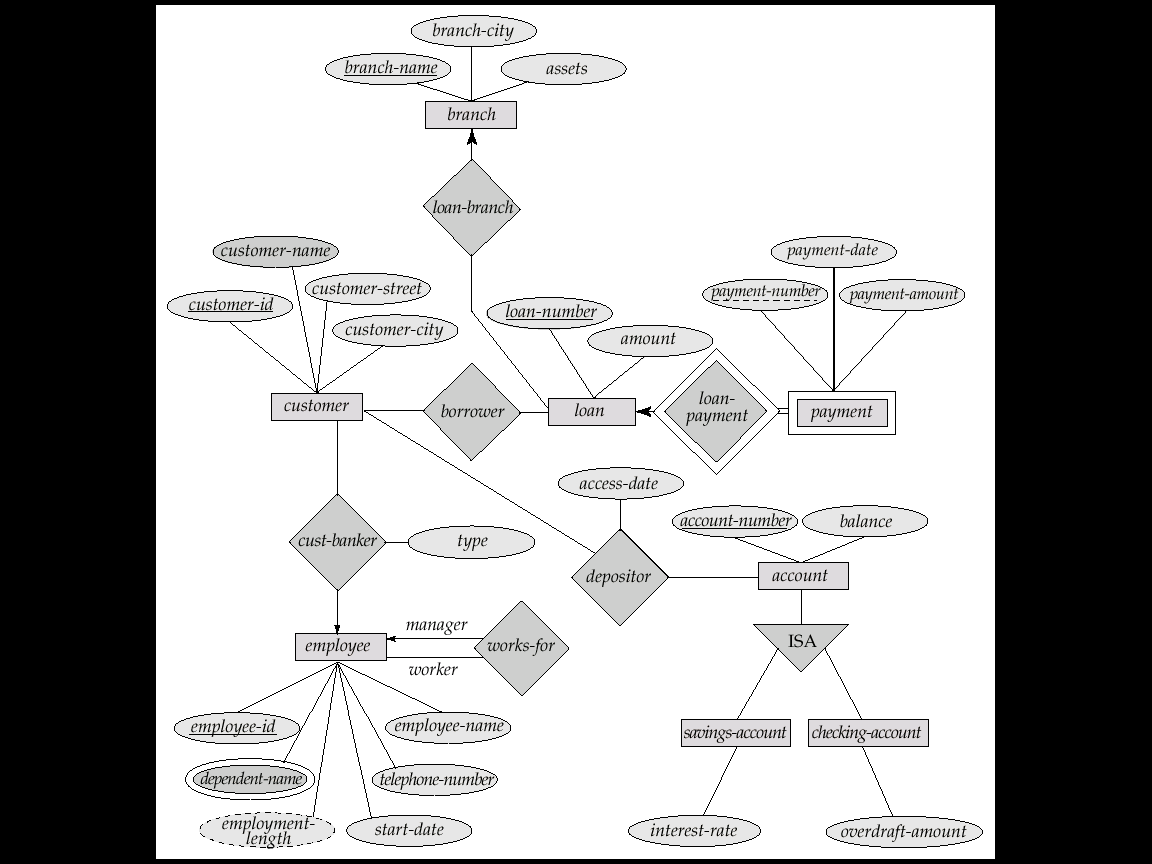
|  |  |
| --- | --- |
| **Mr. Sanjay Kr. Yadav**  **( Lecturer )** |  |
| **Department of Computer Science** | |

**ER DiagRAM:**

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of [data](http://www.webopedia.com/TERM/D/data.html) within [databases](http://www.webopedia.com/TERM/D/database.html) or information systems. An entity is a piece of data-an [object](http://www.webopedia.com/TERM/O/object.html)or concept about which data is stored.

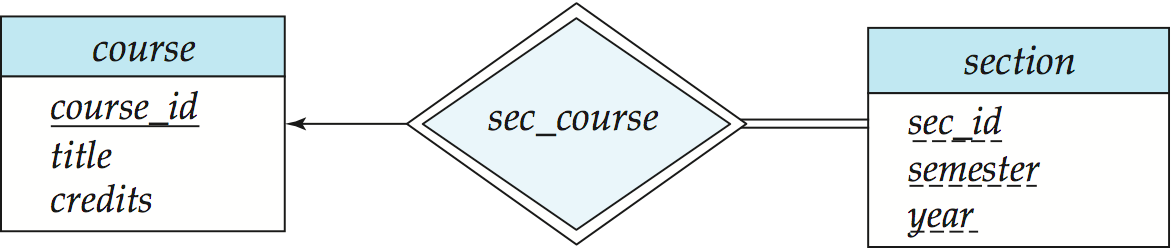






Weak Entity Set:

* An entity set that does not have a primary key is referred to as a weak entity set.
* The existence of a weak entity set depends on the existence of a identifying entity set
  + it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  + Identifying relationship depicted using a double diamond
* The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
* The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set’s discriminator.
* We underline the discriminator of a weak entity set with a dashed line.
* We put the identifying relationship of a weak entity in a double diamond.
* Discriminator of the weak entity set is underlined by dashed lines
* Primary key for section – (course\_id, sec\_id, semester, year)



**Functional Design:**

Functional Design is a paradigm used to simplify the design of hardware and software devices such as computer software and increasingly, 3D models. A functional design assures that each modular part of a device has only one responsibility and performs that responsibility with the minimum of side effects on other parts. Functionally designed modules tend to have low coupling.

The advantage for implementation is that if a software module has a single purpose, it will be simpler, and therefore easier and less expensive, to design and implement.

Systems with functionally designed parts are easier to modify because each part does only what it claims to do.

Since maintenance is more than 3/4 of a successful system's life,this feature is a crucial advantage. It also makes the system easier to understand and document, which simplifies training. The result is that the practical lifetime of a functional system is longer.

In a system of programs, a functional module will be easier to reuse because it is less likely to have side effects that appear in other parts of the system.

**Database Design:**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database.

**Conceptual Database Design**

The requirement analysis is modeled in this conceptual design. The ER Model is used at the conceptual design stage of the database design. The ER diagram is used to represent this conceptual design. ER diagram consists of Entities, Attributes and Relationships.

**Logical Database Design**

Once the relationships and dependencies are identified the data can be arranged into logical structures and is mapped into database management system tables. Normalization is performed to make the relations in appropriate normal forms.

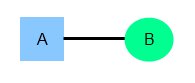
**Physical Database Design**  
It deals with the physical implementation of the database in a database management system. It include the specification of data elements, data types, indexing etc. All these information are stored in the data dictionary .

**Relationships Between Entities**

A relationship is how the data is shared between entities. There are three types of relationships between entities:

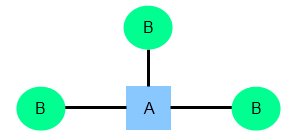
**1. One-to-One**

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



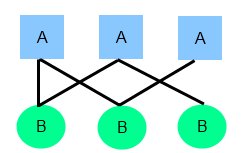
**2. One-to-Many**

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



## 3. Many-to-Many

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



**Multi Valued Attribute:**

* A multivalued attribute M of an entity E is represented by a separate schema EM
  + Schema EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
  + Example: Multivalued attribute phone\_number of instructor is represented by a schema:  
     inst\_phone= ( ID, phone\_number)
  + Each value of the multivalued attribute maps to a separate tuple of the relation on schema EM
    - For example, an instructor entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:   
       (22222, 456-7890) and (22222, 123-4567)
* Attributes (like phone numbers) that are explicitly repeated in a class definition aren’t the only design problem that we might have to correct. Suppose that we want to know what hobbies each person on our contact list is interested in (perhaps to help us pick birthday or holiday presents). We might add an attribute to hold these. More likely, someone else has already built the database, and added this attribute without thinking about it.

