# Week 5: Oct 12th

- Covered in 3530 so far
  - Relational Model
  - Query languages
    - Relational Algebra
    - SQL DML
    - Will cover SQL DDL in Assignment2 and SQL DCL towards the end
- Next Step: Design
  - Part I: Entity Relationship model
  - Part II: Normalization

# Chapter

## Entity Relationship Model

# Chapter 1 : Objectives

- Main Phases of Database Design
- Characteristics of ER Model
- Components of ER Model
- Definitions
- Classification of Attributes
- Entity Types
- Entity Set
- Key attribute

# Chapter 1 : Objectives (Continued.)

- Relationship Types
- Recursive Relationships
- Structural Constraints on Relationship Types
- Attributes on Relationship Types
- Weak Entity Type
- Problems with ER Models

# Database design:

- Why do we need it?
- must agree on structure of the database before implementing it

- What entities to model
- How entities are related
- What constraints exist in the domain
- How can we get a good design

# Main Phases of Database Design

1. Requirements Gathering
2. Conceptual Model (high-level)
   - ER Model
3. Logical Model
   - Relational Model
4. Physical Model (not a part of 3530)

# Introduction

- E-R model facilitates database design by allowing the specification of an “enterprise schema” which represents the overall logical structure of a database.

- The E-R model is extremely useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

# Characteristics of ER Model

- Most common tool for conceptual database design
- No link to any physical DBMS
- Chen's model (model used in 3530)- Proposed by Dr. Peter Chen
  - The E/R model is one of the most cited articles in Computer Science - "The Entity-Relationship model – toward a unified view of data" Peter Chen, 1976
  - Alternate model is crow's foot model

# Components of ER Model

The "world" is expressed in terms of

- Entity, Entity Type, Entity Set
- Attribute
- Key (Identifier)
- Relationship, Relationship Type, Relationship Set
  - Structural Constraints on relationships

- ER Model is visualized as an ER diagram.

# Entity

- In an ER model, we will model the database as :
  - a collection of entities
  - relationships among entities.
- Entity:
  - An object in the real world with an independent existence
  - Is distinguishable from other objects
  - could have a physical existence or a conceptual existence

# Example: Find entities

- Find the entities in the requirement given:

- A student in UofG can take several courses but is taught by just one instructor at any given point in time.

Entities (something that can exist independently):

- student e.g. John Smith, 1112222, johns@uoguelph.ca
- course e.g. CIS3500, Intro to DB, Elmasri Navathe
- Instructor e.g. Ritu, J. D. M 213A, chaturvr@uoguelph.ca

# Example: Find entities?

- Requirements are not necessarily given in a nicely-written statement (as in slide 10). Find the entities from UofG's grad course waiver form.

- Grad course waiver form
  - URL (https://www.uoguelph.ca/registrar/sites/undergraduate/files/forms/graduate_course_waiver_request.html)

# Entity Types

Note that an entity is part of an actual instance. When designing a database, we really don't care about the instance (or actual values) - though we may use an instance for better understanding.

A collection of entities that have the same attributes is known as an entity type and that is of interest in the design process.

ER notation : $\boxed{\text{STUDENT}}$

# Entity Set

Entity Set : The collection of all entities of a particular entity type in the database at any point in time

- Similar to a relation (with 1 or more tuples and 1 or more attributes) in the relational model

# Entity Set

Entity Set : The collection of all entities of a particular entity type in the database at any point in time

STUDENT

John Smith, 1112222, johns@uoguelph.ca

Ritu Chaturvedi, 9991188, chaturvr@uoguelph.ca

…….

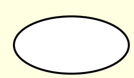
*It is common to use Entity, Entity Set and Entity type interchangeably – but always remember that entity type is what is used in the design.*

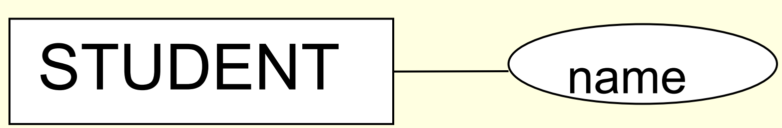
# Attribute

Attributes: are properties possessed by all members of an entity set.

Examples:

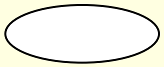
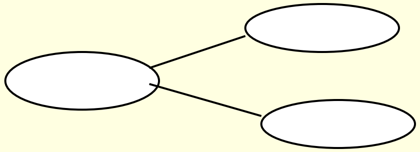
| Entity type. | Attributes |
|---|---|
| Student | name, id, email, courses taken |
| Course | title, number, textbook, credits |
| Instructor | name, office, email, courses taught |

ER notation : 



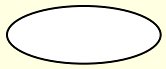
# Classification of Attributes

1.Simple ~ Composite

- Simple Attribute : ER notation 
  - Attribute that is a single component with an independent existence.
- Composite Attribute: ER notation 
  - Attribute composed of multiple components, each with an independent existence.
    - For example, *address* of instructor may consist of 4 attributes: house number, street, city, pincode

# Classification of Attributes

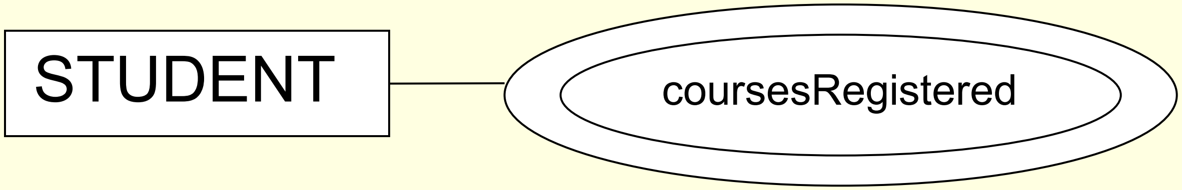
2. Single-Valued ~ Multi-Valued

Single-valued Attribute 

- Attribute that holds a single value for each occurrence of an entity type.

Multi-valued Attribute (MVA),

- Attribute that holds multiple values for each occurrence of an entity type.
- Example of MVA:
  - *courses* of entity type student



# Classification of Attributes

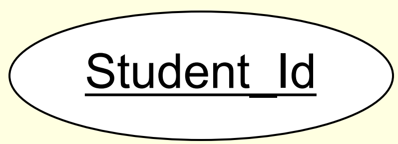
3. Stored ~ Derived

Derived Attribute

- Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type.

- No special notation in ER model
- Example: *age* is a derivable attribute, if date_of_birth is already stored as an attribute

# Key attribute

- Key attribute : An attribute of an entity type whose values are distinct for each individual entity in the collection.
- ER : 

# Example : University Staff Database

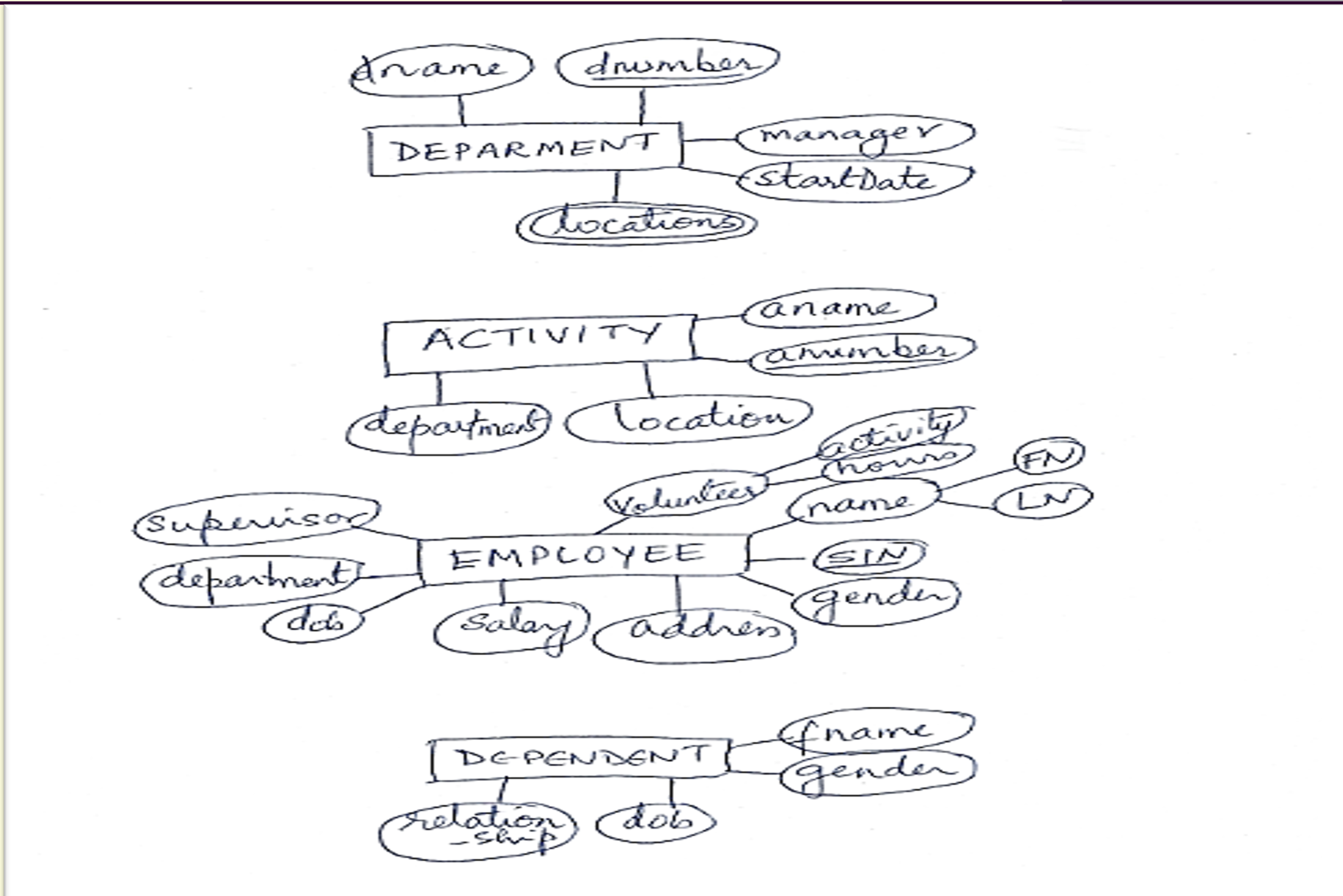
1. Each department of the University has a unique name, a unique number, and a manager (who is also an employee). For each manager, the start date when he / she was hired as a manager is stored. A department may have several locations.
2. A department runs a number of activities to support its staff and its community, each of which has a name, a unique activity number, and a single location where it is hosted.
3. Each employee's name, social insurance number, address, salary, gender and birth date is stored. An employee works for a department but may volunteer for several activities. We keep track of the number of hours per week that an employee volunteers on each activity. Each employee has a supervisor.
4. An employee may have 0 or more dependents. We keep each dependent's first name, gender, birth date, and relationship to the employee (for insurance purposes).

# Preliminary design of entity types for the University Database

Note that {} is used to indicate a multivalued attribute

- DEPARTMENT
  - **dname, dnumber, {locations}, manager, startDate**
- ACTIVITY
  - **aname, anumber, location, department**
- EMPLOYEE
  - **name(FN, LN), SIN, gender, address, salary, dob, department, supervisor, {volunteer(activity, hours)}**
- DEPENDENT
  - **fname, gender, dob, relationship**

# Preliminary design of entity types for the University Staff Database



# Relationship

- A relationship is an association among entities.
- Example:
-

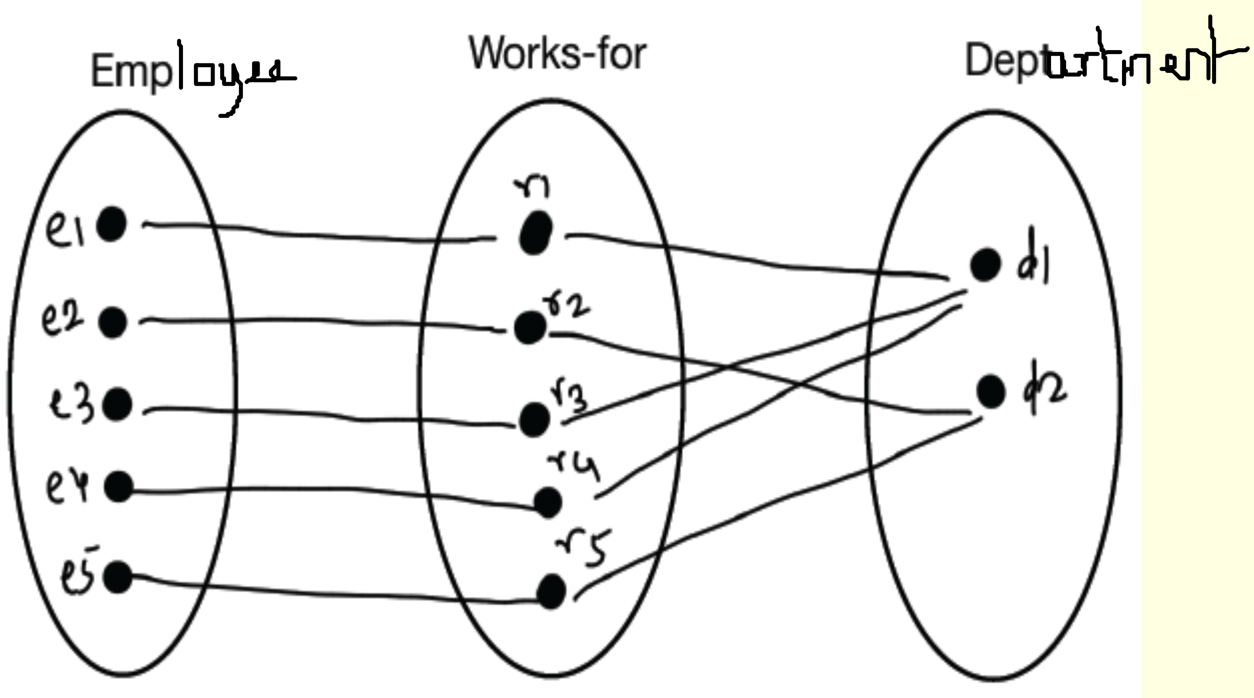
| | | |
|---|---|---|
| Ritu | Volunteers | UnitedWayCampaign |
| Employee | Relationship | Activity |

# Example : University Staff Database

1. Each **department** of the University has a unique name, a unique number, and a manager (who is also an employee). For each manager, the start date when he / she was hired as a manager is stored. A department may have several locations.
2. A department runs a number of activities to support its staff and its community, each of which has a name, a unique activity number, and a single location where it is hosted.
3. Each employee's name, social insurance number, address, salary, gender and birth date is stored. **An employee works for a department** but may volunteer for several activities. We keep track of the number of hours per week that an employee volunteers on each activity. Each employee has a supervisor.
4. An employee may have 0 or more dependents. We keep each dependent's first name, gender, birth date, and relationship to the employee (for insurance purposes).

# Example – relationship instances

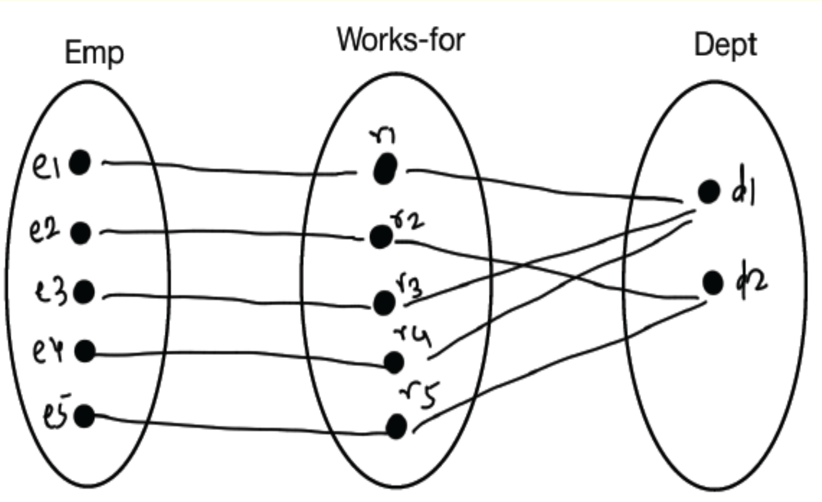
- An employee works for a department




- – note that these instances are only for understanding – how it is represented in the ER model is defined and shown in the next few slides

# Relationship Types

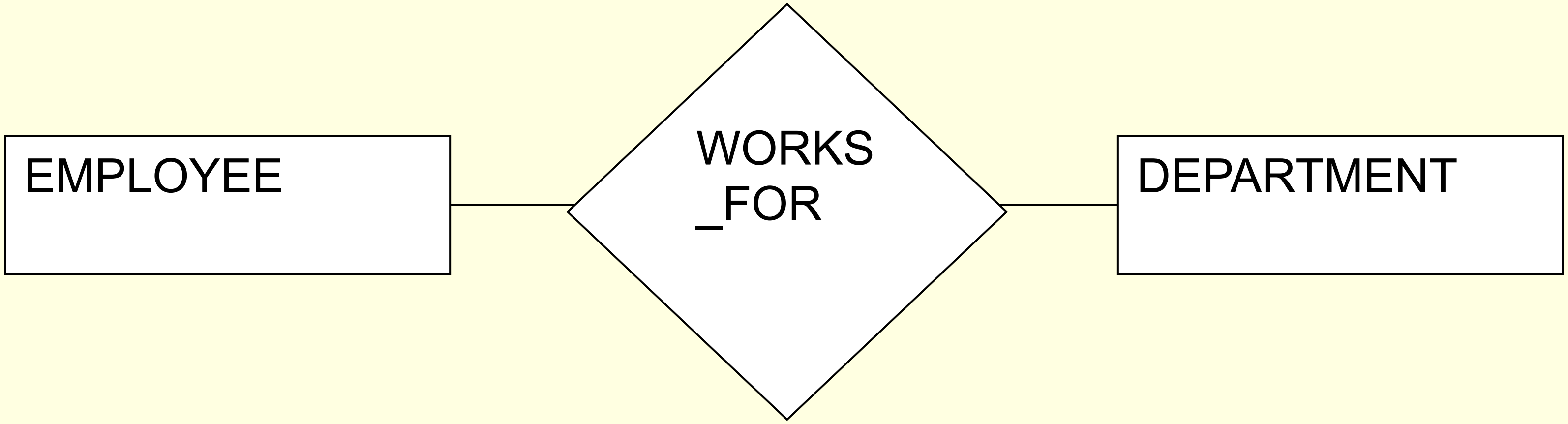
A relationship type R among n entity types $E_1$ .. $E_n$ is a set of relationship instances $r_i$ where each $r_i$ associates n individual entities ($e_1$, $e_2$, …. $e_n$) and each entity $e_j$ in $r_i$ is a member of entity type $E_j$, $1 <= j <= n$. $E_j$ is called the <u>participating entity type</u>.



# Relationship type in ER diagram WORKS_FOR

Relationship type: WORKS_FOR

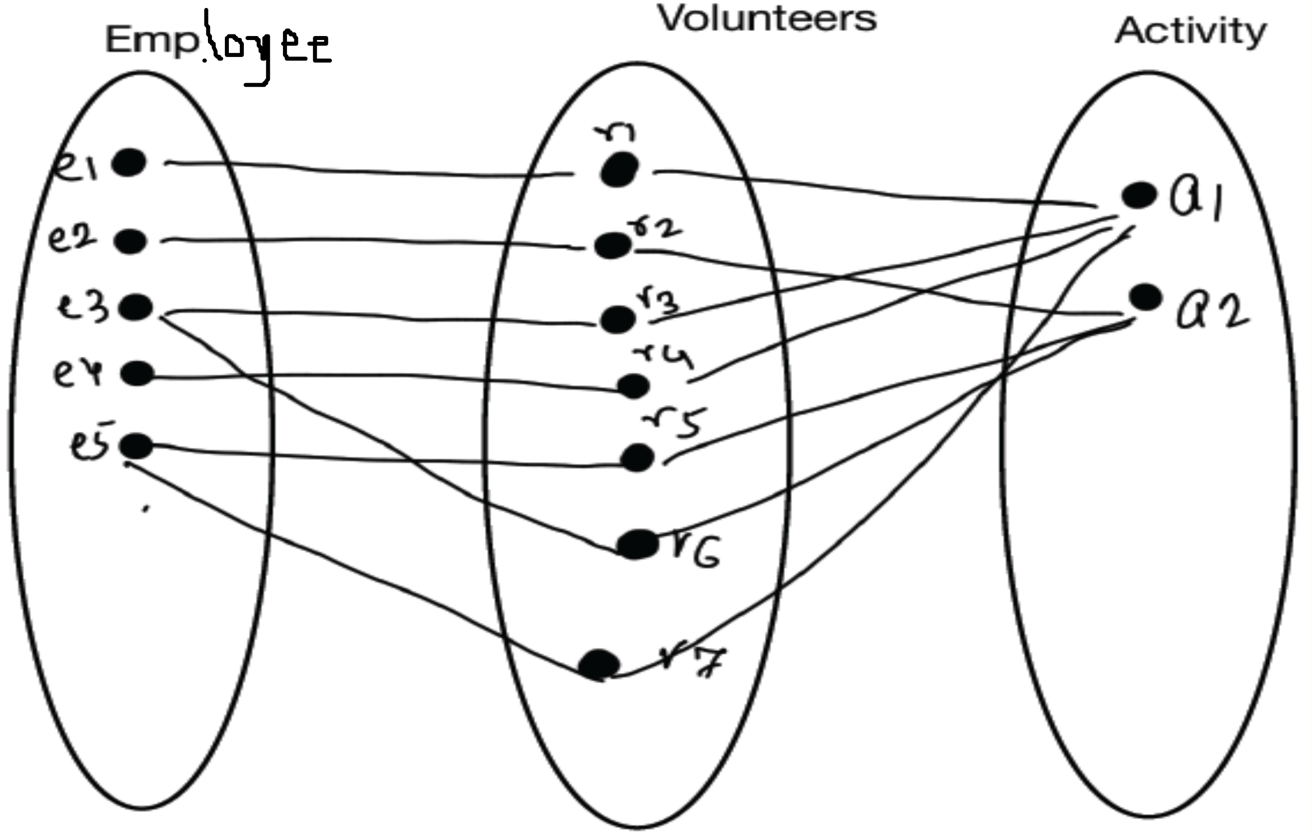
- ER :



# Example : University Staff Database

1. Each department of the University has a unique name, a unique number, and a manager (who is also an employee). For each manager, the start date when he / she was hired as a manager is stored. A department may have several locations.
2. A department runs a **number of activities** to support its staff and its community, each of which has a name, a unique activity number, and a single location where it is hosted.
3. Each employee’s name, social insurance number, address, salary, gender and birth date is stored. **An employee** works for a department but may **volunteer for several activities**. We keep track of the number of hours per week that an employee volunteers on each activity. Each employee has a supervisor.
4. An employee may have 0 or more dependents. We keep each dependent’s first name, gender, birth date, and relationship to the employee (for insurance purposes).

# Relationship type Volunteers




- ER diagram?

# Friday Oct 15th

# W6 Monday

- A2 coming up – will be posted by Friday (due date is Friday Nov 9th)
  - Email me by Thursday Oct 18th 11:55pm, if you plan to work in pairs
- Midterm October 27th
  - 10:00am G TH 1307
- Lab 4 (Week of Oct 29th)

# Midterm (Oct $27^{th}$)

- Relational model
- Relational Algebra
- SQL DML
  - SELECT, INSERT, DELETE, UPDATE

- ER Modeling
- Normalization

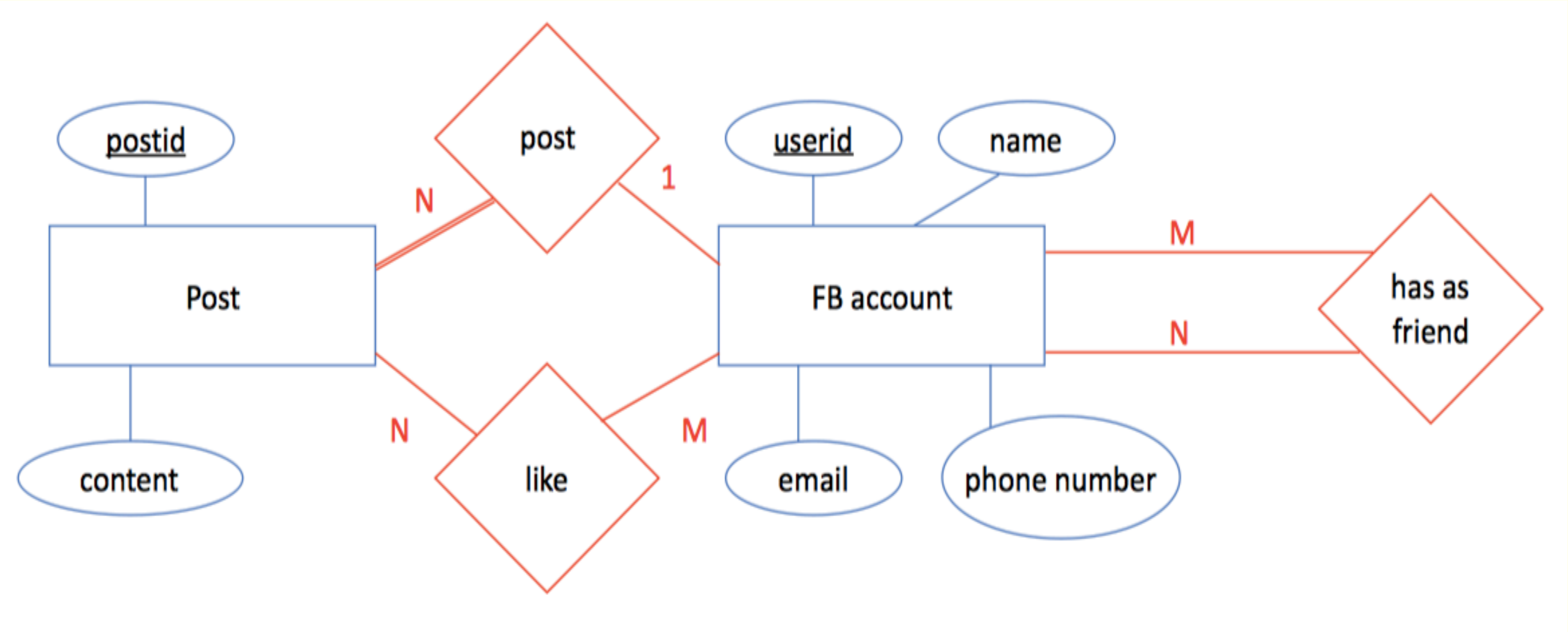
- MCQ + Short answer questions

# ER model

- Entities
  - Strong
  - Weak
- Attributes
  - Simple, composite, single_valued, multi_valued, key, relationship attributes
- Relationships
  - Binary
  - Recursive
  - Structural constraints
    - Degree (1:1, 1:N, M:N)
    - Participation (total, partial)

# Worksheets 5 and 6

A facebook user must have an account that stores the userid, name, email and phone number of the user. A user can have friends, who are also facebook users. Users can post messages and can like like posts of other facebook users. Each post has an id and its content.



# Worksheets 5 and 6

# Main Phases of Database Design

1. Requirements Gathering
2. Conceptual Model
   - ER Model
3. Logical Model
   - Relational Model
4. Physical Model

# Map ER Model to Relational model

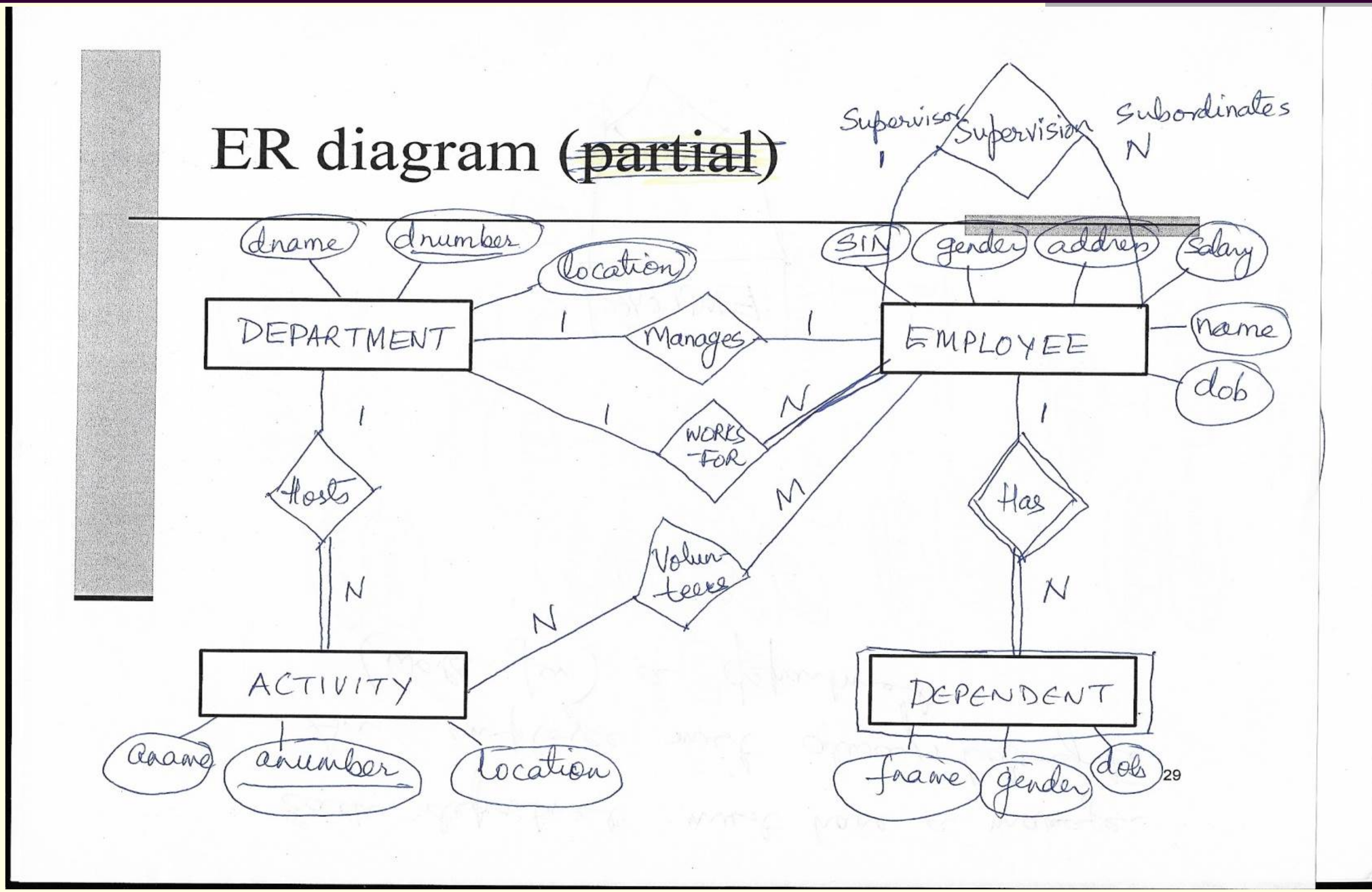
Input: ER Model

Output: Relational Model

General Idea:

- Each entity type (ET) becomes a relation.
- Only the simple components of any composite attribute are taken.
- Each 1:1 and 1:N relationship adds an attribute (as foreign key) to an existing ET.
- Each M:N relationship becomes a new relation
- Each multi-valued attribute becomes a new relation

# University Staff DB ER model - complete



# ER-to-Relational Mapping :Step 1

For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

# ER-to-Relational Mapping :Step 2

- For each weak entity type W in the ER schema with owner entity type E, create a relation R, and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

# ER-to-Relational Mapping :Step 3

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Choose one of the relations—S, say—and include as foreign key in S the primary key of T. It is better to choose an entity type with *total participation* in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

# ER-to-Relational Mapping :Step 4

- For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the *N-side* of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; this is because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.

# ER-to-Relational Mapping :Step 5

- For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations—as we did for 1:1 or 1:N relationship types—because of the M:N cardinality ratio.

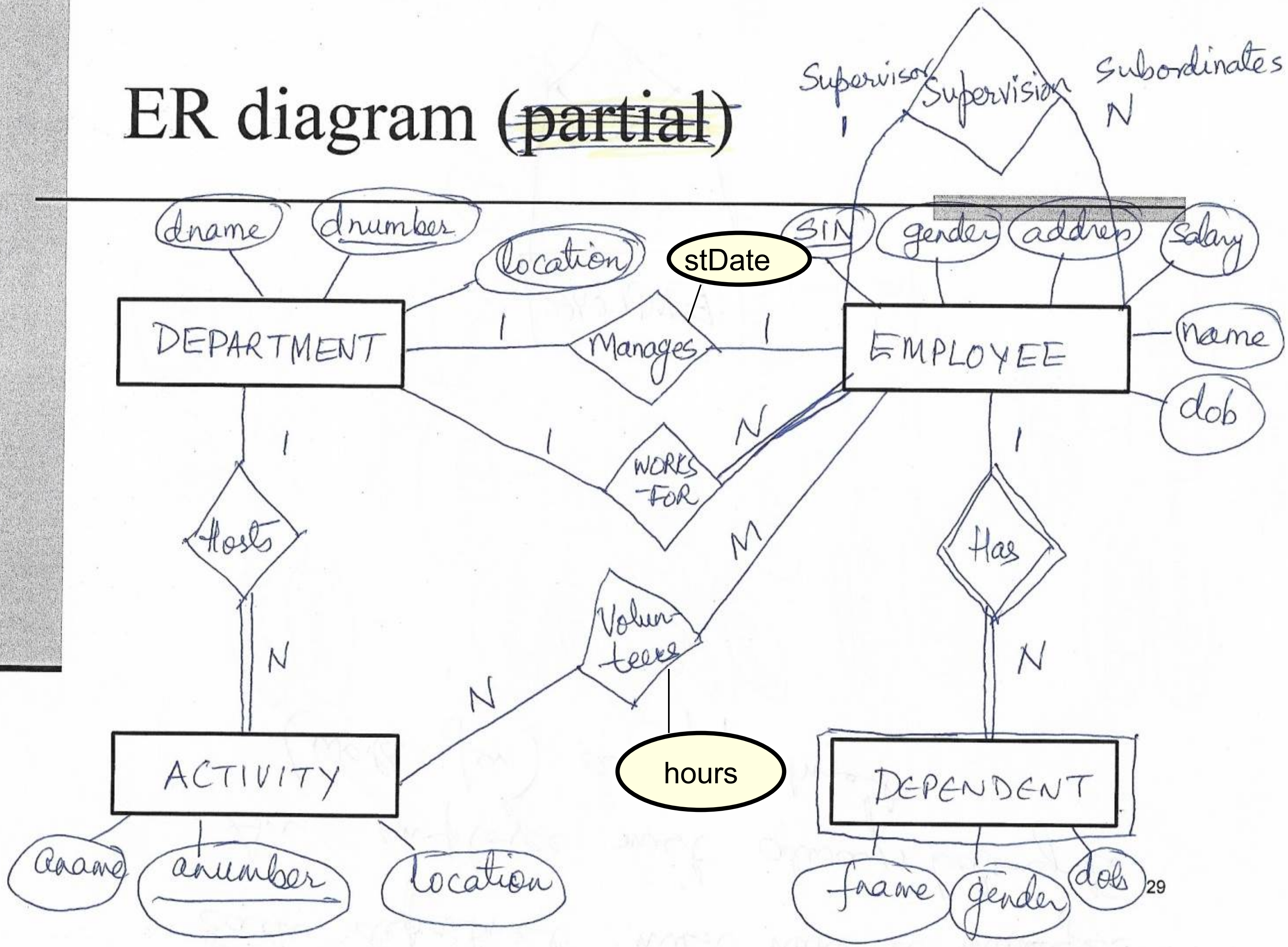
# ER-to-Relational Mapping :Step 6

- For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components

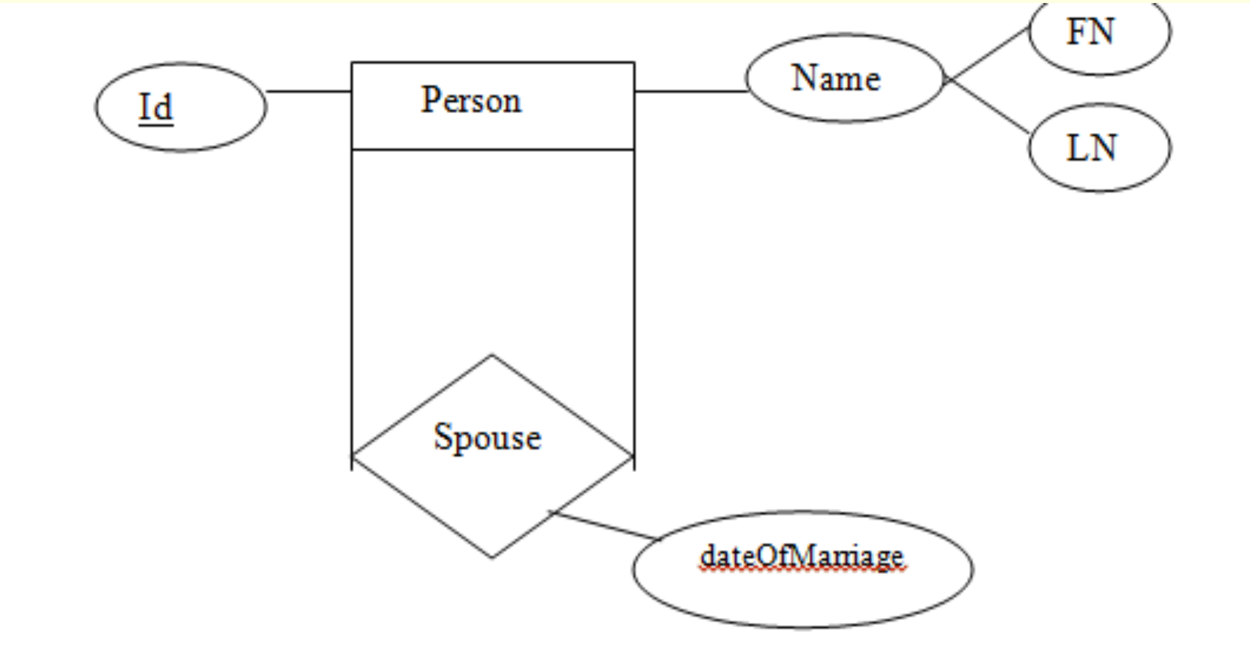
# ER-to-Relational Mapping :Step 7

- (I discussed only binary relationships in class). For each n-ary relationship type R, where n > 2, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types. However, if the cardinality constraints on any of the entity types E participating in R is 1, then the primary key of S should not include the foreign key attribute that references the relation E’ corresponding to E .This concludes the mapping procedure.

# ER diagram (~~partial~~)



# Complete and Convert




This ER diagram represents the relationship between a wife and a husband.

Constraint : A wife can have several husbands, though a husband is allowed to have only one wife . A husband must have a wife !

# Reverse process: from Relational to ER (assume implicit meaning of course and section)

Convert to ER model:

- COURSE(**<u>CRS_CODE</u>**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**<u>CRS_CODE, SECTION#</u>**, YEAR, SEM, CLASS_TIME, ROOM_CODE, PROF_NUM)

# Reverse process: from Relational to ER

Convert to ER Model:

- COURSE(**<u>CRS_CODE</u>**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**<u>CLASS_CODE</u>,** CRS_CODE, SECTION#, CLASS_TIME, ROOM_CODE, PROF_NUM )

# Example (Contd).

- COURSE(**<u>CRS_CODE</u>**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**<u>CLASS_CODE</u>**, CRS_CODE, SECTION#, CLASS_TIME, ROOM_CODE, PROF_NUM )

- CLASS_CODE is a surrogate key
- Let's assume that the CRS_CODE FK in the SECTION is declared to be "not null."

- The SECTION entity is existence-dependent on COURSE. (That's because it is reasonable to assume that COURSE is mandatory to SECTION, since a section does not appear in the class schedule unless there is a course description for it in the course catalog. Note that the "not null" FK requirement means that the CRS_CODE FK in the CLASS entity is mandatory.)

- SECTION is not a weak entity (That's because although the SECTION is existence-dependent on COURSE -- the SECTION entity's PK does not contain the PK of the COURSE entity.)