

# Entity-Relationship Diagram



- Press `Space` to navigate through the slides
- Use `Shift+Space` to go back

# Quick Recap

- Normalisation
  - 1NF
  - 2NF (be 1NF and remove **partial dependency**)
  - 3NF (be 2NF and remove **transitive dependency**)
- Data modelling
  - Conceptual, Logical, Physical
- CA1 released
  - The descriptor is [here \(https://moodle.cct.ie/mod/resource/view.php?id=49275\)](https://moodle.cct.ie/mod/resource/view.php?id=49275)
  - Submission in next week on **Friday 18/10/2019 @5pm**

# Quick Recap

- **Relations & Keys**
  - Terminology: **Relation, Tuple, Attribute, Cardinality, Degree, Primary key, Domain**
  - Keys: **Candidate, Primary, Foreign, Alternate**
  - Keys constructs: **Single, Compound, Composite**
- **Schemas**
  - Internal, Conceptual, External
- **Relational Integrity Constraints**
  - Domain constraints, Key constraints, Referential integrity

# Notations

- A modeling notation is a set of graphical elements (and instructions on how to use them) that help us represent the structure or functioning of something.
- ERD is created using some notation to represent graphically the list of requirements and business rules of whatever we are building.
- Currently, there is no standard notation for an ER model.
- The original notation developed by **Peter Chen** in 1976 is still very popular.
- Numerous other alternatives exist, and many extensions were added to Chen's original one, which arose out of the need to capture more semantic elements (mainly constraints).
- Popular notations:
  - **Chen**
  - Crow's Foot
  - Barker
  - UML
  - Bachman

# Notations (Examples)

Chen



Crow's Foot



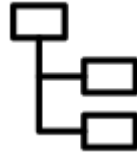
Barker's



Bachman



# ERD Components



An Entity Relationship Diagram (ERD) is a type of diagram that lets you see how different entities (e.g. people, customers, or other objects) relate to each other in an application or a database.

- **Entities** are the main item from which information is collected.
  - It can be an object, person, idea or concept of the real world and must have attributes.
- **Attributes** are entities' properties that provide descriptive information about them.
  - There are two classes of attributes: identifiers and descriptors.  
Additionally, and depending on the notation, there are different attribute constraints.
- **Relationships** are the link that joins two or more entities.
  - They represent associations in the real world and do not have a physical existence.
  - Their role is essential since they serve several purposes:
  - A Relationship has **Cardinality**

# Entity

- A real-world thing either living or non-living that is easily recognisable.
- It is anything in the enterprise that is to be represented in our database.
- It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database.
- The characteristics of entities are must have an attribute, and a unique key.
- Every entity is made up of some 'attributes' which represent that entity.

**Examples of entities:**

- **Person**
  - Employee, Student, Patient
- **Place**
  - Store, Building
- **Object**
  - Machine, Product and Car
- **Event**
  - Sale, Registration, Renewal
- **Concept**
  - Account, Course



# Entity

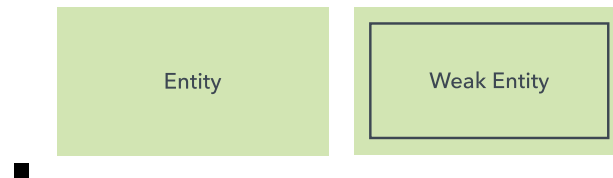
- They are usually represented as rectangles on an ERD with the entity name inside the rectangle
- We will use Chen Notation on our course



Entity


# Entity


- An entity can also be a **strong** entity or a **weak** entity. What's the difference?
  - A **strong entity** has an identifier (a primary key) and does not depend on any other entities for it to exist.
    - For example, a student may be a **strong entity**, as it can have a primary key and does not depend on any other entities for it to exist.
  - A **weak entity** is one that depends on a strong entity for existence, i.e. it has a foreign key to another entity.
    - For example, an enrolment of a student may be a **weak entity**, as an enrolment cannot exist without a student.



- An **associative entity** is a table that associates two other tables in a many to many relationship.



actor		
 actor_id	int	
name	text	

film		
 film_id	int	
title	text	
genre	text	

actor_film_mapping		
actor_id	int	
film_id	int	



# Attributes

- An attribute is a property of an entity or something that can be used to describe an entity.

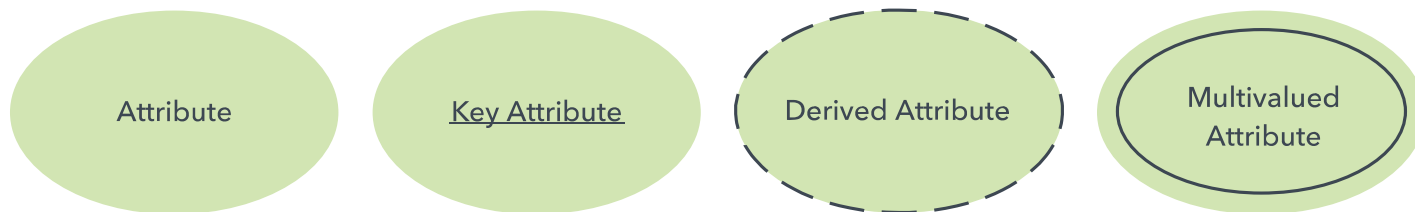


- There are several different types of attributes represented on an ERD:
  - **Simple:** an attribute that cannot be split into other attributes, such as a first name.
  - **Composite:** an attribute that can be split into other attributes, such as name being split into first, middle, and last name.
  - **Derived:** an attribute that is calculated or determined from another attribute, such as the age of record being calculated from the created date.
- An attribute can also be single-value or multi-value:
  - **Single-value:** an attribute that is only captured once for an entity.
  - **Multi-Value:** an attribute that can be captured more than once for an entity, such as multiple phone numbers.

entity, such as multiple phone numbers.

# Attributes ○

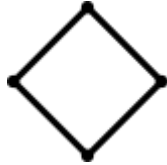
- They are often represented as ovals.



Types of Attributes    Description

Simple attribute| Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value. Composite attribute| It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name. Derived attribute| This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee. Multivalued attribute| Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

# Relationship

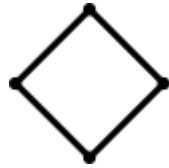


- Relationship represents how entities act upon each other or are associated with each other.
- It is commonly thought as a **verb** connecting the **entities** or **nouns**.
- It is an association among two or more entities, e.g. Students **sits** in Class
  - For example, the named student might register for a course.
  - The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way.
- Relationships are typically shown as diamonds or labels directly on the connecting lines.



Relationship

# Relationship



- **Strong relationship**

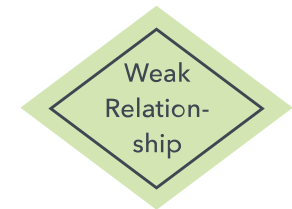
- A relationship where entity is existence-independent of other entities, and PK of Child doesn't contain PK component of Parent Entity.



- A strong relationship is represented by a single rhombus:

- **Weak (identifying) relationship**

- A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity.



- This relationship is represented by a double rhombus:



# Cardinality of Relationship

1:1

1:N

N:N

- Cardinality is a number of items that must be included in a relationship
- An entity in a relationship with minimum cardinality of zero plays an optional role in the relationship
- An entity with a minimum cardinality of one plays a mandatory role in the relationship
- The degree of relationship (cardinality) is represented by characters “1”, “N” or “M” usually placed at the ends of the relationships

# Cardinality

1:1

1:N

N:N

- **one-to-one (1:1)**

- The employee can manage only one department, and each department can be managed by one employee only:



- **one-to-many (1:N)**

- The customer may place many orders, but each order can be placed by one customer only:



# Cardinality



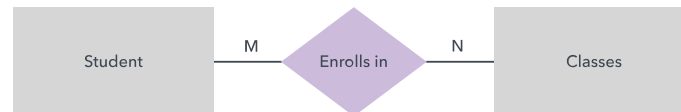
- **many-to-one (N:1)**

- Many employees may belong to one department, but one particular employee can belong to one department only:



- **many-to-many (M:N)**

- One student may belong to more than one student organisations, and one organisation can admit more than one student:

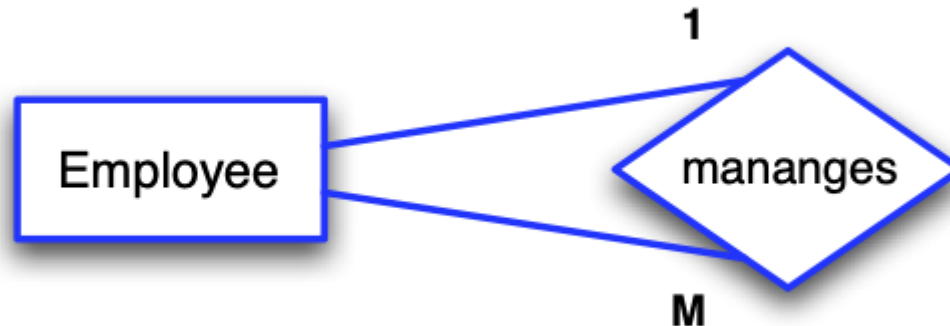


# Degree of Relationship



There are three Degree of Relationships in ERD notation, namely:

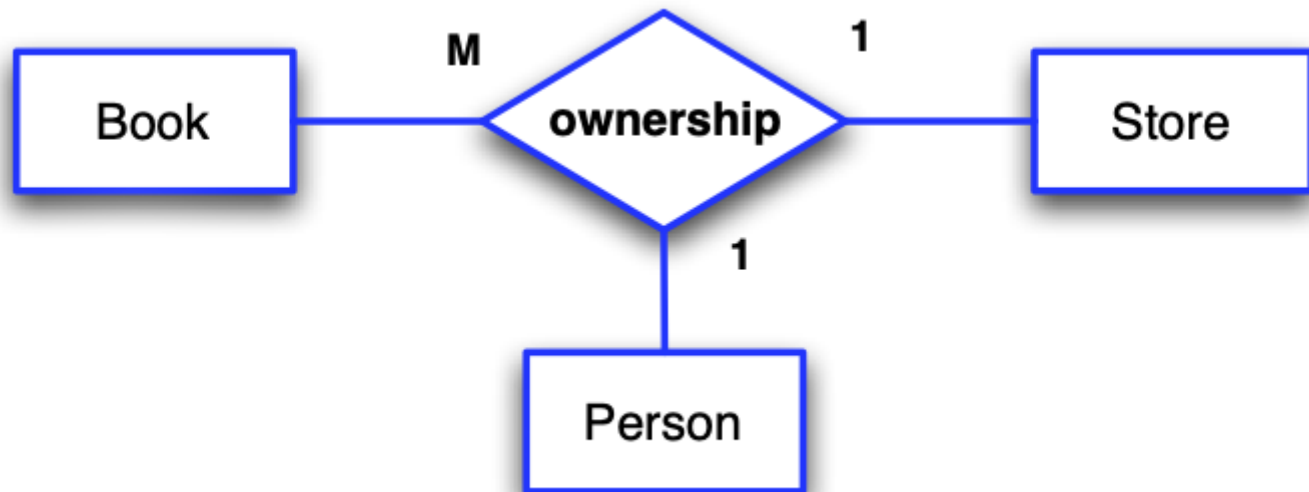
- Unary



- Binary



- Ternary



# Participation Constraints



An entity set may participate in a relation either totally or partially.

- **Total participation** means that every entity in the set is involved in the relationship, e.g., each student must be guided by a professor (there are no students who are not guided by any professor).
  - This kind of relation is depicted as a **double line**.
- **Partial participation** means that not all entities in the set are involved in the relationship, e.g., not every professor guides a student (there are professors who don't).
  - A partial participation is represented by a **single line**.



- The relationship shown above means that each student, without exception, must be guided by one chosen professor, and one – but not every – professor can guide many students.
- So there is no student that is not guided by a professor, and on the other hand there can be professors who don't guide any students.

# Optionality of a Relationship



- Optionality of a relationship refers to whether a relationship instance can exist without an entity in a given role.

- A **mandatory** relationship is represented by a solid line:

—————  
mandatory relationship

- An **optional** relationship is represented by a dashed line:

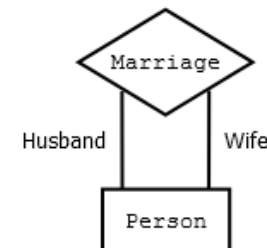
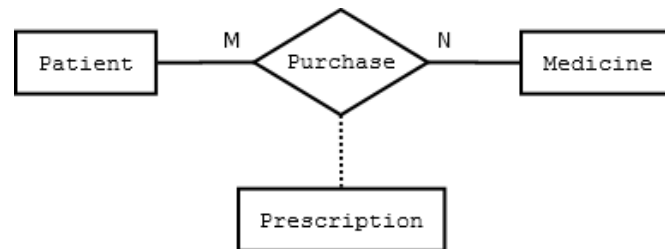
- - - - -  
optional relationship



# Optionality of a Relationship (Example)



- Consider a **marriage**, a type of relationship with **two mandatory roles** filled by the same entity set.
- In most relationships, the entity sets also define the roles, but when an entity set appears multiple times in a single relationship, we distinguish them in different roles.
- Patient can Purchase Medicine with or without a Prescription.
- A Purchase can't exist without a Patient and Medicine, but a Prescription is optional (overall, though it may be required in specific cases).

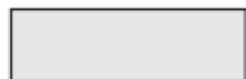


# Chen Notation

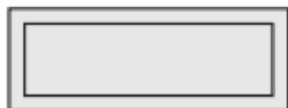
- **Rectangles**
  - This symbol represent entity types
- **Ellipses**
  - Symbol represent attributes
- **Diamonds**
  - This symbol represents relationship types
- **Lines**
  - It links attributes to entity types and entity types with other relationship types
- **Primary key**
  - attributes are underlined
- **Double Ellipses**
  - Represent multi-valued attributes

## Symbol

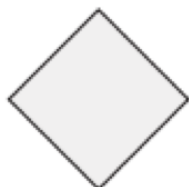
## Meaning



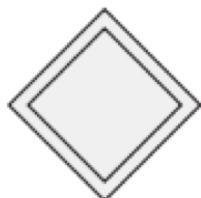
Entity



Weak Entity



Relationship



Identifying Relationship



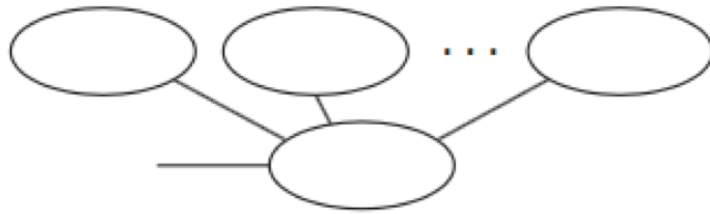
Attribute



Key Attribute



Multivalued Attribute

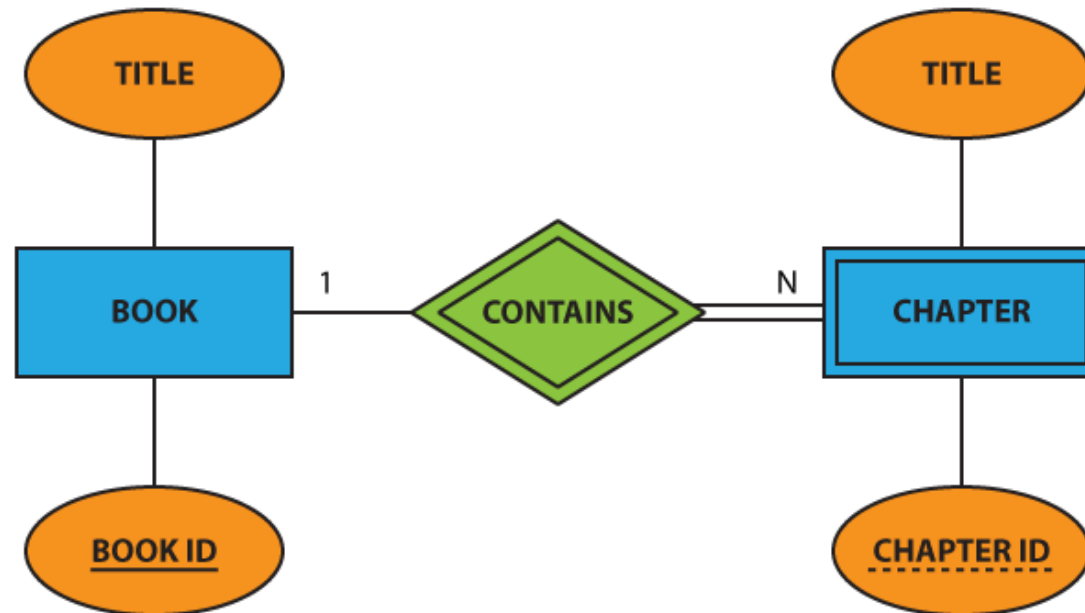
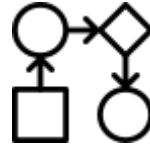


**Composite Attribute**



**Derived Attribute**

# Chen Notation (Example)



# Limitations of ER Diagrams and Models



- **Only for relational data**
  - Understand that the purpose is to show relationships. ER diagrams show only that relational structure.
- **Not for unstructured data**
  - Unless the data is cleanly delineated into different fields, rows or columns, ER diagrams are probably of limited use. The same is true of semi-structured data, because only some of the data will be useful.
- **Difficulty integrating with an existing database**
  - Using ER Models to integrate with an existing database can be a challenge because of the different architectures.

# How to Draw a Basic ER Diagram



1. **Purpose and scope:** Define the purpose and scope of what you're analyzing or modeling.
2. **Entities:** Identify the entities that are involved. When you're ready, start drawing them in rectangles (or your system's choice of shape) and labeling them as nouns.
3. **Relationships:** Determine how the entities are all related. Draw lines between them to signify the relationships and label them. Some entities may not be related, and that's fine. In different notation systems, the relationship could be labeled in a diamond, another rectangle or directly on top of the connecting line.
4. **Attributes:** Layer in more detail by adding key attributes of entities. Attributes are often shown as ovals.
5. **Cardinality:** Show whether the relationship is 1-1, 1-many or many-to-many.

# Tips for ER Diagrams

1. Show the level of detail necessary for your purpose. You might want to draw a conceptual, logical or physical model, depending on the detail needed. (See above for descriptions of those levels.)
2. Watch for redundant entities or relationships.
3. If you're troubleshooting a database problem, watch for holes in relationships or missing entities or attributes.
4. Make sure all your entities and relationships are labeled.
5. You can translate relational tables and ER diagrams back and forth, if that helps you achieve your goal.
6. Make sure the ER diagram supports all the data you need to store.
7. There may be different valid approaches to an ER diagram. As long as it provides the necessary information for its scope and purpose, it's good.



# ERD Naming and Refinement



- When designing a database schema, the choice of names for entity types, attributes, relationship types and (particularly) roles is not always straightforward.
- One should choose names that convey, as much as possible, the **meanings** attached to the different constructs in the schema.
- The cardinality ratio and participation constraint of each relationship type are **determined** from the requirements.
  - If some cardinality ratio or dependency cannot be determined from the requirements, the users must be questioned further to determine these structural constraints.
- We choose to use **singular names** for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type.
- It is occasionally difficult to decide whether a particular concept should be modeled as an entity type, an attribute or a relationship type.
- In general, the schema design process should be considered an **iterative refinement** process, where an initial design is created and then iteratively refined until the most suitable design is reached.

# Your First ERD

The company you work for wants to digitise their time cards.

You have been asked to design the database for submitting and approving time cards.

- A **timecard** should have hours worked and date submitted
- Each **timecard** is associated with exactly one **employee**
- Each **timecard** should have a unique **id**
- Each **timecard** has a **status**: it is either **approved**, **not approved** or **pending**
- Each **employee** has a unique **id**
- Each **employee** has a name and **address**.
- Each **employee** submits a time card **every pay period**, i.e. in 1 year, they will submit multiple time cards
- Each **employee** either has **direct deposit** or **physical cheque** as their method of payment
- Each **employee** is associated with exactly one **manager**
- Each **manager** has a unique **id** and a **name**
- Each **manager** is in charge of **multiple employees**
- Each **manager** approves **time cards** for **multiple employees**

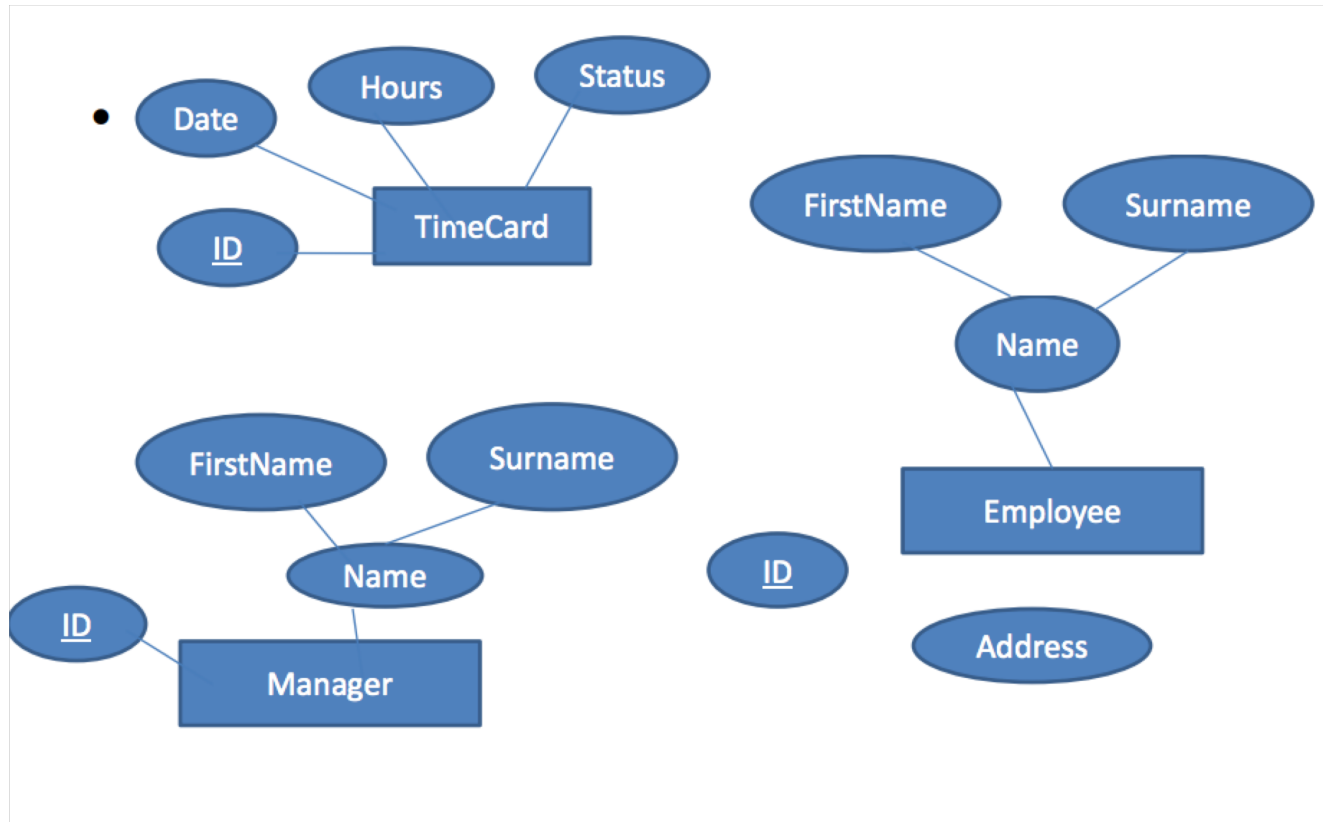
# Define the Entities

TimeCard

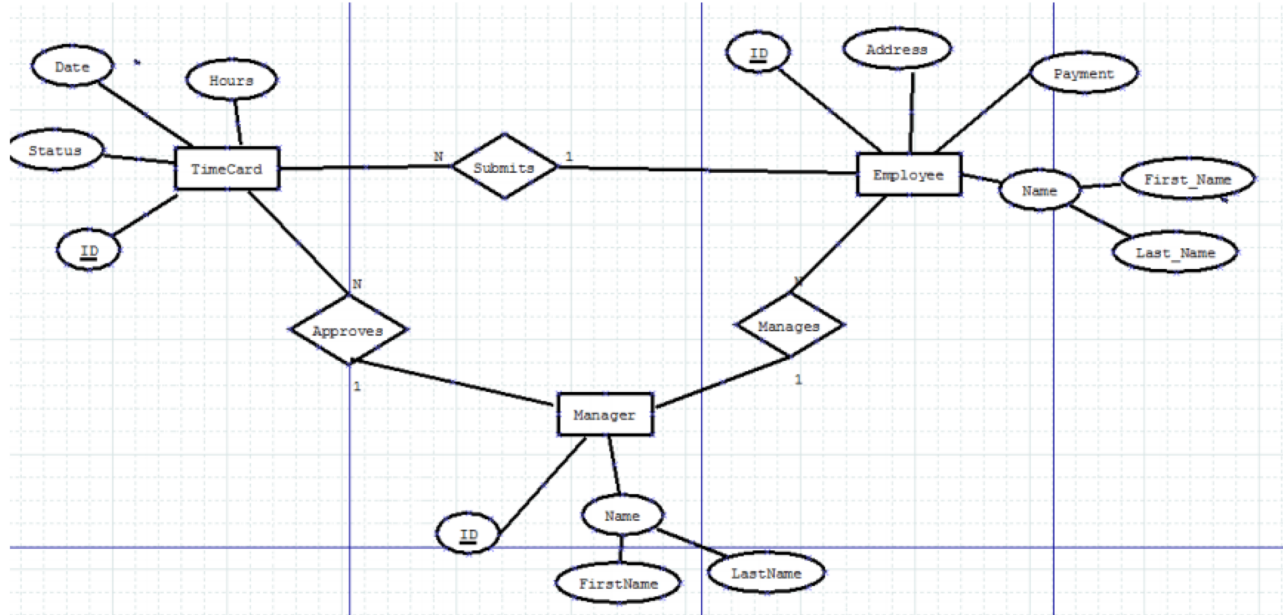
Manager

Employee

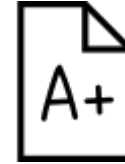
# Define the Attributes ○



# Define the Cardinalities **N:N**

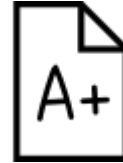


# Exam Question – January Exam 2013



- **Design an ERD according to the provided specifications:**
  - The database must store book, author, publisher and warehouse information.
  - For every book you must capture the title, isbn, year and price information. The isbn value is unique for a book.
  - For every author you must store an id, name, address and the URL of their homepage.
  - Each author can write many books, and each book can have many authors, for example.
  - For every publisher you must store an id, name, address, phone number and an URL of their website.
  - Books are stored at several warehouses, each of which has a code, address and phone number.
  - A book has only one publisher.
  - The warehouse stocks many different books. A book may be stocked at multiple warehouses.
  - The database records the number of copies of a book stocked at various warehouses.

## Exercise 2 – August Exam 2014



- **Design an ERD according to the provided specifications:**
  - United Direct Artists (UDA) is an insurance broker that specialise in insuring paintings for galleries. You are required to design a database for this company.
  - The database must store painters, paintings, and galleries information.
  - Painters have a unique number, Name, and phone number
  - Paintings have unique number, title and price
  - Galleries have unique number, owner, phone number, commission rate and address
  - A painting is painted by a particular artist, and that painting is exhibited in a particular gallery. A gallery can exhibit many paintings, but each painting can be exhibited in only one gallery. Similarly, a painting is painted by a single painter, but each painter can paint many paintings.



# Tutorial

- For previous **Exam Questions**, sketch out your initial ERD with <https://draw.io> (<https://draw.io>).
- Your ERD must show entities, attributes and the relationships between entities.
- Document any assumptions that you make

# Summary



- **Components**
  - **Entity**
  - **Attribute**
  - **Relationship**
    - Cardinality
    - Degree
    - Participation constraints
    - Optionality
- **Chen Notation**
  - Limitations, Tips, Drawing, Naming, Refinement

In [ ]: