DATABASE SYSTEM CONCEPTS, ARCHITECTURE AND DATA MODELS

ARCHITECTURE

DBMS packages has evolved from the early monolithic systems. Now a days, it is on the client-server architecture.

A **client module** is works on the workstations or PCs, and contains application programs and user interfaces (GUIs etc.).

A **server** handles data storage, access, search, and other functions.

DATA MODELS

A **data model** is a precise description of information content. **Data Model** is a representation of the entire information content of the database in a form that is somewhat abstract in comparison with the way which data is physically stored. A database model is a collection of logical constructs (concepts) used to represent the data structure and the data relationships found within database.

Types of data models:

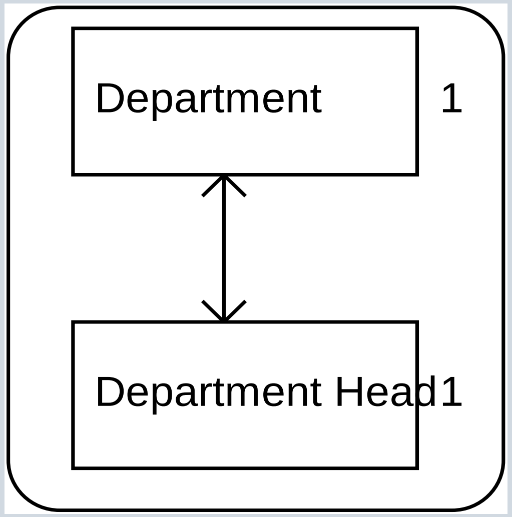
The Conceptual Model (or named as high level data models) is a tool to specify the data requirements of an enterprise. It focuses on the logical nature of the data representation. It is concerned with ***what*** **is represented** in the database rather than *how* it is represented. (in terms that users will understand)

Logical Data Model: in terms that *a relational database system will understand*.
The Physical Data Model (or named as low-level data models) provide concepts that describe the details of *how data is stored* in the computer. They mean something to computer specialist but not to users. (in terms of the underlying computer hardware and operating system)

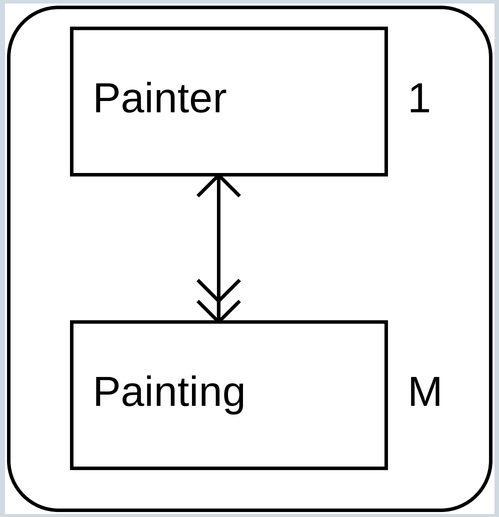
The Implementation Model places the emphasis on *how the data are represented* in the database or how data structures are implemented to represent what is modeled. (Implementation model includes: HIERARCHICAL DB MODEL, NETWORK DB MODEL**,** RELATIONAL DB MODEL).

Conceptual models include the **Entity Relationships(E-R)** model and the Object Oriented Model. Conceptual Models use three types of relationships to describe the associations among data:

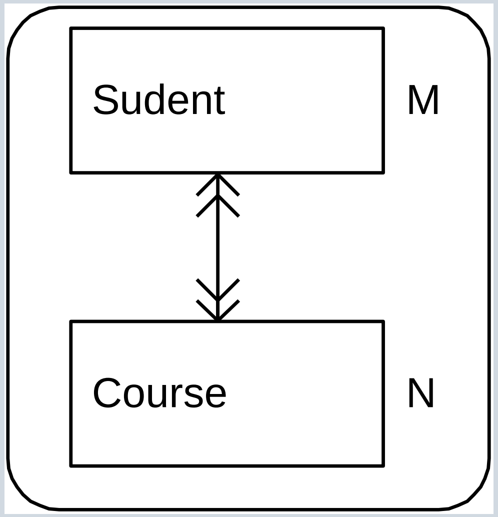
1-One-to-one Relationship:



2-One-to-many Relationship:



3- Many-to-many Relationship



Degree of a Relationship Set

- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. This is the example for Ternary Relationship:

  "*students* work on research *projects* under the guidance of an *instructor*."

  This relationship is a ternary relationship between *instructor, student,* and *Project.*

SCHEMAS, INSTANCES, and DATABASE STATE

The description of a database is called the **database schema,** which is specified during database design and is not expected to change frequently. A displayed schema is called a **schema diagram** which displays the structure of each record type but not the actual instances of the records. We called each object in the schema (such as student or course) **a schema construct**. The data in the database at a particular moment in time is called a **database state** or **snapshot.** It is also called as the current set of **occurrences or instances** in the database. The DBMS stores the descriptions of the schema constructs and constraints- also called the **meta data-** in the DBMS catalog. The schema sometimes called **intension** and a database state an **extension** of the schema. Making changes on the schema is called as **schema evolution.**

A Schema is a data model that is intended to be used with a database system

Internal Level : Describes the physical storage structure of the database. Internal schema defines the representation used by the database server to store the tables in memory or files

External level: The user's view of db. This level describes the part of the database, which is relevant to a particular user. External schemas are defined for the users of a database

Conceptual level: The community view of the database. This level describes what data is stored in the database and the relationships among the data.

Logical schema defines the representation as a collection of tables that are stored in a database server

THE ENTITY RELATIONSHIP DATA MODEL (ER Data Model)

Entity Relationship (E-R) data model is based on objects (entities) and Relationships among these objects. E-R Model is a high level conceptual data model developed by Chen(1976) to facilitate database design.

E-R scheme contains entities and their relationships. Mapping between entity sets can be in three different ways. (one-to-many, one-to-one, many-to-many)

- The classes of thing ....………**THE ENTITY CLASSES / ENTITY SET / ENTITY TYPE**

Each **ENTITY** will ideally be presented by a single type of data record or segment. Since an entity is the basic unit of modelling , the model is sometimes called an entity model.

Entity is an object or concept that is uniquely identifiable. (***entity instance)***

Example: A customer and the characteristics of a customer that are of interest is *an entity*.
All of the potentials of customers. The description of the properties and the information about them is *entity class.*

- The most fundamental logical links between entities...................**DEPENDENCIES** Relationship types between classes

- The properties of each entity type …………………………… **ATTRIBUTES**

Example :first name or last name or age is *an attribute*
first name is ‘Jane‘ or last name is ‘Black’ is *an attribute value***.**

Attribute types:
•Simple and **composite** attributes.
•Single-valued and **multivalued** attributes
•**Derived** attributes: Can be computed from other attributes

- **Domain**–the set of permitted values for each attribute

Entity Sets

▪An **entity** is an object that exists and is distinguishable from other objects.

- Example: specific person, company, event, plant

▪An **entity set** is a set of entities of the same type that share the same properties.

- Example: set of all persons, companies, trees, holidays

▪An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.

- Example:

instructor = (*ID, name, salary*)
course= (*course_id, title, credits*)

▪A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.

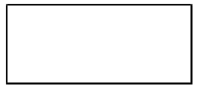
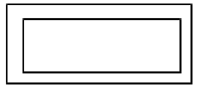
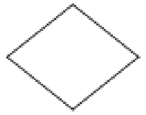
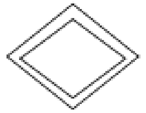
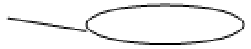
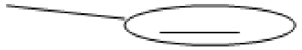
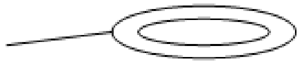
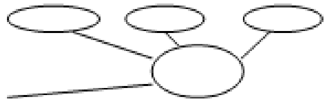
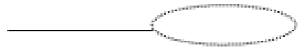
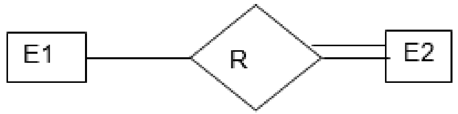
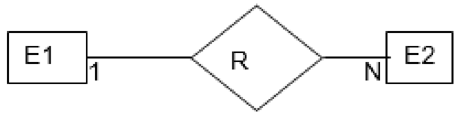
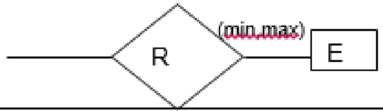
- Constraints for ER diagrams:

- Types of attributes
- Designation of key attributes
- Cardinalities of relationship types

Example: *Constraint for age:* Let's say age of the customers should greater than 18 and after 70, they can not be customer too. This range is the constraint for age attribute.
Domain for age: Since above constraint given, domain of age attribute will be all the numbers between 18 to 70

Data Dictionary: A table that contains the description of classes and the types, descriptions, and constraints on attributes of an information system.

A **relationship type R** among n entity types E1,E2,…,En defines a set of associations or a **relationship set** among entities from these types.

SYMBOL	MEANING
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIPS
	ATTRIBUTE (STORED ATTRIBUTE)
	KEY ATTRIBUTE
	MULTIVALUED
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E2 IN R
	CARDINALITY RATIO 1:N For F1:E2 IN R
	STRUCTURAL CONSTRAINT (min,max) ON PARTICIPATION OF E IN R

The **degree** of a relationship type is the number of participating entity types. (**unary**, **binary** for degree two, **ternary** for degree three)

The **Cardinality ratio** for a binary relationship specifies the number of relationship instances that an entity can participate in. Possible cardinality ratios for the binary relationship types are 1:1 , 1:N , N:1 , M:N

The **participation Constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type. **Total participation (total dependency)** is the one where an entity instance can exists if and only if participates in an other relationship instance and this is also called as **existence dependency.** (If existence of the entity X depends on the existence of entity Y, this is **called *Existence Dependency***.*)*If some or part of the instances of an entity are related to another entity or relationship instance, this is called as **partial participation. (partial dependency).** Cardinality ratio and participation constraints are called as **structural constraints.**

Entity types:

A strong entity type is one that exists independently of other entity types. These entities having a key attribute. They also called as **regular entity types** .

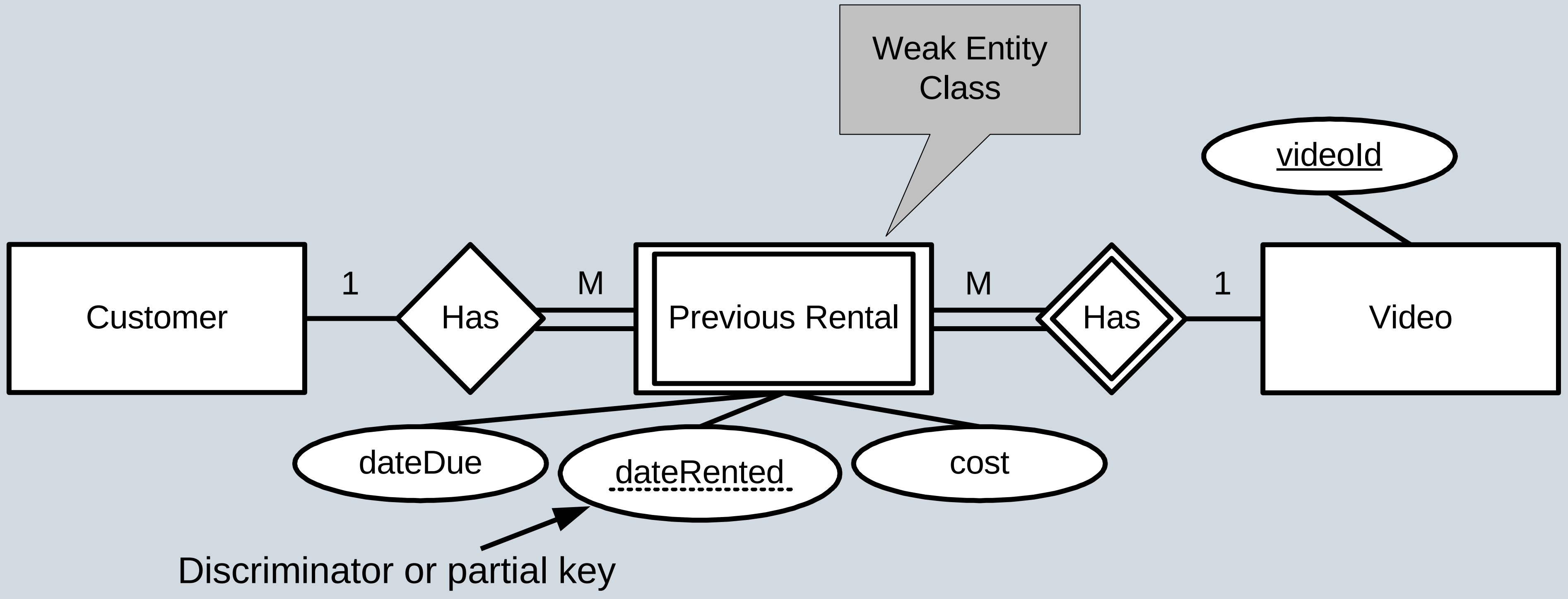
A weak entity type is an entity type whose existence depends on some other entity type. These entity types that do not have key attributes of their own. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with some of their attribute values. We call this entity type as **identifying** or **owner entity type,** and we call the relationship type that relates a weak entity type to its owner **the identifying relationship** of the weak entity type. A weak entity type always total participation constraint with respect to its identifying relationship. A weak entity type normally has a **partial key** which is the set of attributes that can uniquely identify weak entities that are related to the same owner entity. The partial key is defined by dashed underline.
A weak entity type normally has a **partial key**, which is the set of attributes that can uniquely identify weak entities that are related to the same owner entity.

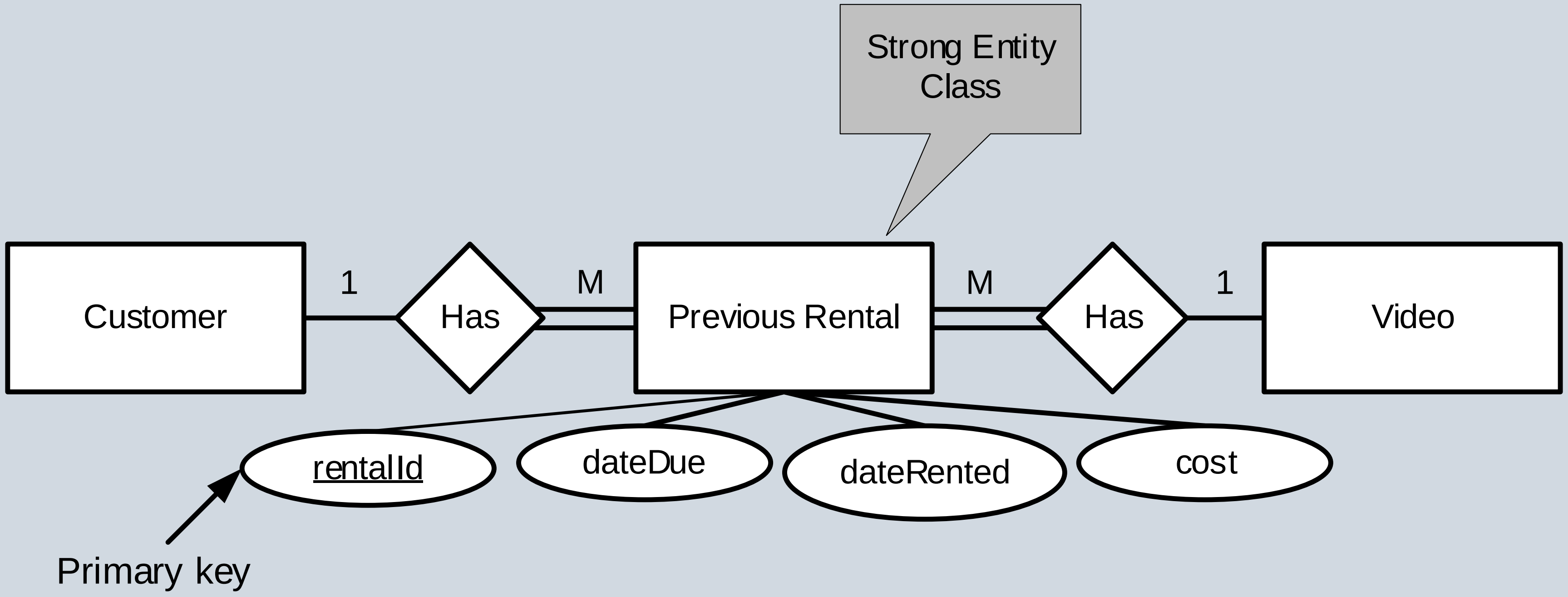
A weak entity class is:
- An entity class with no key of its own
- An entity class whose entities cannot exists without being related to another entities.

An identifying relationship type is:
- A relationship type that determines the keys of weak entities
- Always -to one cardinality.

Partial Key is:
Attributes that help uniquely identifying an entity among all those related to specific entity.

The weak entity class becomes strong by introducing an artificial key attribute.




Strong Entity Class
Customer
1
Has
M
Previous Rental
M
Has
1
Video
rentalId
dateDue
dateRented
cost
Primary key

ATTRIBUTES:

An attribute is a property or characteristics of an entity or relationship type that is of interest to the organization. An attribute that must have a value for every entity (or relationship) instance with which it is associated called as **required attribute**. An attribute that may not have a value for every entity is called **optional attribute**.

An attribute that cannot be broken down into smaller components that are meaningful for the organization called as **simple (or atomic) attribute**. **Composite attributes** are the ones that has meaningful component parts (or attributes-**component attributes**).

Multivalued attribute is an attribute that may take more than one value for a given entity(or relationship) instance.

An attribute which is stored actually in the database table as an attribute of an entity is called as **actual attribute**.

An attribute whose values can be calculated from related attribute values called as **derived attribute**.

An attribute (or combination of attributes) whose values distinguishes instances of an entity type is called as **identifier.**

Each simple attribute of an entity type is associated with a value set (domain) , which specifies the set of values that may be assigned to that attribute for each individual entity.

For enhanced ER Diagrams:

Generalization is the result of taking the union of two or more (lower level) entity sets to produce higher-level entity set. It is the process of defining a more general entity type from a set of more specialized entity types.

Specialization is the result of taking a subset of a higher-level entity set to form a lower level entity set. It is the process of defining one or more subtypes of the super-type and forming supertype/subtype relations.

DISCOVERING RELATIONSHIPS

A **relationship** is a connection between entities. Each entity plays a specific **role** in the relationship. Roles are particularly important in relationship types that relate an entity class to itself.

CARDINALITIES OF RELATIONSHIPS

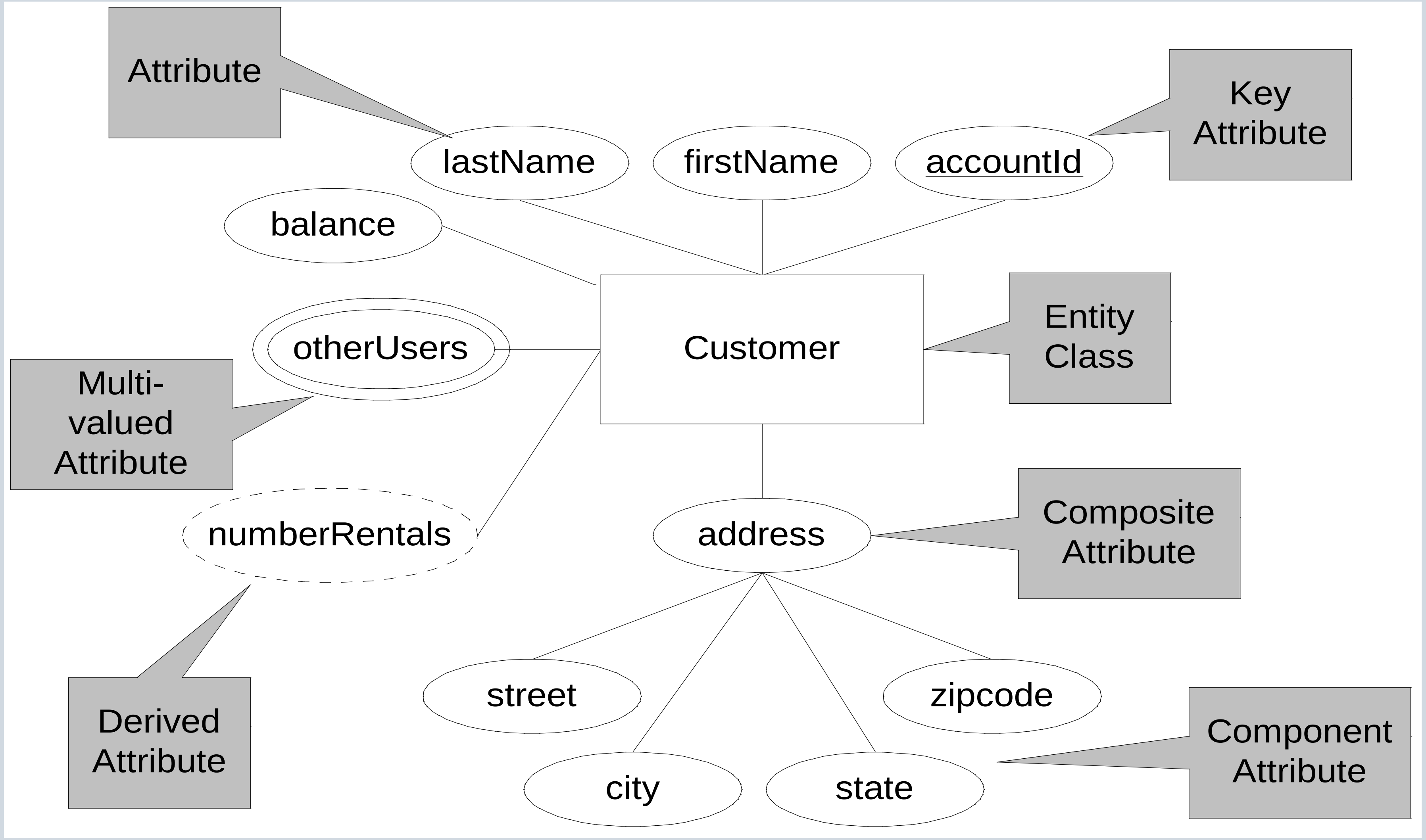
A relationship type may be limited in how many relationships an object may have.

The **<u>cardinality</u>** of a relationship type may be:

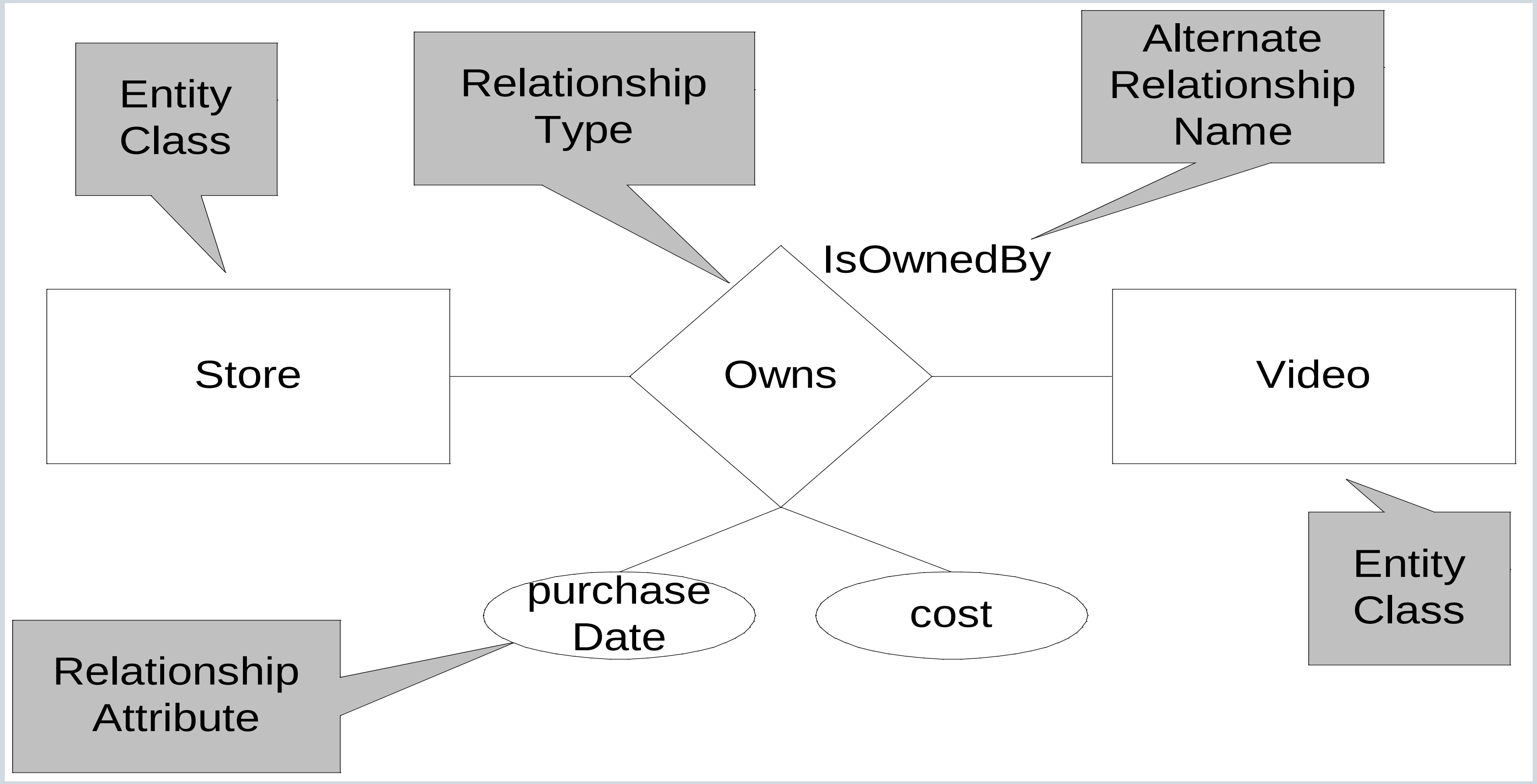
<u>One-to-one</u>: Each object of a class may have no more than one related object of the other class (e.g., marriage relationship)

<u>One-to-many</u>: One of the classes allows an entity to have any number of related entities, but the other class restricts its entities to be related to no more than one entity

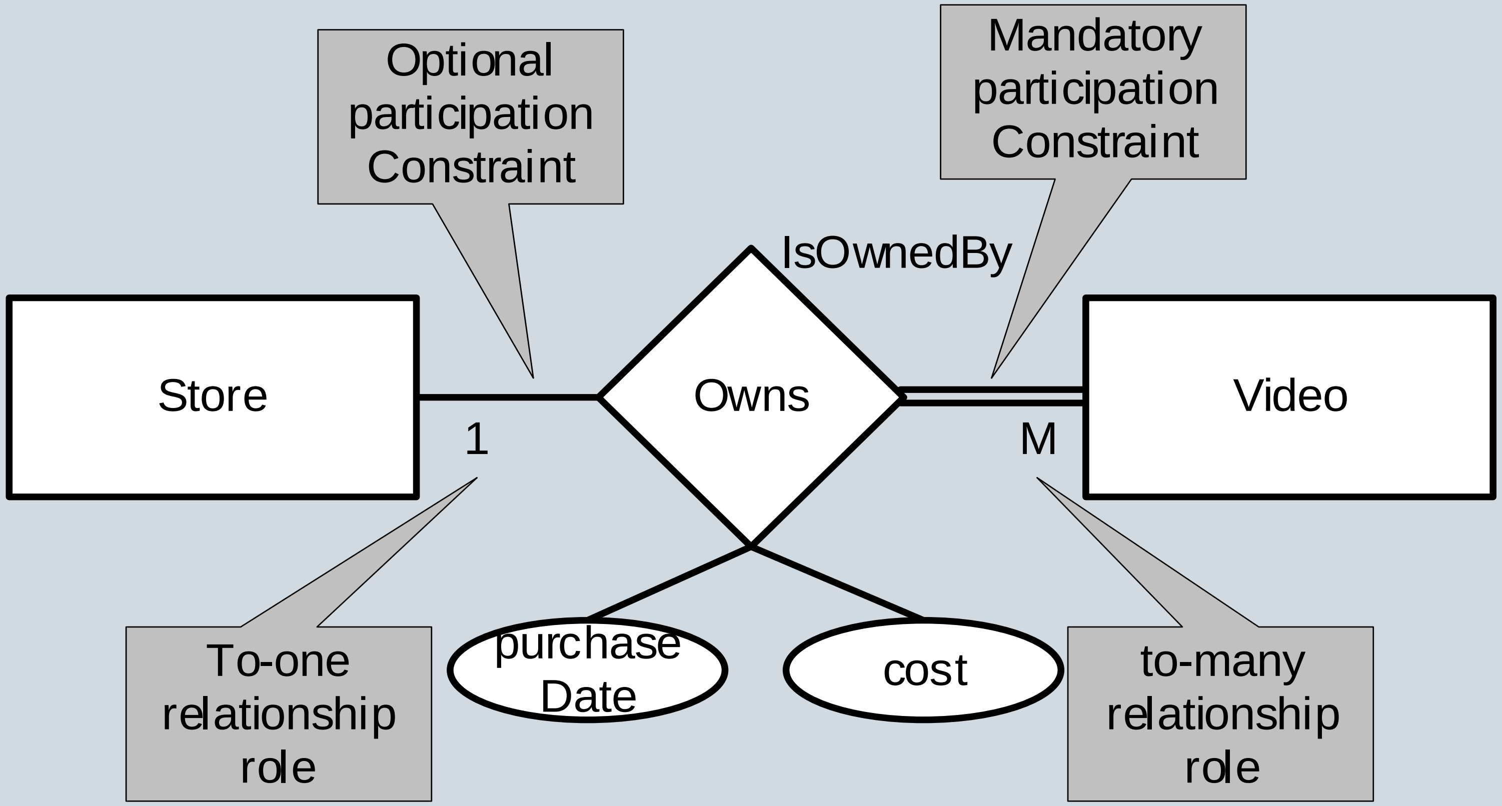
<u>Many-to-many</u>: Entities of both classes may have any number of related entities of the other class.

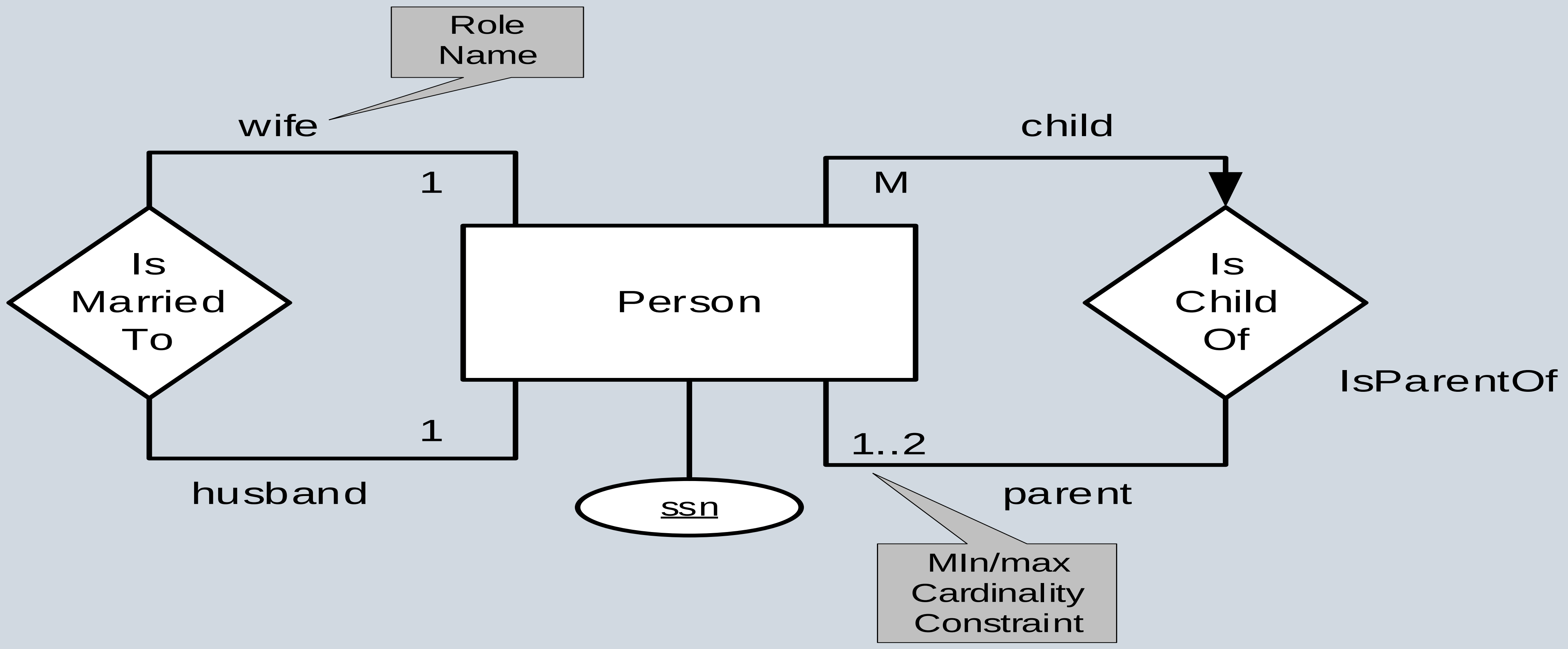

Attribute
lastName
firstName
accountId
Key Attribute
balance
Customer
Entity Class
otherUsers
Multi-valued Attribute
numberRentals
address
Composite Attribute
Derived Attribute
street
zipcode
city
state
Component Attribute

Relationship Type Example:



Cardinality and Participation Constraints:


Optional participation Constraint
Mandatory participation Constraint
IsOwnedBy
Store
Owns
Video
1
M
purchase Date
cost
To-one relationship role
to-many relationship role


Role Name
wife
child
1
M
Is Married To
Person
Is Child Of
IsParentOf
1
1..2
husband
ssn
parent
MIn/max Cardinality Constraint

EXAMPLE: THE COMPANY DATABASE
Requirements:

The company database keeps track of a company’s employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provided the following description of the part of the company to be represented in the database:

1. The company organized into departments. Each department has a unique name and number. A department may have several locations.
   - A particular employee **manages** the department. We keep track of the start date when that employee began managing the department.

2. Each project has a unique name, a unique number and a single location.
   - A department **controls** a number of projects

3. We store each employee’s name, social security number address, salary, sex, and birthdate.
   - An employee is **assigned** to one department
   - Employees may **work on** several projects, which are not necessarily controlled by the same department.
   - We keep track of the **direct supervisor** of each employee.

4. We keep dependents of each employee’s first name, sex, birth date, and relationships to the employee.
   - Employee **has** dependents, we want to keep track of the dependents for insurance purposes.

Preliminary design

Entity Types and Attributes:
These are defined according to the requirements.

- DEPARTMENT: Strong Entity type
  Name: actual attribute, unique, candidate key
  Number: actual attribute, unique, primary key
  Location: actual attribute, multivalued attribute
  Number of employees: derived attribute

- PROJECT : Strong Entity type
  Name: actual attribute, unique, candidate key
  Number: actual attribute, unique, primary key
  Location : actual attribute
  Number of employees: derived attribute

- EMPLOYEE : Strong Entity type
  Name: actual attribute, composite attribute. Components: First name, Middle name, Last name
  SSN: actual attribute, unique, primary key
  Sex: actual attribute
  Address: actual attribute
  Salary: actual attribute
  Birth Date: actual attribute
  Number of projects: derived attribute
  Number of dependents: derived attribute

- DEPENDENT : Weak Entity type
  DependentName: actual attribute, partial key
  Sex: actual attribute
  BirthDate: actual attribute
  Relationships (to the employee): actual attribute

So far, we have not represent the fact that an employee can work on several projects, nor have we represented the number of hours per week an employee works on each project. This characteristic is listed as part of requirement can be represented by a multivalued composite attribute of employee called WorksOn with the simple components. (Project, Hours)

Relationship Types

The cardinality ratio and participation constraint of each relationship type are determined from the requirements.

The following relationship types are specified.

- MANAGES: A 1:1 relationship type between EMPLOYEE and DEPARTMENT.
    - Employee participation is partial. (all of them can not be a manager)
    - Department participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation.
    - The attribute StartDate is assigned to this relationship type.

- WORKS_FOR: A 1:N relationship type between DEPARTMENT and EMPLOYEE.
    - Both participations are total.

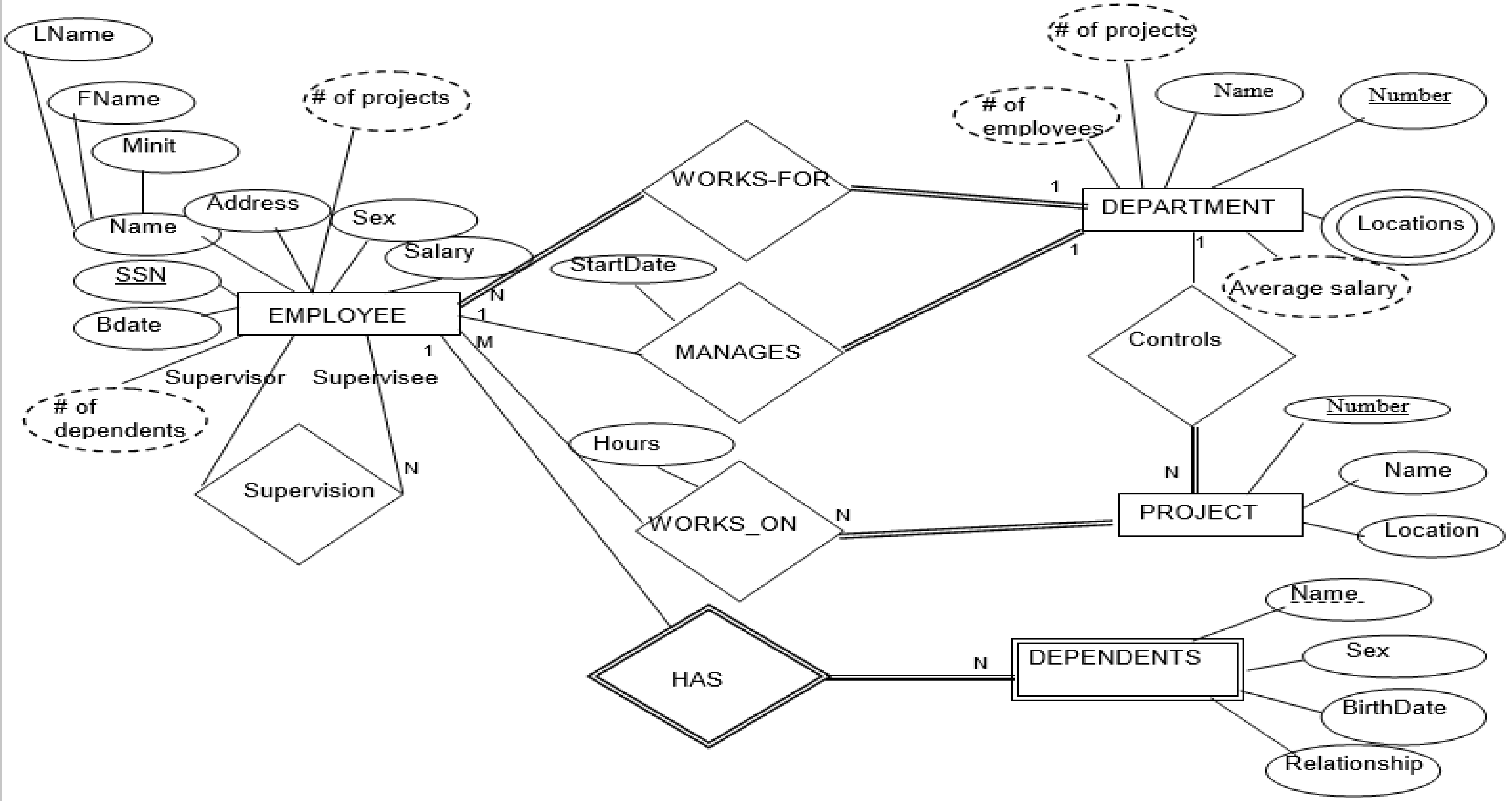
- CONTROLS: A 1:N relationship type between DEPARTMENT and PROJECT.
    - The participation of project is total,
    - Participation of department is determined to be partial. The users indicate that some departments may control no projects.

- SUPERVISE: A 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).
    - Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.

- WORKS_ON: determined to be an M:N relationship type between EMPLOYEE AND PROJECT with an attribute hours.
    - After the users indicate that a project can have several employees working on it, projects participation is determined to be total.
    - But some of the employees may not works on the projects, so participation of Employee is partial.
    - Hours attribute which shows how many hours each employee works on each project is assigned to this relationship type.

- DEPENDENTS_OF / HAS: A 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT.

  - The participation of employee is partial.
  - The participation of DEPENDENT is total.


LName
FName
Minit
Name
Address
Sex
Salary
# of projects
SSN
Bdate
EMPLOYEE
# of dependents
Supervisor
Supervisee
Supervision
1
N
WORKS-FOR
N
1
StartDate
MANAGES
1
M
Hours
WORKS_ON
N
HAS
N
DEPENDENTS
Name
Sex
BirthDate
Relationship
# of projects
# of employees
Name
Number
DEPARTMENT
Locations
1
1
1
Average salary
Controls
N
Number
Name
PROJECT
Location

CONVERTING AN E-R DIAGRAM TO RELATIONAL DB

To obtain table (relational database) from E-R diagrams, every attribute will be one filed of a table(column) , name of entity sets will be name of table.

1. All entities ➔A table. Define a relation with the same name and create attribute for each single valued attributes. And define a primary key.
2. Weak entities➔A new table and should take the primary key of the entity that they being in relation, as part of their primary key.
3. For Multivalued attributes ➔There should be a new table in which the primary key consists of pk. of the main table+multivalued attribute.
4. For composite attributes➔Define an attribute for each component attribute. If appropriate, use the name of composite attribute as part of the name of an attribute (eg.LastName)
5. For derived attributes➔They are not defined as an attribute but, they defined for a specific function, which will derive their values. But in special cases, for example if it takes too much time to derive the attribute(eg. Needs to read many records to calculate the value) it may be preferred to keep it as an attribute, and update it when the values of attributes which affects to value of this derived attribute is changed.

6. 1-1 relationships➔As an attribute in total dependency site.(Eg: Department-manager relation represented as an attribute in Department table)
7. 1-M relationships➔ An attribute in M site. (Eg. Mother-child relation represented as an attribute in child table) If M site has partial dependency, and the density of this dependency is very poor, then prefer to have a new table with primary key composed of primary keys of the two site(entity).
8. M:N relationship ➔ In a new table which has a primary key composed of the primary keys of two entity-table. (Eg: Student-Course relation in a new table let's say stucrs, with primary key <u>stid,crscode</u>)

E-R Diagrams Lecture Exercises

1. CONCERT SEASON

- CONCERT-SEASON: The season during which a series of concerts will be performed. Identifier is Opening-date, which includes Month, Day and Year.
- CONCERT: A given performance of one or more compositions. Identifier is Concert-Number. Another important attribute is Concert-Date. Each concert typically has more than one concert date.
- COMPOSITION: Compositions to be performed at each concert. Identifier is Composition-Id. Composer-Name and Composition-Name and Movement-Id (which consists of Movement-Number and Movement-Name) are the other attributes. Many, but not all, compositions have multiple movements.
- CONDUCTOR: Person who will conduct the concert. Identifier is Conductor-ID. Another attribute is Conductor-Name.
- SOLOIST Solo artist who performs a given composition on a particular concert. Identifier is Soloist -ID. Another attribute is Soloist-Name.
- A concert season schedules one or more concerts. A particular concert is scheduled for only one concert season.
- A concert includes the performance of one or more compositions. A composition may be performed at one or more concerts, or may not be performed.
- For each concert there is one conductor. A conductor may conduct any number of concerts, or may not conduct any concerts.
- Each composition may require one or more soloists, or may not require a soloist. A soloist may perform one or more compositions at a given concert, or may not perform any composition. The symphony orchestra wishes to record the date when a soloist last performed a given composition.(Date-Last-Performed)

2. REALESTATE

- The firm has a number of sales offices in several states. Attributes of sales office include Office-number (identifier) and Location.
- Each sales office is assigned one or more employees. Attributes of employee include Employee-ID (identifier) and Employee-Name. An employee must be assigned to only one sales office.
- For each sales office, there is always one employee assigned to manage that office . An employee may manage only the sales office to which she is assigned.
- The firm has property for sale. Attributes of property include Property-id (identifier) and Location. Components of Location include Address, City, State and Zip Code.
- Each unit of property must be listed with one (and only one) of sales offices. A sales office may have any number of properties listed, or may have no properties listed.
- Each unit of property has one or more owners. Attributes of owners are Owner-Id (identifier) and Owner-Name. An owner may own one or more units of property. An attribute of the relationship between property and owner is Percent-Owned.

3. SUPPLIER

- A supplier may supply many items and all items may supplied by many suppliers. Items has item id, name and total quantity on hand. Supplier has supplier id name and location..
- An item must be used in at least one product. Each product must use one or more items. Product has product id, name and quantity on stock.
- Each supplier may make shipments. Each shipment must be done by exactly one supplier.
- A shipment must include one or more items. Shipment date and number should be recorded.
- A customer (customer identified by customer no, name and address) may order many different kinds of products (as part) in different quantities in one order. And customer can make many order. Order defined by order no and date.

4. SUPPLIER II

- A supplier may supply many items and all items may supplied by many suppliers. Items has item id, name and total quantity on hand. Supplier has supplier id name and location..

- An item must be used in at least one product. Each product must use one or more items. Product has product id, name and quantity on stock.

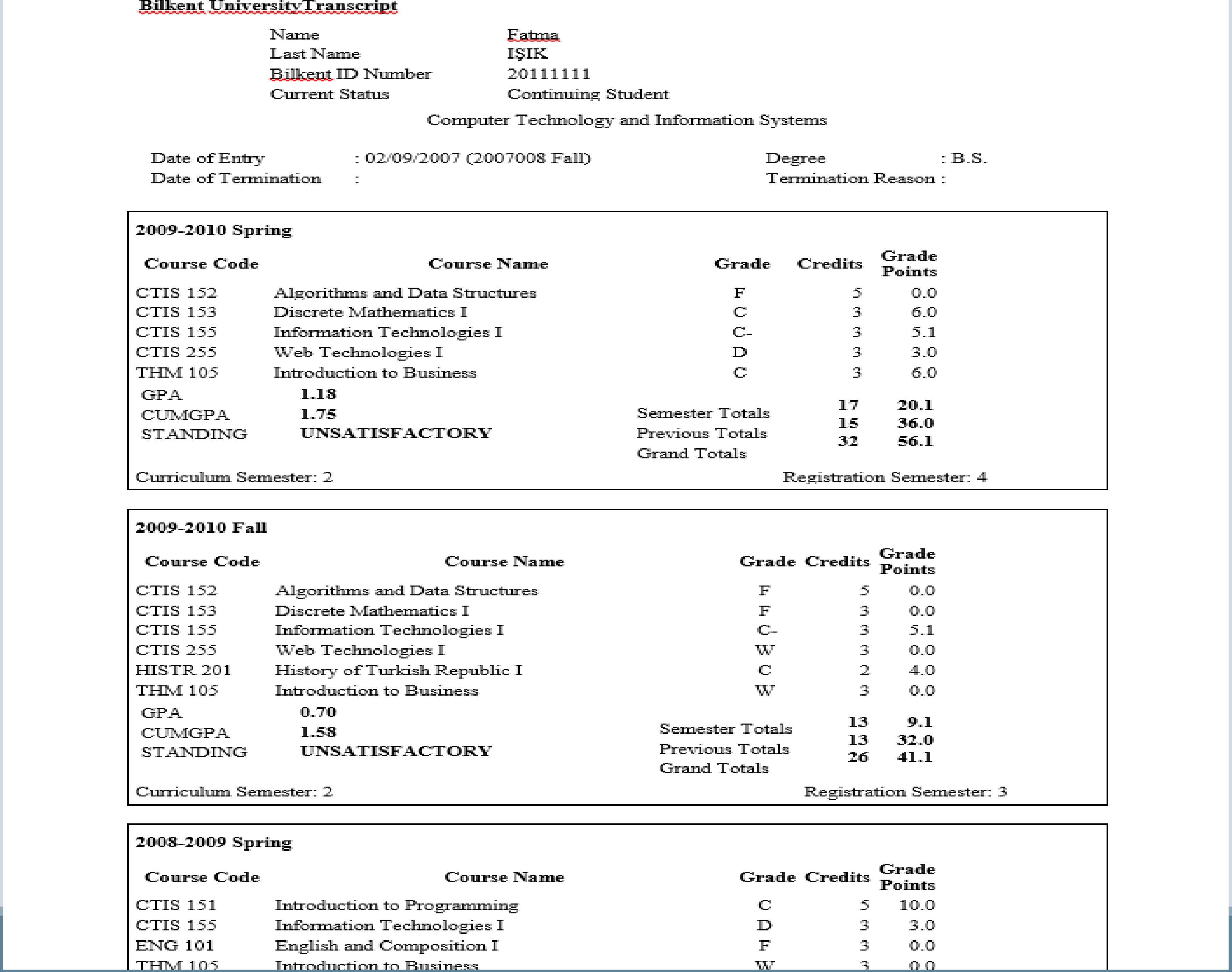
- Shipments of the suppliers are also saved in the system. Each supplier may make shipments. Each shipment must be done by exactly one supplier.

A shipment must include one or more items. Shipment date and the number of shipment on that day by that supplier will be recorded. This means that there can be same shipment number for different suppliers on same date.

- A customer (customer identified by customer no, name and address) may order many different kinds of products(as part) in different quantities in one order. And customer can make many order. Order defined by order no and date.

- In some reports, the number of order made by customer, and the number of products bought in each order wants to be report.

- In some other reports, number of items supplied by each supplier should also be included.



Bilkent UniversityTranscript

Name	Fatma
Last Name	IŞIK
Bilkent ID Number	20111111
Current Status	Continuing Student

Computer Technology and Information Systems

Date of Entry	: 02/09/2007 (2007008 Fall)	Degree	: B.S.
Date of Termination	:	Termination Reason	:

2009-2010 Spring

Course Code	Course Name	Grade	Credits	Grade Points
CTIS 152	Algorithms and Data Structures	F	5	0.0
CTIS 153	Discrete Mathematics I	C	3	6.0
CTIS 155	Information Technologies I	C-	3	5.1
CTIS 255	Web Technologies I	D	3	3.0
THM 105	Introduction to Business	C	3	6.0

GPA	**1.18**			
CUMGPA	**1.75**	Semester Totals	**17**	**20.1**
STANDING	**UNSATISFACTORY**	Previous Totals	**15**	**36.0**
		Grand Totals	**32**	**56.1**

Curriculum Semester: 2 Registration Semester: 4

2009-2010 Fall

Course Code	Course Name	Grade	Credits	Grade Points
CTIS 152	Algorithms and Data Structures	F	5	0.0
CTIS 153	Discrete Mathematics I	F	3	0.0
CTIS 155	Information Technologies I	C-	3	5.1
CTIS 255	Web Technologies I	W	3	0.0
HISTR 201	History of Turkish Republic I	C	2	4.0
THM 105	Introduction to Business	W	3	0.0

GPA	**0.70**			
CUMGPA	**1.58**	Semester Totals	**13**	**9.1**
STANDING	**UNSATISFACTORY**	Previous Totals	**13**	**32.0**
		Grand Totals	**26**	**41.1**

Curriculum Semester: 2 Registration Semester: 3

2008-2009 Spring

Course Code	Course Name	Grade	Credits	Grade Points
CTIS 151	Introduction to Programming	C	5	10.0
CTIS 155	Information Technologies I	D	3	3.0
ENG 101	English and Composition I	F	3	0.0
THM 105	Introduction to Business	W	3	0.0