DBMS

Created by	Ⓑ Borhan
Last edited time	@October 21, 2024 9:29 PM
Tag	

Resources:

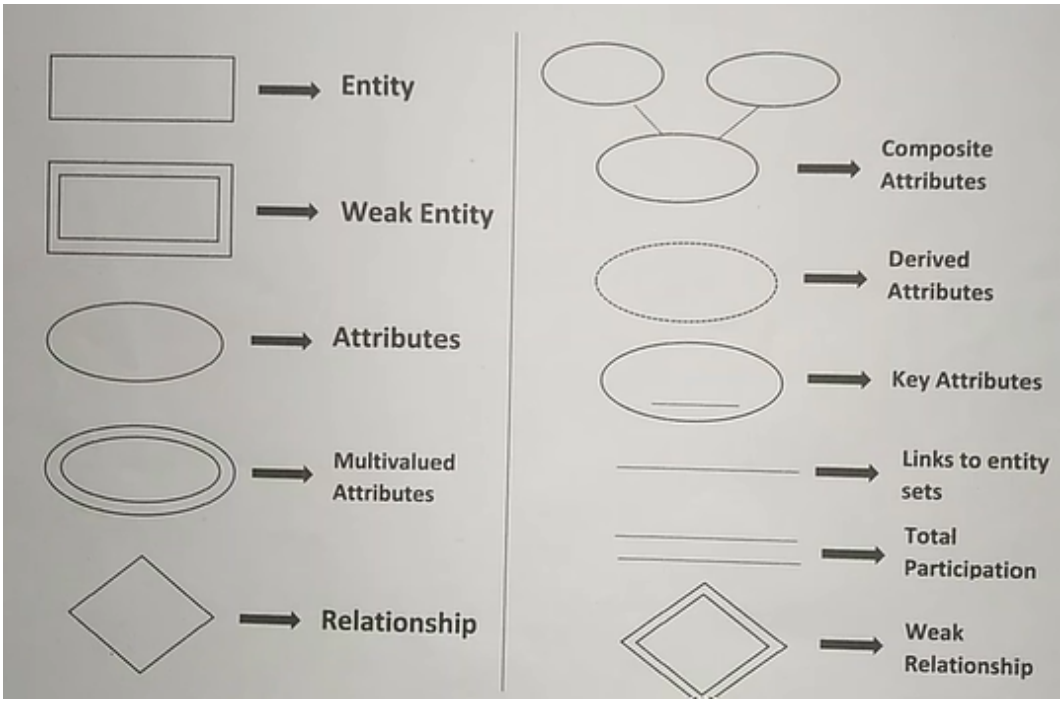
- **Videos :**
  - Bangla → https://www.youtube.com/watch?v=S4cBelRt2D8&list=PLgrAmbRAezuhRWTyOKPxvoiyNQfPo6Iv3,
  - To Understand normalization, FD → https://www.youtube.com/playlist?list=PLxCzCOWd7aiFAN6I8CuViBuCdJgiOkT2Y,
  - Understanding Discriminator : https://www.youtube.com/watch?v=qgoXPY2DmO0&ab_channel=RituKapurClasses,
  - ERD to Relational Schema : https://www.youtube.com/watch?v=OwdFzygGZqk&list=PLKDJE8BkZ4wzsaO-eutJTega6iXhDqjNC&ab_channel=OrangeOutputs,

- **PDF and Articles:**
  - Best One : https://www.poriyaan.in/paper/database-management-system-79/
  - Theory → https://www.mbbcollege.in/db/notes/233.pdf, https://www.du.ac.in/du/uploads/departments/Operational Research/24042020_E-R Model.pdf
  - DIU : https://elearn.daffodilvarsity.edu.bd/course/view.php?id=10248
  - Have some good Example → https://kp.kiit.ac.in/pdf_files/06/4th-Sem_CSE_Database-Management-System.pdf
  - ER Model Example → https://www.vssut.ac.in/lecture_notes/lecture1423726199.pdf

- ERD to Relational Schema Example : https://mebrahimii.github.io/comp440-fall2020/lecture/week_12/Lecture16.pdf
- Example on Normalization : https://cse.poriyaan.in/topic/example-on-normalization-50872/
- Integrity Constraints : https://www.linkedin.com/pulse/integrity-constraints-dbms-sarbani-sahoo-cvobe/, https://medium.com/@reetesh043/types-of-constraints-in-dbms-73cce32aca80#:~:text=A key constraint in a,in establishing relationships between tables
- Anomalies : https://testbook.com/gate/anomalies-in-dbms-notes#:~:text=Last Updated on Jul 31,data loss%2C and incorrect data,
- Degree of relationship with example : https://afteracademy.com/blog/what-is-the-degree-of-relation-in-dbms/#:~:text=degree of relationship.-,Degree of Relationship,the degree of that relationship

ER Diagram : Entity Relationship Diagram

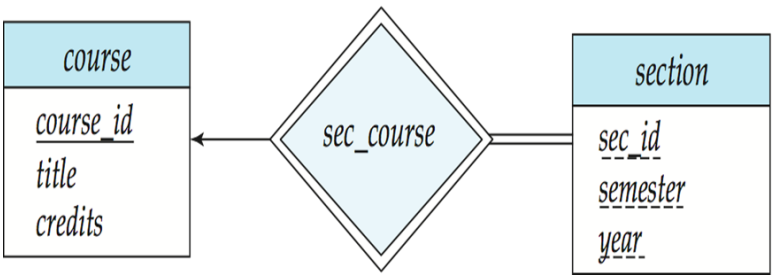
- ER diagram shows the relationship among the entity sets and also helps to visualize the logical structure of database.
- purpose of **defining the relationship between the database entities**.



Entity

- An entity is an object that exists and is distinguishable from other objects. Example : Company, Student. (It is the logical representation of table.)
- **Types:**
  - **Strong entity:** which have a primary key
  - **Weak entity**
    - doesn't have a PK
    - dependent on a strong entity
    - A weak entity set doesn't have a primary key, but we need a means of distinguish between all entities in the weak entity set that depend/linked with a particular strong entity.

Example:



Discriminator/Partial Key: The discriminator of a weak entity set is a set of attributes that allows the distinguish to be made. For an example, we may have same *course_id* multiple times in “section” table, but how could we uniquely identify each of them ? We can find them out by *sec_id, semester, year,* so they are discriminator for this weak entity. *sec_id, semester, year* can be same for each of “*course_id*”, but a “*course_id”* must not have same *sec_id, semester, year* for multiple times.

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

Attributes

Attributes are the properties which define the entity type. *Example: Roll_No, Name, DOB, Age.*

Types of Attributes:

Type	Description
Simple Attribute/ Atomic Attribute	- Can’t be sub-divided further into any other components. - Example : Roll, reg_no
Composite Attribute	- Can further sub-divided into different components - combination of other attributes - Example: name →{ first_name, last_name}
Single-valued	- Consists of a single value for each entity. - Cannot store more than one value.

Type	Description
	Example : Roll_no, DOB, reg; Not: Phone number, because each might have more than one phone number
Multi-valued	- Can store more than one value - Represented by double ellipse/ ovals Example : Phone, email
Derived	- Can be derived from the values of other attributes - Represented by dashed ellipse/oval Example : Age. We can find out the age from DOB.
Key	- can uniquely identify an entity from an entity set - identify each tuple according to this key - represented by underlining Example : Roll, ID
Stored	- fixed value, that won't be changed in future Example : DOB

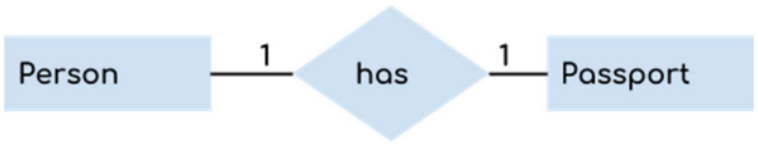
Relationship and Mapping Cardinality

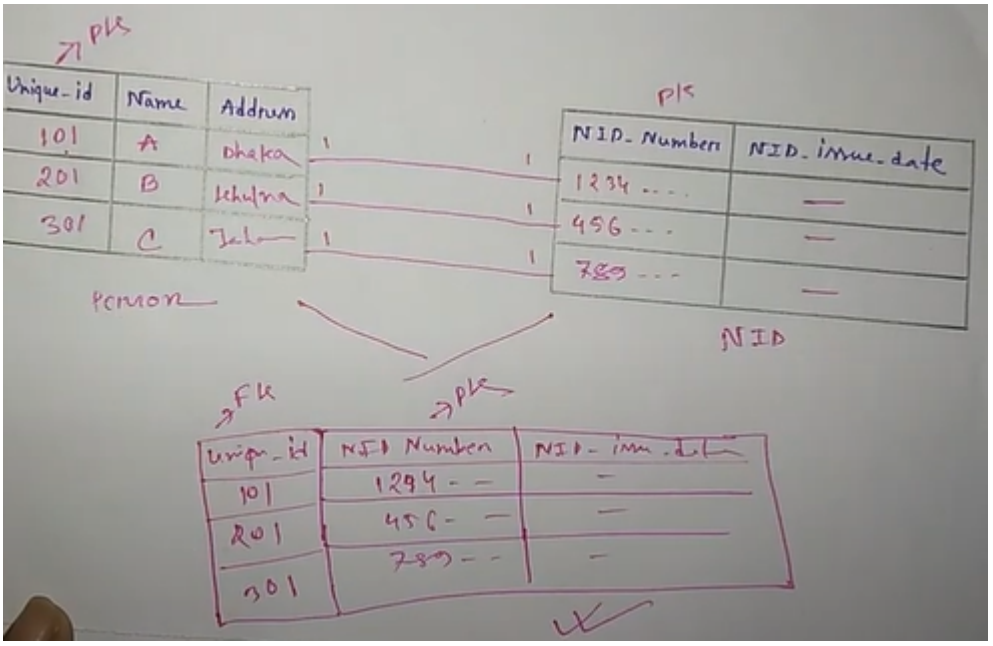
Relationships are association between or among entities.

Mapping Cardinality represents **the number of entities** to which another entity can be associated via a relationship set.

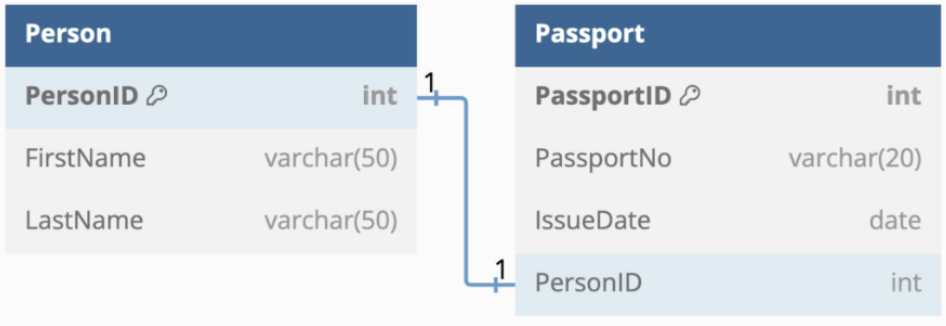
Types of Relationship:

- **One to One (1:1) :** One instance in an entity (parent) refers to one and only one instance in the related entity (child). For example, a person (parent table) has only one passport and a passport/NID (child) is given to one person.



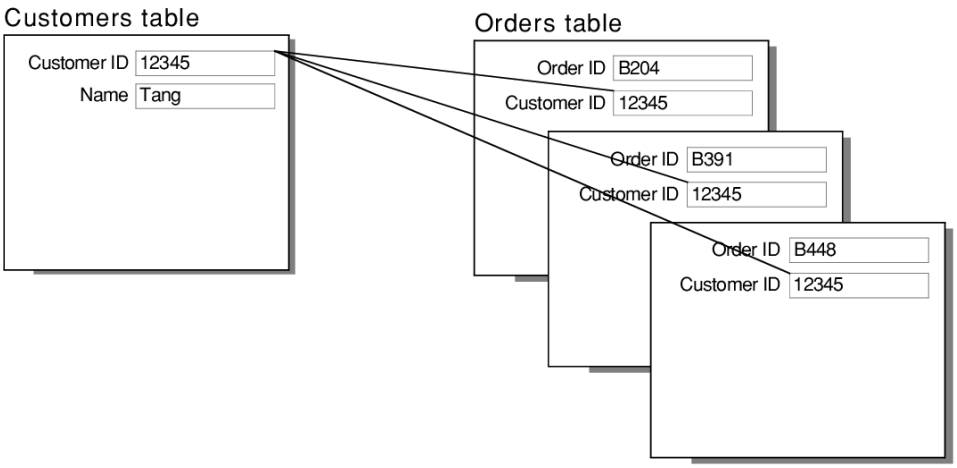


(As a rule of thumb, you should add a foreign key on the child table referencing the parent table) [Ref]



(Here, the `Person` table has a `PersonID` column that's referenced by the `PersonID` column in the `Passport` table. The diagram shows a one-to-one relationship by having a `1` against each column reference. From : Database Guide)

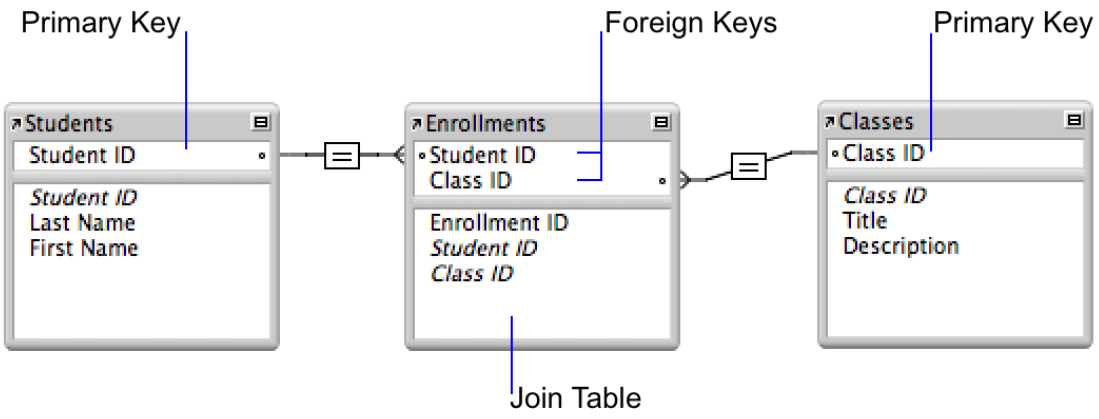
- **One to Many (1:M):** One instance in an entity (parent) refers to one or more instances in the related entity (child). For example – a customer can place many orders but a order cannot be placed by many customers.



(In this example the primary key field in the Customers table, *Customer ID*, is designed to contain unique values. The foreign key field in the Orders table, *Customer ID*, is designed to allow multiple instances of the same value.) [Ref]

- **Many to One (M:1) :** When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.

- **Many to Many (M:N) :** Many to Many relationship exists when one instance of the first entity (parent) can relate to many instances of the second entity (child), and one instance of the second entity can relate to many instances of the first entity. For example, a can be assigned to many classes and a class can be assigned to many students.



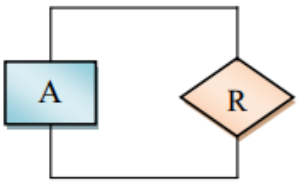
(The primary key Student ID uniquely identifies each student in the Students table. The primary key Class ID uniquely identifies each class in the Classes table. The Enrollments table contains the foreign keys Student ID and Class ID.) [Ref]

Degree of Relationships

It represents **the number of entity types** that associate in a relationship.

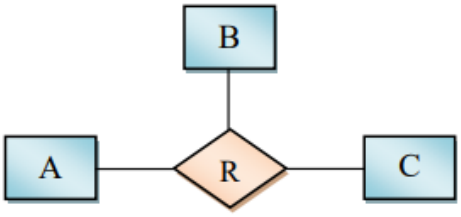
Types:

1. **Unary :** when both the participating entity type are the same, degree: 1
2. **Binary :** when exactly two entity type participates, degree: 2
3. **Ternary:** when exactly three entity type participates, degree: 3
4. **N-ary:** when exactly *n* entity type participates, degree: *n*

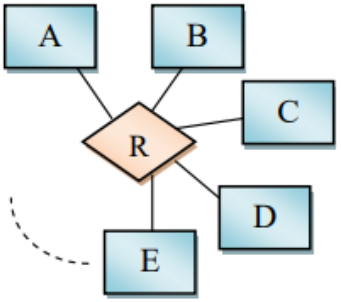


1. **Unary Relationship Set**

2. **Binary Relationship Set**



3. **Ternary Relationship Set**



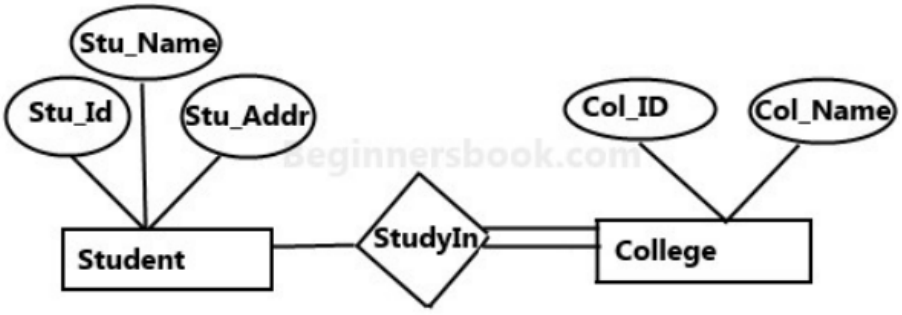
4. **n-ary Relationship Set**

Participation Constraints

Participation constraints define the **least number of relationship instances** in which an entity must compulsorily participate.

Types:

1. **Total/Mandatory Participation:** each entity in the entity set must compulsorily participate in at least one relationship instance. It represents by double lines, minimum cardinality 1.
2. **Partial participation:** each entity in the entity set may or may not participate in the relationship instance in that relationship set. It represents by single line, cardinality 0.



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

Relational Model

Integrity Constraints

Integrity constraints are a set of rules defined in a database to maintain **data accuracy, consistency and reliability**.

Types of Integrity Constraints

- **Domain Constraint** : This constraint is related to attributes and ensures the data entered into a column satisfies the rules defined for that column. Example: age column can never accept negative integers.
- **Entity Integrity Constraint** : A table should have at least one primary key which will uniquely identify each row, a primary key cannot be null.
- **Relational Integrity Constraint:** This is totally related to foreign key. This integrity is applied to establish relationship between tables in a database. two tables are related to each other through foreign key. the primary key of one table serves as the foreign key for its related table.
- **Key Constraint :** A key constraint in a Database Management System (DBMS) refers to a set of rules applied to one or more columns in a

database table to ensure the uniqueness and integrity of data. Keys are used to uniquely identify rows in a table, and they play a fundamental role in establishing relationships between tables.

Normalization

- Normalization is a database design technique which organizes tables in a manner that **reduces redundancy and dependency of data**
- It divides larger tables to smaller tables and links them using relationships.

Anomaly

Anomalies in the relational model refer to **inconsistencies or errors that can arise when working with relational databases.**

Types of Anomalies:

[The table has four columns: *e_id (employee's id)*, *e_name (employee's name)*, *e_address (employee's address)*, *and e_dept (employee's department)*.
Suppose the table has not been normalized and has redundant data.]

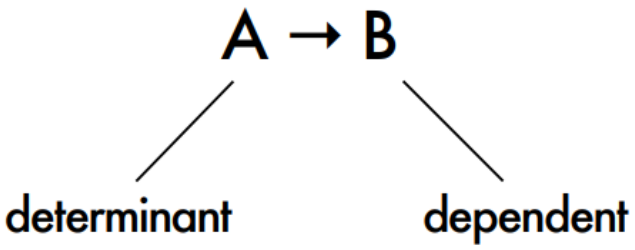
1. **Insert Anomaly:**
    a. Insert anomalies occur when certain attributes cannot be inserted into the database due to missing additional data.
    b. For example, if a new employee is not assigned a department, their data cannot be inserted into the table if the department field does not allow null values.
2. **Update Anomaly:**
    a. Update anomaly is something when we are trying to update some records in table, and that update is causing data inconsistency.
    b. For example, if an employee's address changes and the update is made in one row but not in others, the database will contain inconsistent data
3. **Delete anomaly**

a. Delete anomaly is something when we delete some data from the table, and due to that delete operation we loss some other useful data.

b. For example, if a department is shut down and all rows containing that department are deleted, the data of employees working solely in that department will also be deleted.

Prerequisite for Normal forms:

- **Functional Dependency :** relationship between attributes that if we are given the value of one of the attributes we can look up the value of the other. Example : ***s_id → s_name****, s_id* determines *s_name.*
  - Functional dependency describes relationship between attributes. For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted A → B), if each value of A in R is associated with exactly one value of B in R



  - **Properties of FD:**
    - **Reflexivity**: If *Y* is a subset of *X*, then *X* → *Y*
    - **Augmentation**: If *X* → *Y*, then *XZ* → *YZ*
    - **Transitivity**: If *X* → *Y* and *Y* → *Z*, then *X* → *Z*
    - **Decomposition: If** A→BC holds, then A→B holds and A→C holds.
    - **Union rule:** If A→B holds and A→C then A→BC holds
    - **Pseudo transitivity rule:** If A→B holds and BC→D holds, then AC→D holds.
  - **Types:**
    - **Trivial :** This type occurs when a set of attributes is functionally dependent on a subset of itself. Example : {name, roll} → {name}

- **Non-Trivial**: This type occurs when a set of attributes is functionally dependent on another set of attributes that is not a subset of itself. Example: {roll} → {name}
- **Multivalued:** In Multivalued functional dependency, entities of the dependent set are not dependent on each other. Example : a → {b, c} and there exists no functional dependency between b and c.
- **Transitive:** If *X* → *Y* and *Y* → *Z*, then *X* → *Z*
- **Partial Dependency:**
  - Partial dependency in a relational database occurs when a non-prime attribute (i.e., not part of any candidate key) is functionally dependent on only a part of the candidate key, rather than the entire candidate key. [Ref]
  - Given a relation dependencies F defined on the attributes of R and K as a candidate key ,if X is a proper subset of K and if F|= X→A, then A is said to be partial dependent on K

  [Ref]

- **Closure Method**
  - Helps to find all candidate keys of a table

  $R(ABCD)$

  FD: $\{A \to B, B \to C, C \to D\}$

  Closure of A, $A^+ = \{ABCD\}$, $B^+ = BCD, C^+ = CD, D^+ = D$

  Explanation of A^+ : A can determines itself. A determines B so B is here. B determines C and A determines B, by transitive property A determines C too. By the same way, A can determine D.

  Hence, A can determines all attributes, so it is a Candidate Key.

  $CK = A$

  $\text{Prime attribute} = A$, which is used in making of the candidate key.

  $\text{Non-prime} = B, C, D$, which is not used in making of the candidate key.

  $(AB)^+ = ABCD$, super key.

  $FD = \{A \to B, B \to C, C \to D, D \to A, \}$

$A^+ = B^+ = C^+ = D^+ = ABCD$, so all of them are candidate key.

$CK = A, B, C, D$

$\text{Prime attribute (PA)} = A, B, C, D$

$\text{Non-prime attribute (NPA)} = \{\}$

How to find CK ?

- Find the attribute which are absent in the right side.

$R(ABCDE)$

$FD = \{A \to B, BC \to D, E \to C, D \to A\}$

E is absent in right side. So, each CK must contain E.

$E^+ = EC$, so it can't determines all attribute. We have to add another attribute from left side.

$(AE)^+ = ABECD$, is a CK. Now we can check BE, CE, DE etc. But, there is another easy way to find it out, just replace the A with X when $X \to A$. Do it recursively.

So, $CK = \{AE, DE, BE\}$

- **Decompositions**
  - A decomposition of a relation schema R consists of replacing the relation schema by two (or more) relation schemas that each contain a subset of the attributes of R and together include all attributes in R.
  - **Types**
    - **Lossless-join Decomposition**
      - A decomposition of a relation scheme $R < S, F >$ into the relation schemes $R_i (1 <= i <= n)$ is said to be a lossless join decomposition or simply lossless if for every relation R that satisfies the FDs in F, the natural join of the projections or R_i gives the original relation R, i.e,

        $R = \prod_{R1}(R) \prod_{R2}(R) \ldots\ldots\ldots \prod_{Rn}(R)$

To be lossless, $R_1, ...R_n$ should have a common attribute which is a CK or Super Key of either $R_1, or ...R_n$ or both.

Example : Let R(A,B,C) AND F={A→B}. Then the decomposition of R into R1(A,B) and R2(A,C) is lossless because the FD { A→B} is contained in R1 and the common attribute A is a key of R1.

- **Lossy Decomposition**
  - if R is a subset of $\prod_{R1}(R) \prod_{R2}(R) \ldots\ldots\ldots \prod_{Rn}(R)$

  The common attribute of $R_1, ...R_n$ is not a CK, Super Key.

  Example: Let R(A,B,C) AND F={A→B}. Then the decomposition of R into R1(A,B) and R2(B,C) is not lossless because the common attribute B does not functionally determine either A or C. i.e, it is not a key of R1 or R 2.

- **Dependency Preserving Decomposition:**
  - If we decompose R into $R_1, R_2, ...R_n$, then

  $FD_1\ U\ FD_2 ...\ U\ FD_n = FD^+$, then dependency is preserved.

Normal forms

Normal forms are series of guidelines that help to ensure that the design of a database is **efficient, organized and free from data anomalies.**

1. **First Normal Form:**
   a. The value of each **attribute is atomic.**
   b. Each record needs to be unique
   c. Each column must have a unique name
   d. Attribute domain should be same

student	courses
Jane Smith	Databases, Mathematics
John Lipinsky	English Literature, Databases
Dave Beyer	English Literature, Mathematics

Not in First normal form

student	course
Jane Smith	Databases
Jane Smith	Mathematics
John Lipinsky	English Literature
John Lipinsky	Databases
Dave Beyer	English Literature
Dave Beyer	Mathematics

In First normal form

- **Second Normal Form**
  a. Table should be in 1NF first.
  b. Partial dependency is not allowed = All the non-prime attributes should be fully functional dependent on CK =
  $\text{NOT } (PartOfCK(\text{proper subset of CK}) \rightarrow NPA)$

Courses

Course Title	Fee	Qualification	Teacher ID	Teacher Name
Computer Science	£2000	Advanced Level	1	Miss Lovelace
Mathematics	£2500	Advanced Level	2	Mr Pascal
Physics	£1800	Advanced Level	3	Mr Einstein
Chemistry	£1800	Advanced Level	4	Mr Bunsen
Music	£1200	Diploma	5	Miss Holiday
Biology	£1000	Certificate	6	Mr Darwin
Economics	£1500	Diploma	7	Mr Keynes

Course Title → Teacher ID → Teacher Name
Course Title → Teacher Name → Teacher ID

Courses

Course Title	Fee	Qualification	Teacher
Computer Science	£2000	Advanced Level	1
Mathematics	£2500	Advanced Level	2
Physics	£1800	Advanced Level	3
Chemistry	£1800	Advanced Level	4
Music	£1200	Diploma	5
Biology	£1000	Certificate	6
Economics	£1500	Diploma	7

Teachers

Teacher ID	Teacher Name
1	Miss Lovelace
2	Mr Pascal
3	Mr Einstein
4	Mr Bunsen
5	Miss Holiday
6	Mr Darwin
7	Mr Keynes

$R(ABCDEF)$

$FD = \{C \rightarrow F, E \rightarrow A, EC \rightarrow D, A \rightarrow B\}$

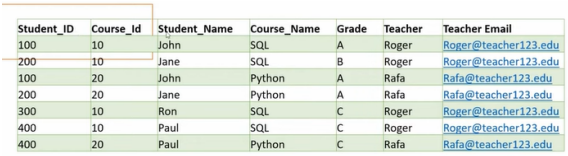
$EC^{+} = ABCDEF$, why EC ? because they are absent in right side.

$CK = EC,$
$NPA = A, B, D, F,$

$PA = E, C$

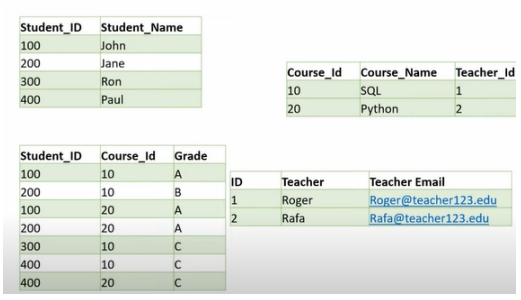
Now, $C \rightarrow F, E \rightarrow A$ where a part of CK determines a NPA. So, there are partial dependency.

- **Third Normal Form**
  - The table should be 2NF
  - There should be no transitive dependency in table = NO $NPA \rightarrow NPA$ in functional dependency = $A \rightarrow B$, A candidate key or B PA



Student_ID	Course_Id	Student_Name	Course_Name	Grade	Teacher	Teacher Email
100	10	John	SQL	A	Roger	Roger@teacher123.edu
200	10	Jane	SQL	B	Roger	Roger@teacher123.edu
100	20	John	Python	A	Rafa	Rafa@teacher123.edu
200	20	Jane	Python	A	Rafa	Rafa@teacher123.edu
300	10	Ron	SQL	C	Roger	Roger@teacher123.edu
400	10	Paul	SQL	C	Roger	Roger@teacher123.edu
400	20	Paul	Python	C	Rafa	Rafa@teacher123.edu

[Ref]



Student_ID	Student_Name
100	John
200	Jane
300	Ron
400	Paul

Course_Id	Course_Name	Teacher_Id
10	SQL	1
20	Python	2

Student_ID	Course_Id	Grade
100	10	A
200	10	B
100	20	A
200	20	A
300	10	C
400	10	C
400	20	C

ID	Teacher	Teacher Email
1	Roger	Roger@teacher123.edu
2	Rafa	Rafa@teacher123.edu

$FD = AB \rightarrow C, C \rightarrow D$

$CK = AB$

$PA = A, B$

$NPA = C, D$

This is in 2nd NF. Hence, $C \rightarrow D$, is $NPA \rightarrow NPA$ so there is transitive dependency. It is not in 3rd NF.

$FD = AB \rightarrow CD, D \rightarrow A$

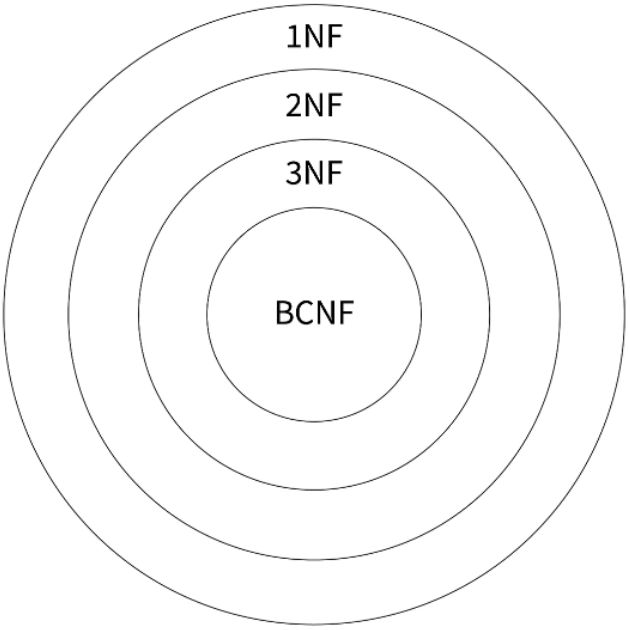
$CK = AB$

$PA = A, B$

$NPA = C, D$

This is in 2nd NF . Hence, there is no $NPA \rightarrow NPA$, = No transitive dependency. So, it is in 3rd NF.

- **BCNF/Boyce Codd Normal Form/3.5 NF**
  - Table should be in 3NF
  - For any function dependencies, $A \rightarrow B$, A must be a super key



Sr. No.	3NF	BCNF
1.	3NF stands for third normal form.	BCNF stands for Boyce Codd normal form.
2.	The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold : (i) X is a superkey, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
3.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
4.	Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to obtain in BCNF.
5.	3NF can be achieved without loosing any information from the old table.	For obtaining BCNF we may loose some information from old table.

- **Fourth Normal Form**
  - Table should be in BCNF
  - There mustn’t multi-value dependency

  Multi-valued Dependency

- Table should have at least 3 columns
- For a FD A→ B, if for a single value of A, multiple values of B exists
- For a relation R(A, B, C), if there is multi-valued dependency between A and B , A and C, then B and C should be independent of each other.

sid	Course	Skill
1	C C++	English German
2	Java	English French

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Here sid and course are dependent but the Course and Skill are independent. The multivalued dependency is denoted as :

sid→ Course

sid→ Skill

Convert it to 4 NF:

sid	Course
1	C
1	C++
2	Java

sid	Skill
1	English
1	German
2	English
2	French

- **Fifth Normal Form**
  - It should be in 4NF
  - it is a lossless decomposition

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Britania	Biscuits

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuit

Company_Product

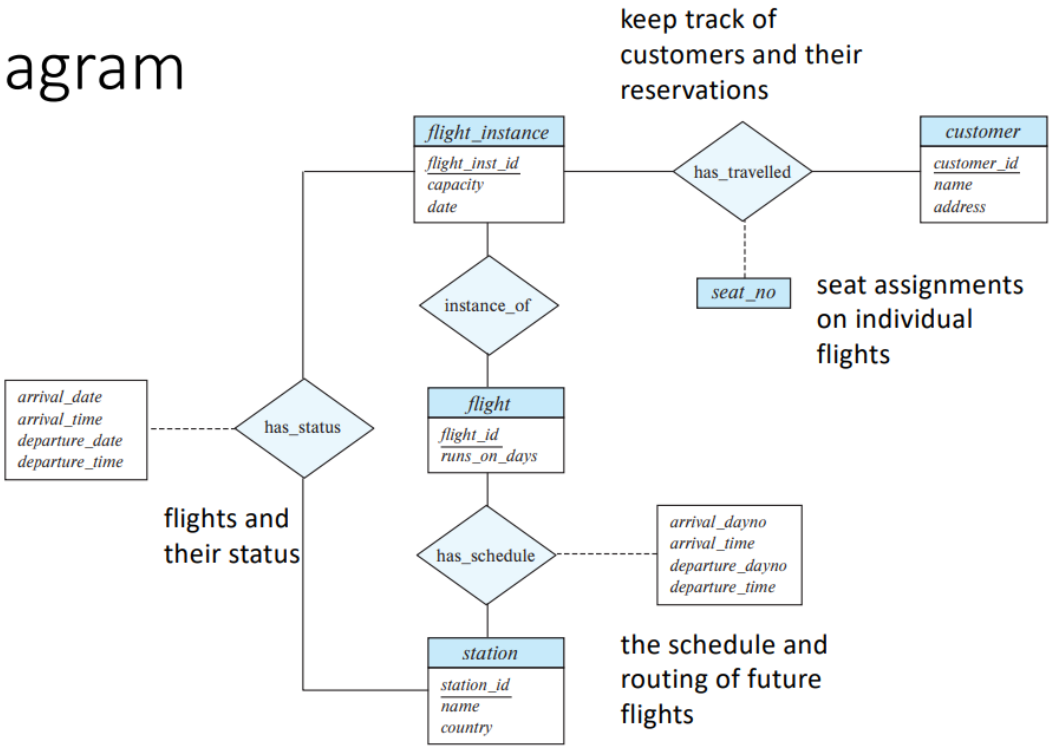
Company	Product
Godrej	Cinthol
Dabur	Honey
Dabur	HairOil
Dabur	RoseWater
Amul	Icecream
Britania	Biscuit

5.	a) What is the main purpose of an ER (Entity Relationship) diagram? Shortly explain the difference between a weak and a strong entity set.	2
	b) What is purpose of mapping cardinality? Also shortly explain the cardinalities of relationships in an ER diagram with illustrative example.	2
	c) Design a database for an airline. The database must keep track of customers and their reservations, flights and their status, seat assignments on individual flights, and the schedule and routing of future flights. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.	6
6.	a) A weak entity set can always be made into a strong entity set by adding to its attributes the primary-key attributes of its identifying entity set. Outline what sort of redundancy will result if we do so.	3
	b) What are the different steps involved in normalizing a database? Why do you need to use functional dependencies while normalizing databases?	4
	c) Given a relation R(A, B, C, D) and Functional Dependency set FD ={AB→CD, B →C}, determine whether the given R is in 2NF? If not convert it into 2NF.	3

(5)

(c)

ER Diagram



flight_instance(flight_inst_id, capacity, date)
customer(customer_id, customer_name, address)
flight(flight_id, runs_on_days)
airport(airport_id, name, country)
has_traveled(flight_inst_id, customer_id, seat_number,
foreign key *flight_inst_id* **references** *flight_instance,*
foreign key *customer_id* **references** *customer)*
instance_of(flight_inst_id, flight_id,
foreign key *flight_inst_id* **references** *flight_instance,*
foreign key *flight_id* **references** *flight)*
has_schedule(flight_id, airport_id, arrival_time, departure_time,
foreign key *flight_id* **references** *flight,*
foreign key *airport_id* **references** *airport)*
has_status(flight_inst_id, airport_id, arrival_time, departure_time
foreign key *flight_inst_id* **references** *flight_instance,*
foreign key *airport_id* **references** *airport)*

(6)

(a) A weak entity set can always be made into a strong entity set by adding to its attributes the primary-key attributes of its identifying entity set. Outline what sort of redundancy will result if we do so.

The redundancy resulting from this conversion leads to the unnecessary duplication of primary key attributes in both the weak entity set and the

relationship set.

The primary key of a weak entity set can be inferred from its relationship with the strong entity set. If we add primary key attributes to the weak entity set, they will be present in both the entity set and the relationship set and they have to be the same. Hence there will be redundancy.

(b) What are the different steps involved in normalizing a database? Why do you need to use functional dependencies while normalizing databases?

Normalization is a database design technique which organizes tables in a manner that **reduces redundancy and dependency of data.**

Step:

1. Unnormalized Form
2. 1NF
3. 2NF
4. 3NF
5. BCNF
6. 4NF
7. 5NF
8. 6NF

Why do we need to use functional dependency

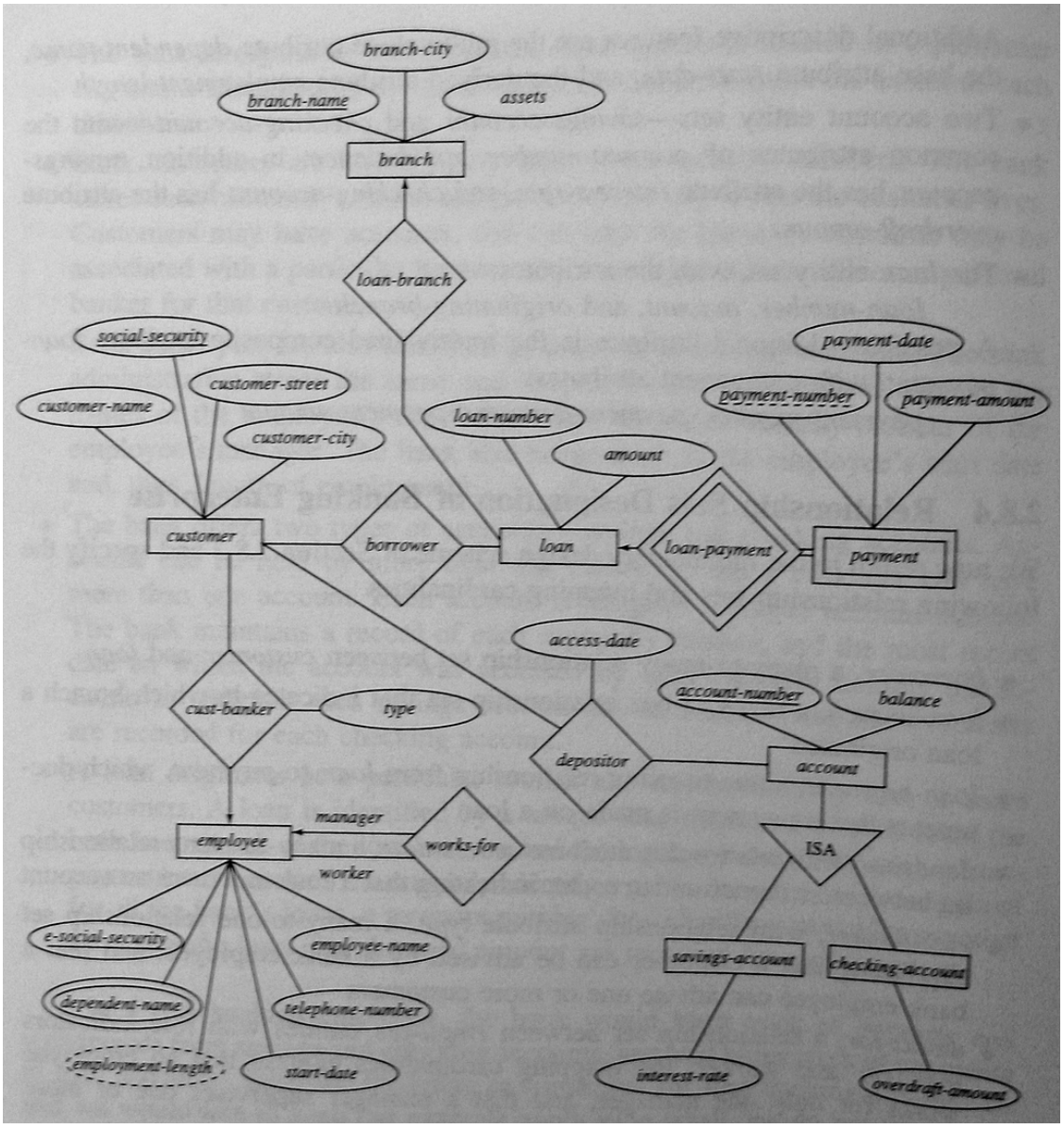
- Eliminating Redundancy: By analyzing FDs, we can break tables into smaller, related tables to reduce redundancy.
- Ensuring Data Integrity: When a functional dependency is violated, data integrity problems can occur.
- Detecting Partial Dependencies: 2NF
- Detecting Transitive Dependencies: 3NF
- Candidate Key Determination: All NF
- Achieving Higher Normal Forms: To check the highest NF of a relation

(c)

4.a) Read the following simple banking scenario. Now your task is to draw the database ER diagram with the following: 5

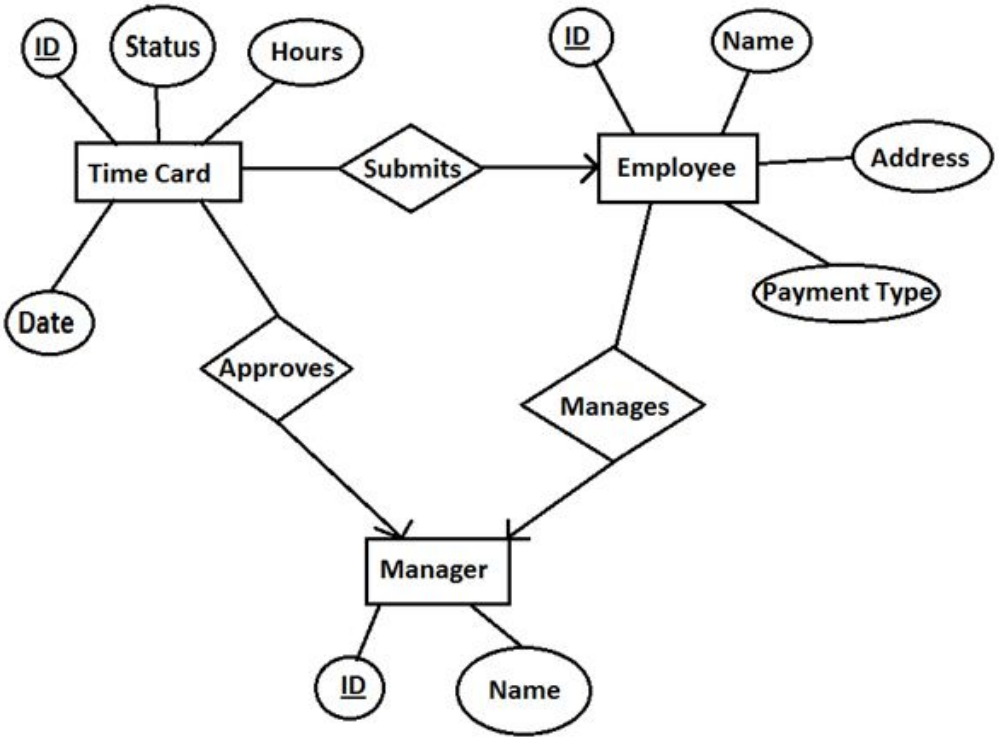
i. Customer has unique customer_id, name, street and city.
ii. A customer may have accounts and can take out loans.
iii. Account has unique account_number, balance. Accounts can be held by more than one customer, and a customer can have more than one account. The bank maintains the most recent date on which the account was accessed by each customer holding the account.
iv. The bank offers two types of accounts- savings and checking accounts. Each savings account has an interest rate and overdrafts are recorded for each checking account.
v. Loan is identified by unique loan_number, amount.
vi. Loan Payment includes payment_number, payment_date, payemnt_amount which is associated with loan. payment_number does not uniquely identify a particular payment among those for all the bank's loan.

5.a) Why is database normalization so important? Differentiate 1NF, 2NF, 3NF and BCNF with proper example. 5



Class Test: 2, Session: 2016-2017, Set: B

1	Why mapping cardinality is used? Explain cardinality constrains with example.	5
2	The company you work for wants to digitize their time cards. You have been asked to design the database for submitting and approving time cards. Draw the database ER diagram and also convert it to relational schema with the following i. A timecard should have hours worked and date submitted ii. Each timecard is associated with exactly one employee iii. Each timecard should have a unique id iv. Each timecard has a status: it is either approved, not approved, or pending v. Each employee has a unique id vi. Each employee has a name and address. vii. Each employee submits a time card every pay period. i.e. In 1 year, they will submit multiple time cards viii. Each employee either has direct deposit or physical check as their method of payment ix. Each employee is associated with exactly one manager x. Each manager has a unique id and a name xi. Each manager is in charge of multiple employees xii. Each manager approves time cards for multiple employees If you feel that you must make some assumptions, please state them clearly so that they are easily understood by the graders. Remember to indicate the key for each entity, as well as the multiplicity of each relationship (e.g. one-to-many) using the appropriate notation.	15
3	Write the conditions of 1NF, 2NF, 3NF and BCNF.	5



- Each apartment has one or more units of different sizes.
- Each unit has a number unique within each apartment, and the unit can be leased by one or more tenants. Each tenant is not allowed to rent more than one unit in an apartment for availability reasons.
- Each unit has the number of rooms, the number of bathrooms, and the total area in square feet.
- Tenant's social security number, name and birthday must be recorded in the database.
- Leasing contracts are identified by their contract ID and the rental rate. Each contract consists of one unit and at least one tenant. We do not consider leasing parking spaces to simplify the problem.
- There are two kinds of leasing contracts: long term and monthly. They need to be separated because they will be treated differently. Long term lease will have specific start and end dates, and short term lease will only have a start date.

Design E/R diagram & Convert to a relational database schema.

- A property management company has a unique name and a contact phone number, and may manage one or more apartments.
- An apartment has a unique address, the number of units, a leasing office, and one or more parking spaces. There is exactly one leasing office per one apartment.
- Each leasing office has a unique phone number and the number of staff.
- Parking space has a unique number and information on whether it is covered or not. An apartment has many parking spaces, and each parking space cannot be shared between different apartments even if they are close by, and each parking space number is unique for each apartment.
- Each utility company (electricity) is identified by a unique business registration number, has a name, and an address. Each electricity provider may supply electricity to multiple apartments, and an apartment has one supplier.

Design E/R diagram & Convert to a relational database schema.