Assignment

# Assignment	1
Field	CS
Galala ID	223106831
Instructor	Shaker EL-Sappagh
Student Name	Mahros AL-Qabasy
Subject	Database Systems (CSE 221)

1. Consider the ER diagram in Figure 3.21, which shows a simplified schema for an airline reservations system. Extract from the ER diagram the requirements and constraints that produced this schema. Try to be as precise as possible in your requirements and constraints specification.

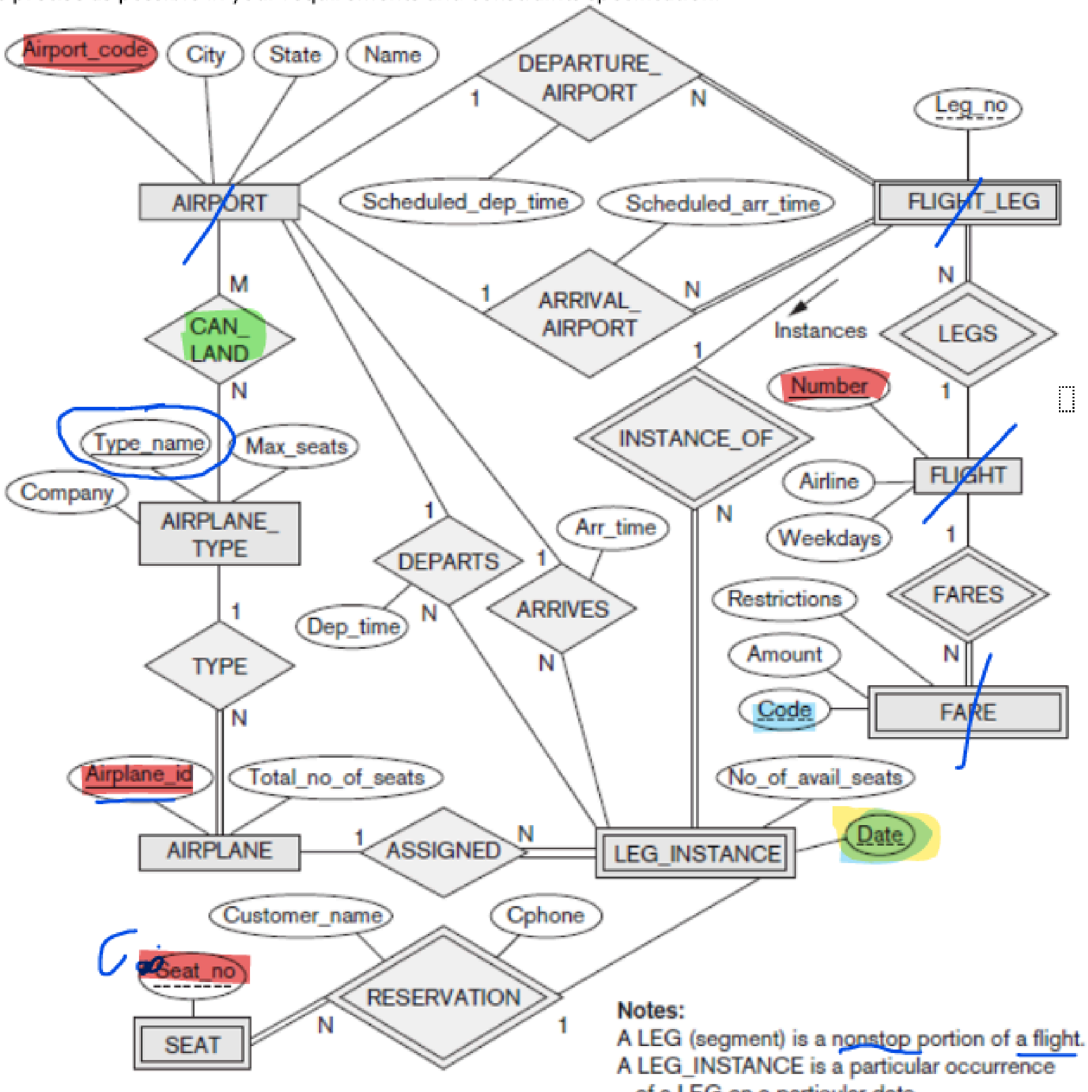


Figure 3.21
An ER diagram for an AIRLINE database schema.

Entities

1. <AIRPORT>

- **Attributes**
    - Airport_Code → **PrimaryKey**
    - City → *PlainText*

- `State` → *PlainText*
- `Name` → *PlainText*

- **Requirement**
  - Each `<AIRPORT>` is uniquely identified by its `Airport_Code`.
  - An `<AIRPORT>` can serve as a departure or arrival point for many `<LEG_INSTANCE>` occurrences.

---

2. `<AIRPLANE_TYPE>`

- **Attributes**
  - `Type_Name` → **PrimaryKey**
  - `Max_Seats` → *Integer*
  - `Company` → *PlainText*
- **Requirement**
  - Each `<AIRPLANE_TYPE>` entry describes a class of airplane, uniquely identified by `Type_Name`.
  - An `<AIRPLANE_TYPE>` can land at multiple airports (via the `<CAN_LAND>` relationship).

---

3. `<AIRPLANE>`

- **Attributes**
  - `Airplane_ID` → **PrimaryKey**
  - `Total_no_of_seats` → *Integer*
- **Requirement**
  - Each `<AIRPLANE>` references exactly one `<AIRPLANE_TYPE>` (via the `<TYPE>` relationship).
  - `Total_no_of_seats` should not exceed the `Max_Seats` defined by the associated `<AIRPLANE_TYPE>`.

---

4. `<FLIGHT>`

- **Attributes**

- `Number` → **PrimaryKey**
- `Airline` → *PlainText*
- `Weekdays` → *List of Days*
- `Restrictions` → *PlainText*
- `Amount` → *Decimal*
- `Code` → *PlainText*

- **Requirement**
  - Each `<FLIGHT>` is uniquely identified by `Number`.
  - A `<FLIGHT>` can be composed of multiple legs (`<FLIGHT_LEG>`) but must have at least one.

---

5. `<FLIGHT_LEG>`

- **Attributes**
  - `Leg_no` → **PrimaryKey**
- **Requirement**
  - A `<FLIGHT_LEG>` represents a *nonstop segment* of a flight.
  - Each `<FLIGHT_LEG>` belongs to exactly one `<FLIGHT>` (via the `<LEGS>` relationship).

---

6. `<LEG_INSTANCE>`

- **Attributes**
  - *(Leg_no, Date)* → **Composite PrimaryKey** (or an internal ID)
  - `No_of_avail_seats` → *Integer*
  - `Dep_time` → *Time*
  - `Arr_time` → *Time*
- **Requirement**
  - A `<LEG_INSTANCE>` is a particular occurrence of a `<FLIGHT_LEG>` on a specific date.

- Each `<LEG_INSTANCE>` must have exactly one departure `<AIRPORT>` and one arrival `<AIRPORT>`.
- `No_of_avail_seats` tracks how many seats remain for that specific date.

---

7. `<FARE>`

- **Attributes**
  - `Code` → Normal Text
  - `Amount` → *Decimal*
  - `Restrictions` → *Text*
- **Requirement**
  - Each `<FARE>` must be associated with exactly one `<FLIGHT>` (via the `<FARES>` relationship).
  - The **double line** in the diagram indicates *total participation*: a `<FARE>` cannot exist independently of a `<FLIGHT>`.

---

8. `<SEAT>`

- **Attributes**
  - `Seat_no` → *PlainText*
- **Requirement**
  - Each `<SEAT>` record identifies a physical seat that can be assigned to a reservation.
  - Typically, seats are linked to `<AIRPLANE>` or to `<LEG_INSTANCE>` through an assignment or reservation relationship.

---

9. `<RESERVATION>`

- **Attributes**
  - `Customer_name` → *PlainText*
  - `Cphone` → *PlainText*
- **Requirement**

- A `<RESERVATION>` indicates that a passenger (identified by name and phone) has booked one or more seats on specific `<LEG_INSTANCE>` dates.
- The actual **primary key** might be `Reservation_ID` (not shown explicitly), or a composite including `Customer_name` + `Cphone` if assumed unique.

Relationships

1. `<CAN_LAND>`

- **Entities Involved:** `<AIRPORT>` ↔ `<AIRPLANE_TYPE>`
- **Cardinalities**: Many to Many
- **Requirements**:
  - An `<AIRPORT>` can accommodate multiple `<AIRPLANE_TYPE>` objects.
  - An `<AIRPLANE_TYPE>` can be compatible with multiple `<AIRPORT>` locations.

---

2. `<TYPE>`

- **Entities Involved:** `<AIRPLANE_TYPE>` ↔ `<AIRPLANE>`
- **Cardinalities**: 1:N —one `<AIRPLANE_TYPE>` can have many `<AIRPLANE>` instances
- **Requirements**:
  - Each `<AIRPLANE>` is of exactly one `<AIRPLANE_TYPE>`.
  - `Total_no_of_seats` in `<AIRPLANE>` must be ≤ `Max_Seats` in `<AIRPLANE_TYPE>`.

---

3. `<LEGS>`

- **Entities Involved:** `<FLIGHT>` ↔ `<FLIGHT_LEG>`
- **Cardinalities**: 1:N one `<FLIGHT>` has many `<FLIGHT_LEG>` segments
- **Requirements**:
  - Each `<FLIGHT_LEG>` belongs to a single `<FLIGHT>`.
  - A `<FLIGHT>` with multiple stops has multiple `<FLIGHT_LEG>` entries.

---

4. `<INSTANCE_OF>`

- **Entities Involved:** `<FLIGHT_LEG>` ↔ `<LEG_INSTANCE>`

- **Cardinalities**: 1:N one `<FLIGHT_LEG>` can have many `<LEG_INSTANCE>` over different dates, `i don't know why!!?`
- **Requirements**:
  - Each `<LEG_INSTANCE>` is a date-specific occurrence of a `<FLIGHT_LEG>`.
  - Must store departure time, arrival time, and number of available seats for that date.

---

5. `<DEPARTS>` and `<ARRIVES>`

- **Entities Involved:** `<LEG_INSTANCE>` ↔ `<AIRPORT>`
- **Cardinalities**: 1:1 on `<LEG_INSTANCE>` side, M:1 on
- **Requirements**:
  - Each `<LEG_INSTANCE>` has exactly one departure `<AIRPORT>` and one arrival `<AIRPORT>`.
  - An `<AIRPORT>` can serve as a departure or arrival point for many `<LEG_INSTANCE>` entries.

---

6. `<FARES>`

- **Entities Involved:** `<FLIGHT>` ↔ `<FARE>`
- **Cardinalities**: 1:N (one `<FLIGHT>` has many `<FARE>` options)
- **Requirements**:
  - Each `<FARE>` belongs to exactly one `<FLIGHT>`.
  - The double line to `<FARE>` indicates *mandatory* participation: no `<FARE>` can exist without its parent `<FLIGHT>`.

---

7. `<ASSIGNED>`

- **Entities Involved:** `<SEAT>`, `<LEG_INSTANCE>`, `<RESERVATION>`
- **Cardinalities**: This can be modeled as a ternary relationship or multiple binary relationships.
- **Requirements**:
  - A seat on a specific `<LEG_INSTANCE>` is allocated to a particular `<RESERVATION>`.

Q2:

2. Design an ER schema for keeping track of information about votes taken in the U.S. House of Representatives during the current two-year congressional session. The database needs to keep track of each U.S. STATE's Name (e.g., 'Texas', 'New York', 'California') and include the Region of the state (whose domain is {'Northeast', 'Midwest', 'Southeast', 'Southwest', 'West'}). Each CONGRESS_PERSON in the House of Representatives is described by his or her Name, plus the District represented, the Start_date when the congressperson was first elected, and the political Party to which he or she belongs (whose domain is {'Republican', 'Democrat', 'Independent', 'Other'}). The database keeps track of each BILL (i.e., proposed law), including the Bill_name, the Date_of_vote on the bill, whether the bill Passed_or_failed (whose domain is {'Yes', 'No'}), and the Sponsor (the congressperson(s) who sponsored—that is, proposed—the bill). The database also keeps track of how each congressperson voted on each bill (domain of Vote attribute is {'Yes', 'No', 'Abstain', 'Absent'}). Draw an ER schema diagram for this application. State clearly any assumptions you make.

Entities

1. STATE

- **Attributes**
  - ID → **Primary Key**
  - Name → Text

1. REGION

- **Attributes**
  - ID → **Primary Key**
  - Name → *Text*

2. CONGRESS_PERSON

- **Attributes**
  - ID → **Primary Key**
  - Name → *Text*

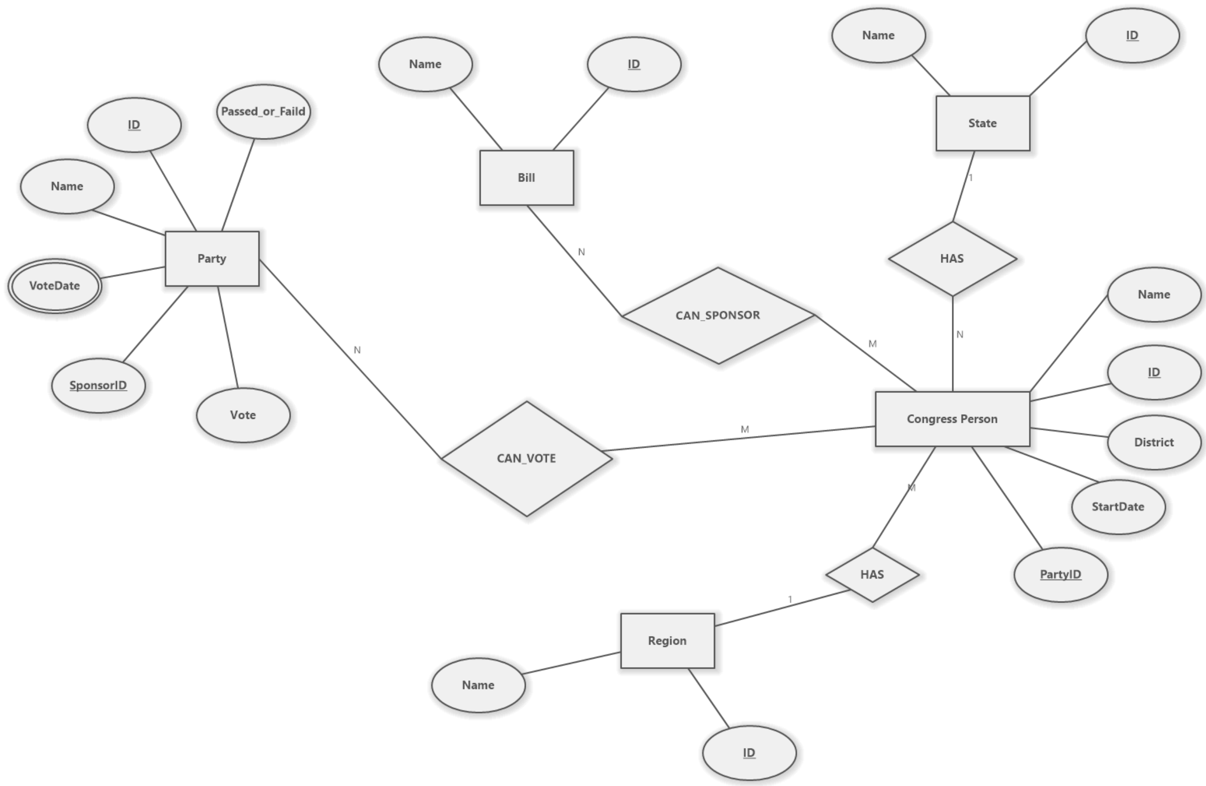
- `District` → Integer
- `StartDate` → Date
- `Party_ID` → Integer

3. `PARTY`

- **Attributes**
  - `ID` → Primary Key
  - `Domain` → Text

4. `BILL`

- **Attributes**
  - `ID` → Primary Key
  - `Bill_Name` → Text
  - `Date_of_Vote` → Date
  - `Passed_or_Faild` → Bool
  - `Sponsor_ID` → Congress Person Integer
  - `Vote` → Multi Values Attribute

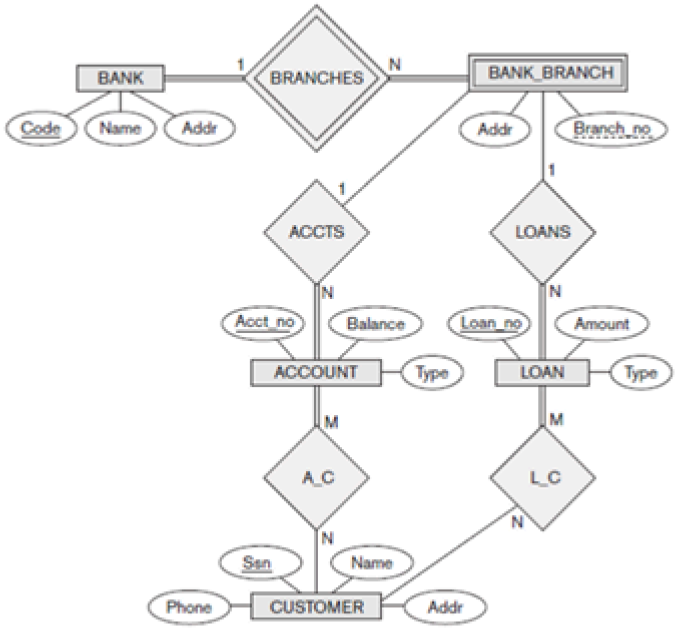

Name
ID
Name
ID
State
ID
Passed_or_Faild
Name
Bill
1
Party
HAS
Name
VoteDate
N
CAN_SPONSOR
N
N
M
ID
SponsorID
Vote
M
Congress Person
CAN_VOTE
District
M
StartDate
HAS
PartyID
1
Region
Name
ID

Q3

3. Consider the ER diagram shown in Figure 3.22 for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.

   a. List the strong (nonweak) entity types in the ER diagram.
   b. Is there a weak entity type? If so, give its name, partial key, and identifying relationship.
   c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?
   d. List the names of all relationship types, and specify the (min, max) constraint on each participation of an entity type in a relationship type. Justify your choices.

Figure 3.22
An ER diagram for a BANK database schema.



a. Strong Entities

- `Customer`
- `Bank`
- `Account`
- `Loan`

b. Week Entities

- `BankBranch`
    - Partial Participation with `Loans`
    - Has `One-to-Many` relationship with `Loan`

c. Partial keys constrains

- `BankBranch` has many `Loans` but `Loan` has only one `BankBranch`
- `Customer` has many `L_C`

d. Relationships

- `Branches`
    - Type: Week
    - Min, Max: 1 Bank
    - Min, Max: 1 Branch, In Finite Branches
-