# Database Systems

**Part 2:** Develop a logical data model based on the following requirements:

a. Derive relations from the conceptual model.

| | |
|---|---|
| **Client** (clientNo, clientfName, clientlName, clientAddress, clientTelNo)<br>**Primary Key:** clientNo | **Services** (serviceID, startDate, startTime, duration, comments, clientNo)<br>**Primary Key:** serviceID<br>**Foreign Key:** clientNo **references** Client (clientNo) |
| **Employee** (empNo, empfName, emplName, empAddress, salary, empTelNo)<br>**Primary Key:** empNo | **Equipment** (equipID, description, usage, cost)<br>**Primary Key:** equipID |
| **EquipmentUsage** (serviceID, equipID)<br>**Primary Key:** serviceID, equipID<br>**Foreign Key:** serviceID **references** Services (serviceID)<br>**Foreign Key:** equipID **references** Equipment (equipID) | **Hiring** (serviceID, empNo)<br>**Primary Key:** serviceID, empNo<br>**Foreign Key:** serviceID **references** Services (serviceID)<br>**Foreign Key:** empNo **references** Employee (empNo) |

b. Validate the logical model using normalization to 3NF.

***Functional Dependencies***:

***0NF to 1NF*** (Remove replicate groups)

There are no replicated groups in any relation.

***1NF to 2NF*** (Remove Partial Dependencies)

There are no partial dependencies in any relation.

***2NF to 3NF*** (Remove Transitive Dependencies)

There are no transitive dependencies in any relation.

***3NF*** (Consider Primary key and Candidate key dependencies)

**Client:** clientNo->clientfName, clientlName, clientAddress, clientTelNo (Primary Key)

Assumption: Each client has a unique telephone number and no client lives with another client.

clientTelNo->clientfName, clientlName, clientAddress (Candidate Key)

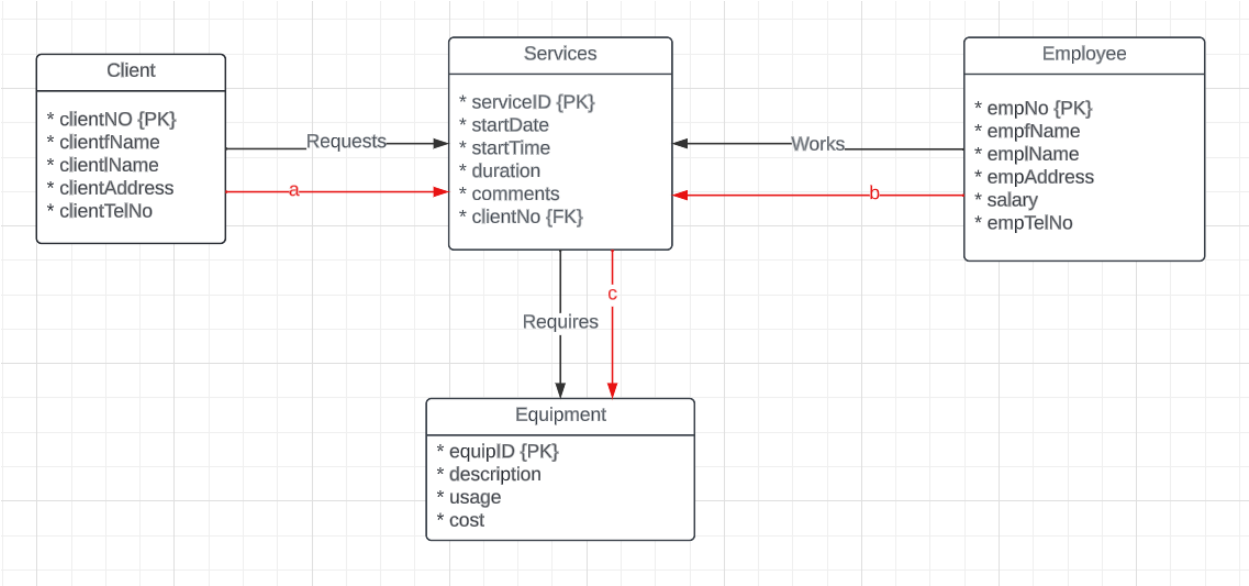
**Services:** serviceID->startDate, startTime, duration, comments, clientNo (Primary Key)

**Employee:** empNo->empfName, emplName, empAddress, salary, empTelNo (Primary Key)
Assumption: Each employee has a unique telephone number and no employee lives with another employee.
empTelNo->empfName, emplName, empAddress (Candidate key)

**Equipment:** equipID->description, usage, cost (Primary Key)

(Relations from Many to Many but will not be included in final logical model)
**EquipmentUsage:** (serviceID, equipID) (Primary Key)
**Hiring:** (serviceID, empNo) (Primary Key)

c. Validate the logical model against user transactions.




Transaction: Client requests a service: -> a
Transaction: Employee works for a service: -> b
Transaction: Service has a task that requires equipment: -> c
Transaction: Client uses a service and the service needs equipment to perform the task: -> a -> c
Transaction: Employee works for a service and the service needs equipment to perform the task: -> b -> c

Transaction: SELECT * FROM Services WHERE clientNo == 2;
(Find service attributes like startDate, startTime, etc. for the second client (assuming clientNo increments by one per new client)).
Transaction: SELECT a.empfName, a.emplName FROM Employee WHERE salary > 30,000;

(Find employees who work for Super Maids Cleaning Company whose salary is greater than 30,000).
Transaction: SELECT equipID FROM Equipment WHERE usage >= 10 AND cost < 5,000;
(Find the equipment identifier where it has been used at least 10 times or more and costs less than $5,000 to use)

d. Define integrity constraints:
   i. Primary key constraints.
   **Client:** clientNo (PK), constraint: Integer bigger than 0, UNIQUE, NotNull
   **Services:** serviceID (PK), constraint: Integer bigger than 0, UNIQUE, NotNull
   **Employee:** empNo (PK), constraint: Integer bigger than 0, UNIQUE, NotNull
   **Equipment:** equipID (PK), constraint: Integer bigger than 0, UNIQUE, NotNull
   ii. Referential integrity/Foreign key constraints.
   **Client:** No Foreign Key(s)
   **Services:** clientNo (FK), constraint: Integer bigger than 0, UNIQUE, must match clientNo in Client relation
   **Employee:** No Foreign Key(s)
   **Equipment:** No Foreign Key(s)
   iii. Alternate key constraints (if any).
   **Client:** clientTelNo (Alternate Key), constraint: VARCHAR, must not be NULL.
   Assumption: Each client has a unique telephone number/doesn't share with anyone else.
   **Services:** No Alternate Key
   **Employee:** empTelNo (Alternate Key), constraint: VARCHAR, must not be NULL.
   Assumption: Each employee has a unique telephone number/doesn't share with anyone else.
   **Equipment:** No Alternate Key
   iv. Required data.
   Assumptions: Clients and Employees do not necessarily have to have a telephone, they can be reached via their address (mail or in person).
   clientAddress
   - Cannot be null since this is a cleaning service and the employees need an address to go to.

   startDate
   - This is a scheduled start date that is filled out when the service is created. Cannot be null since the service must start at a certain date.

   startTime

- This is a scheduled start time that is filled out when the service is created. Cannot be null since the service must start at a certain time.

usage

- Usage is an integer from 0 to positive infinity that specifies how many times a piece of equipment has been used, so it is never null or negative.

Cost

- Cost is a dollar amount from $0.00 to an infinite positive dollar amount, so it is never null or negative.

empfName, emplName, empAddress

- Employees are hired by the service and therefore must provide this information. Cannot be null.

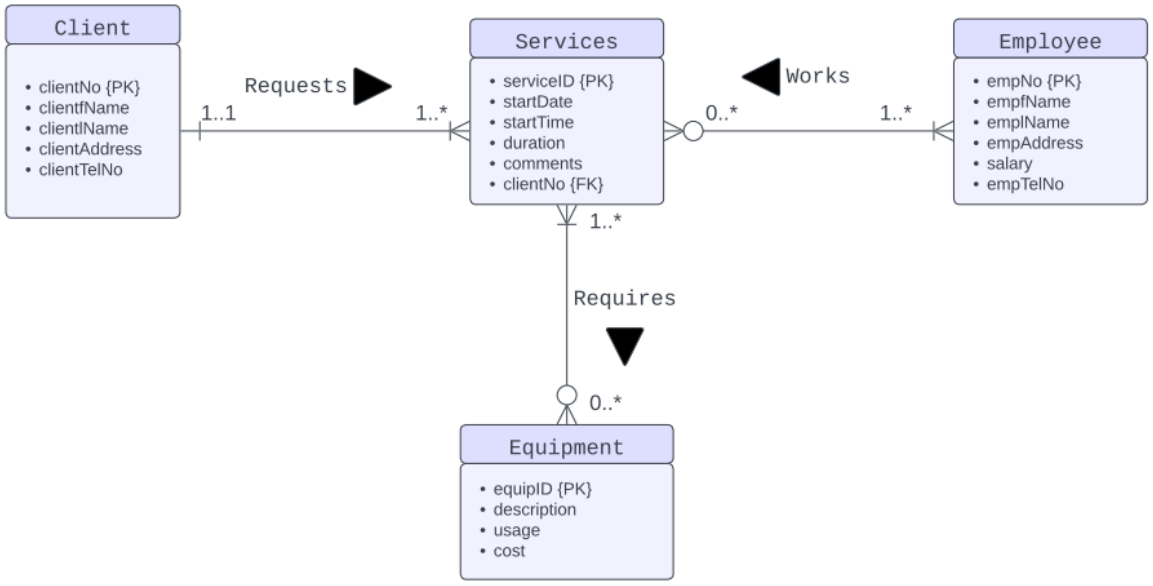
Salary

- Employees are paid a dollar amount so this cannot be null.

v. Attribute domain constraints.
**Client:** clientNo (Integer bigger than 0), clientfName (VARCHAR (50)), clientlName (VARCHAR (50)), clientAddress (VARCHAR (100)), clientTelNo (VARCHAR (11))
**Services:** serviceID (Integer bigger than 0), startDate (DATE), startTime (TIME), duration (TIME), comments (VARCHAR (100)), clientNo (Integer bigger than 0)
**Employee:** empNo (Integer bigger than 0), empfName (VARCHAR (50)), emplName (VARCHAR (50)), empAddress (VARCHAR (100)), salary (FLOAT), empTelNo (VARCHAR (11))
**Equipment:** equipID (Integer bigger than 0), description (VARCHAR (100)), usage (Integer bigger than 0), cost (FLOAT)

vi. General constraints (if any).

e. Generate the E-R diagram for the logical level (contain FKs as attributes).
Assumption: An employee can have a minimum of 0 services if they are a new employee with no services yet and a max of * services.


Client
• clientNo {PK}
• clientfName
• clientlName
• clientAddress
• clientTelNo
Requests
1..1
1..*
Services
• serviceID {PK}
• startDate
• startTime
• duration
• comments
• clientNo {FK}
Works
0..*
1..*
Employee
• empNo {PK}
• empfName
• emplName
• empAddress
• salary
• empTelNo
1..*
Requires
0..*
Equipment
• equipID {PK}
• description
• usage
• cost