

**Project: Design, development and implementation of a relational
Database - Part 3: Logical Data Model Translated for Oracle Enterprise DBMS**

Andrea A. Venti Fuentes, Jeremiah Moise

University of Miami

CSC 423 – Database Systems

Dr. Vanessa Aguiar

December 07, 2025

Case Study: SuperMaids Cleaning Company

The SuperMaids Cleaning Company specializes in providing cleaning services for clients. Each type of client has a set of requirements. For example, The Cardboard Box Company requires cleaning services from Monday to Friday 7am until 9am and 5pm until 7pm each day, but P. Nuttall only requires cleaning services on a Wednesday from 10am until 1pm. Each requirement will have a unique identifier and the following information will be stored: start date, start time, duration and comments.

Whenever a new client is taken on, it is determined whether any special equipment is required and when. For example, three industrial floor cleaners may be needed on two out of five occasions for one client. Therefore, the following information will be stored for each equipment, in addition to the equipment identifier: description, usage and cost. For each employee, the following data will be stored: staff number (uniquely identifies an employee), first and last name, address, salary and telephone number. For each client, the following data will be stored: client number (uniquely identifies a client), first and last name, address and telephone number.

Part 1

a. Identify the main entities.

- **CLIENT** - Customers who require cleaning services
- **EMPLOYEE** - Staff who provide cleaning services
- **REQUIREMENT** - The cleaning schedules for each client
- **EQUIPMENT** - Special equipment needed for cleaning

b. Identify the main relationships between the entities identified in "a".

- **HAS** - between CLIENT and REQUIREMENT
- **REQUIRES** - between REQUIREMENT and EQUIPMENT
- **ASSIGNED_TO** - between EMPLOYEE and REQUIREMENT

c. Determine the multiplicity constraints for each relationship identified in "b".

E1	M1	Relationship	M2	E2	Type of Relationship
CLIENT	1..1	HAS	1..*	REQUIREMENT	1:*
REQUIREMENT	0..*	REQUIRES	0..*	EQUIPMENT	*:*
EMPLOYEE	0..*	ASSIGNED_TO	1..*	REQUIREMENT	*:*

d. Identify attributes and associate them with entities or relationships.

Entity Attributes:

- CLIENT

- clientNo
- fName
- lName
- address
- telephoneNo

- EMPLOYEE

- employeeNo
- fName
- lName
- address
- salary
- telephoneNo

- REQUIREMENT

- requirementNo
- sDate
- sTime
- duration
- comments

- **EQUIPMENT**

- equipmentNo
- description
- usage
- cost

Relationship Attributes:

- **REQUIRES**

- quantity

e. Determine candidate and primary key attributes for each (strong) entity.

- **CLIENT:**

- Primary Key: clientNo
- Candidate Keys: clientNo

- **EMPLOYEE**

- Primary Key: employeeNo
- Candidate Keys: employeeNo

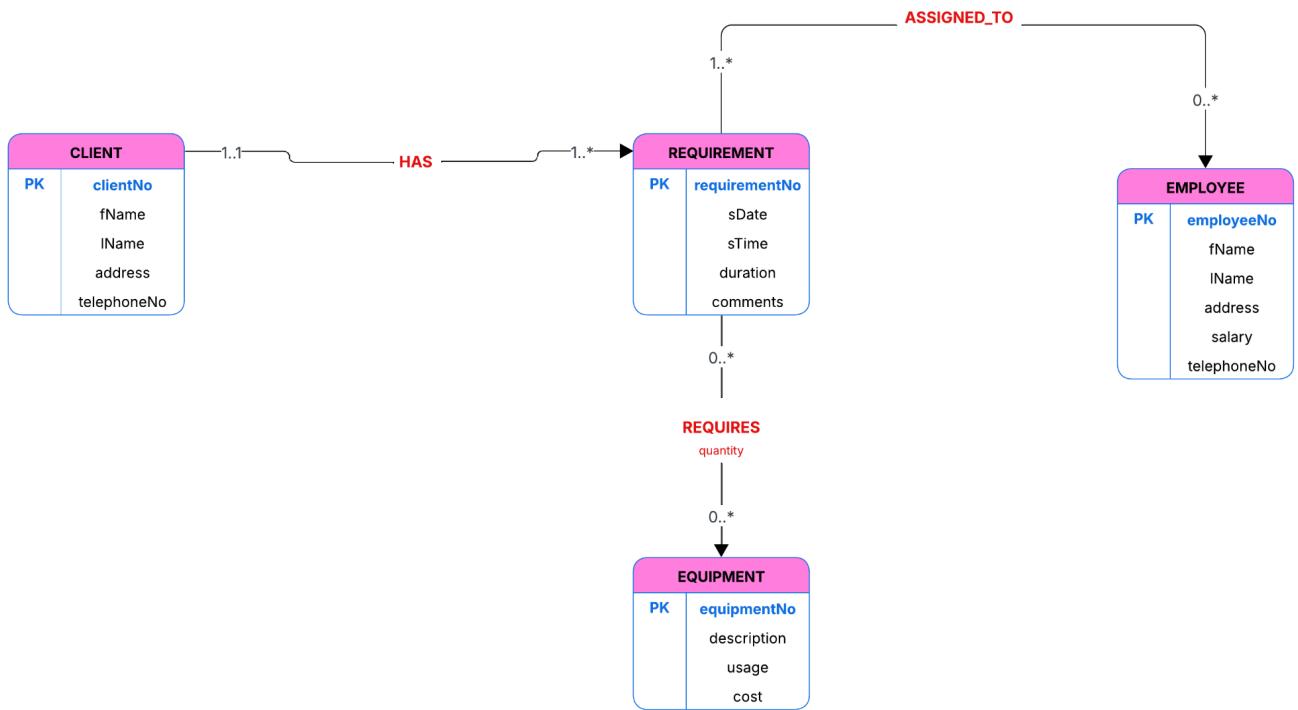
- **REQUIREMENT**

- Primary Key: requirementNo
- Candidate Keys: requirementNo

- **EQUIPMENT**

- Primary Key: equipmentNo
- Candidate Keys: equipmentNo

f. Generate the E-R diagram for the conceptual level (no FKs as attributes).



Part 2

a. Derive relations from the conceptual model.

CLIENT(
 clientNo PK,
 fName,
 lName,
 address,
 telephoneNo
)

EMPLOYEE(
 employeeNo PK,
 fName,
 lName,
 address,
 salary,
 telephoneNo
)

EQUIPMENT(
 equipmentNo PK,
 description,
 usage,
 cost
)

REQUIREMENT(
 requirementNo PK,
 clientNo FK → CLIENT(clientNo),
 sDate,
 sTime,
 duration,
 comments
)

REQUIRES(

 requirementNo FK → REQUIREMENT(requirementNo),
 equipmentNo FK → EQUIPMENT(equipmentNo),
 quantity,
 PK (requirementNo, equipmentNo)
)

ASSIGNED_TO(

 requirementNo FK → REQUIREMENT(requirementNo),
 employeeNo FK → EMPLOYEE(employeeNo),
 PK (requirementNo, employeeNo)
)

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

CLIENT

Functional Dependencies:

`clientNo → fName, lName, address, telephoneNo`

- All attributes are atomic: **1NF**
- PK is a single attribute ('clientNo'); no partial dependencies: **2NF**
- Atomic attributes, single-attribute PK, no transitive dependencies: **3NF**

EMPLOYEE

Functional Dependencies:

`employeeNo → fName, lName, address, salary, telephoneNo`

- All attributes are atomic: **1NF**
- PK is a single attribute ('employeeNo'); no partial dependencies: **2NF**
- Atomic attributes, single-attribute PK, no transitive dependencies: **3NF**

EQUIPMENT

Functional Dependencies:

`equipmentNo → description, usage, cost`

- All attributes are atomic: **1NF**
- PK is a single attribute ('equipmentNo'); no partial dependencies: **2NF**
- Atomic attributes, single-attribute PK, no transitive dependencies: **3NF**

REQUIREMENT

Functional Dependencies:

`requirementNo → clientNo, sDate, sTime, duration, comments`

- All attributes are atomic: **1NF**
- PK is a single attribute ('requirementNo'); no partial dependencies: **2NF**
- Atomic attributes, single-attribute PK, no transitive dependencies: **3NF**

REQUIRES (Associative entity/Joint Table)

Functional Dependencies:

`{requirementNo, equipmentNo} → quantity`

- All attributes are atomic: **1NF**
- Composite PK; quantity depends on full key; no partial dependency: **2NF**
- No transitive dependencies, only one non-key attribute: **3NF**

ASSIGNED_TO (Associative entity/Joint Table)

Functional Dependencies:

`{requirementNo, employeeNo} → (no non-key attributes)`

- All attributes are atomic: **1NF**
- Composite PK; no partial dependency: **2NF**
- No non-key attributes: automatically **3NF**

c. Validate the logical model against 5 user transactions. (Note: These will be then implemented in 3c).

Using the first approach, we check that all the information (entities, relationships, and their attributes) required by each transaction is provided by the model, by documenting a description of each transaction's requirements.

Transaction 1: Add a new client and their first cleaning requirement

The details of clients are held in the CLIENT entity and the details of cleaning requirements are held in the REQUIREMENT entity. In this case, we can use the **CLIENT HAS REQUIREMENT** relationship to record a new client along with their first requirement. The clientNo from CLIENT is used as a foreign key in REQUIREMENT to establish this connection.

Transaction 2: Record which equipment is needed for a requirement, including quantity

The details of cleaning requirements are held in the REQUIREMENT entity and the details of special equipment are held in the EQUIPMENT entity. The quantity of each equipment item needed is specific to each requirement-equipment pairing. In this case, we can use the REQUIREMENT **REQUIRES EQUIPMENT** relationship (through the REQUIRES junction table) to record which equipment is needed for a specific requirement and in what quantity.

Transaction 3: Assign employees to a requirement

The details of employees are held in the EMPLOYEE entity and the details of cleaning requirements are held in the REQUIREMENT entity. In this case, we can use the EMPLOYEE **ASSIGNED_TO REQUIREMENT** relationship (through the ASSIGNED_TO junction table) to record which employees are assigned to work on a specific cleaning requirement.

Transaction 4: List all requirements (with times) for a given client

The details of clients are held in the CLIENT entity and the details of their cleaning requirements (including sDate, sTime, duration, and comments) are held in the REQUIREMENT entity. In this case, we can use the CLIENT **HAS REQUIREMENT** relationship to produce the required list of all requirements for a specified client, accessing the temporal attributes stored in REQUIREMENT.

Transaction 5: Find all employees and equipment assigned to a given requirement

The details of employees are held in the EMPLOYEE entity, the details of equipment are held in the EQUIPMENT entity, and the details of requirements are held in the REQUIREMENT entity. In this case, we can use:

- The EMPLOYEE **ASSIGNED_TO** REQUIREMENT relationship to identify all employees working on the specified requirement
- The REQUIREMENT **REQUIRES** EQUIPMENT relationship to identify all equipment needed for the specified requirement, including the quantity attribute stored in the REQUIRES relationship

Conclusion: All main operations described in the case study (recording clients, employees, requirements, equipment usage and assignments, and querying them) are supported by the entities and relationships in the logical model. The model successfully passes validation against these user transactions.

d. Define integrity constraints

i. Primary Key Constraints

Relation	Primary Key
CLIENT	clientNo
EMPLOYEE	employeeNo
REQUIREMENT	requirementNo
EQUIPMENT	equipmentNo
REQUIRES	(requirementNo, equipmentNo)
ASSIGNED_TO	(employeeNo, requirementNo)

ii. Referential Integrity/Foreign Key Constraints

Child Relation	Foreign Key	References	Description
REQUIREMENT	clientNo	CLIENT(clientNo)	Each requirement must belong to an existing client.
REQUIRES	requirementNo	REQUIREMENT(requirementNo)	Equipment requirements must reference an existing requirement.
REQUIRES	equipmentNo	EQUIPMENT(equipmentNo)	Required equipment must exist in the equipment table.
ASSIGNED_TO	employeeNo	EMPLOYEE(employeeNo)	Assigned employees must exist in the employee table.
ASSIGNED_TO	requirementNo	REQUIREMENT(requirementNo)	Assignments must reference an existing requirement.

iii. Alternate Key Constraints

None. Each entity has only one candidate key, which serves as the primary key.

iv. Required Data (NOT NULL Constraints)

CLIENT: clientNo, fName, lName, address, telephoneNo

EMPLOYEE: employeeNo, fName, lName, address, salary, telephoneNo

REQUIREMENT: requirementNo, clientNo, sDate, sTime, duration

EQUIPMENT: equipmentNo, description, cost

REQUIRES: requirementNo, equipmentNo, quantity

ASSIGNED_TO: requirementNo, employeeNo

v. Attribute Domain Constraints

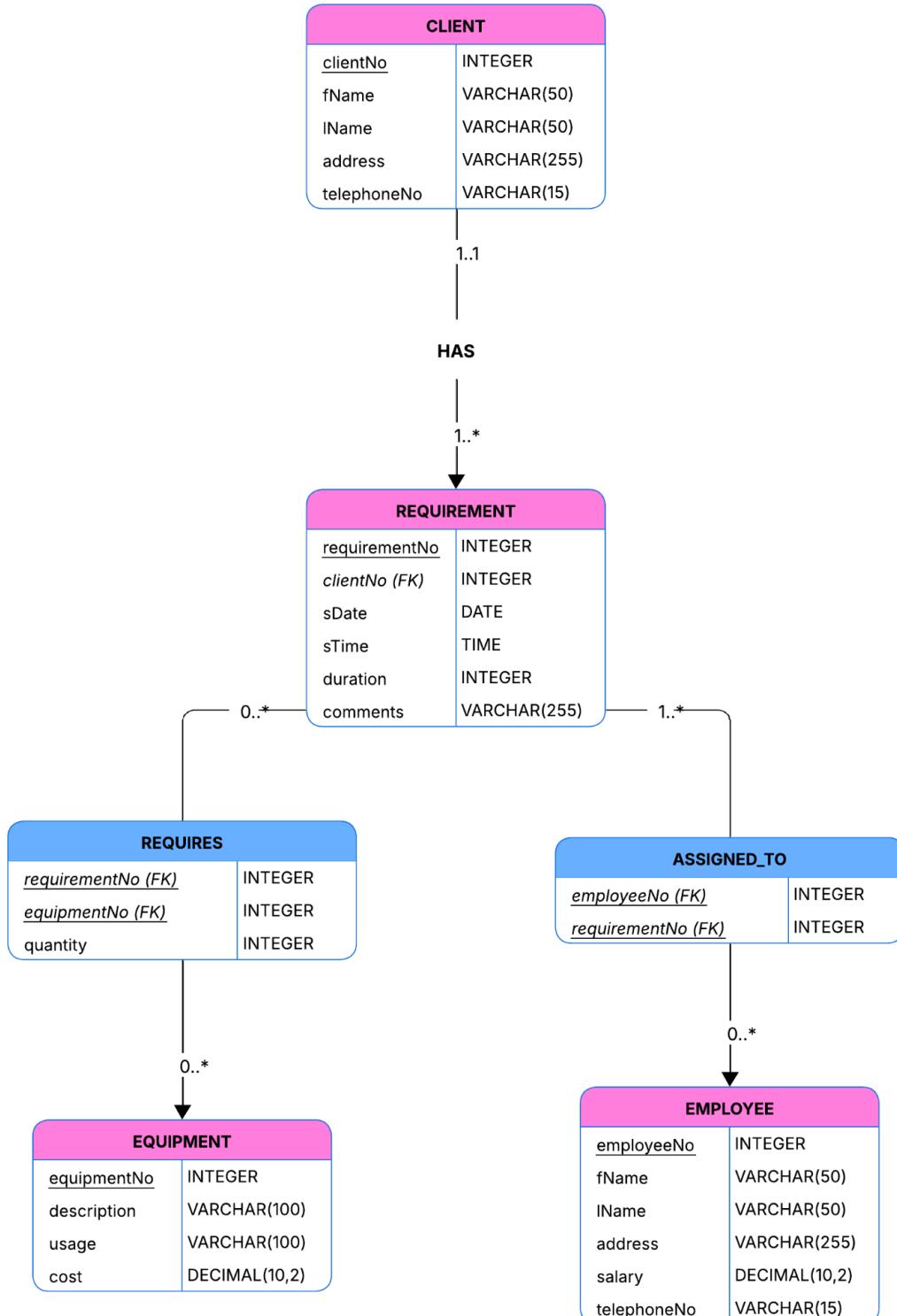
Attribute	Domain / Constraint
clientNo, employeeNo, equipmentNo, requirementNo	INTEGER
fName, lName	VARCHAR(50)
address	VARCHAR(255)
telephoneNo	VARCHAR(15)
salary	DECIMAL(10,2), CHECK (salary \geq 0)
sDate	DATE
sTime	TIME
duration	INTEGER, CHECK (duration > 0)
comments	VARCHAR(255), NULL allowed
description	VARCHAR(100)
usage	VARCHAR(100), NULL allowed
cost	DECIMAL(10,2), CHECK (cost \geq 0)
quantity	INTEGER, CHECK (quantity > 0)

vi. General Constraints

Business Rules:

- A requirement cannot exist without an assigned client (enforced by FK constraint and NOT NULL on clientNo).
- Each requirement must have at least one employee assigned (enforced by application logic or trigger).
- Equipment quantities must be positive integers.
- Duration must be expressed in minutes and must be positive.
- All monetary values (salary, cost) must be non-negative.

e. Generate the E-R diagram for the logical level (contains FKs as attributes).



Part 3

<https://github.com/av1155/CSC423-SuperMaids-Cleaning-Company>