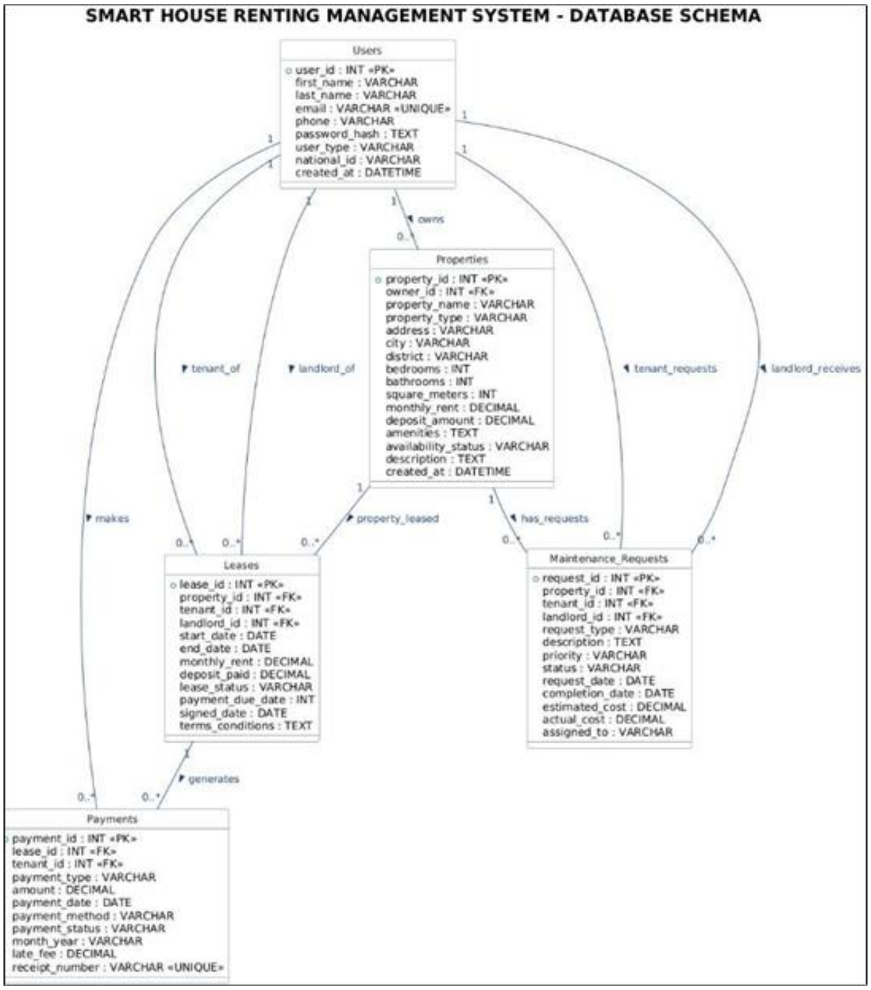
PHASE III: Smart House Renting Management System Logical Model Design

Prepared by: MWUNGERI Bonheur ID: 29337

Course: Database Development with PL/SQL (INSY 8311)

Lecturer: Eric Maniraguha



1. Introduction

This document outlines the detailed logical model design for the **Smart House Renting Management System** database. The design is based on the Entity-Relationship (ER) diagram developed from the project's requirements (Phase I & II). This logical model serves as the blueprint for building a **robust, production-ready Oracle database solution** with enhanced analytical capabilities (Business Intelligence).

---

2. Project Overview

The Smart House Renting Management System aims to automate and streamline the entire process of listing, leasing, and managing properties. The system's goal is to provide **Property Owners/Landlords** and **Property Managers** with a centralized, data-driven system to track performance, financial status, and maintenance needs, thereby minimizing manual errors and maximizing efficiency.

---

3. Entity Relationship Analysis

3.1 Core Entities

The logical model is constructed around five primary entities that manage the rental lifecycle:

1. **Users:** Represents all system actors (Tenants, Landlords, Managers, Admins).
2. **Properties:** Represents the physical accommodation units for rent.
3. **Leases:** Manages the legal and contractual agreements between a tenant and a landlord.
4. **Payments:** Tracks all financial transactions (rent, fees, deposits).
5. **Maintenance_Requests:** Handles property repair and service requests.

3.2 Relationships

The critical relationships between these entities are defined by Foreign Keys (FKs) to ensure **referential integrity**:

Relationship	Cardinality	Description
Users to Properties	1:M (One-to-Many)	One **User** (as the `Owner`) can manage multiple **Properties**.
Properties to Leases	1:M (One-to-Many)	One **Property** can have multiple **Leases** over its lifetime.
Users to Leases	1:M (One-to-Many)	One **User** (as the `Tenant`) is tied to one **Lease** contract at a time.

Relationship	Cardinality	Description
Leases to Payments	1:M (One-to-Many)	One **Lease** agreement generates multiple recurring **Payments**.
Properties to Maintenance_Requests	1:M (One-to-Many)	One **Property** can generate multiple **Maintenance_Requests**.

4. Detailed Entity Definitions (Data Dictionary)

The model is defined using standard Oracle data types and is enforced by strict constraints.

4.1 Users Entity (Central Actor Management)

Attribute	Data Type	Constraints	Description
`user_id`	`NUMBER`	**PRIMARY KEY**	Unique identifier for all users.
`first_name`	`VARCHAR2(50)`	`NOT NULL`	User's given name.
`email`	`VARCHAR2(100)`	`NOT NULL`, **UNIQUE**	Unique email address for login.
`user_type`	`VARCHAR2(20)`	`NOT NULL`	Role: 'Tenant', 'Landlord', 'Manager', 'Admin'.
`national_id`	`VARCHAR2(20)`	**UNIQUE**	National ID for verification (if required).

4.2 Properties Entity (The Asset)

Attribute	Data Type	Constraints	Description
`property_id`	`NUMBER`	**PRIMARY KEY**	Unique identifier for each property.
`owner_id`	`NUMBER`	**FOREIGN KEY** (Users)	Links property to its owning user.

Attribute	Data Type	Constraints	Description
`monthly_rent`	`DECIMAL(10,2)`	`NOT NULL, CHECK (> 0)`	Base cost of monthly rent.
`address`	`VARCHAR2(255)`	`NOT NULL`	Full physical address.
`availability_status`	`VARCHAR2(15)`	`NOT NULL`	Status: 'Available', 'Occupied', 'Pending'.
`district`	`VARCHAR2(50)`	`NOT NULL`	District/Sector for BI reporting.

4.3 Leases Entity (The Contract)

Attribute	Data Type	Constraints	Description
`lease_id`	`NUMBER`	**PRIMARY KEY**	Unique identifier for the contract.
`property_id`	`NUMBER`	**FOREIGN KEY** (Properties)	The property under lease.
`tenant_id`	`NUMBER`	**FOREIGN KEY** (Users)	The user renting the property.
`start_date`	`DATE`	`NOT NULL`	Lease start date.
`end_date`	`DATE`	`NOT NULL`, `CHECK (> start_date)`	Lease end date.
`lease_status`	`VARCHAR2(20)`	`NOT NULL`	Status: 'Active', 'Expired', 'Pending'.
`payment_due_date`	`INT`	`NOT NULL`, `CHECK (1-28)`	Day of the month rent is due (e.g., 5th).

4.4 Payments Entity (The Transaction)

Attribute	Data Type	Constraints	Description
`payment_id`	`NUMBER`	**PRIMARY KEY**	Unique identifier for the transaction.

Attribute	**Data Type**	**Constraints**	**Description**
`lease_id`	`NUMBER`	**FOREIGN KEY** (Leases)	Links payment to the specific contract.
`amount`	`DECIMAL(10,2)`	`NOT NULL`, `CHECK (> 0)`	The amount paid.
`payment_date`	`DATE`	`NOT NULL`	Date the payment was processed.
`receipt_number`	`VARCHAR2(20)`	`NOT NULL`, **UNIQUE**	System-generated, unique receipt ID.
`late_fee`	`DECIMAL(10,2)`	`DEFAULT 0`	Automatically calculated late fee.

4.5 Maintenance_Requests Entity (Service Management)

Attribute	**Data Type**	**Constraints**	**Description**
`request_id`	`NUMBER`	**PRIMARY KEY**	Unique identifier for the request.
`property_id`	`NUMBER`	**FOREIGN KEY** (Properties)	The property needing service.
`tenant_id`	`NUMBER`	**FOREIGN KEY** (Users)	The user who submitted the request.
`status`	`VARCHAR2(20)`	`NOT NULL`	Status: 'New', 'In Progress', 'Completed'.
`estimated_cost`	`DECIMAL(10,2)`	`NULLABLE`	Manager's estimated cost for the repair.

5. Normalization Analysis (3NF Compliance)

The model is designed to minimize data redundancy and insertion/update anomalies by ensuring compliance with **Third Normal Form (3NF)**.

5.1 Third Normal Form (3NF) Justification

- **1NF & 2NF:** All tables have a Primary Key, and all non-key attributes in all tables depend fully on the Primary Key.
- **3NF:** No transitive dependencies exist. For example, in the `Properties` table, attributes like `monthly_rent` depend directly on `property_id`. We avoid storing derived values or information that can be found in a separate lookup table (e.g., storing the owner's name in the `Properties` table, which would depend on `owner_id`, not `property_id`). The structure ensures every non-key attribute is dependent only on the key, the whole key, and nothing but the key.

6. Business Intelligence (BI) Strategy

The logical design is optimized for future analytical queries and data warehousing:

BI Concept	Renting System Implementation	Analytical Purpose
Fact Table	**Payments**	Measures are aggregated to find **Total Revenue**, **Bad Debt**, and **Late Fee** statistics over time.
Dimension Tables	**Properties**, **Users**, **Leases**	Used to slice and dice the financial facts (e.g., filter revenue by `district`, `property_type`, or `lease_status`).
Key Performance Indicators (KPIs)	**Occupancy Rate**, **Payment Delinquency Rate**, **Average Maintenance Cost per Property**.	These KPIs, calculated via PL/SQL Functions, directly inform property owners' strategic decisions on pricing and maintenance budgets.

7. PL/SQL Implementation Plan (Phases VI & VII)

The following PL/SQL components will be developed based on this robust logical model:

Component	Example Logic (Phase VI)	Security/Auditing (Phase VII)
Procedure	`proc_process_rent_payment`**:** Inserts a row into the `Payments` table and checks for late status.	`proc_create_audit_log`**:** Called by triggers to record DML activity.
Function	`func_calculate_late_fee`**:** Returns the penalty amount based on	`func_is_transaction_allowed`**:** Checks system date against

Component	Example Logic (Phase VI)	Security/Auditing (Phase VII)
	`payment_date` vs. `payment_due_date`.	weekdays/holidays and returns TRUE/FALSE.
Trigger	`trg_update_availability`: Updates `Properties.availability_status` when a new lease is created.	**Compound Trigger:** Implements the **CRITICAL REQUIREMENT** to block DML (INSERT/UPDATE/DELETE) on sensitive tables during weekdays/holidays, logging denied attempts.
Package	`PKG_FINANCIAL_UTILS`: Groups all payment-related functions and procedures for better error handling.	`PKG_SECURITY`: Contains the restriction function and audit procedure.

---

8. Conclusion

This logical model design provides a comprehensive, 3NF-compliant foundation for implementing the **Smart House Renting Management System** database. The structure ensures data integrity, minimizes redundancy, and explicitly incorporates the necessary entities and attributes to support key business processes, automated PL/SQL routines, and critical Business Intelligence reporting. The model is prepared for physical implementation in Oracle.