**1. ManagementCompany**

**Purpose:** Represents the parent company that runs the property management business.

* **Attributes:**
  + CompanyID (int, PK): Unique identifier for the company.
  + Name (varchar): The name of the company.
  + Tier (varchar): The service level of the company (e.g., "Premium," "Basic").
  + is\_active (boolean): Whether the company is currently active in the system.

**Relationships:**

* A **ManagementCompany** can have **one or many** PropertyManagement contracts. **(1 to Many)**
* A **ManagementCompany** can have **one or many** Users. **(1 to Many)**
* A **ManagementCompany** can have **one or many** AuditLog entries. **(1 to Many)**

**2. PropertyManagement**

**Purpose:** Represents a management contract for a specific property or portfolio.

* **Attributes:**
  + PropertyManagementID (int, PK): Unique identifier for the contract.
  + CompanyID (int, FK): Links to the ManagementCompany.
  + management\_fee (decimal): The fee charged for management services.
  + start\_date & end\_date (datetime): The duration of the contract.

**Relationships:**

* A **PropertyManagement** contract is associated with **one** ManagementCompany. **(Many to 1)**
* A **PropertyManagement** contract can be the subject of **one or many** MessageThreads. **(1 to Many)**

**3. User**

**Purpose:** Represents any person who can log into the system (employees, tenants, etc.).

* **Attributes:**
  + UserID (int, PK): Unique identifier for the user.
  + CompanyID (int, FK): The company the user belongs to.
  + email, first\_name, last\_name, phone (varchar): Personal details.

**Relationships:**

* A **User** belongs to **one** ManagementCompany. **(Many to 1)**
* A **User** can be **one** Tenant. **(1 to 1)** (A User can optionally be a Tenant).
* A **User** can send **many** Messages. **(1 to Many)**

**4. Tenant**

**Purpose:** Represents a person who rents a unit. A specialized type of User.

* **Attributes:**
  + TenantID (int, PK): Unique identifier for the tenant.
  + UserID (int, FK): Links to the User account for login.
  + employer (varchar), income (decimal), credit\_score (int): Rental application info.
  + move\_in\_date (datetime): When the tenant first moved in.

**Relationships:**

* A **Tenant** is **one** User. **(1 to 1)**
* A **Tenant** can have **one or many** Leases. **(1 to Many)** (over time).
* A **Tenant** can make **one or many** Payments. **(1 to Many)**

**5. Property**

**Purpose:** Represents a building (e.g., an apartment complex) that is being managed.

* **Attributes:**
  + PropertyID (int, PK): Unique identifier for the property.
  + CompanyID (int, FK): The company that manages this property.
  + Name, Address (varchar): The property's details.
  + Type (varchar): e.g., "Residential," "Commercial."

**Relationships:**

* A **Property** is managed by **one** ManagementCompany. **(Many to 1)**
* A **Property** contains **one or many** Units. **(1 to Many)**

**6. Unit**

**Purpose:** Represents an individual rentable space within a property (e.g., Apartment 101).

* **Attributes:**
  + UnitID (int, PK): Unique identifier for the unit.
  + PropertyID (int, FK): The parent Property this unit is in.
  + Name (string): e.g., "Unit 101".
  + is\_occupied (boolean): Whether the unit is currently rented out.

**Relationships:**

* A **Unit** belongs to **one** Property. **(Many to 1)**
* A **Unit** can have **one or many** Leases. **(1 to Many)** (over time).
* A **Unit** can have **many** Amenities. This is a **Many-to-Many** relationship, resolved by the UnitAmenity junction table.

**7. Lease**

**Purpose:** Represents a rental agreement between a Tenant and a Unit for a specific period.

* **Attributes:**
  + LeaseID (int, PK): Unique identifier for the lease.
  + TenantID (int, FK), UnitID (int, FK): Links to the Tenant and Unit.
  + start\_date, end\_date (datetime): The lease term.
  + monthly\_rent, deposit (decimal): Financial terms.

**Relationships:**

* A **Lease** is for **one** Tenant and **one** Unit. **(Many to 1** for each).
* A **Lease** can have **one or many** Invoices. **(1 to Many)**

**8. Amenity**

**Purpose:** Represents a feature or service (e.g., "Swimming Pool," "In-Unit Washer").

* **Attributes:**
  + AmenityID (int, PK): Unique identifier.
  + scope (varchar): Defines if it's for a "Property" or "Unit".
  + category (varchar): e.g., "Appliance," "Recreational".
  + is\_active (boolean): Whether the amenity is currently available.

**Relationships:**

* An **Amenity** can be linked to **many** Units. This is a **Many-to-Many** relationship, resolved by the UnitAmenity junction table.

**9. UnitAmenity**

**Purpose:** A junction table that links Units to Amenities and specifies unit-specific details.

* **Attributes:**
  + UnitID (int, PK/FK)
  + AmenityID (int, PK/FK)
  + is\_included (boolean): Whether the amenity is included in the rent for this specific unit.

**Relationships:**

* Resolves the **Many-to-Many** relationship between Unit and Amenity.

**10. Vendor**

**Purpose:** Represents an external contractor hired for maintenance or services.

* **Attributes:**
  + VendorID (int, PK): Unique identifier.
  + Name, email, phone (varchar): Contact information.

**Relationships:**

* A **Vendor** can be assigned to **one or many** MaintenanceRequests. **(1 to Many)**

**11. MaintenanceRequest**

**Purpose:** A work order for a repair or service request.

* **Attributes:**
  + RequestID (int, PK): Unique identifier for the request.
  + UnitID (int, FK), TenantID (int, FK), VendorID (int, FK): Who submitted it, for which unit, and which vendor is assigned.
  + category, description (varchar): Details of the problem.
  + status (varchar): e.g., "Open," "In Progress," "Completed".

**Relationships:**

* A **MaintenanceRequest** is submitted for **one** Unit by **one** Tenant. **(Many to 1)**
* A **MaintenanceRequest** can be assigned to **one** Vendor. **(Many to 1)**

**12. Invoice**

**Purpose:** A bill sent to a tenant (e.g., for rent or fees).

* **Attributes:**
  + InvoiceID (int, PK): Unique identifier.
  + LeaseID (int, FK): Links the invoice to a specific lease.
  + amount (decimal), due\_date (date), status (varchar): Billing details.

**Relationships:**

* An **Invoice** is generated for **one** Lease. **(Many to 1)**
* An **Invoice** can receive **one or many** Payments. **(1 to Many)** (e.g., a partial payment).

**13. Payment**

**Purpose:** A record of a payment made by a tenant.

* **Attributes:**
  + PaymentID (int, PK): Unique identifier.
  + TenantID (int, FK), InvoiceID (int, FK): Who paid and for which invoice.
  + amount (decimal), date (datetime), method (varchar): Payment details.

**Relationships:**

* A **Payment** is made by **one** Tenant against **one** Invoice. **(Many to 1)** for each.

**14. MessageThread**

**Purpose:** A conversation topic linking multiple messages.

* **Attributes:**
  + ThreadID (int, PK): Unique identifier.
  + PropertyManagementID (int, FK): Links the thread to a management contract.
  + subject (varchar), created\_at (datetime): Thread details.

**Relationships:**

* A **MessageThread** belongs to **one** PropertyManagement contract. **(Many to 1)**
* A **MessageThread** contains **one or many** Messages. **(1 to Many)**

**15. Message**

**Purpose:** An individual message within a conversation thread.

* **Attributes:**
  + MessageID (int, PK): Unique identifier.
  + UserID (int, FK): The sender of the message.
  + ThreadID (int, FK): The thread this message belongs to.
  + body (varchar), sent\_at (datetime), is\_read (boolean): Message content and status.

**Relationships:**

* A **Message** is sent by **one** User. **(Many to 1)**
* A **Message** belongs to **one** MessageThread. **(Many to 1)**

**16. AuditLog**

**Purpose:** A security log that tracks all significant actions in the system.

* **Attributes:**
  + AuditID (int, PK): Unique identifier.
  + CompanyID (int, FK): The company where the action occurred.
  + actor (varchar): Who performed the action (e.g., a user ID).
  + entity (varchar): What was acted upon (e.g., "Invoice").
  + action (varchar): What was done (e.g., "CREATE", "UPDATE").
  + created\_at (datetime): When it happened.

**Relationships:**

* An **AuditLog** entry is recorded for **one** ManagementCompany. **(Many to 1)**

**17. Role**

**Purpose:** Defines user permissions and access levels in the system.

* **Attributes:**
  + RoleID (int, PK): Unique identifier for the role.
  + role\_name (varchar): The name of the role (e.g., "Manager," "Tenant," "Maintenance").
  + scope (varchar): Defines the access level or permissions scope.

**Relationships:**

* A **Role** can be assigned to **many** Users. **(1 to Many)**

**18. Document**

**Purpose:** Stores important files like lease agreements, invoices, or maintenance reports.

* **Attributes:**
  + DocumentID (int, PK): Unique identifier for the document.
  + LeaseID (int, FK): Links the document to a specific lease agreement.
  + due\_date (datetime): When the document needs to be completed or signed.
  + status (varchar): The current state (e.g., "Pending," "Signed," "Archived").

**Relationships:**

* A **Document** is associated with **one** Lease. **(Many to 1)**

**19. PropertyAmenity**

**Purpose:** A junction table that links Properties to Amenities available at the property level.

* **Attributes:**
  + PropertyID (int, PK/FK)
  + AmenityID (int, PK/FK)
  + is\_included (boolean): Whether the amenity is available for the entire property.

**Relationships:**

* Resolves the **Many-to-Many** relationship between Property and Amenity.

**Conclusion**

Our new ERD creates a clear and organized blueprint for the property management system. It neatly defines all the important parts like users, properties, tenants, and leases and shows how they connect. By carefully structuring the entities and their relationships, the design ensures that data is consistent, easy to find, and can grow with the business.

**Rationale for Design Changes**

We updated the ERD to fix the professor's main points. We added start and end dates to track when management contracts and leases are active. We also made sure that amenities can be assigned separately to a whole property or to just one unit, which is how it works in the real world. Finally, we defined all the data types and keys to make the model more precise and ready to be built. The new design is much simpler and more logical than the old one.