Wild Wood Apartments: Database Management System, Law/Ethics/Security Plan

Ivan Novasak

Southern New Hampshire University

IT 650: Principles of Database Design

Dr. Douglass Smith

2024 July 14

# Wild Wood Apartments: Database Management System, Law/Ethics/Security Plan

**Organization**

**Problem**

***Challenges***

According to Steve Conger in *Hands-On Database*, Wild Wood Apartments has 20 different apartment complex buildings spread across 4 states: California, Idaho, Oregon, and Washington (Conger, 2013, p 17). Each complex building has between 10 and 60 apartment units of sizes that vary (Conger, 2013, p 17). Each apartment unit is leased for either 6 or 12 months (Conger, 2013, p 17). The company hires one of each building’s tenants to manage the building and in exchange gets free rent and a stipend for maintenance and repair expenses (Conger, 2013, p 18). The property manager’s responsibilities include (Conger, 2013, p 18):

* Admit new tenants to the building
* Collect rent payments from tenants
* Close out leases
* Execute maintenance repairs and replacements or renovations
* Bill the headquarters office for the maintenance, replacements, and renovations
* Mail a report to the headquarters office every quarter containing ‘the occupancy rate, the total revenues in rent, the total expenses in maintenance and repairs’

As it stands now, the information for the quarterly report is contained in 3 spreadsheets as well as things the property manager remembers from their experience over time, but any numeric quantities needs to be manually calculated based on the values in the sheet as well as paper checks that the manager received from each tenant every month (Conger, 2013, pp 37/38). These spreadsheets are one for tracking leases, one for tracking rent payments, and another for tracking the maintenance requests and responses (Conger, 2013, pp 37/38). The type of data stored in these spreadsheets is as follows (Conger, 2013, pp 37/38):

Leases spreadsheet:

* Apartment Number
* Lease Number
* Lessee Name
* Start Date
* End Date
* Rent Amount ($)
* Deposit($)
* Current (Occupancy)

Rent payments spreadsheet:

* Date
* Name
* Apartment
* Lease Number
* Amount Paid($)
* Late

Maintenance requests and responses spreadsheet:

* Apartment Number
* Date
* Problem
* Type
* Resolution
* Resolution Date
* B Expense ($)
* T Expense ($)

Some of these entries will require clarification from the property manager, such as the B and ‘T Expense’ columns, the ‘Date’ column for the maintenance spreadsheet, the lack of a lease number in some records in the rent payments spreadsheet, and the ‘Current’ column in the leases spreadsheet.

During a job shadow on 1 March 2013 with Joe Kindel, the manager for Eastlake Apartments, the following observations were made:

1. Tenants pay by personal check, and Kindel manually enters the tenants’ names, apartment numbers, payment amounts into a spreadsheet on his computer (Conger, 2013, pp 37/38).
2. This particular month, 3 tenants were late in paying Kindel the rent, and so Kindel decided to call these tenants to see what’s going on (Conger, 2013, p 38). The first and third tenants got no answer, though the first one has always paid their rent within a 5-day grace period (Conger, 2013, p 38). The second tenant answered the phone and said her Social Security check comes in on the 10th of the month and asked if it was OK to wait till then (Conger, 2013, p 38). Kindel was fine with it and will not assess a $100 penalty that normally is charged if a tenant is beyond 5 days late on paying rent. The headquarters company allows Kindel to give an extra grace period to her as she has always been a good tenant and lived there a long time (Conger, 2013, p 38). The third tenant both did not answer and was at least 2 months behind on the rent, prompting a consideration of eviction proceedings (Conger, 2013, p 38).
3. During the job shadowing, a tenant in apartment 211 called Kindel to tell him the stove was not functional (Conger, 2013, p 38). Kindel wrote in both a second spreadsheet for repairs and on a notepad about this new event, then promised to the tenant he will be by in the afternoon to check it out (Conger, 2013, p 38).

Between both the job shadow as well as the headquarters requirements for the quarterly form, the main challenges appear to be that the information needed to create the report is stored in several different places: multiple spreadsheets, paper checks, a notepad, a property manager’s memory, and whatever the headquarters office uses for storing records there. This makes the calculation and compilation of the required data for the quarterly report very time-consuming and difficult (Conger, 2013, p 18).

***Business Requirements***

The objective the headquarters office is proposing is “to develop a centralized database that can be used by the managers to track the daily business of their apartment building and to prepare their reports” (Conger, 2013, p 18). The specific content these quarterly reports require are as follows (Conger, 2013, p 38):

* Building number
* Address (block number, street name, city, state, ZIP code)
* Quarter (Spring/Summer/Autumn/Winter)
* Year (4 digits)
* Total number of apartments in the building
* Number of currently occupied apartments
* Percentage of the total apartments that are occupied
* Number of changing tenants
* Total revenues
* Total expenses
* Individual expense breakdown:
  + Utilities
  + Maintenance
  + Repairs
  + Insurance
  + New tenant cleaning
  + Wages
  + Unrecovered rents
* Total profit/loss amount

***Limitations of the Current System***

According to Conger (2023, p 18), the managers at the corporate headquarters currently have no way of verifying the accuracy of the quarterly reports, given the setup relies on so much manual calculations at each apartment complex, paperwork, data stored in multiple places, and reliable timely mail delivery. If a manager at the headquarters needed to verify a specific detail at a complex, they would need to call the property manager, then the property manager would need to check any of multiple locations the data are stored, then report back. The information Conger gathered (tenant that always pays rent during the grace period but after the official due date, another tenant who gets an extra grace period due to Social Security) at the Job Shadow on 1 March 2013 were also based a lot on the verbal agreements between tenants and the property manager, with little that is formally established in writing, let alone in a computer system (Conger, 2013, p 38). A database being built will need to be designed with the flexibility for these matters in mind, such as a default grace period but the ability to adjust it based on extenuating circumstances.

**Departments and Operations**

According to Thierry Tremblay, of Kohezion, the five main things using a centralized online database for property management will allow are (Tremblay, 2023):

1. Quick online rent collection
2. Easier communication between tenants and the manager or maintenance staff
3. Automation of tasks including generating reports and reminders of events like inspection
4. Automatic backup to a cloud server that provides access to the database from anywhere one has an Internet connection
5. More efficient property management, allowing owners to scale up their business and manage more properties

Online rent collection opens the ability to accept rent payments from anywhere and digitally, a method of purchase more of the youth market are familiar with. An online database will allow automated reminders to tenants for late payments, autopay, see in real-time who has and has not yet paid rent, as well as the possibility of late fee calculators in some products (Tremblay, 2023).

An online property management database allows tenants to send their requests more quickly and have it entered into the tracking system automatically. The database can also group tenants and maintenance crew so tenants can post their problems directly in the relevant section for the maintenance crew to see it directly, rather than needing to go through the property manager first (Tremblay, 2023). The database can handle a sudden rush on requirements faster than a person who needs to write down each request (Tremblay, 2023). The database also allows storage of contacts’ emails and phone numbers, so a maintenance staff member can view it and contact the tenant directly about any issues (Tremblay, 2023).

Report generation is one of the big time savers of using an online database system because having all the data stored in it, the system can automatically generate aggregate reports off its stored data (Tremblay, 2023). Aryan Gupta, writing for Simplilearn.com, mentions 5 aggregate functions that are especially useful when combined with the GROUP BY and HAVING clauses in a database using the SQL language: COUNT, SUM, AVG. MIN, MAX (Gupta, 2023). The COUNT function displays the number of rows (records) in a table (Gupta, 2023). The SUM function calculates the sum of all non-null numeric values for a particular table’s column (Gupta, 2023). The AVG function calculates the arithmetic mean of all non-null numeric values for a table’s column (Gupta, 2023). The MIN and MAX functions display the smallest and largest respective numeric values from a particular table’s column (Gupta, 2023). When combined with the GROUP BY clause, the results can be displayed specifically by each group, such as if one wanted to display the total number of occupied apartments for each state (Gupta, 2023). The HAVING clause allows filtering on specific conditions but for aggregate functions (the WHERE clause does not work here) (Gupta, 2023).

The online functionality can allow both the property managers as well as the corporate managers to see the same data in real-time, allowing any discrepancies to be addressed immediately rather than having to call the property manager and wait on them gathering up the data manually to verify the reports’ contents (Tremblay, 2023).

Backup and recovery are essential, and having an online database means since the data are stored in a cloud server, there is less worry about natural disasters or fire, or even malicious people damaging valuable data about the business (Tremblay, 2023).

**Analysis and Design**

**Conceptual Model**

## *Problem Description*

According to Steve Conger in *Hands-On Database*, Wild Wood Apartments owns 20 apartment buildings across four states and requires property managers of each building to compile a report every quarter that states the apartment building’s occupancy ratio as a percentage and as a raw number, how many tenants changed, total rent revenue, total expenses, individual expense by category, unrecovered rents, as well as the total profit (or loss) in dollars (Conger, 2013, pp 17/18/37). Property managers can live in an apartment rent-free in exchange for collecting rents, admitting new tenants, closing out leases, writing up the quarterly reports, and hiring the maintenance crew to perform repairs and upgrades (Conger, 2013, p 18). Property managers also get a stipend that is based on the apartment building’s size (Conger, 2013, p 18). Currently, the quarterly reports are written manually on paper forms and mailed back to the corporate headquarters, however the property managers are finding that collecting the data to compile every quarter is time-consuming and difficult, and the managers at the corporate office are worried about the accuracy of the reports, as well as not having any way to verify the data that is used to write these reports (Conger, 2013, p 18). To solve these problems, the corporate headquarters is proposing to make a centralized database that will store all this data about each apartment and building, so the reports will be easier to generate and have a way to verify their accuracy (Conger, 2013, p 18).

***Business Rules***

The types of data that are currently being stored include three spreadsheets: one for information about leases, one for payment information, and another for maintenance requests and responses (Conger, 2013, pp 37/38). Conger shadowed Joe Kindel, the property manager for Eastlake Apartments, in order to find out about other types of information that should be tracked within the database: months in arrears and one person who was given an exception to pay later than a 5-day grace period due to her being a good tenant for a long time (Conger, 2013, pp 38/39).

The three spreadsheets mentioned above will track the following data:

* Leases spreadsheet (Conger, 2013, p 37):
  + Apartment Number
  + Lease Number
  + Lessee Name
  + Start Date
  + End Date
  + Rent Amount
  + Deposit Amount
  + Whether the lessee is current
* Rent payments spreadsheet (Conger, 2013, p 38):
  + [Payment] Date
  + [Tenant] Name
  + Apartment [Number]
  + Lease Number
  + Amount Paid
  + Whether the person is late
* Maintenance requests spreadsheet (Conger, 2013, p 38):
  + Apartment Number
  + [Request] Date
  + Problem
  + Type
  + Resolution
  + Resolution Date
  + “B Expense”
  + “T Expense”

The business rules as stated by the author in (Novasak, 2024a) include:

* Lease duration is either 6 or 12 months (Conger, 2013, p 18)
* The property manager lives rent-free and has a stipend based on the building’s size (Conger, 2013, p 18)
* Payments are submitted by check (Conger, 2013, pp 38/39)
* Tenants who pay after the fifth day of the month are typically charged a $100.00 penalty, though exceptions exist on a case-by-case basis (Conger, 2013, pp 38/39)

***Entities***

The following entities can be determined based off the objects and people involved in the business (Novasak, 2024a, p 13):

* Tenants
* Property Managers
* Maintenance Staff
* Buildings
* Apartments
* Leases
* Rent Payments
* Maintenance Requests

The tenants, maintenance staff, and property managers share many attributes in common, such as name, date of birth, government ID number, phone number, and e-mail address, so another entity will be created to store these attributes in common: People.

***Relationships***

The following relationships between the entities can be established (Novasak, 2024a, pp 12/13):

* A person can be a tenant, property manager, or maintenance employee
* A tenant may have many leases (over time)
* A tenant may make many maintenance requests
* An apartment may have any number of leases over time
* Many apartments may exist in one building
* Each lease may have many rent payments
* An apartment may have many maintenance requests
* One maintenance employee may handle many maintenance requests
* A building has a property manager

***Attributes***

Attributes are what describe an entity and are information about a specific entity (Conger, 2013, p 58). A listing of each entity and their attributes follows:

* People
  + Internal database ID number
  + Surname
  + Given name
  + Middle name (will not always be used)
  + Government ID type
  + Government ID number
  + Date of birth
  + Social Security Number (will not always be used)
  + Phone number
  + E-mail address (will not always be used)
* Tenants
  + Internal tenant ID number (linked from the People entity)
* Property Managers
  + Internal manager ID number (linked from the People entity)
  + Stipend amount
* Maintenance Staff
  + Internal employee ID number (linked from the People entity)
  + Job title
  + License type
  + License number
  + Hourly wage
  + Hire date
* Buildings
  + Building number
  + Building name (will not always be used)
  + Street address
  + City
  + State (will not always be used)
  + ZIP/postal code
  + Country
  + Office phone number
  + Office email address
  + Managed By (linked from the Property Managers entity)
* Apartments
  + Apartment number
  + Floor number (will not always be used)
  + Size (area, in square feet)
  + Rent price
  + Utility costs
  + Insurance costs
  + Building number (linked from the Buildings entity)
* Leases
  + Lease number
  + Start date
  + End date
  + Rent amount
  + Deposit amount
  + Months in arrears
  + Apartment number (linked from the Apartments entity)
  + Tenant ID (linked from the Tenants entity)
* Rent Payments
  + Payment number
  + Payment date
  + Amount paid
  + Amount late
  + Lease number (linked from the Leases entity)
* Maintenance Requests
  + Internal request number
  + Request date
  + Problem
  + Request Type
  + Resolution
  + Resolution date
  + B Expense
  + T Expense
  + Repair duration in hours
  + Apartment number (linked from the Apartments entity)
  + Tenant ID (linked from the Tenants entity)
  + Staff ID (linked from the Maintenance Staff entity)

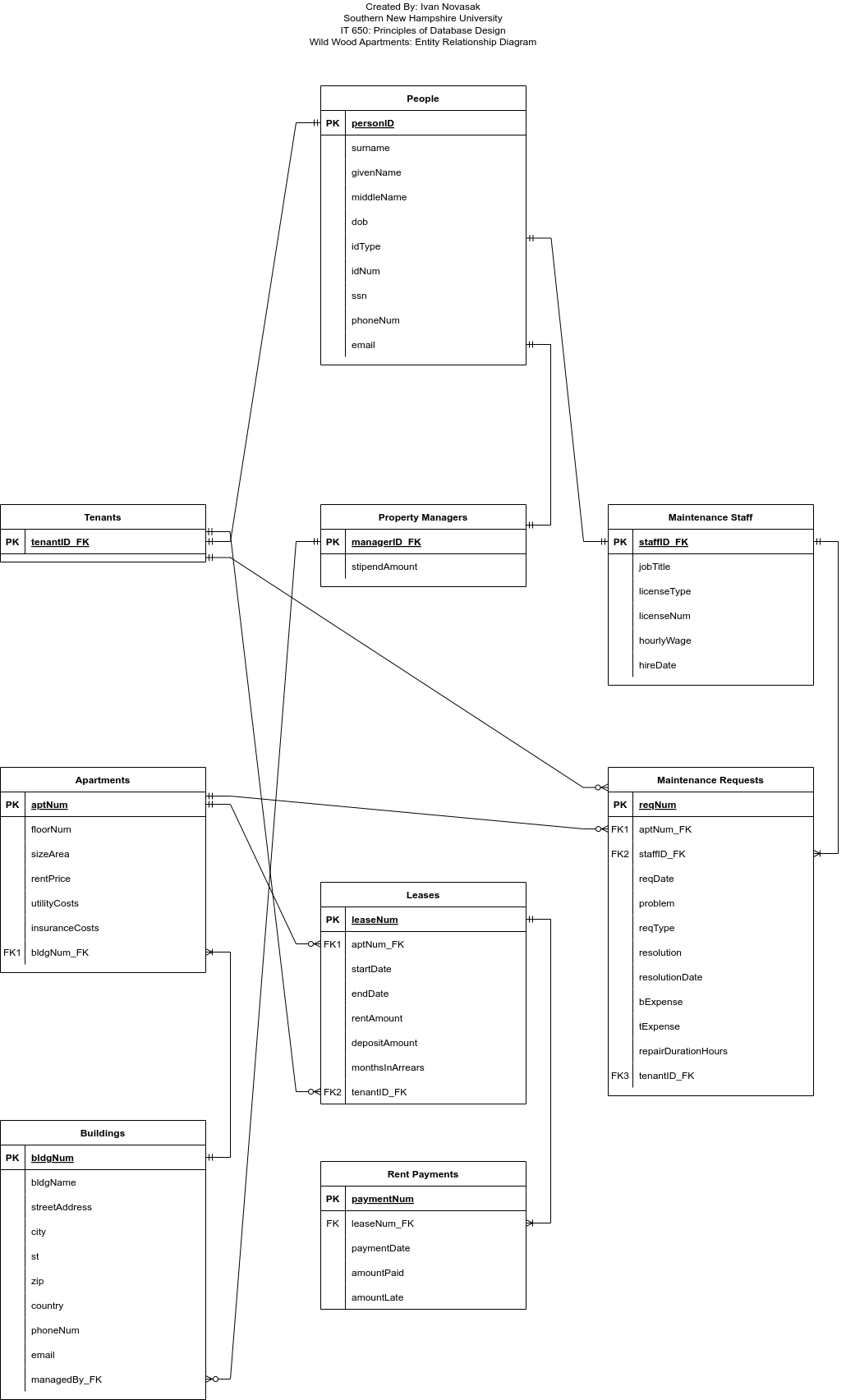
**Logical Model**

***Entity-Relationship Diagram***

The Entity-Relationship diagram (ERD), in Crow’s Foot format, is shown in Figure 1. It is based on the design by Vertabelo Team (Vertabelo Team, 2017).

**Figure 1**

*ERD for Wild Wood Apartments*



**Physical Design**

**Table 1**

*List of Tables and Their Descriptions*

|  |  |
| --- | --- |
| **Entity/Table** | **Description** |
| Person | Stores data about people that have overlapping attributes, such as legal name, government ID number, phone number, and e-mail address. |
| Tenants | Stores the tenant ID number and has room for expansion to store any future data that is unique to a tenant. |
| Property Managers | Stores the property manager ID number and their stipend amount. |
| Maintenance Staff | Stores employee ID number, their license data, job title, date hired, and their wage. |
| Buildings | Stores data about the buildings themselves, such as their location, name, office phone number, e-mail address, and who manages the building. |
| Apartments | Contains data about individual apartments, such as their size, what floor they are on, rent price, utility costs, insurance costs, and which building it is at. |
| Leases | Stores leases’ start dates, end dates, rent amounts, the tenant ID number, the apartment number, deposit amount, and how many months in arrears the tenant is. |
| Rent Payments | Stores the payment date, amount late, and amount paid, as well as for which lease number the payment is for. |
| Maintenance Requests | Stores the request date, problem, problem type, whether it was resolved, when it was resolved, how long the work took, the tenant requesting the work, the employee doing the work, expenses, and which apartment the work was done at. |

***Data Dictionary***

A data dictionary is a listing of all tables (entities), their columns (attributes), and data types for each attribute (Conger, 2013, p 124). According to Wesley Chai, of TechTarget, the purpose of a data dictionary is to describe tables’ attributes and state what data types are appropriate, whether an attribute is a primary key (the attribute that uniquely identifies a given record in the table), a foreign key (the attribute that links one table to another via the other table’s primary key), and whether or not null values are to be allowed (Chai, 2022). The database dictionary for Wild Wood Apartments is the same as in (Novasak, 2024b, pp 7–15) and is as follows:

**Table 2**

*People*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| personID | Internal ID number for a person | INT | > 0 | Yes | No | No |
| surname | A person’s surname (last name) | VARCHAR | 50 Characters | No | No | No |
| givenName | A person’s given name (first name) | VARCHAR | 50 Characters | No | No | No |
| middleName | A person’s middle name (first name) | VARCHAR | 50 Characters | No | No | Yes |
| dob | Date of birth in YYYY-MM-DD format | DATE | > 1880-01-01 & < 9999-12-31 | No | No | No |
| idType | Type of government ID | VARCHAR | 50 Characters | No | No | No |
| idNum | The person’s government ID number | VARCHAR | 50 Characters | No | No | No |
| ssn | The person’s Social Security Number | CHAR | 9 Characters | No | No | Yes |
| phoneNum | The person’s phone number | VARCHAR | 30 Characters | No | No | No |
| email | The person’s email address | VARCHAR | 100 Characters | No | No | Yes |

**Table 3**

*Tenants*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| tenantID | Internal ID number for a tenant | INT | > 0 | Yes | Yes | No |

**Table 4**

*Property Managers*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| managerID | Internal ID number for a property manager | INT | > 0 | Yes | Yes | No |
| stipendAmount | Amount of money for the stipend to manage a building | DOUBLE | >= 0.00 | No | No | Yes |

**Table 5**

*Maintenance Staff*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| staffID | Internal ID number for a maintenance employee | INT | > 0 | Yes | Yes | No |
| jobTitle | Job title for the employee | VARCHAR | 30 Characters | No | No | Yes |
| licenseType | Type of license the employee has | VARCHAR | 50 Characters | No | No | Yes |
| hourlyWage | Wage the employee is paid per hour | DOUBLE | > 0.00 | No | No | No |
| hireDate | Date the employee was hired in YYYY-MM-DD format | DATE | > 1900-01-01 & < 9999-12-31 | No | No | No |

**Table 6**

*Buildings*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| bldgNum | Internal number of the building | INT | > 0 | Yes | No | No |
| bldgName | Name of the building, if one exists | VARCHAR | 100 Characters | No | No | Yes |
| streetAddress | Number and street address of the apartment building (no city/state/ZIP here) | VARCHAR | 100 Characters | No | No | No |
| city | City the building is located in | VARCHAR | 100 Characters | No | No | No |
| st | State the apartment is located in | CHAR | 2 Characters | No | No | Yes |
| postalCode | Postal or ZIP code of the building’s address | VARCHAR | 30 Characters | No | No | No |
| country | Country the building is located | VARCHAR | 100 Characters | No | No | No |
| bldgPhoneNum | Office phone number of the building | VARCHAR | 30 Characters | No | No | No |
| bldgEmail | E-mail address of the office for the building | VARCHAR | 100 Characters | No | No | No |
| managedBy\_FK | Internal ID number for the property manager | INT | > 0 | No | Yes | No |

**Table 7**

*Apartments*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| aptNum | Number of the apartment | INT | > 0 | Yes | No | No |
| floorNum | Floor the apartment is on, if used in the building | INT | Any | No | No | Yes |
| sizeArea | Size of the apartment in square feet | DOUBLE | >= 10.0 | No | No | Yes |
| rentPrice | Current price of rent in the apartment | DOUBLE | >= 0.00 | No | No | No |
| utilityCosts | How much the utilities cost | DOUBLE | >= 0.00 | No | No | No |
| insuranceCosts | How much insurance costs for this apartment | DOUBLE | >= 0.00 | No | No | No |
| bldgNum\_FK | Internal number of the building | INT | > 0 | No | Yes | No |

**Table 8**

*Leases*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| leaseNum | Internal ID number for the lease | INT | > 0 | Yes | No | No |
| startDate | The date the lease starts, in YYYY-MM-DD format | DATE | > 1900-01-01 & < 9999-12-31 | No | No | No |
| endDate | The date the lease ends, in YYYY-MM-DD format | DATE | > 1900-01-01 & < 9999-12-31 | No | No | No |
| rentAmount | Monthly rent on this lease | DOUBLE | >= 0.00 | No | No | No |
| depositAmount | Deposit amount on this lease | DOUBLE | >= 0.00 | No | No | No |
| monthsInArrears | Number of months the tenant is behind on rent | INT | >= 0 | No | No | No |
| aptNum\_FK | Apartment number | INT | > 0 | No | Yes | No |
| tenantID\_FK | Internal tenant ID number | INT | > 0 | No | Yes | No |

**Table 9**

*Rent Payments*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| paymentNum | Internal payment number for the payment | INT | > 0 | Yes | No | No |
| paymentDate | Date the payment was made | DATE | > 1900-01-01 & < 9999-12-31 | No | No | No |
| amountPaid | Amount of the payment | DOUBLE | > 0.00 | No | No | No |
| amountLate | Amount the tenant is late | DOUBLE | > 0.00 | No | No | No |
| leaseNum\_FK | Internal lease number | INT | > 0 | No | Yes | No |

**Table 10**

*Maintenance Requests*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Attribute Description** | **Data Type** | **Range** | **Is PK?** | **Is FK?** | **Allow Null?** |
| reqNum | Internal request number | INT | > 0 | Yes | No | No |
| reqDate | Date the request was made, in YYYY-MM-DD format | DATE | > 1900-01-01 & < 9999-12-31 | No | No | No |
| problem | What kind of problem the tenant is having | VARCHAR | 250 Characters | No | No | No |
| reqType | Type of request | VARCHAR | 50 Characters | No | No | Yes |
| resolution | How the problem was resolved | VARCHAR | 250 Characters | No | No | Yes |
| resolutionDate | Date the problem was resolved | DATE | > 1900-01-01 & < 9999-12-31 | No | No | Yes |
| bExpense | B Expense | DOUBLE | >= 0.00 | No | No | Yes |
| tExpense | T Expense | DOUBLE | >= 0.00 | No | No | Yes |
| aptNum\_FK | Internal apartment number for this request | INT | > 0 | No | Yes | No |
| staffID\_FK | Employee ID number of who will be taking care of this request | INT | > 0 | No | Yes | No |
| tenantId\_FK | Internal tenant ID of who needs help | INT | > 0 | No | Yes | No |

**DBMS**

# Research on Relational and Non-relational Databases and Some Popular Database Systems

According to AltexSoft, the two main types of databases are relational and non-relational (AltexSoft, 2023). Relational databases are also known as Structured Query Language (SQL) databases because that is usually the scripting language used to develop and maintain these databases (AltexSoft, 2023). Relational databases usually are for storing data that can be represented as text and are organized into tables, columns, and rows (AltexSoft, 2023). Database management systems (DBMS) are software that bridges the user’s commands with the database itself and they may use a command line interface (CLI) or graphical user interface (GUI) (AltexSoft, 2023). These tables represent entities with multiple instances to be tracked and their rows represent instances of that entity (AltexSoft, 2023). The columns represent the entity’s attributes like name, phone number, e-mail address, vehicle identification number, manufacturer, etc. (AltexSoft, 2023). The tables are linked by the concept of primary and foreign keys, where the primary key of one table becomes the foreign key in the linked table (AltexSoft, 2023). A primary key in a relational database is an attribute that has a unique representation for every instance of an entity (AltexSoft, 2023). In a relational/SQL database, the data resides on a single server, and access or write times are improved by adding more RAM, CPU, or GPU to that specific server (AltexSoft, 2023). There can be some downtime when switching components on the server, as the data are stored in just that one central server (AltexSoft, 2023). If the data set is small to medium-sized, an SQL database has high read and write speed via adding indices to the data fields (columns/attributes) for querying and joining tables (AltexSoft, 2023). Performance may fall if the data set and number of user requests gets high (AltexSoft, 2023). Relational/SQL databases are quite secure by their nature of being integrated and having their data stored on only one server (AltexSoft, 2023). SQL is compliant with ACID, which stands for “Atomicity, Consistency, Isolation, Durability,” so SQL is suitable for eCommerce or financial usage (AltexSoft, 2023). A listing of some popular relational database management systems (RDBMS) are (AltexSoft, 2023):

* MySQL
* MariaDB
* Oracle
* PostgreSQL
* MSSQL
* SQLite

Non-relational databases, also known as NoSQL, are used for storing unstructured data, such as pictures, videos, PDF files, and other arbitrary data that cannot be easily classified into tables, rows, and columns (AltexSoft, 2023). The data models used in a NoSQL database include document-oriented, key-value pairs, node-edge graph structures, and a wide-column tabular format that allows for varying-width columns (AltexSoft, 2023). Unlike relational/SQL databases, with non-relational/NoSQL databases, the data are often stored across multiple servers where each server contains only some of the data to allow higher performance by spreading the load evenly across the different servers (AltexSoft, 2023). Due to this distributed design, NoSQL databases are ideal for very large sets of data and numerous simultaneous requests (AltexSoft, 2023). They also are flexible on switching data types, should the need arise when working with the database (AltexSoft, 2023). Security takes a hit because the data are spread around different servers and so can only be ACID-compliant within a single server partition (AltexSoft, 2023). NoSQL databases may be ideal for applications like streaming or broadcast audio/video, where the data are numerous, unstructured, yet may have many simultaneous users. A listing of some non-relational/NoSQL database systems follows (AltexSoft, 2023):

* MongoDB
* Redis
* Cassandra
* Elasticsearch
* Firebase
* Amazon DynamoDB

## *Analysis/Comparison of Products*

Due to the centralized nature of the proposed database Wild Wood Apartments is looking for, as well as the financial nature of the data to be stored within it (Conger, 2013, p 18), a relational/SQL database looks to be the way to go, rather than a NoSQL database. So, this report will compare database management systems in the relational/SQL category. In order of popularity, the SQL databases AltexSoft lists are PostgreSQL (46.48%), MySQL (45.68%), SQLite (30.83%), Microsoft SQL Server (28.77%), MariaDB (17.91%), and Oracle (11.79%) (AltexSoft, 2023).

MySQL used to be open source but is now owned by Oracle (AltexSoft, 2023). Despite this, it still is under the GNU Generally Public License and is supported on Windows, MacOS, Linux, and IRIX (AltexSoft, 2023). It is free to download the community edition to start using, and paid enterprise and cluster editions exist for larger companies (AltexSoft, 2023). MySQL has cloud compatibility and has an easy learning curve, so it is good for users who need both an easy-to-use popular database system that has online support (AltexSoft, 2023). Not all of MySQL is open source, so support and improvement of MySQL lies with Oracle rather than the community (AltexSoft, 2023). MySQL is also tricky to scale up (AltexSoft, 2023). MySQL’s ideal use cases include small eCommerce stores, online analytical processing (OLAP), and storing Internet-of-Things (IoT) sensor data (AltexSoft, 2023).

MariaDB is a free and open-source implementation of MySQL that was created by the original developers of MySQL and originally intended to be a drop-in replacement to MySQL (MariaDB, 2024). Some pros for MariaDB are encryption via PAM and LDAP authentication, Kerberos, encrypted tablespaces, and logs (AltexSoft, 2023). MariaDB has recently been extended to support GIS (for location queries) and NoSQL databases in addition to SQL (AltexSoft, 2023). Performance is enhanced via thread pool management and engine-independent table statistics (AltexSoft, 2023). The learning curve is just as easy as with MySQL, but MariaDB is easier to scale up to larger systems (AltexSoft, 2023). The same use cases for MySQL apply for MariaDB (AltexSoft, 2023). Some popular large users of MariaDB include Wikipedia, Wordpress, and Google (MariaDB, 2024).

Oracle is a proprietary database management system that is ideal for large corporations with extremely large data sets (AltexSoft, 2023). They have begun to refocus on cloud computing since release 12c (AltexSoft, 2023). Oracle has a difficult learning curve, though does have extensive technical support (AltexSoft, 2023). The costs of deployment for a business start at $17,500 per unit, with the Enterprise Edition costing $47,000 per unit, though additional costs are likely due to the recommendation of hiring a certified Oracle DB engineer (AltexSoft, 2023). The use cases for Oracle databases are ideal for large enterprises, government, healthcare, transportation systems, and national security (AltexSoft, 2023).

PostgreSQL is another SQL RDBMS that is extremely popular and supports user-defined objects in addition to tables for making more complex data structures (AltexSoft, 2023). PostgreSQL is open source and is supported on Windows, MacOS, Linux, BSD, and Solaris (PostgreSQL.org, n.d.). PostgreSQL is ideal for organizations with database needs that scale high and are complex, as this system supports JSON, H-Store, XML, and other custom data types in addition to third-party extensions (AltexSoft, 2023). It is also ideal if the database needed is a mixture of relational and non-relational data types (AltexSoft, 2023). The learning curve of PostgreSQL is difficult due to lack of consistent support and documentation being non-uniform, with different types of support spread around the community (AltexSoft, 2023). A major void in PostgreSQL is the lack of an auditing system, which means users need to always manually check the database for changes (AltexSoft, 2023). Popular use cases for financial institutions, telecommunications firms, and any organization that needs custom interfaces for processing large data sets like in data analysis/warehousing while having ACID compliance and the SQL engine (AltexSoft, 2023).

Microsoft SQL Server (MSSQL) is Microsoft’s proprietary SQL database product, which uses an extension to the base SQL named Transact-SQL (AltexSoft, 2023). It comes in a variety of editions: Express (free and ideal for small entry-level databases), Developers (for building and testing applications without a production server license), Web, Standard, and Enterprise (AltexSoft, 2023). Support from both documentation and the community are wide, and MSSQL integrates with other Microsoft products, so it is ideal for businesses who already use a lot of other Microsoft products (AltexSoft, 2023). MSSQL supports cloud databases via “Microsoft Cloud, Azure SQL Database, or SQL Server on Azure Virtual Machines” (AltexSoft, 2023). MSSQL has some downsides not found with the open-source database products like PostgreSQL and MariaDB: high costs that start at $15,123 per core with a two-core minimum and floating license conditions that make it difficult to know precisely what one is getting with each edition when new version debut (AltexSoft, 2023). Finally, the learning curve for MSSQL is on the difficult side, so it is not ideal for beginners who are new to databases (AltexSoft, 2023).

SQLite is a public-domain database system that is serverless and ACID-compliant (AltexSoft, 2023). It is engineered to optimize storage and memory usage, making it ideal for mobile and embedded applications like with IoT (AltexSoft, 2023). SQLite’s designers intend for this system to platform-agnostic and fully compatible through the year 2050 by using aviation-grade testing, extensive documentation, liberal use of commenting, cloning of source code across data centers on different continents to protect against large natural disasters, cryptographic protection of source code history, and programmers avoiding coding fads (SQLite Consortium, 2018). Due to this robust design, the US Library of Congress recommends SQLite for preserving digital content (SQLite Consortium, 2018). Several cons of SQLite include no support for concurrency, access by multiple users, stored procedures, triggers, or user-defined functions (AltexSoft, 2023).

**Recommendation**

MariaDB is the RDBMS product recommended to deploy for the Wild Wood Apartments database. The reasons why include its open-source nature, simplicity in deploying, and the author’s familiarity with the product in a prior database course.

The author tried MSSQL as was demoed in (Conger, 2013, pp 104–155) but encountered error messages when attempting to establish relationships between the database’s tables and changing which columns are allowed to have null values. Creating the tables and their relationships via command line in MSSQL also did not work at the time of testing, so the author decided to implement the database using MariaDB, which uses a very similar SQL syntax anyway, as it was based on MySQL. Even if it ran without issue, Microsoft’s products have a hefty cost factor and unknown, changing licensing terms to consider, so their solution may not have been ideal for a small-to-medium-sized apartment company who is just getting started in the world of databases.

Oracle and PostgreSQL are designed for much more complicated and varied data sets, and Oracle also has a very high cost of entry as well as both DBMS requiring a database expert to use. Those factors make Oracle and PostgreSQL not very ideal for Wild Wood Apartments.

MySQL itself seems ideal given its open-source nature, but since it is owned by Oracle and is not fully open source, the author has concerns MySQL may turn proprietary due to being owned by Oracle. It is also not free if one needs the Enterprise of Cluster edition, so should Wild Wood Apartments grow in future, they may need to incur another unforeseen cost of switching MySQL editions.

SQLite’s lack of multi-user support or a server makes this option not suitable for Wild Wood Apartments because it closes the possibility of online payments and remote access by multiple users. So MariaDB it is.

**Hardware and Software**

According to Global Data 365, the following hardware specifications are recommended:

* 64 GB RAM
* At least 8 cores CPU
* 3 high-speed storage drives (hard drive or SSD): one for data, one for auditing log, and one for OS data

***RAID Arrays***

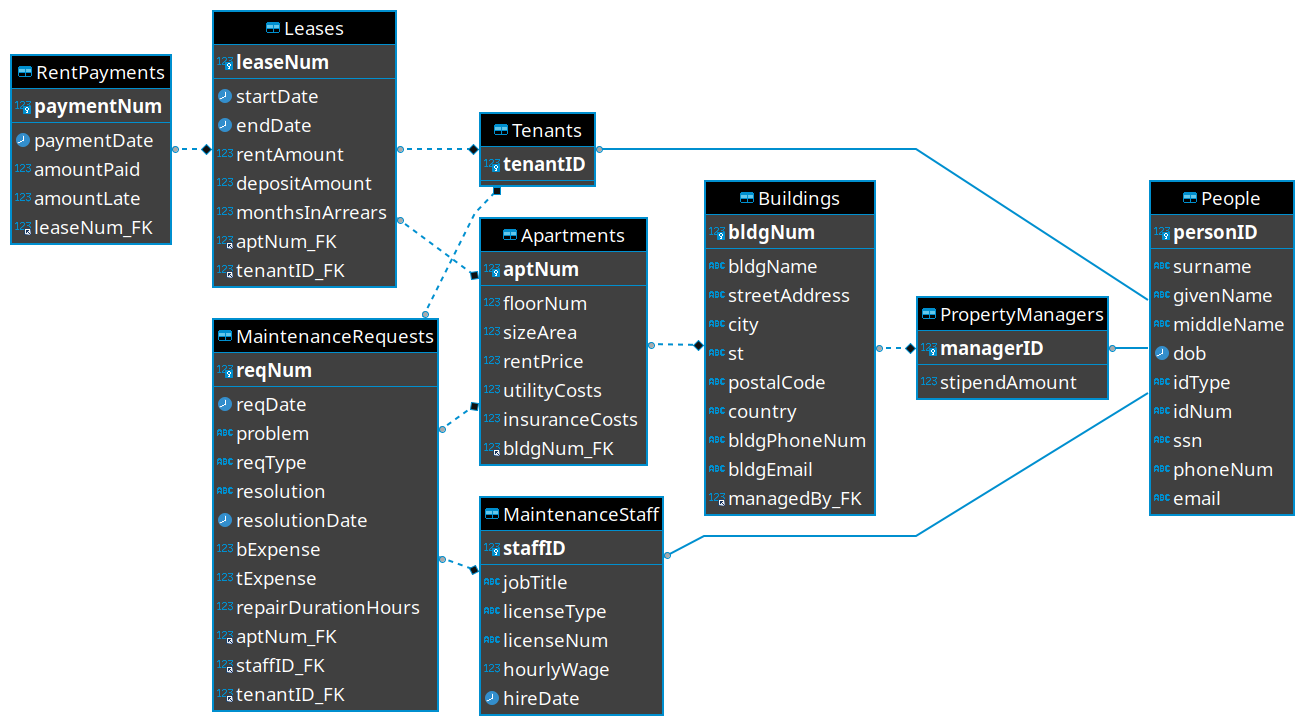
In addition to these is a recommendation of a RAID 1 or 5 array which provides redundancy for the three storage drives, as stated by Alexander S. Gillis, Erin Sullivan, and Brien Posey in a 2023 TechTarget article on RAID arrays (Gillis et al, 2023). RAID allows redundancy of data should a drive fail and while it is not an alternative to backing up data, it can allow the database to remain functional without data loss until such time the IT staff can arrive to swap out the failed drive for a new one (Gillis et al, 2023). RAID 1 is simple mirroring, where each drive’s data are cloned to another drive 1:1, requiring double the drives (Gillis et al, 2023). RAID 5, parity, is where a minimum of three but recommended of at least five hard disks are used to stripe error-correcting data across the entire array, so should a drive fail, the array can rebuild the missing data onto a new drive (Gillis et al, 2023).

***GUI***

If the user would like a graphical interface to display and manage MariaDB database, they could use DBeaver (DBeaver, n.d.). Besides running queries and displaying data graphically, DBeaver can generate Entity-Relationship Diagrams from the database contents. Figure 2, below, shows an example of this with the latest incarnation of a sample database for Wild Wood Apartments. Figure 3 shows a screenshot of DBeaver displaying the content of a sample table with data in it.

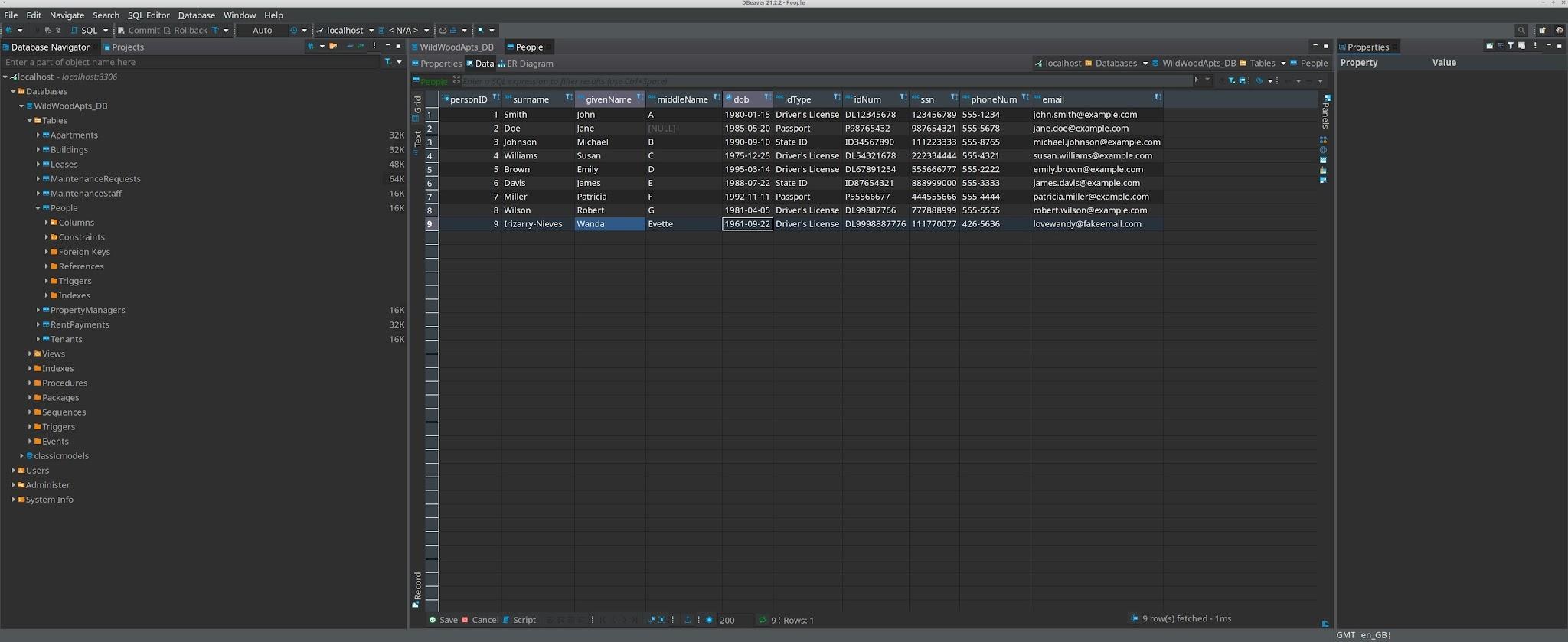
**Figure 2**

*A screenshot of an ER Diagram that was generated by DBeaver for the Wild Wood Apartments database.*



**Figure 3**

*A screenshot of DBeaver showing the People table’s sample data.*



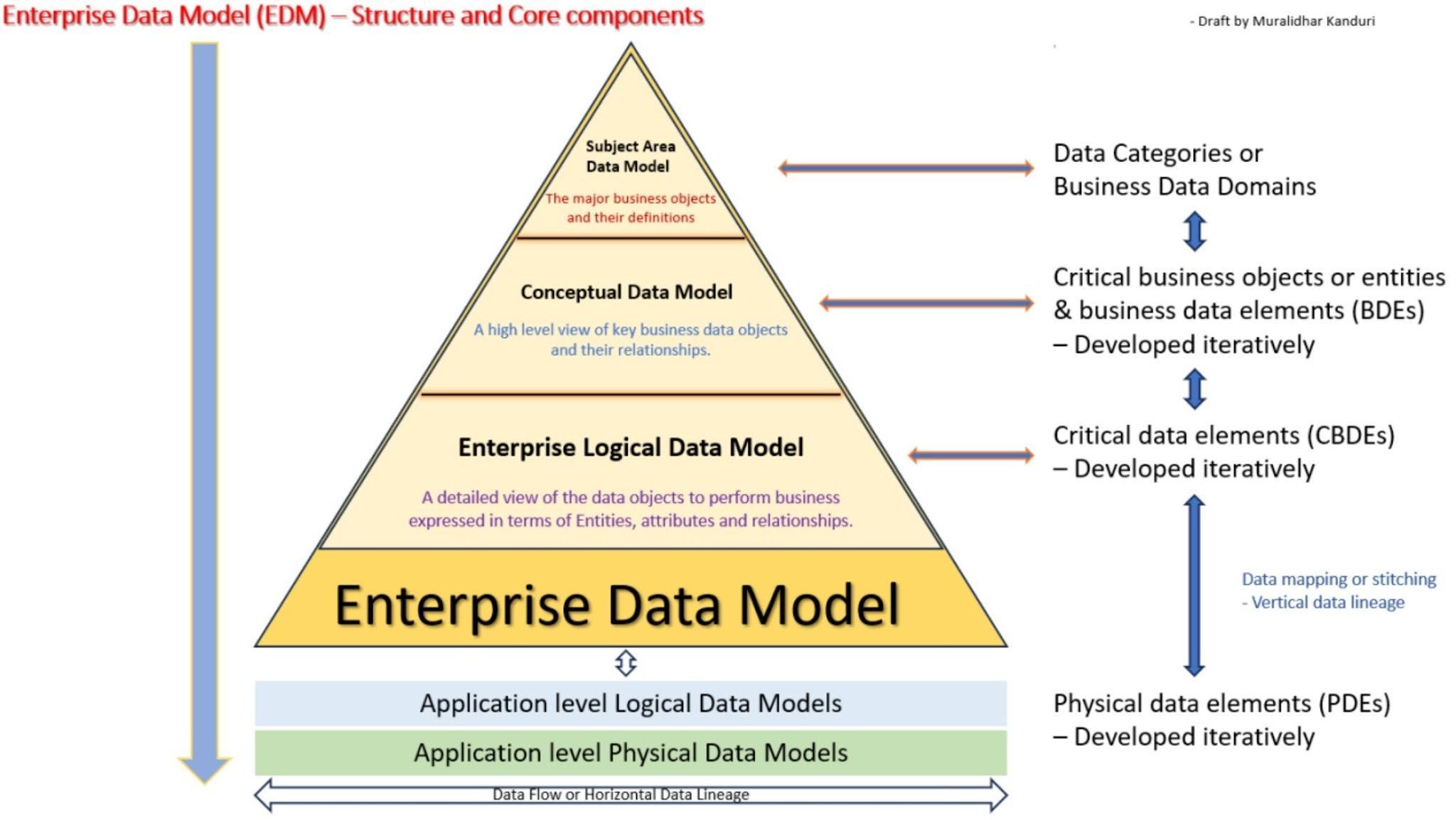
**Data Model**

**Enterprise Data Model**

According to Muralidhar Kanduri of LinkedIn, an enterprise data model is “a structured framework for managing data, ensuring consistency, accuracy, and accessibility while enabling informed decision-making” (Kanduri, 2023). It features a data dictionary, where entities attributes, and relationships are defined; data taxonomy structure; formats and naming conventions; an entity-relationship diagram; metadata management; security/access control; and data quality tools (Kanduri, 2023). A diagram of the components that make up an enterprise data model is shown in Figure 4 (Kanduri, 2023).

**Figure 4**

*Enterprise Data Model*



*NOTE*: Image is Copyright Muralidhar Kanduri, 2023 (Kanduri, 2023).

**Operating Rules**

A property manager oversees adding new tenants, taking rent payments, making maintenance requests, and compiling quarterly reports for corporate management. A sequence of steps these processes go through from the perspective of the property manager follows.

***Sequence of Steps for Adding a New Tenant***

1. A prospective tenant enters the leasing office and speaks to the property manager about renting an apartment.
2. The property manager has the prospective tenant fill out an application and consents to a background and credit check.
3. The prospective tenant either passes or fails the background and credit checks:
   1. If they pass, proceed to step 4.
   2. If they fail, the property manager declines the tenant.
4. The property manager gives the tenant some tours of vacant units and the tenant makes a selection.
5. The property manager has the tenant fill a lease form out.
6. The property manager reviews the lease form, signs it, and gives the tenant a move in date, lease duration, and due date of the first payment.
7. The property manager enters the new tenant’s contact data into the database in the Person and Tenant tables.
8. The property manager enters the new lease data into the Lease table.
9. The tenant leaves and returns on move-in day to move in and make their first payment.

***Taking Rent Payments***

1. Property manager bills the tenant for rent on the first day of each month.
2. Tenant either pays by the fifth day of the month or does not:
   1. If the tenant pays on time, proceed to step 3.
   2. If the tenant pays after the fifth day but before two months elapse, require the tenant pay an extra $100.00 penalty in addition to the rent, then proceed to step 3.
   3. If the tenant does not pay within 60 days and does not speak to the property manager, notify them on possible eviction.
3. After accepting the payment, the property manager needs to enter the payment date and amount data into the RentPayments table. If the tenant is still in arrears, proceed to step 4. Otherwise, this sequence is concluded.
4. Update the arrears amount and time in the Leases and RentPayments tables.

***Creating Maintenance Requests***

1. The tenant calls or walks into the office needing a problem addressed.
2. The property manager notes the problem and enters its request date, problem type, problem description, and who will be contracted to fix the problem into the MaintenanceRequests table.
3. The property manager contacts the maintenance staff to set an appointment date to fix the problem.
4. The maintenance crew visits the tenant’s apartment to fix the problem.
   1. If the problem is resolved, update the MaintenanceRequests table with the resolution date and work that was performed.
   2. If the problem was not yet resolved, update the MaintenanceRequests table with the work that was performed and has yet to be performed.
5. Property manager accepts the bill from the maintenance staff and inserts that data into the MaintenanceRequests table.

***Compiling Quarterly Reports***

1. Create an SQL View called myRevenue that sums the payment amounts in the RentPayments table for lease numbers associated with the property the manager is in charge of for the desired date range that reflects the quarter the report is for.
2. Create a second SQL View called myExpenses that sums the utility costs, insurance costs, amount late, maintenance staff wages, and maintenance costs associated with the property the manager oversees for the desired date range that reflects the quarter the report is for.
3. Create a third SQL View called myProfitLoss that calculates the expression myRevenue - myExpenses, where the expression is the output of the first two SQL Views. The output value should be the final profit/loss amount.
4. Run a query to count all leases having start or end date between the desired quarter’s start and end date for apartments in the managed building then divide this number by the total number of apartments and multiply that value by 100 to get the occupancy ratio as a percentage. For example, if the quarter in question is the 2nd quarter of 2023, which starts 2023-04-01 and ends on 2023-06-30, query for a lease start date of after 2023-03-31 and before 2023-07-01, and a end date of after 2023-04-30.

**Rule Reflection**

A restatement of the business rules (Novasak, 2024a, pp 4/5) is:

* A lease’s duration is either 6 or 12 months (Conger, 2013, p 18).
* The property manager lives rent-free and has a stipend based on the building’s size (Conger, 2013, p 18).
* Payments are submitted by check (Conger, 2013, pp 38/39).
* Tenants who pay after the 5th day of the month are typically charged a $100.00 penalty, though exceptions exist on a case-by-case basis (Conger, 2013, pp 38/39).
* The property manager of each building must compile a report every quarter with the occupancy quantity and percentage, revenues, itemized and total expenses, as well as the profit/loss.

The following relationships exist (including their cardinalities) and are shown visually in Figures 1 and 2 (Hernandez, 2021, pp 403–405):

* A person must be either a tenant, property manager, or maintenance staff (1:1, both mandatory).
* A maintenance staff employee may fulfill many maintenance requests (1:M, 1 mandatory/many optional).
* A tenant may place many maintenance requests (1:M, 1 mandatory/many optional).
* An apartment may have multiple maintenance requests (1:M, 1 mandatory/many optional).
* A tenant may have many leases over time (1:M, 1 mandatory/many optional).
* An apartment may be associated with many leases (over time) (1:M, 1 mandatory/many optional).
* One building may have many apartments (1:M, 1 mandatory/many optional).
* One lease may have many payments associated with it (1:M, 1 mandatory/many optional).
* One property manager may manage many buildings over time (1:M, 1 mandatory/many optional).

These business rules, combined with the relationships between the entities, represent the Conceptual Data Model stated by Kanduri (2023).

**Law, Ethics, and Society**

The next portion of this document is for analyzing the impact of Wild Wood Apartments database on society and making sure that the database complies with privacy and security laws and sound ethical practices.

**Legal and Ethical Standards**

There exist legal and ethical standards for use of the database. These are to be covered next. According to Yifat Perry of NetApp, relevant laws and standards concerning data storage about individuals include (Perry, 2019):

* The General Data Protection Regulation (GDPR) for the European Union
* California Consumer Privacy Act (CCPA)
* Personal Information Protection and Electronic Documents (PIPEDA) for Canada
* The Brazilian General Data Protection Act (LGPD)
* The Australian Consumer Data Right (CDR)
* The Protection of Personal Information Act (POPI) of South Africa
* Payment Card Industry Data Security Standards (PCI-DSS).

The first six entries in the list are laws in different regions of the world that, in summary, require that people (residents and citizens) know why a specific piece of data is being collected, people are ensured that any data collected about them are protected from unauthorized gathering, people have the right to have their data erased, and they have the right to be notified if there was a data breach that endangers their freedoms and rights (Perry, 2019). The government ID type would be handy for determining which of these laws will need to be referred to in more detail if one of those countries’ citizens were to move into a Wild Wood Apartments property. The data collected that are affected by these regulations include anything organizations collect that can identify a person and can range from names, addresses, phone numbers, Social Security numbers, national identification numbers, internet activity, biometrics, and IoT devices inside peoples’ homes (Perry, 2019). Fines for violating these laws can be high and range from $50,000 USD per incident for HIPAA violations to “€20 million or 4% of worldwide annual turnover” and the “right of Data Protection Authorities to prevent a company from collecting or processing personal data while a suspected non-compliance or breach is being investigated” both for GDPR data breaches (Perry, 2019).

PCI-DSS is less of a law and more in the way of “contractual commitments maintained and enforced by the Payment Card Industry Security Standards Council (PCI SSC), an independent global body established in 2006” (Perry, 2019). It concerns credit card data and payments and is especially crucial for organizations who store credit card data for future payment usage (Perry, 2019). The six control objectives of PCI-DSS are (Perry, 2019):

* Have a secure network and systems
* Protect cardholders’ data
* Vulnerability management program
* Implementation of strong access control measures
* Monitor and test data networks regularly
* Have and maintain an information security policy.

Yet, Wild Wood Apartments is not taking credit cards for payment, so some of these objectives may not apply while others still make good security sense, such as having a secure network/systems, monitoring and testing networks, access control measures, and an information security policy. They all apply once the complex starts accepting cards for payment. If Wild Wood Apartments starts taking cards for payment and stores the card data, they first need to “submit once per quarter to a vulnerability scan run by an Approved Scanning Vendor” that scans the network, devices, and software in a non-intrusive way for security vulnerabilities once per quarter (Perry, 2019). Secondly, there is an annual assessment that companies must undergo that is based on number of processed transactions per year:

* If the merchant processes under 6 million transactions per year, they “must submit an annual Self-Assessment Questionnaire (SAQ) or a Report on Compliance (ROC)” (Perry, 2019).
* If the merchant processes over 6 million transactions per year, they “must be audited on-site by a Qualified Security Assessor (QSA) certified by the PCI SCC” (Perry, 2019).

If the merchant does not comply with PCI-DSS, the acquiring bank can be fined $5,000 to $100,000 per month which is typically passed onto the merchant, along with termination of the relationship between the merchant and the acquiring bank and reputational damage should this become public knowledge (Perry, 2019).

**Best Practices in Design, Data Use, and Storage to Ensure Legal Compliance**

According to Amit Ashbel of NetApp, there are different aspects of database security that need to be accounted for to ensure the database is secure and complies with privacy laws. These aspects are data visibility, data mapping, database queries, and technical/organizational measures (Ashbel, 2021).

Data visibility is the knowledge of what specific personal data is being stored in the database (Ashbel, 2021). In the case of Wild Wood Apartments, this is covered in the data dictionary and the .sql file that accompanies this document.

Data mapping is how data is identified, what data the organization chooses to keep, and how it moves across the organization (Ashbel, 2021). Data mapping includes an inventory of what kinds of sensitive data are stored about people within the organization, where it is stored, what categories of the data subjects, and any retention policies that exist for the data (Ashbel, 2021). For Wild Wood Apartments, the sensitive data being stored are peoples’ names, Social Security numbers, phone numbers, e-mail addresses, physical addresses, government ID types and numbers, date of birth, lease start and end dates, amounts people are paying on those leases, and whether they are in arrears. The ER diagrams shown in Figures 1 and 2 show how all the data, including the sensitive data, are organized within the database and how the data flows through the database via primary and foreign keys. Surrogate primary keys such as an internal ID number are used where possible to avoid using a natural primary key that is composed of sensitive information like Social Security or government ID numbers. This could eliminate the need to store Social Security numbers and government ID information within the database, but more information will be needed from the corporate headquarters as to the need of storing these pieces of information within the database. One could make the case that at least storing a government ID number and type/jurisdiction is fine in case emergency medical services (EMS) needs that data.

SQL queries are how data is accessed within the database, but Ashbel says, “as a tool for keeping tabs on your personal data, it can provide only very limited value” (Ashbel, 2021). Ashbel says that column names and other metadata cannot always be relied on for getting all of a person’s information out of a database and so one would need to query several columns and tables then do pattern recognition to identify the different data that pertains to someone (Ashbel, 2021). They then explain about how queries are restricted only to text-based data and how many enterprise databases are moving to NoSQL which mainly use APIs for accessing data (Ashbel, 2021), though Wild Wood Apartments is using an SQL database with only text-based data, so that portion of what Ashbel says about queries not being so useful is irrelevant. The plan is to use SQL Views to generate result sets that are restricted to the data that specific users have permission to access and no more.

***Organizational and Technical Measures***

For organizational and technical measures, these include encryption, transport layer security (TLS), physical security, network security, access controls, backups and replication, patching and software updates, as well as activity monitoring and vulnerability scanning (Ashbel, 2021).

Encryption would be both for the data itself via an encrypted filesystem such as using one of the techniques JumpCloud describes that allows one to set up an encrypted filesystem when installing a Linux distribution (JumpCloud, 2023), as well as encrypting network data via using a VPN if the database will be cloud-based. TLS replaced Secure Socket Layer (SSL) and provides a symmetric key encryption algorithm, certificate-based authentication, forward secrecy (meaning even if a private key is compromised, past data can not be decrypted as each key is unique to the data connection session), and the TLS handshake protocol (GeeksforGeeks, 2024).

Physical security of a database concerns protecting the computer devices that the database requires from theft or damage via locks, bolting the devices to a wall or other fixed installation, and surveillance cameras in the areas where physical access is possible. Physical security can extend to having generators and water-tight seals on the doors for extra protection against natural disasters. Network security is another essential category for keeping a database secure. According to Nick Barney and Ben Lutkevich of TechTarget, it can range from using devices like firewalls that block malicious traffic to network segmentation which is where the database would be on a separate network from the other computers in the building, as well as multi-factor authentication (MFA) and requiring the use of VPNs on the premises (Barney & Nutkevich, 2022).

Access controls are what “limit[s] user access to resources based on job role or business function” (Ashbel, 2021). For the Wild Wood Apartments database, the permissions described in (Novasak, 2024c, pp 2–7) and restated below in Tables 11–14 define the levels of access to the database each type of user will have.

**Table 11**

*Tenant Permissions*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** | **Constraints** |
| People | Yes\* | No | No | No | \*Only for their own data |
| Tenants | Yes\* | No | No | No | \*Only for their own data |
| PropertyManagers | No | No | No | No |  |
| MaintenanceStaff | No | No | No | No |  |
| Buildings | No | No | No | No |  |
| Apartments | No | No | No | No |  |
| Leases | Yes\* | No | No | No | \*Only for their own data |
| RentPayments | Yes\* | No | No | No | \*Only for their own data |
| MaintenanceRequests | Yes\* | No | No | No | \*Only for their own data |

Table 12 covers the permissions the property managers have. A property manager manages only one building (Conger, 2013, p 18), so they are expected to be restricted to the data for the building they manage. The maintenance staff may be outside contractors, so property managers can view/insert/update those entries. Deletions are banned from this role to prevent accidental data loss.

**Table 12**

*Property Manager Permissions*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** | **Constraints** |
| People | Yes\* | Yes | Yes | No | \*Only data for the building the property manager is managing |
| Tenants | Yes\* | Yes\* | Yes\* | No | \*Only data for the building the property manager is managing |
| PropertyManagers | Yes\* | No | No | No | \*Only data for the building the property manager is managing |
| MaintenanceStaff | Yes | Yes | Yes | No |  |
| Buildings | Yes | No | No | No |  |
| Apartments | Yes\* | Yes\* | Yes\* | No | \*Only data for the building the property manager is managing |
| Leases | Yes\* | Yes\* | Yes\* | No | \*Only data for the building the property manager is managing |
| RentPayments | Yes\* | Yes\* | Yes\* | No | \*Only data for the building the property manager is managing |
| MaintenanceRequests | Yes\* | Yes\* | Yes\* | No | \*Only data for the building the property manager is managing |

Table 13 covers permissions for the maintenance staff. They can view their own tasks, the contact data for the tenants and property managers related only to the jobs they have requested for and update only their own maintenance requests.

**Table 13**

*Maintenance Staff Permissions*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** | **Constraints** |
| People | Yes\* | No | No | No | \*Only for their own or a client’s data |
| Tenants | Yes\* | No | No | No | \*Only for a client’s data |
| PropertyManagers | Yes\* | No | No | No | \*Only for the places they are scheduled to work |
| MaintenanceStaff | Yes | No | No | No | \*Only for their own data |
| Buildings | Yes\* | No | No | No | \*Only for the places they are scheduled to work |
| Apartments | Yes\* | No | No | No | \*Only for the places they are scheduled to work |
| Leases | No | No | No | No |  |
| RentPayments | No | No | No | No |  |
| MaintenanceRequests | Yes\* | No | Yes\* | No | \*Only for their own requests |

Table 14 defines the corporate managers’ permissions. These employees have permissions concerning more than one building as they deal with the company. They have nearly full permissions but must speak with IT/the database administrator first before deleting any data.

**Table 14**

*Corporate Management Permissions*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** | **Constraints** |
| People | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| Tenants | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| PropertyManagers | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| MaintenanceStaff | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| Buildings | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| Apartments | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| Leases | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| RentPayments | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |
| MaintenanceRequests | Yes | Yes | Yes | Yes\* | \*Caution - speak with the database administrator about deletions first |

Backup and replication policies and procedures concern data safety that have both natural disasters and human-caused factors. The use of RAID was outlined in this document in the section on RAID Arrays and is an example of replication to ensure continuity of operations should a hard drive or SSD fail (Ashbel, 2021). The backup policies and procedures are defined in (Novasak, 2024c, pp 9–11) and are restated as follows:

Backup Policies:

* Operating system and its associated programs, the production database, and the database’s backup/audit log files shall be stored on separate physical drives.
* Have the IT staff run an antivirus/anti-malware software prior to backing up the database.
* No sharing of passwords amongst database users - the person logged in must be the person using the database.
* Disconnect the database’s server from the network during any backup or restoration.
* A full backup for the database and logs/audit files shall be performed once per week, with the full backups stored on external hard drives that are stored at another secure location away from the database’s main location.
* An incremental backup with daily changes since the last full backup shall be performed daily.
* Label the backup drive with the most recent backup date in yyyy-mm-dd format after each backup. Replace this label each time a new backup is made.
* Use RAID-1 mirroring to ensure a lower probability of data loss should a hard drive fail (Gillis et al, 2021).

For the procedures, these follow for each activity.

For the backup procedure, every Sunday at 23:00 / 11.00 p.m.:

1. Disconnect computers from the network.
2. Run antivirus/anti-malware software to scan for and rid computers of any possible malware.
3. Backup the full contents of the database and log/audit files to two separate external hard drives.
4. Label these external drives with a sticker label in the following format:

**Wild Wood Apartments**

**Database Hard Drive**

**Full Backup Dated yyyy-mm-dd HH:mm**

1. Transport the hard drives to the secure off-site location and store in a safe.

The incremental backup is similar, except done nightly on another set of hard drives that will remain onsite for faster restoration. The same steps except #5 is changed to storage in a locked file cabinet with only the database administrator’s access.

For recovery, the following steps need to be performed:

1. Rebuild/reconnect relevant computer components.
2. Restore RAID-1 array.
3. Retrieve the most recent “Full Backup Dated” hard drives from the safe.
4. Retrieve the most recent “Incremental Backup Dated” hard drives from the locked file cabinet.
5. Restore the Full Backup hard drives first.
6. Next, restore data from each incremental backup hard drive’s data.
7. Manually enter into the database any transactions that were written down during the downtime.
8. Verify that the database is functioning.
9. Perform another full backup according to the backup procedure.
10. Reconnect the database computer to the Internet if it is an online database.

If a hard drive in a RAID-1 array fails:

1. Note which hard drive and purchase a new one of the same make/model/capacity.
2. Disconnect the database computers from the Internet.
3. Perform a full backup from the drives that are still functional.
4. Remove the damaged hard drive and install the new hard drive to the computer system in its place.
5. Restore the data to the new hard drive that was purchased.
6. Dispose of the damaged hard drive wherever sensitive computer equipment is disposed of in the local area. For the city of Philadelphia, one such center is located at (eForce Recycling, n.d.):

**eForce Recycling**

**1229 South Napa Street**

**Philadelphia, PA 19146**

**Tel: +1 215 964 6665**

**E-mail:** [**recycle@eforcecompliance.com**](mailto:recycle@eforcecompliance.com)

**Hours: Monday to Friday 08:00 to 14:00 / 8.00 a.m. to 2.00 p.m.**

1. Restart the database and ensure it is running properly.
2. Reconnect the computer to the Internet if the database is an online one.

MoldStud. (2024, February 1). Database development and Ethical data handling Practices. MoldStud. https://moldstud.com/articles/p-database-development-and-ethical-data-handling-practices Patching and software updates keep the database secure from vulnerabilities that cyber criminals may take advantage of (Ashbel, 2021). This should be performed after the backup but before the database is put back online to ensure that once it is running, the database server’s operating system already has the latest security updates. Activity monitoring and network scanning are covered in the audit log files the database should already be configured to store, as well as antivirus software and network devices like firewalls that keep out malicious traffic.

**Ethical Practices**

According to MoldStud, it is important to handle data ethically because without doing so, one runs the risk of data breaches that can result in financial losses and reputation damage, privacy violations that can cause loss in customer trust and have legal consequences, unauthorized data usage such as selling of data, and unfair or inaccurate decisions or biased algorithms made based on insufficient and inaccurate data (MoldStud, 2024). Ethical data handling begins with consent and transparency (MoldStud, 2024). Consent is where the person requesting data from another person gets their express permission that it is OK to collect that data (MoldStud, 2024). Transparency is giving an accurate reason as to why these data are needed and for what purposes (MoldStud, 2024). Data minimization is another ethical practice which is the practice of collecting only the data that are necessary to conduct business (MoldStud, 2024). Anonymize data whenever possible and have policies and procedures for data deletion when it is no longer needed (MoldStud, 2024). Anonymization is where names and other personal identifiers are removed from an aggregate data set before using in a report or other purpose that does not require that level of detail and may risk people’s privacy (MoldStud, 2024). Having encryption, access controls, and security audits are vital to ensure the data are safe and not able to leak to unauthorized and malicious parties (MoldStud, 2024). Finally, training staff on ethical handling of data is important so everyone knows how important it is to be ethical with people’s data (MoldStud, 2024).

At Wild Wood Apartments, all data collection will require signature confirmation, preferably via Docusign so both the apartment complex and the tenant or maintenance staff have acknowledged they consented to this collection of data, and each has a digital record of it. For instance, they will be required to sign documentation for applying for an apartment, confirming a lease, confirming maintenance work, and renewing or ending a lease. For applying for the apartment, the prospective tenant’s name, date of birth, Social Security number, government ID, phone number, and e-mail address are not entered into the database until they accept an offer and are opening their first lease, or if the person is a maintenance employee, when they are hired and have a start date. This prevents collection of data until such time as it is really needed. Financial data will be retained for seven years from the last usage as per US SEC 68 FR 4861 Rule 20.6 (US SEC, 2003).

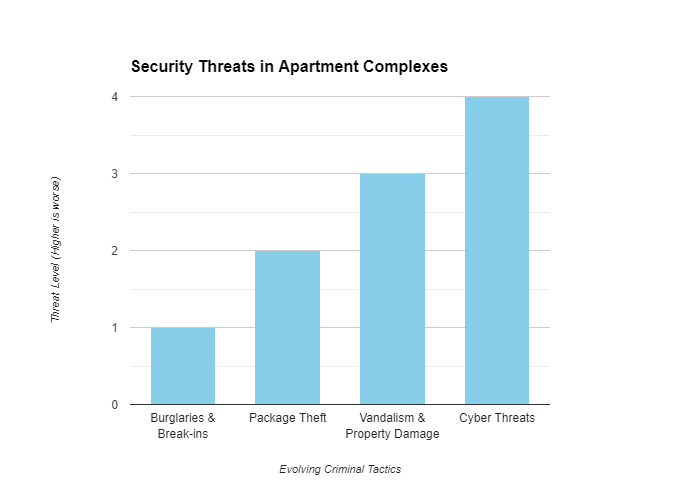
**Security Needs of the Solution**

***The Property Manager’s Role***

This section concerns the specific security needs of the property manager role at Wild Wood Apartments. As discussed in the Organizational and Technical Measures section, property managers have permissions to select, update, and insert data to/from tables but do not have deletion permissions. In addition, property managers have their allocated permissions for only the building they manage. Property managers will not oversee backup or restoration of the database, nor encryption or network security - these aspects of the database are to be handled by IT/database engineers. They will be responsible for informing the IT staff as soon as possible should anything go wrong with the database’s operations, such as a database table update not being applied. Property managers will, however, oversee ensuring appropriate security guards are present particularly during after-hours, weekends, holidays, as well as any other day the property manager is not present at the premises. The guards ensure the manager’s office is not entered by anyone other than the manager or law enforcement should the need arise. According to Citiguard, cyber threats are a higher threat than burglaries/break-ins, package theft, and vandalism/property damage (Citiguard, 2024). Figure 5, below, shows the comparison of four different threats and their threat levels on bar graphs.

**Figure 5**

*Security Threats Comparison*



*NOTE*: Image is Copyright Citiguard, Inc., 2024.

***Scanning of Paper Records***

Besides building security, property managers would also oversee scanning the existing paper records with software such as VueScan, by Hamrick Software, which supports any of up to 7,750 scanners from 42 different manufacturers and runs in Windows, Mac OS, and Linux (Hamrick Software, n.d.). The scanned records will need to be uploaded to whatever encrypted cloud-based file storage account the headquarters of Wild Wood Apartments uses. Property managers will be responsible for writing paper records in the event of an electricity outage which will need to be scanned and added into the database once electricity service resumes. The records will need to be stored in a fire-proof safe when not being accessed.

***Emergencies and Natural Disasters***

According to Aly J. Yale of Multifamily Executive, natural disasters are one of the biggest threats to multifamily apartment complexes, so having a plan is essential (Yale, 2022). In 2021, “major weather events impacted over 14.5 million single- and multifamily homes,” causing $57 billion in property damage (Yale, 2022). The first thing to note are the types of natural disasters that are likely to take place in the region the apartment complex is located. Given that Wild Wood Apartments have complexes that are located in different states, each state may have its own threats that are relevant only there. This is where the property manager’s knowledge of the local area can come in handy. Yales says that maintaining the property and paying close attention to drain systems when storm season and even individual storms are likely to approach the region (Yale, 2022). Making sure insurance is up to date and covers all likely disasters is crucial, and getting a public adjuster like Kenny Taylor of The Greenspan Co./Adjusters International can help keep costs low for line items in claims (Yale, 2022). It is a good idea to call the insurance company if a big storm, like a hurricane or windstorm, is approaching the area so the complex will not be waiting a long time for claims to be processed (Yale, 2022). It is smart to arrange for contractors ahead of time and from another region where work may be in less demand for cleanup and recovery processes to begin soon after the event passes so as to not suffer any mold or mildew damage or other structural issues that may happen if not repaired post-storm (Yale, 2022). Having emergency supplies on hand “like flashlights, glow sticks, batteries, radios, and first aid kits, as well as more specific items, like sandbags for areas where flooding is common or wood for boarding up windows in hurricane-prone areas” is vital in the time both during and immediately after the event passes, as it may take some time for emergency crews to get to the area (Yake, 2022). Backup power like generators or solar panels can help after a storm and this will help using other useful equipment such as “air blowers, shop vacs to suck up water, and dehumidifiers” that can be used for some cleanup after a storm (Yale, 2022). Training staff and regularly conducting fire drills and informing residents the correct course of action and people to contact after a natural disaster, in addition to calling 911 should their life be in danger, is vital so residents are prepared and will not panic (Yale, 2022).

Besides natural disasters like hurricanes, tornadoes, and windstorms, fire is another, more common disaster that can happen and property managers as well as the maintenance crew need to be equipped to deal with it. Being prepared for fire is as important as storm preparedness. According to the National Fire Protection Association, “almost 86,000 fires occurred in multifamily properties in 2020,” and “[t]hese properties also accounted for 10% of all civilian fire deaths and 19% of all fire injuries” (Yale, 2022). Yale shares a list of tips she got from Christopher Alker, who is vice-president of building operations at AKAM (Yale, 2022):

1. Make sure that all doors in the building are self-closing. Closed doors help prevent a fire from spreading.
2. Have nothing blocking the path in hallways like welcome mats and shoes at unit entrances.
3. Post in common areas and distribute a fire evacuation plan yearly to all residents and employees of the complex.
4. Make sure all fire extinguishers on the premises are regularly inspected, charged, and tested.
5. Have no open electrical breaker boxes, exposed electrical wiring, or exposed light fixtures.
6. Have, check, and regularly test emergency lighting for the hallways; smoke can limit visibility fast, necessitating the need for extra light.
7. Change all batteries in the smoke and carbon monoxide detectors yearly and replace all such detectors every 10 years. Test them regularly to make sure they are functional.
8. Check to make sure all electrical access panels are unobstructed, labeled, and accessible.
9. Store any combustible/flammable liquids in a non-combustible cabinet in only the amounts legally allowed.

Always remember to dial 911 (or your location’s appropriate emergency phone number) if a fire breaks out or if someone gets hurt. The property manager’s responsibilities to keep the building they manage are complete.

**Database Security Plan**

This final section covers Wild Wood Apartments as a whole and recaps what staff needs to do to ensure their database is secure from outside threats. Encryption, anonymization of aggregate data, collecting only the minimal data needed for a specific purpose, network security, and disaster protection were covered in the preceding sections. The staff at the headquarters will need to supply the property managers with important phone numbers for insurance claims, IT/database engineers, and security guard company should a security guard not show up. The IT staff must ensure the RAID-1 array is working and continue to do full and incremental backups of the system. They will need to consult with a data recovery company should any media fail between backups like in a major natural disaster. A two/multi-factor authentication system such as Google Authenticator should be used if the database will be connected to the Internet to ensure the person is who they claim to be when logging in. The various roles Tenant, Property Manager, Maintenance Staff, Corporate Management, and IT/Database Engineer have their database permissions defined in the Organizational and Technical Measures section of this document.

# References

AltexSoft. (2023, May 16). *Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others*. AltexSoft. <https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>

Ashbel, A. (2021, April 21). *Database Compliance with Data Privacy Regulations*. NetApp, Inc. <https://bluexp.netapp.com/blog/blg-database-compliance-with-data-privacy-regulations>

Barney, N., & Lutkevich, B. (2022, October 5). *network security*. TechTarget. <https://www.techtarget.com/searchnetworking/definition/network-security>

Chai, W. (2022, December 13). *data dictionary*. TechTarget. <https://www.techtarget.com/searchapparchitecture/definition/data-dictionary>

Citiguard. (2024, February 9). *Why Security Guard Services Are Essential for Apartments?* EMSAR DESIGN LTD. <https://www.mysecurityguards.com/blog/why-security-guard-services-are-essential-for-apartments/>

Conger, S. (2013). *Hands-On Database: An Introduction to Database Design and Development* (Second Edition). Pearson Education, Inc.

DBeaver. (n.d.). *DBEaver Community | Free Universal Database Tool*. DBeaver Corporation. <https://dbeaver.io/>

eForce Recycling (n.d.). *IT Asset Disposition (ITAD)*. eForce Recycling. <https://eforcecompliance.com/it-asset-disposition-itad.php>

GeeksforGeeks. (2024, May 23). *Transport Layer Security (TLS)*. GeeksforGeeks. [https://www.geeksforgeeks.org/transport-layer-security-tls/#](https://www.geeksforgeeks.org/transport-layer-security-tls/)

Gillis, A. S., Sullivan, E., & Posey, B. (2021, August 18). *RAID (redundant array of independent disks)*. TechTarget. <https://www.techtarget.com/searchstorage/definition/RAID>

Global Data 365. (2023, December 5). *Server Hardware sizing Recommendations*. Global Data 365.

Gupta, A. (2021, September 27). *The best Guide to understand SQL having Clause*. Simplilearn Americas Inc. <https://www.simplilearn.com/tutorials/sql-tutorial/sql-having>

Gupta, A. (2023, August 18). *SQL Aggregate Functions: Explore 5 types of functions*. Simplilearn Americas Inc. <https://www.simplilearn.com/tutorials/sql-tutorial/sql-aggregate-functions>

Hamrick Software. (n.d.). *Supported Scanners*. Hamrick Software. <https://www.hamrick.com/vuescan/supported-scanners.html>

JumpCloud. (2023, June 5). *Configure data encryption for Linux devices*. JumpCloud. <https://jumpcloud.com/support/configure-data-encryption-for-linux-devices>

MariaDB. (2024, June 12). *About MariaDB Server - MariaDB.org*. MariaDB.org. <https://mariadb.org/about/>

MoldStud. (2024, February 1). *Database development and Ethical data handling Practices.* MoldStud. <https://moldstud.com/articles/p-database-development-and-ethical-data-handling-practices>

Novasak, I. (2024a, June 2). *Wild Wood Apartments - Scenario Activity: Database Design*. Southern New Hampshire University.

Novasak, I. (2024b, June 8). *Wild Wood Apartments - Scenario Activity: Database Design*. Southern New Hampshire University.

Novasak, I. (2024c, June 23). *Wild Wood Apartments Scenario: Is It Secure?* Southern New Hampshire University.

Perry, Y. (2019, September 17). *Meeting Data Compliance with a Wave of New Privacy Regulations: GDPR, CCPA, PIPEDA, POPI, LGPD, HIPAA, PCI-DSS, and More*. NetApp, Inc. <https://bluexp.netapp.com/blog/data-compliance-regulations-hipaa-gdpr-and-pci-dss>

PostgreSQL.org. (n.d.). *PostgreSQL: Downloads*. The PostgreSQL Global Development Group. <https://www.postgresql.org/download/>

SQLite Consortium. (2018, May 31). *Long term support*. SQLite.org. <https://sqlite.org/lts.html>

Tremblay, T. (2023, May 2). *Property Management Databases: Top 5 benefits*. Kohezion. <https://www.kohezion.com/blog/property-management-database>

U.S. Securities and Exchange Commission (US SEC). (2003, January 30). *Retention of Records Relevant to Audits and Reviews*. U.S. Securities and Exchange Commission. <https://www.sec.gov/rules-regulations/2003/01/retention-records-relevant-audits-reviews>

Vertabelo Team. (2017, September 27). *A data model for a leasing office*. Vertabelo SA. <https://vertabelo.com/blog/a-data-model-for-a-leasing-office/>

Yale, A. J. (2022, April 19). *The natural disaster threat*. Multifamily Executive. <https://www.multifamilyexecutive.com/business-finance/the-natural-disaster-threat_o>