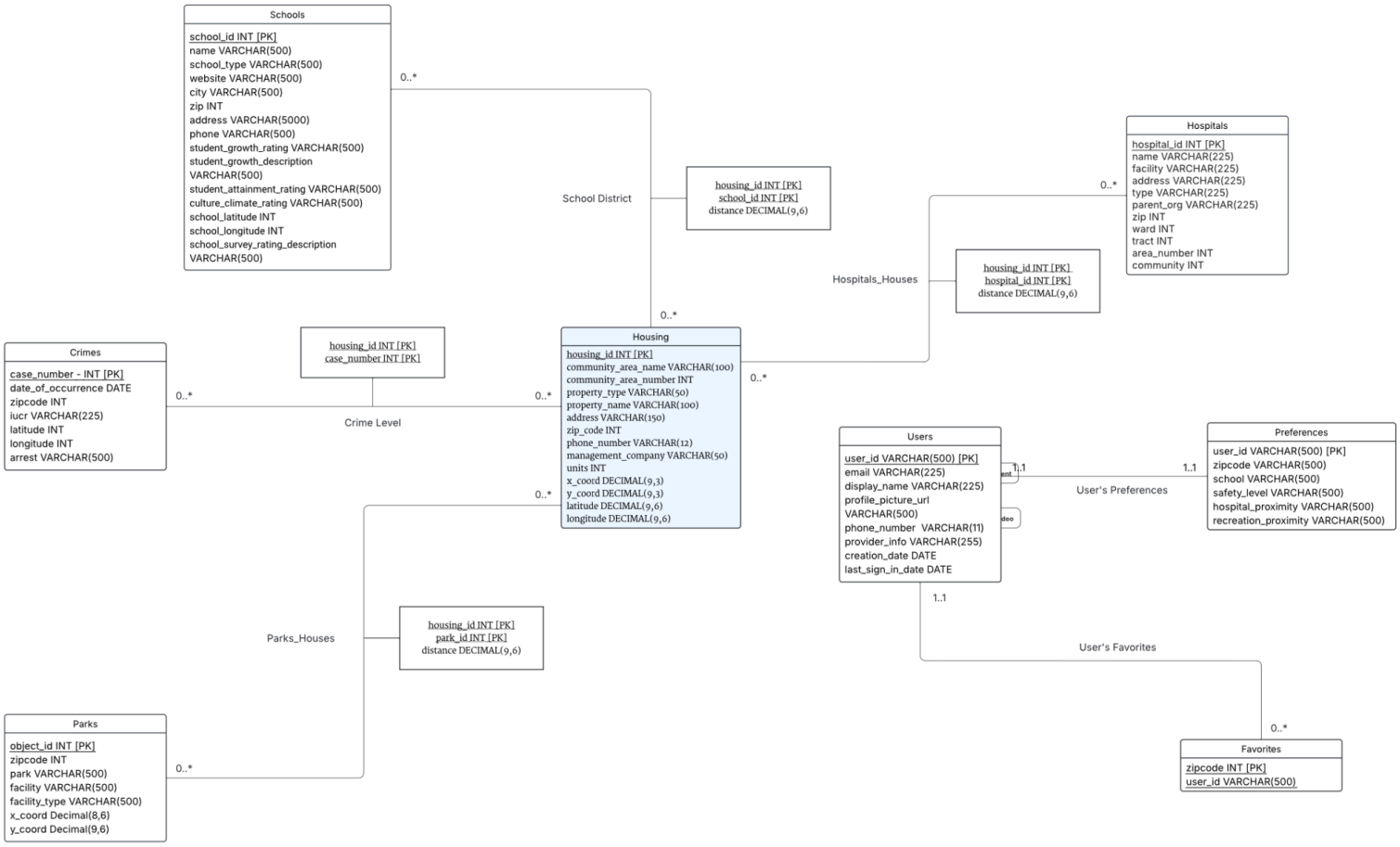
UML diagram



Logical Database Design

Schools(

school_id: INT [PK],

name: VARCHAR(100),

school_type: VARCHAR(100),

website: VARCHAR(100),

city: VARCHAR(100),

zip_code: INT [FK to Housing.zip_code],

address: VARCHAR(250),

phone: VARCHAR(12),

student_growth_rating: VARCHAR(500),

student_growth_description: VARCHAR(500),

student_attainment_rating: VARCHAR(500),

culture_climate_rating: VARCHAR(500),

school_latitude: DECIMAL(9,6),

school_longitude: DECIMAL(9,6),

school_survey_rating_description: VARCHAR(500)

)

Housing(

housing_id: INT [PK],

community_area_name: VARCHAR(100),

community_area_number: INT,

property_type: VARCHAR(50),

property_name: VARCHAR(100),

address: VARCHAR(150),

zip_code: INT,

phone_number: VARCHAR(12),

management_company: VARCHAR(50),

units: INT,

x_coord: DECIMAL(9,3),

y_coord: DECIMAL(9,3),

latitude: DECIMAL(9,6),

longitude: DECIMAL(9,6)

)

Hospitals(

hospital_id: INT [PK],

name: VARCHAR(225),

facility: VARCHAR(225),

address: VARCHAR(225),

type: VARCHAR(225),

parent_org: VARCHAR(225),

zip_code: INT [FK to Housing.zip_code],

ward: INT,

tract: INT,

area_number: INT,

community: INT

)

Crimes(

case_number: INT [PK],

date_of_occurrence: DATE,

zip_code: INT [FK to Housing.zip_code],

iucr: VARCHAR(225),

latitude: DECIMAL(9,6),

longitude: DECIMAL(9,6),

arrest: VARCHAR(500)

)

Parks(

object_id: INT [PK],

zip_code: INT [FK to Housing.zip_code],

park: VARCHAR(500),

facility: VARCHAR(500),

facility_type: VARCHAR(500),

x_coord: DECIMAL(8,6),

y_coord: DECIMAL(9,6)

)

Users(

user_id: VARCHAR(500) [PK],

email: VARCHAR(225),

display_name: VARCHAR(225),

profile_picture_url: VARCHAR(500),

phone_number: VARCHAR(11),

provider_info: VARCHAR(255),

creation_date: DATE,

last_sign_in_date: DATE

)

Favorites(

housing_id: INT [PK] [FK to Housing.housing_id],

user_id: VARCHAR(500) [PK] [FK to Users.user_id]

)

Preferences(

user_id: VARCHAR(500) [PK] [FK to Users.user_id],

housing_id: INT [FK to Housing.housing_id],

school: VARCHAR(500),

safety_level: VARCHAR(500),

hospital_proximity: VARCHAR(500),

recreation_proximity: VARCHAR(500)

)

INTERMEDIATE TABLES

School_District(

housing_id: INT [PK] [FK to Housing.housing_id],

school_id: INT [PK] [FK to Schools.school_id],

distance: DECIMAL(9,6)

)

Hospitals_Houses(

housing_id: INT [PK] [FK to Housing.housing_id],

hospital_id: INT [PK] [FK to Hospitals.hospital_id],

distance: DECIMAL(9,6)

)

Parks_Houses(

housing_id: INT [PK] [FK to Housing.housing_id],

park_id: INT [PK] [FK to Parks.object_id],

distance: DECIMAL(9,6)

)

Crime_Level(

housing_id: INT [PK] [FK to Housing.housing_id],

case_number: INT [PK] [FK to Crimes.case_number]

)

Database Normalization Analysis

Housing Table

Primary Key: housing_id

Functional Dependencies:

housing_id → community_area_name, community_area_number, property_type, property_name, address, zip_code, phone_number, management_company, units, x_coord, y_coord, latitude, longitude

Schools Table

Primary Key: school_id

Functional Dependencies:

school_id → name, school_type, website, city, zip, address, phone, student_growth_rating, student_growth_description, student_attainment_rating, culture_climate_rating, school_latitude, school_longitude, school_survey_rating_description

Hospitals Table

Primary Key: hospital_id

Functional Dependencies:

hospital_id → name, facility, address, type, parent_org, zip_code, ward, tract, area_number, community

Crimes Table

Primary Key: case_number

Functional Dependencies:

case_number → date_of_occurrence, zip_code, iucr, latitude, longitude, arrest

Users Table

Primary Key: user_id

Functional Dependencies:

user_id → email, display_name, profile_picture_url, phone_number, provider_info, creation_date, last_sign_in_date

Favorites Table

Composite Primary Key: (housing_id, user_id)

Functional Dependencies:

(housing_id, user_id) → favorites relationship (tracks which users have favorited which housing)

Parks Table

Primary Key: object_id

Functional Dependencies:

object_id → zip_code, park, facility, facility_type, x_coord, y_coord

Preferences Table

Primary Key: user_id

Functional Dependencies:

user_id -> housing_id, school, safety_level, hospital_proximity, recreation_proximity

INTERMEDIATE TABLES

School_District Table

Primary Key: (housing_id, school_id)

Functional Dependencies:

(housing_id, school_id) → distance

Hospitals_Houses Table

Primary Key: (housing_id, hospital_id)

Functional Dependencies:

(housing_id, hospital_id) → distance

Parks_Houses Table

Primary Key: (housing_id, park_id)

Functional Dependencies:

(housing_id, park_id) → distance

Crime_Level Table

Primary Key: (housing_id, case_number)

Functional Dependencies:

(housing_id, case_number) → tracks the crime level associated with a given housing

Analysis of the normal forms on basis of dependencies

Applying 3NF to our schema:

In order to satisfy the 3NF requirements, we must also ensure that our schema adheres to the 1NF and 2NF requirements. We know we are satisfying the 1NF requirements because all of the attributes are atomic. Additionally, we know that we are meeting the 2NF requirements because there aren't any partial dependencies in our schema. This essentially means that each non-key attribute is fully dependent on the whole primary key and requires it entirely to be uniquely determined. Coming back to the 3NF requirements, all of the LHS of our functional dependencies are superkeys. All of this shows that our schema meets the criteria of being 3NF.

Assumptions and Justifications for Each Entity and Relationship

The logical database design in the UML diagram represents a system where users can explore housing options, schools, hospitals, crimes, parks, and personal preferences related to housing locations. The structure ensures data integrity, normalization, and

efficient querying. Below, we explain why each entity exists independently rather than as an attribute and clarify the cardinality of relationships.

1. Housing (Primary Entity)

Housing is the central entity in the system, representing real estate properties. Each property has attributes like price, size, style, and location. Housing properties have multiple attributes, including images, price history, and tax assessments. Keeping them as separate fields prevents redundancy. Since a house can be related to multiple other entities (schools, hospitals, crimes, parks), keeping it separate simplifies relationships.

Cardinality:

- **Schools:** A school can be associated with multiple houses within the same zip code, but each house relates to the schools within its zip code.
- **Hospitals:** A hospital serves multiple houses within its zip code.
- **Crimes:** Many crime incidents can occur in the same zip code.
- **Parks:** Parks serve multiple houses in the same area.

2. Schools

Schools are independent entities because their attributes include ratings, locations, and descriptions. If schools were just attributes of housing, there would be data duplication as multiple houses relate to the same school.The education-related attributes (growth ratings, culture climate rating, survey rating) require independent tracking.

Cardinality:

- **Many-to-Many with Housing:** A school can serve many houses, but a house is only related to the schools in its zip code. The **School_District** intermediate table represents the many-to-many relationship between housing and schools, with the added benefit of tracking the distance between a housing unit and its corresponding school(s).

3. Hospitals

Hospitals are distinct entities because they serve multiple houses and have independent attributes. Each hospital has unique attributes such as facility type, ward, and community classification.Users might want to filter houses based on proximity to hospitals.

Cardinality:

- **Many-to-Many with Housing:** Each hospital serves multiple houses within a zip code. These attributes require independent tracking to avoid redundancy, and hospitals are linked to housing via the **Hospitals_Houses** intermediate table, which tracks the proximity between housing units and hospitals.
- **Preferences:** Users may set preferences for proximity to hospitals.

4. Crimes

Crime incidents are recorded independently because they vary by date, type, and location. If crimes were attributes of housing, updating crime data would be inefficient.Crime records require detailed tracking of incident reports (date, arrest status, latitude/longitude).

Cardinality:

- **Many-to-Many with Housing:** A zip code may have multiple crime reports, but each crime record belongs to a single zip code. The **Crime_Level** intermediate table has been added to track crime incidents for each housing unit, linking housing with specific crime cases.
- **Preferences:** Users may filter properties based on crime safety levels.

5. Parks

Parks exist as independent entities since they serve multiple houses and include specific attributes. Parks have names, facilities, and geographic coordinates that must be stored separately.If parks were just an attribute of housing, there would be redundant data.

Cardinality:

- **Many-to-Many with Housing:** Multiple houses in the same zip code can be near the same park. The **Parks_Houses** intermediate table tracks the proximity between parks and housing units, enabling efficient queries related to park availability near housing.
- **Preferences:** Users may filter properties based on parks in the vicinity.

6. Users

Users need unique profiles for tracking preferences, favorites, and interactions.Each user has a unique email, display name, and profile settings.Users interact with multiple properties and save favorites.

Cardinality:

- **One-to-One with Preferences:** Each user has one set of preferences.

- **Zero-to-Many with Favorites:** A user can save multiple favorite locations. Users can explore multiple houses.
- **Many-to-Many with Housing:** The **Houses_Favorited** intermediate table has been introduced to track these many-to-many relationships, enabling efficient querying of users' favorite housing units.

7. Favorites

Users can bookmark or save preferred properties.Storing favorites as an entity allows tracking multiple saved properties per user.A separate entity enables efficient querying of user preferences.

8. Preferences

Users define specific preferences (schools, safety, recreation) related to housing. Storing preferences separately prevents redundant data storage. Users can change their preferences without affecting other data.

Cardinality:

- **One-to-One with Users:** Each user has a unique preference set.

Cardinality Discussion:

- The Users and Preferences have a one-to-one relationship because each user will have a single set of preferences
- The Users and Favorites have a zero-to-many relationship because a single user can have multiple favorites. On the other hand, each favorite will be linked to a specific user.
- Schools: Schools serve multiple houses, and a house can be near multiple schools. It is a many-to-many relationship
- Hospitals: Hospitals are similar and also have a many-to-many relationship.
- Crimes: Crimes occur at specific times and locations, so they can be mapped to multiple locations, and there can be multiple crimes in the same location. So, this is a many-to-many relationship.