**Carnegie Mellon University Qatar**
**15-415 – Database Applications**
**Recitation 2**

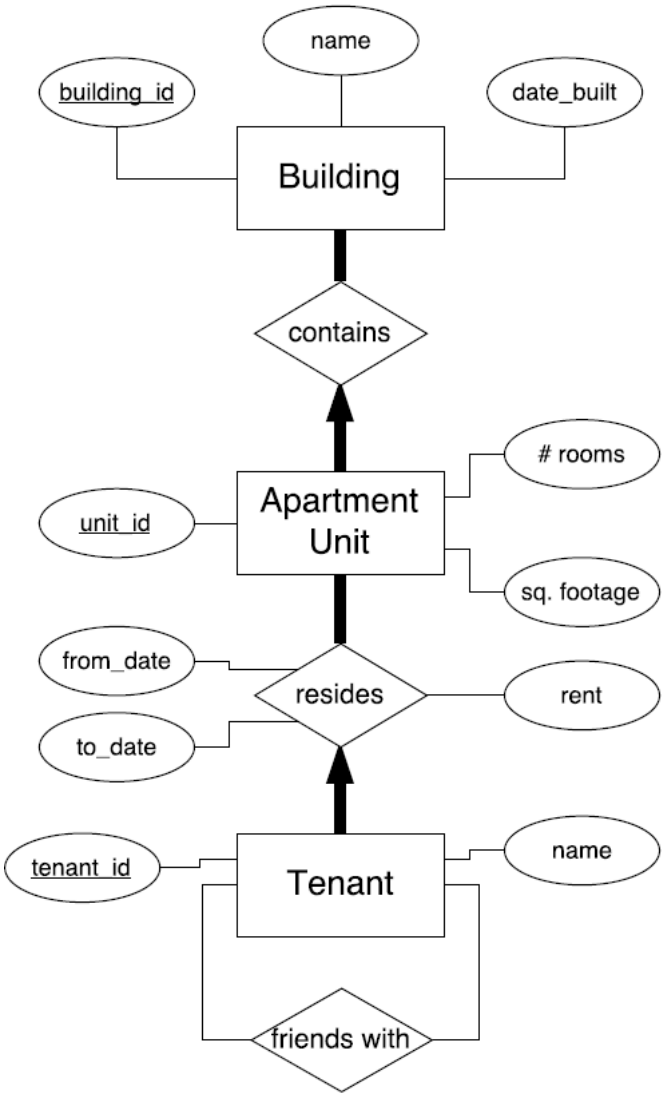


Figure 1: Apartment-tracking application

Consider the above ER diagram for an apartment-tracking system that stores data about apartment buildings, units and tenants. Your task is to write the DDL statements and build a simple PostgreSQL database to represent our DDL.

1. **To start,** we need to conceptually design our tables and their relations (one-table vs. two tables, the keys, constraints, how to translate a self-relationship, etc.)

2. **After we have the tables' conceptual design,** we will define our DDL to create our database in PostgreSQL. For example, we will define the table for the buildings entity as:

```
CREATE TABLE Building (
      building_id INTEGER,
      name CHAR(20),
      date_built DATE,
      PRIMARY KEY (building_id)
);
```

   We need to define the same for the rest of our ER diagram: A*partment* Unit, *Contains*, *Resides*, *Tenant*, *Friends with*.

3. Let's get our hands dirty and practice basic postgres commands:
   **a.** Login to your VM using your Andrew credentials.
   **b.** Start PostgreSQL from the command line using the below command and you should be direct to the default postgres database (postgres=#):

```
sudo –u postgres psql
```

   **c.** Create a new database for our apartment tracking system:

```
CREATE DATABASE <db_name>;
```

   **d.** You can list the databases you have to make sure your database was created:

```
\list
```

   **e.** Connect to the database you have just created using:

```
\connect <db_name>
```

4. Now we have a database ready to work with. Let's create our relations in our <db_name>
5. You can use the below commands to make sure your tables are created and to describe a table:
   a. List all relations: `\dt`
   b. Describe a table: `\d <table_name>`

   After creating the apartment-tracking database, we need to populate it with some meaningful data. For example, we can insert single or multiple records at a time using (Note that the types has to be matching with the type domain)

```
INSERT INTO Building VALUES
(101, 'The Dome','2018-01-20'),
(102, 'Red Tower', '2007-09-02'),
(103, 'Blue Pyramid', '1994-12-17');
```

**6.** We need to query the database to find the below data. We can always query with this syntax:

```
Select <fields to return>
From <which tables>
Where <condition>;
```

Aliases are a good way to make your queries neat. For example,

```
Select T.name
From Tenant as T
Where T.id = 100;
```

Now let's write some queries to find the bellow information:

i) All information about the buildings.

ii) The names of tenants whose contracts are already expired! You can use built-in helpers such as:

```
now()
current_date
current_time
```

iii) The name of tenants who could make friends. You can use the keyword `DISTINCT` to filter duplicates.