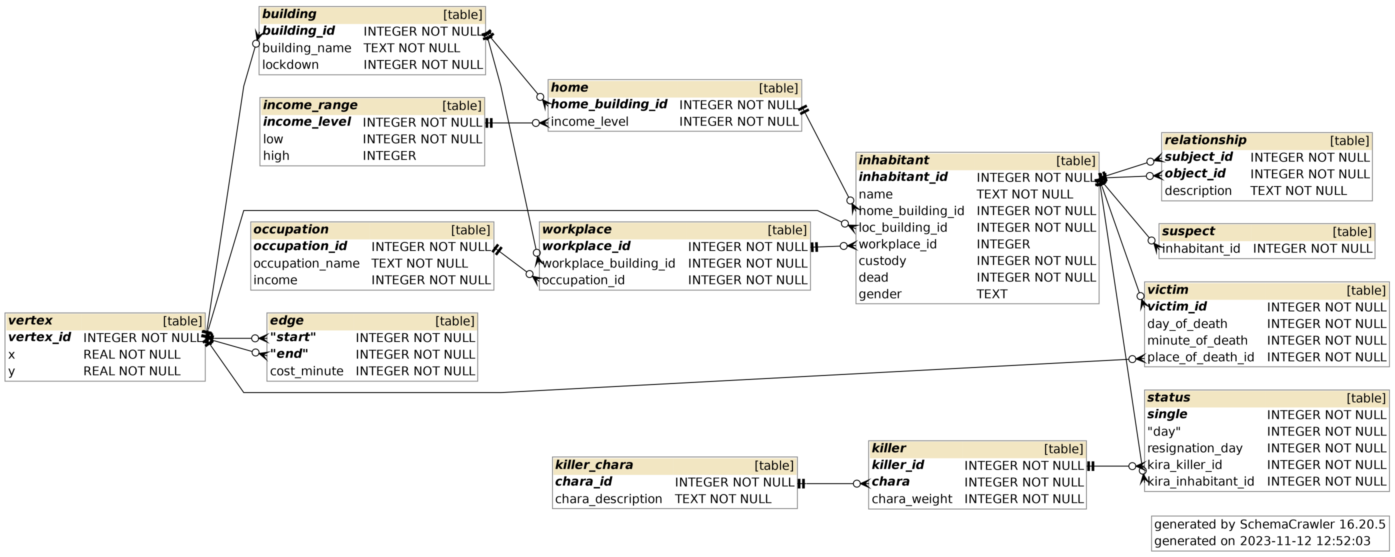
Database Design

D Simulator

Issac Yang, Yuxiang Lin, Shanruo Xu

Duke Kunshan University

1 ER Diagram


building [table]
building_id INTEGER NOT NULL
building_name TEXT NOT NULL
lockdown INTEGER NOT NULL
income_range [table]
income_level INTEGER NOT NULL
low INTEGER NOT NULL
high INTEGER
home [table]
home_building_id INTEGER NOT NULL
income_level INTEGER NOT NULL
inhabitant [table]
inhabitant_id INTEGER NOT NULL
name TEXT NOT NULL
home_building_id INTEGER NOT NULL
loc_building_id INTEGER NOT NULL
workplace_id INTEGER
custody INTEGER NOT NULL
dead INTEGER NOT NULL
gender TEXT
relationship [table]
subject_id INTEGER NOT NULL
object_id INTEGER NOT NULL
description TEXT NOT NULL
suspect [table]
inhabitant_id INTEGER NOT NULL
occupation [table]
occupation_id INTEGER NOT NULL
occupation_name TEXT NOT NULL
income INTEGER NOT NULL
workplace [table]
workplace_id INTEGER NOT NULL
workplace_building_id INTEGER NOT NULL
occupation_id INTEGER NOT NULL
victim [table]
victim_id INTEGER NOT NULL
day_of_death INTEGER NOT NULL
minute_of_death INTEGER NOT NULL
place_of_death_id INTEGER NOT NULL
vertex [table]
vertex_id INTEGER NOT NULL
x REAL NOT NULL
y REAL NOT NULL
edge [table]
"start" INTEGER NOT NULL
"end" INTEGER NOT NULL
cost_minute INTEGER NOT NULL
status [table]
single INTEGER NOT NULL
"day" INTEGER NOT NULL
resignation_day INTEGER NOT NULL
kira_killer_id INTEGER NOT NULL
kira_inhabitant_id INTEGER NOT NULL
killer_chara [table]
chara_id INTEGER NOT NULL
chara_description TEXT NOT NULL
killer [table]
killer_id INTEGER NOT NULL
chara INTEGER NOT NULL
chara_weight INTEGER NOT NULL
generated by SchemaCrawler 16.20.5
generated on 2023-11-12 12:52:03

2 Database Schema

This database is to be implemented in SQLite. SQLite is dynamic-typed with only these five storage classes: **NULL, INTEGER, REAL, TEXT, BLOB**. Therefore, all string data would be simply stored as **TEXT**. For boolean type, it is stored as an **INTEGER** constrained to be either 0 or 1 to represent **TRUE** or **FALSE**. All SQLite tables have a **ROWID** column by default unless **WITHOUT ROWID** is specified. The specification of a column as **INTEGER PRIMARY KEY** usually results in an alias to **ROWID**, which is a 64-bit integer that uniquely identifies each row.

```
CREATE TABLE vertex(
        vertex_id INTEGER NOT NULL,
        x         REAL NOT NULL,
        y         REAL NOT NULL,
                  PRIMARY KEY(vertex_id)
);
```

Stores information about vertices representing locations.

```
CREATE TABLE edge(
        start       INTEGER NOT NULL,
        end         INTEGER NOT NULL,
        cost_minute INTEGER NOT NULL CHECK(cost_minute > 0),
                    PRIMARY KEY(start, end),
                    FOREIGN KEY(start) REFERENCES vertex(vertex_id),
                    FOREIGN KEY(end)   REFERENCES vertex(vertex_id)
);
```

Describes edges connecting vertices, including the cost in minutes to traverse the edge.

```
CREATE TABLE building(
        building_id   INTEGER NOT NULL,
        building_name TEXT NOT NULL,
        lockdown      INTEGER NOT NULL CHECK(lockdown IN (0, 1)),
                      FOREIGN KEY(building_id) REFERENCES vertex(vertex_id),
                      PRIMARY KEY(building_id)
);
```

Represents information about homes, linking them to specific income levels.

```
CREATE TABLE income_range(
        income_level  INTEGER NOT NULL,
        low           INTEGER NOT NULL CHECK(low > 0),
        high          INTEGER,
                      PRIMARY KEY(income_level),
                      CHECK(high >= low)
);
```

Stores income range levels along with corresponding low and high values.

```
CREATE TABLE home(
        home_building_id  INTEGER NOT NULL,
        income_level      INTEGER NOT NULL,
                          PRIMARY KEY(home_building_id),
                          FOREIGN KEY(home_building_id) REFERENCES building(building_id),
                          FOREIGN KEY(income_level)     REFERENCES income_range(income_level)
);
```

Represents information about homes, linking them to specific income levels.

```
CREATE TABLE occupation(
        occupation_id   INTEGER NOT NULL,
        occupation_name TEXT    NOT NULL,
        income          INTEGER NOT NULL CHECK(income > 0),
                        PRIMARY KEY(occupation_id)
);
```

Contain information about occupation as well as their income level.

```
CREATE TABLE workplace(
        workplace_id          INTEGER NOT NULL,
        workplace_building_id INTEGER NOT NULL,
        occupation_id         INTEGER NOT NULL,
                              UNIQUE(workplace_building_id, occupation_id),
                              PRIMARY KEY(workplace_id),
                              FOREIGN KEY(workplace_building_id) REFERENCES building(building_id),
                              FOREIGN KEY(occupation_id)         REFERENCES occupation(occupation_id)
);
```

Link occupation accordingly to their specific workplace.

```
CREATE TABLE inhabitant(
        inhabitant_id    INTEGER NOT NULL,
        name             TEXT    NOT NULL,
        home_building_id INTEGER NOT NULL,
        loc_building_id  INTEGER NOT NULL,
        workplace_id     INTEGER,
        custody          INTEGER NOT NULL CHECK(custody IN (0, 1)),
        dead             INTEGER NOT NULL CHECK(dead IN (0, 1)),
        gender           TEXT             CHECK(gender IN('m','f')),
                         PRIMARY KEY(inhabitant_id),
                         FOREIGN KEY(home_building_id) REFERENCES home(home_building_id),
                         FOREIGN KEY(loc_building_id)  REFERENCES vertex(vertex_id),
                         FOREIGN KEY(workplace_id)     REFERENCES workplace(workplace_id)
);
```

Include information about the inhabitants, including homes workplace, dead status, gender, etc.

```
CREATE TABLE relationship(
        subject_id  INTEGER NOT NULL,
        object_id   INTEGER NOT NULL,
        description TEXT NOT NULL,
                    PRIMARY KEY(subject_id, object_id),
                    FOREIGN KEY(subject_id) REFERENCES inhabitant(inhabitant_id),
                    FOREIGN KEY(object_id)  REFERENCES inhabitant(inhabitant_id)
);
```

Record and describe the relationship between inhabitants.

```
CREATE TABLE victim(
        victim_id          INTEGER NOT NULL,
        day_of_death       INTEGER NOT NULL,
        minute_of_death    INTEGER NOT NULL,
        place_of_death_id  INTEGER NOT NULL,
                           PRIMARY KEY(victim_id),
                           FOREIGN KEY(victim_id)          REFERENCES inhabitant(inhabitant_id)
                           FOREIGN KEY(place_of_death_id) REFERENCES vertex(vertex_id)
);
```

Contains information about victims, including the time and place of death.

```
CREATE TABLE killer(
        killer_id    INTEGER NOT NULL,
        chara        INTEGER NOT NULL,
        chara_weight INTEGER NOT NULL,
                     PRIMARY KEY(killer_id, chara),
                     FOREIGN KEY(chara) REFERENCES killer_chara(chara_id)
);
```

Stores information about killers and their characteristics.

```
CREATE TABLE killer_chara(
        chara_id INTEGER NOT NULL,
        chara_description TEXT NOT NULL,
        PRIMARY KEY(chara_id)
);

CREATE TABLE suspect(
```

Store the killer characters that will influence the potential victims, as well as the detailed description about how exactly it will influence this choice.

```
CREATE TABLE suspect(
        inhabitant_id INTEGER NOT NULL,
                      FOREIGN KEY(inhabitant_id) REFERENCES inhabitant(inhabitant_id)
);
```

Put the suspect the player chose to the list

```
CREATE TABLE status(
        single             INTEGER DEFAULT 0 NOT NULL CHECK(single = 0),
        day                INTEGER NOT NULL,
        resignation_day    INTEGER NOT NULL,
        kira_killer_id     INTEGER NOT NULL,
        kira_inhabitant_id INTEGER NOT NULL,
                           PRIMARY KEY(single),
                           FOREIGN KEY(kira_killer_id)     REFERENCES killer(killer_id)
                           FOREIGN KEY(kira_inhabitant_id) REFERENCES inhabitant(inhabitant_id)
) WITHOUT ROWID;
```

Stores status information, including relationship status and details related to a specific killer. This is forced to be a single row as these all the "global" attributes. By forcing the special primary key **single** to always be 0 and disable **ROWID**, such a single row constraint is enforced.