Database Design Proposal
Christopher Lee

Disclaimer: The ctOS and Blume logos are owned by Ubisoft Montreal.
Executive Summary

The Central Operating System (ctOS) is responsible for the management and facilitation of the city of Chicago and its over 2.7 million citizens. The city of Chicago requires a database to catalogue the various functionalities controlled by the operating system. Due to federal regulations, the data must be accurate and consistent.

This document outlines the structure and entities involved in the design and implementations of a database system for ctOS. The purpose of this database is to enable cataloging of the various functionalities of the operating system such as management of the roadways, the electric grid, the subway system, the security camera system, the citizen Profiler, and more.

This database will allow administration to create useful information from queries that provide valuable statistics and other facts from the catalogued data.

An overview of the database will be presented, followed by the details of every individual database table for each of the systems managed by ctOS. Purposes of each table will be suggested and triggers will be explained to reinforce the data integrity of the database. For each of the individual parts, sample reports will be shown.

This design was targeted for and tested on PostgreSQL 9.4.1, released on Feb 5, 2015.
Infrastructure Table: Stores valid, unique identification numbers for the city's various infrastructure such as bridges, traffic lights, electric grid, etc.

CREATE TABLE IF NOT EXISTS infrastructure (  
    ifst_id SERIAL NOT NULL UNIQUE,  
    type VARCHAR(25) NOT NULL,  
    description VARCHAR(50) NOT NULL,  
    PRIMARY KEY (ifst_id)  
);

Functional Dependencies
ifst_id -> type, description

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bridge</td>
<td>Washington Bridge</td>
</tr>
<tr>
<td>2</td>
<td>Traffic Light</td>
<td>Brown St and Park Ave</td>
</tr>
<tr>
<td>3</td>
<td>Transformer</td>
<td>152 Pensacola St</td>
</tr>
<tr>
<td>4</td>
<td>Security Camera</td>
<td>29 Myers Rd</td>
</tr>
<tr>
<td>5</td>
<td>Traffic Light</td>
<td>Atkins St and Bay Ave</td>
</tr>
<tr>
<td>6</td>
<td>Bridge</td>
<td>Bayview Bridge</td>
</tr>
<tr>
<td>7</td>
<td>Transformer</td>
<td>2 Blake Ct</td>
</tr>
</tbody>
</table>
**Bridges Table:** Contains the list of bridges crossing the Chicago River. Type refers to the bridge's structure or any other significant descriptions.

```sql
CREATE TABLE IF NOT EXISTS bridges (  
    bridge_id SERIAL NOT NULL,  
    name VARCHAR(50) NOT NULL,  
    type VARCHAR(25) NOT NULL,  
    length VARCHAR(25) NOT NULL,  
    daily_traffic INTEGER NOT NULL,  
    year_opened INTEGER NOT NULL,  
    PRIMARY KEY (bridge_id)  
);  

Fuctional Dependencies  
bridge_id -> name, type, length, daily_traffic, year_opened  
```

<table>
<thead>
<tr>
<th>bridge_id</th>
<th>name</th>
<th>type</th>
<th>length</th>
<th>daily_traffic</th>
<th>year_opened</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Michigan Avenue Bridge</td>
<td>bascule</td>
<td>339 ft</td>
<td>49600</td>
<td>1920</td>
</tr>
<tr>
<td>2</td>
<td>La Salle Street Bridge</td>
<td>bascule</td>
<td>242 ft</td>
<td>12050</td>
<td>1928</td>
</tr>
<tr>
<td>3</td>
<td>Nichols Bridgeway</td>
<td>pedestrian</td>
<td>620 ft</td>
<td>8200</td>
<td>2009</td>
</tr>
<tr>
<td>4</td>
<td>Clark Street Bridge</td>
<td>bascule</td>
<td>346 ft</td>
<td>72830</td>
<td>1929</td>
</tr>
<tr>
<td>5</td>
<td>BP Pedestrian Bridge</td>
<td>pedestrian</td>
<td>935 ft</td>
<td>17890</td>
<td>2004</td>
</tr>
<tr>
<td>6</td>
<td>Outer Drive Bridge</td>
<td>bascule</td>
<td>480 ft</td>
<td>40000</td>
<td>1937</td>
</tr>
<tr>
<td>7</td>
<td>Sky Ride</td>
<td>ferry</td>
<td>3200 ft</td>
<td>65000</td>
<td>1933</td>
</tr>
<tr>
<td>8</td>
<td>Kinzie Street Bridge</td>
<td>bascule</td>
<td>196 ft</td>
<td>0</td>
<td>1908</td>
</tr>
</tbody>
</table>
Traffic_Lights Table

CREATE TABLE IF NOT EXISTS traffic_lights (  
    tlight_id SERIAL NOT NULL,  
    location VARCHAR(50) NOT NULL,  
    last_maintained DATE NOT NULL,  
    PRIMARY KEY (tlight_id)  
);

Functional Dependencies  
tlight_id -> location, last_maintained

<table>
<thead>
<tr>
<th>tlight_id</th>
<th>location</th>
<th>last_maintained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Brown St and Park Ave</td>
<td>2007-04-27</td>
</tr>
<tr>
<td>2</td>
<td>N Kennedy St and Fairbanks Ct</td>
<td>2009-06-16</td>
</tr>
<tr>
<td>3</td>
<td>Meyer Ave and Damien Ave</td>
<td>2010-07-02</td>
</tr>
<tr>
<td>4</td>
<td>Atkins St and Bay Ave</td>
<td>2008-02-09</td>
</tr>
<tr>
<td>5</td>
<td>W 38 St and Kemper Pl</td>
<td>2013-05-18</td>
</tr>
<tr>
<td>6</td>
<td>N Emmett St and Felton Ave</td>
<td>2015-05-01</td>
</tr>
<tr>
<td>7</td>
<td>S Independence Blvd and 29 St</td>
<td>2014-12-12</td>
</tr>
</tbody>
</table>
**Security_Cameras Table:** The location of security cameras are usually in intersections, but can be located at the end of certain streets. In this situation, the precise address is recorded.

```sql
CREATE TABLE IF NOT EXISTS security_cameras (
    cam_id SERIAL NOT NULL,
    location VARCHAR(50) NOT NULL,
    last_maintained DATE NOT NULL,
    PRIMARY KEY (cam_id)
);
```

**Functional Dependencies**
cam_id -> location, last_maintained

<table>
<thead>
<tr>
<th>cam_id</th>
<th>location</th>
<th>last_maintained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S Ingleside Ave and Raven Rd</td>
<td>2014-03-17</td>
</tr>
<tr>
<td>2</td>
<td>492 Bandle Pl</td>
<td>2007-05-11</td>
</tr>
<tr>
<td>3</td>
<td>Lawndale Ave and Princeton Ave</td>
<td>2010-07-02</td>
</tr>
<tr>
<td>4</td>
<td>Quinn St and S Prospect Ave</td>
<td>2009-01-02</td>
</tr>
<tr>
<td>5</td>
<td>Atkins St and Bay Ave</td>
<td>2011-05-16</td>
</tr>
<tr>
<td>6</td>
<td>12 W 89th St</td>
<td>2008-11-01</td>
</tr>
<tr>
<td>7</td>
<td>N Kennedy St and Fairbanks Ct</td>
<td>2013-10-12</td>
</tr>
</tbody>
</table>
Transformers Table

CREATE TABLE IF NOT EXISTS transformers (  
    transformer_id SERIAL NOT NULL,  
    location VARCHAR(50) NOT NULL,  
    last_maintained DATE NOT NULL,  
    PRIMARY KEY (transformer_id)  
);

Functional Dependencies  
transformer_id -> location, last_maintained

<table>
<thead>
<tr>
<th>transformer_id</th>
<th>location</th>
<th>last_maintained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42 Riverside Rd</td>
<td>2015-02-14</td>
</tr>
<tr>
<td>2</td>
<td>132 Park Ave</td>
<td>2013-12-03</td>
</tr>
<tr>
<td>3</td>
<td>4 N Kennedy St</td>
<td>2008-09-12</td>
</tr>
<tr>
<td>4</td>
<td>92 Emmett St</td>
<td>2010-10-10</td>
</tr>
<tr>
<td>5</td>
<td>1439 Atkins St</td>
<td>2011-04-20</td>
</tr>
<tr>
<td>6</td>
<td>2 W 89th St</td>
<td>2009-09-11</td>
</tr>
<tr>
<td>7</td>
<td>50 Fairbanks Ct</td>
<td>2012-05-01</td>
</tr>
</tbody>
</table>
**Infrastructure_Bridges Table**

```sql
CREATE TABLE IF NOT EXISTS infrastructure_bridges (  
  ifst_id INTEGER NOT NULL,  
  bridge_id INTEGER NOT NULL,  
  PRIMARY KEY (ifst_id, bridge_id),  
  FOREIGN KEY (ifst_id) REFERENCES infrastructure(ifst_id),  
  FOREIGN KEY (bridge_id) REFERENCES bridges(bridge_id)
);
```

Functional Dependencies
ifst_id, bridge_id ->

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>bridge_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>
**Infrastructure_Traffic_Lights Table**

CREATE TABLE IF NOT EXISTS infrastructure_traffic_lights (  
    ifst_id INTEGER NOT NULL,  
    tlight_id INTEGER NOT NULL,  
    PRIMARY KEY (ifst_id, tlight_id),  
    FOREIGN KEY (ifst_id) REFERENCES infrastructure(ifst_id),  
    FOREIGN KEY (tlight_id) REFERENCES traffic_lights(tlight_id)  
);  

Functional Dependencies  
ifst_id, tlight_id ->

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>tlight_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
**Infrastructure_Security_Cameras Table**

```sql
CREATE TABLE IF NOT EXISTS infrastructure_security_cameras (
    ifst_id INTEGER NOT NULL,
    cam_id INTEGER NOT NULL,
    PRIMARY KEY (ifst_id, cam_id),
    FOREIGN KEY (ifst_id) REFERENCES infrastructure(ifst_id),
    FOREIGN KEY (cam_id) REFERENCES security_cameras(cam_id)
);
```

Functional Dependencies
ifst_id, cam_id ->

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>cam_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
**Infrastructure_Transformers Table**

CREATE TABLE IF NOT EXISTS infrastructure_transformers (
    ifst_id INTEGER NOT NULL,
    transformer_id INTEGER NOT NULL,
    PRIMARY KEY (ifst_id, transformer_id),
    FOREIGN KEY (ifst_id) REFERENCES infrastructure(ifst_id),
    FOREIGN KEY (transformer_id) REFERENCES transformers(transformer_id)
);  

Functional Dependencies
ifst_id, transformer_id ->

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>transformer_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>
**Transportation Table:** Stores valid, unique identification numbers for the city's various transportation such as subway, ferry and buses.

CREATE TABLE IF NOT EXISTS transportation (
    transport_id SERIAL NOT NULL UNIQUE,
    type VARCHAR(25) NOT NULL,
    description VARCHAR(50),
    PRIMARY KEY (transport_id)
);

Functional Dependencies
transport_id -> type, description

<table>
<thead>
<tr>
<th>transport_id</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bus</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Subway</td>
<td>Irving Park - Belmont</td>
</tr>
<tr>
<td>3</td>
<td>Ferry</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bus</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bus</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Subway</td>
<td>Racine - Forest Park</td>
</tr>
</tbody>
</table>
**Subways Table:** Station determines where the subway starts and stops, route determines the time it takes for the subway to perform its route.

```sql
CREATE TABLE IF NOT EXISTS subways (  
  subway_id SERIAL NOT NULL,  
  start_station VARCHAR(50) NOT NULL,  
  end_station VARCHAR(50) NOT NULL,  
  start_time TIME NOT NULL,  
  end_time TIME NOT NULL,  
  frequency VARCHAR(50) NOT NULL,  
  PRIMARY KEY (subway_id)
);
```

Functional Dependencies

`subway_id` -> `start_station`, `end_station`, `start_time`, `end_time`, `frequency`

<table>
<thead>
<tr>
<th>subway_id</th>
<th>start_station</th>
<th>end_station</th>
<th>start_time</th>
<th>end_time</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O'Hare</td>
<td>Logan Square</td>
<td>08:00:00</td>
<td>10:00:00</td>
<td>15 min</td>
</tr>
<tr>
<td>2</td>
<td>Irving Park</td>
<td>Belmont</td>
<td>09:00:00</td>
<td>10:00:00</td>
<td>12 min</td>
</tr>
<tr>
<td>3</td>
<td>Montrose</td>
<td>Jackson</td>
<td>07:00:00</td>
<td>11:00:00</td>
<td>18 min</td>
</tr>
<tr>
<td>4</td>
<td>Logan Square</td>
<td>Racine</td>
<td>11:00:00</td>
<td>14:00:00</td>
<td>13 min</td>
</tr>
<tr>
<td>5</td>
<td>Jackson</td>
<td>Harlem</td>
<td>12:30:00</td>
<td>15:00:00</td>
<td>16 min</td>
</tr>
<tr>
<td>6</td>
<td>Racine</td>
<td>Forest Park</td>
<td>16:00:00</td>
<td>19:00:00</td>
<td>17 min</td>
</tr>
</tbody>
</table>
**Buses Table:** In addition to the hours of operation, route also covers the frequency of buses.

```sql
CREATE TABLE IF NOT EXISTS buses (  
    bus_id SERIAL NOT NULL,  
    start_station VARCHAR(50) NOT NULL,  
    end_station VARCHAR(50) NOT NULL,  
    start_time TIME NOT NULL,  
    end_time TIME NOT NULL,  
    frequency VARCHAR(50) NOT NULL,  
    day VARCHAR(25) NOT NULL,  
    PRIMARY KEY (bus_id)
);
```

**Functional Dependencies**

bus_id -> start_station, end_station, start_time, end_time, frequency, day

<table>
<thead>
<tr>
<th>bus_id</th>
<th>start_station</th>
<th>end_station</th>
<th>start_time</th>
<th>end_time</th>
<th>frequency</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indiana/35th</td>
<td>Union Station</td>
<td>05:40:00</td>
<td>21:00:00</td>
<td>27 min</td>
<td>Weekdays</td>
</tr>
<tr>
<td>2</td>
<td>St. Lawrence</td>
<td>Fairbanks</td>
<td>04:45:00</td>
<td>23:05:00</td>
<td>15 min</td>
<td>Weekdays</td>
</tr>
<tr>
<td>3</td>
<td>South Shore</td>
<td>Wacker</td>
<td>04:00:00</td>
<td>23:45:00</td>
<td>20 min</td>
<td>Weekdays</td>
</tr>
<tr>
<td>4</td>
<td>South Shore</td>
<td>Wacker</td>
<td>04:45:00</td>
<td>00:05:00</td>
<td>22 min</td>
<td>Saturday</td>
</tr>
<tr>
<td>5</td>
<td>Harrison</td>
<td>Michigan</td>
<td>06:10:00</td>
<td>22:05:00</td>
<td>10 min</td>
<td>Weekdays</td>
</tr>
<tr>
<td>6</td>
<td>Halstead</td>
<td>Broadway</td>
<td>04:05:00</td>
<td>00:30:00</td>
<td>12 min</td>
<td>Sunday</td>
</tr>
</tbody>
</table>
Ferries Table: Ferry stations are denominated by direction.

CREATE TABLE IF NOT EXISTS ferries (  
    ferry_id SERIAL NOT NULL,  
    start_station VARCHAR(50) NOT NULL,  
    end_station VARCHAR(50) NOT NULL,  
    frequency VARCHAR(50) NOT NULL,  
    PRIMARY KEY (ferry_id)  
);  

Functional Dependencies  
ferry_id \rightarrow start_station, end_station, frequency

<table>
<thead>
<tr>
<th>ferry_id</th>
<th>start_station</th>
<th>end_station</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Belfast</td>
<td>Harlem</td>
<td>20 min</td>
</tr>
<tr>
<td>2</td>
<td>Boruch</td>
<td>Radon</td>
<td>30 min</td>
</tr>
<tr>
<td>3</td>
<td>Harlem</td>
<td>Belfast</td>
<td>35 min</td>
</tr>
<tr>
<td>4</td>
<td>East Side</td>
<td>Grant</td>
<td>25 min</td>
</tr>
<tr>
<td>5</td>
<td>East Side</td>
<td>West Side</td>
<td>27 min</td>
</tr>
<tr>
<td>6</td>
<td>Radon</td>
<td>Boruch</td>
<td>32 min</td>
</tr>
</tbody>
</table>
**Transport_Subways Table**

CREATE TABLE IF NOT EXISTS transport_subways (
    transport_id    INTEGER    NOT NULL,
    subway_id       INTEGER    NOT NULL,
    PRIMARY KEY (transport_id, subway_id),
    FOREIGN KEY (transport_id) REFERENCES transportation(transport_id),
    FOREIGN KEY (subway_id) REFERENCES subways(subway_id)
);

Functional Dependencies
transport_id, subway_id ->

<table>
<thead>
<tr>
<th>transport_id</th>
<th>subway_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>
Transport_Buses Table

CREATE TABLE IF NOT EXISTS transport_buses (  
    transport_id INTEGER NOT NULL,  
    bus_id INTEGER NOT NULL,  
    PRIMARY KEY (transport_id, bus_id),  
    FOREIGN KEY (transport_id) REFERENCES transportation(transport_id),  
    FOREIGN KEY (bus_id) REFERENCES buses(bus_id)  
);

Functional Dependencies
transport_id, bus_id ->

<table>
<thead>
<tr>
<th>transport_id</th>
<th>bus_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
**Transport_Ferries Table**

CREATE TABLE IF NOT EXISTS transport_ferries (
  transport_id INTEGER NOT NULL,
  ferry_id INTEGER NOT NULL,
  PRIMARY KEY (transport_id, ferry_id),
  FOREIGN KEY (transport_id) REFERENCES transportation(transport_id),
  FOREIGN KEY (ferry_id) REFERENCES ferries(ferry_id)
);

Functional Dependencies
transport_id, ferry_id ->

<table>
<thead>
<tr>
<th>transport_id</th>
<th>ferry_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
**Central_OS Table:** This table is, at its most basic purpose, meant to record all unique ids in one place.

CREATE TABLE IF NOT EXISTS central_os (
    ifst_id INTEGER NOT NULL,
    person_id INTEGER NOT NULL,
    transport_id INTEGER NOT NULL,
    PRIMARY KEY (ifst_id, person_id, transport_id),
    FOREIGN KEY (ifst_id) REFERENCES infrastructure(ifst_id),
    FOREIGN KEY (person_id) REFERENCES profiler(person_id),
    FOREIGN KEY (transport_id) REFERENCES transportation(transport_id)
);

Functional Dependencies
ifst_id, person_id, transport_id ->

<table>
<thead>
<tr>
<th>ifst_id</th>
<th>person_id</th>
<th>transport_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Christopher Lee
Profiler Table: The ctOS Profiler tracks people and keeps records of personal information.

CREATE TABLE IF NOT EXISTS profiler (  
  person_id SERIAL NOT NULL,  
  first_name VARCHAR(50) NOT NULL,  
  middle_name VARCHAR(50),  
  last_name VARCHAR(50) NOT NULL,  
  birth_date DATE NOT NULL,  
  gender CHAR(1) NOT NULL,  
  address VARCHAR(50) NOT NULL,  
  phone_number CHAR(15) NOT NULL,  
  email CHAR(256) NOT NULL,  
  eye_color VARCHAR(25) NOT NULL,  
  hair_color VARCHAR(25) NOT NULL,  
  CONSTRAINT valid_gender CHECK (gender = 'M' OR gender = 'F'),  
  PRIMARY KEY (person_id)  
);

Functional Dependencies  
person_id -> first_name, middle_name, last_name, birth_date, gender, address, phone_number, email, eye_color, hair_color

Sample table on next page →
<table>
<thead>
<tr>
<th>person_id</th>
<th>first_name</th>
<th>middle_name</th>
<th>last_name</th>
<th>birth_date</th>
<th>gender</th>
<th>address</th>
<th>phone_number</th>
<th>email</th>
<th>eye_color</th>
<th>hair_color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob</td>
<td>Randal</td>
<td>Tarly</td>
<td>1989-04-20</td>
<td>M</td>
<td>123 Kenny Ln</td>
<td>312-483-2035</td>
<td><a href="mailto:bob_tarly@icloud.com">bob_tarly@icloud.com</a></td>
<td>Blue</td>
<td>Blonde</td>
</tr>
<tr>
<td>2</td>
<td>Frank</td>
<td>Underwood</td>
<td></td>
<td>1956-08-25</td>
<td>M</td>
<td>426 Rook St</td>
<td>312-928-3058</td>
<td><a href="mailto:funderwood@gmail.com">funderwood@gmail.com</a></td>
<td>Black</td>
<td>Brown</td>
</tr>
<tr>
<td>3</td>
<td>Jaime</td>
<td>Lannister</td>
<td></td>
<td>1967-09-11</td>
<td>M</td>
<td>1 Casterly Rock Rd</td>
<td>312-312-3120</td>
<td><a href="mailto:kingslayer@hotmail.com">kingslayer@hotmail.com</a></td>
<td>Green</td>
<td>Blonde</td>
</tr>
<tr>
<td>4</td>
<td>Grace</td>
<td>Rose</td>
<td>Kelly</td>
<td>1990-07-16</td>
<td>F</td>
<td>92 Flower Ln</td>
<td>312-213-9999</td>
<td><a href="mailto:grace_kelly@gmail.com">grace_kelly@gmail.com</a></td>
<td>Gray</td>
<td>White</td>
</tr>
<tr>
<td>5</td>
<td>Eileen</td>
<td>Calvin</td>
<td>Hobbes</td>
<td>1970-01-01</td>
<td>F</td>
<td>172 Brooks Rd</td>
<td>312-183-5720</td>
<td><a href="mailto:calvin_hobbes@gmail.com">calvin_hobbes@gmail.com</a></td>
<td>Black</td>
<td>Black</td>
</tr>
<tr>
<td>6</td>
<td>Rhodes</td>
<td>Rodney</td>
<td></td>
<td>1930-02-15</td>
<td>M</td>
<td>304 56th St</td>
<td>572-381-3957</td>
<td><a href="mailto:rrodney12@hotmail.com">rrodney12@hotmail.com</a></td>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>7</td>
<td>Susan</td>
<td>Dumont</td>
<td>Morgan</td>
<td>1948-03-10</td>
<td>F</td>
<td>95 Flower Ln</td>
<td>572-395-2934</td>
<td><a href="mailto:susan_morgan@gmail.com">susan_morgan@gmail.com</a></td>
<td>Pale</td>
<td>Red</td>
</tr>
<tr>
<td>8</td>
<td>Cersei</td>
<td>Baratheon</td>
<td></td>
<td>1975-12-25</td>
<td>F</td>
<td>1 King's Landing Rd</td>
<td>312-304-2950</td>
<td><a href="mailto:stupid_queen@gmail.com">stupid_queen@gmail.com</a></td>
<td>Green</td>
<td>Blonde</td>
</tr>
<tr>
<td>9</td>
<td>Jon</td>
<td>Snow</td>
<td></td>
<td>1982-09-09</td>
<td>M</td>
<td>1 Knows Nothing Rd</td>
<td>312-304-5820</td>
<td><a href="mailto:clueless@hotmail.com">clueless@hotmail.com</a></td>
<td>Black</td>
<td>Black</td>
</tr>
</tbody>
</table>
**Employees Table**

CREATE TABLE IF NOT EXISTS employees (  
  person_id INTEGER NOT NULL,  
  hire_date DATE NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  year_wages_usd MONEY NOT NULL,  
  PRIMARY KEY (person_id),  
  FOREIGN KEY (person_id) REFERENCES Profiler(person_id)  
);

Functional Dependencies
person_id -> hire_date, year_wages_usd

<table>
<thead>
<tr>
<th>person_id</th>
<th>hire_date</th>
<th>year_wages_usd</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2002-02-05</td>
<td>42000</td>
</tr>
<tr>
<td>7</td>
<td>2009-04-20</td>
<td>92000</td>
</tr>
<tr>
<td>9</td>
<td>2000-01-17</td>
<td>50000</td>
</tr>
</tbody>
</table>
Affiliates Table: Not to be confused with Blume employees, affiliates are people such as temporary contractors or other such people with temporary connections to Blume.

CREATE TABLE IF NOT EXISTS affiliates ( 
    person_id INTEGER NOT NULL, 
    hire_date DATE NOT NULL DEFAULT CURRENT_TIMESTAMP, 
    contract_length VARCHAR(25) NOT NULL, 
    pay MONEY NOT NULL, 
    PRIMARY KEY (person_id), 
    FOREIGN KEY (person_id) REFERENCES Profiler(person_id)
);

Functional Dependencies
person_id -> hire_date, contract_length, pay

<table>
<thead>
<tr>
<th>person_id</th>
<th>hire_date</th>
<th>contract_length</th>
<th>pay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2001-07-08</td>
<td>16 months</td>
<td>42000</td>
</tr>
<tr>
<td>3</td>
<td>2008-02-16</td>
<td>18 months</td>
<td>50000</td>
</tr>
<tr>
<td>8</td>
<td>2010-10-01</td>
<td>6 months</td>
<td>20000</td>
</tr>
</tbody>
</table>
**EmployeeInformation View:** This view keeps track of every Blume employee's important contact information all in one view display, specifically: full name, phone number, e-mail, hire date, and salary.

```
CREATE OR REPLACE VIEW employeeInformation AS
    SELECT p.first_name,
           p.middle_name,
           p.last_name,
           p.phone_number,
           p.email,
           e.hire_date,
           e.year_wages_usd
    FROM   profiler p,
           employees e
    WHERE  p.person_id = e.person_id
    ORDER BY p.last_name DESC
```

<table>
<thead>
<tr>
<th>first_name</th>
<th>middle_name</th>
<th>last_name</th>
<th>phone_number</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jon</td>
<td>Snow</td>
<td>Snow</td>
<td>312-304-5820</td>
<td><a href="mailto:clueless@hotmail.com">clueless@hotmail.com</a></td>
</tr>
<tr>
<td>Susan</td>
<td>Dumont</td>
<td>Morgan</td>
<td>572-395-2934</td>
<td><a href="mailto:susan_morgan@gmail.com">susan_morgan@gmail.com</a></td>
</tr>
<tr>
<td>Jaime</td>
<td>Lannister</td>
<td></td>
<td>312-312-3120</td>
<td><a href="mailto:kingslayer@hotmail.com">kingslayer@hotmail.com</a></td>
</tr>
</tbody>
</table>
**AffiliateInformation View:** This view keeps track of every Blume affiliate's important contact information all in one view display, specifically: full name, phone number, e-mail, hire date, contract length and salary.

CREATE OR REPLACE VIEW affiliateInformation AS

```
SELECT p.person_id AS Employee ID,
    p.first_name,
    p.middle_name,
    p.last_name,
    p.phone_number,
    p.email,
    a.hire_date,
    a.contract_length,
    a.pay
FROM   profiler p,
    affiliates a
WHERE  p.person_id = a.person_id
ORDER BY p.last_name DESC
```
**TLight_Maintain View:** Traffic lights, electricity transformers and security cameras all need regular maintenance. This view keeps track of the traffic light or transformer that needs the most attention (has the oldest last maintained date).

```
CREATE OR REPLACE VIEW tlight_maintain AS
    SELECT t.tlight_id,
           t.location,
           t.last_maintained
    FROM   traffic_lights t
    ORDER BY t.last_maintained ASC
```

**Transformer_Maintain**

```
CREATE OR REPLACE VIEW transformer_maintain AS
    SELECT t.transformer_id,
           t.location,
           t.last_maintained
    FROM   transformers t
    ORDER BY t.last_maintained ASC
```

**Cam_Maintain**

```
CREATE OR REPLACE VIEW cam_maintain AS
    SELECT s.cam_id,
           s.location,
           s.last_maintained
    FROM   security_cameras s
    ORDER BY s.last_maintained ASC
```
Reports and Queries

**Average Bridge Daily Traffic:** When the traffic load on bridges is especially high on a certain day, this query can be used to find the total average daily traffic in order to manipulate traffic into using certain bridges over others.

```
SELECT b.bridge_id AS BridgeID,
       b.name AS Name,
       avg(b.daily_traffic) AS Avg_Daily_Traffic
FROM   bridges b
WHERE  b.daily_traffic IS NOT NULL
GROUP BY b.bridge_id;
```

**Affiliate Financial Planning:** Blume Corporation hires many affiliates and independent contractors, and the finances to hire and train them (if necessary) must be kept track of at all times.

```
SELECT a.contract_length * a.pay AS Financial_Cost,
       p.first_name,
       p.middle_name,
       p.last_name
FROM   affiliates a,
        profiler p
WHERE  a.person_id = p.person_id
GROUP BY Financial_Cost;
```
**Population Percentage**: For managing censuses and keeping track of population percentages, this query returns the percentage of people under 21.

```sql
SELECT TRUNC ( CAST ( 
    (SELECT COUNT(person_id) as count
     FROM Profiler
     WHERE date_part('year', age( Profiler.birth_date )) < 21
     ) as decimal(5, 2)
    ) / (SELECT COUNT(person_id) as total
     FROM Profiler
     ) * 100
  ) as Underage
```
Potential Criminal Search: Given a set of physical attributes such as eye color, hair color and gender, the Profiler can access all the citizens in Chicago as an initial step to find a potential criminal.

CREATE OR REPLACE FUNCTION potential_crime(eye_color text, hair_color text, gender CHAR(1))
RETURNS TABLE(First_Name text, Middle_Name text, Last_Name text) AS
$BODY$
BEGIN
    SELECT DISTINCT p.first_name, p.middle_name, p.last_name
    FROM Profiler p
    WHERE eye_color = p.eye_color
    AND hair_color = p.hair_color
    AND gender = p.gender
END;
$BODY$
LANGUAGE plpgsql;
**New Employee Hire**: While all citizens are automatically tracked and registered by the ctOS Profiler, new employee and affiliate hires must be managed by the database separately.

```sql
CREATE OR REPLACE FUNCTION new_employee()
RETURNS trigger AS $$
BEGIN
    IF NEW.is_employee = true THEN
        INSERT INTO Employees VALUES(NEW.person_id, NEW.hire_date, NEW.year_wages_usd);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql
```

**New Affiliate Hire**

```sql
CREATE OR REPLACE FUNCTION new_affiliate()
RETURNS trigger AS $$
BEGIN
    IF NEW.is_affiliate = true THEN
        INSERT INTO Affiliates VALUES(NEW.person_id, NEW.hire_date, NEW.contract_length, NEW.pay);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql
```
Triggers

New Employee: This example triggers on a new entry being created for a Blume employee who recently moved to Chicago and a new entry must be created in the Profiler.

CREATE TRIGGER add_employee
AFTER INSERT OR UPDATE ON Profiler
FOR EACH ROW
EXECUTE PROCEDURE new_employee();

New Affiliate

CREATE TRIGGER add_affiliate
AFTER INSERT OR UPDATE ON Profiler
FOR EACH ROW
EXECUTE PROCEDURE new_affiliate();
Security

**Admin**: High ranking officials; access to the entire ctOS database.

```
CREATE ROLE admin;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO admin;
```

**Infrastructure Management Employee**: Employees working in the infrastructure department have access to areas of infrastructure only: broadly speaking, this includes security cameras, electric transformers, bridges, and traffic lights.

```
CREATE ROLE ifst_employee;
GRANT SELECT, INSERT, UPDATE ON infrastructure, security_cameras, transformers, bridges, traffic_lights, infrastructure_security_cameras, infrastructure_transformers, infrastructure_bridges, infrastructure_traffic_lights
TO ifst_employee;
```
Transportation Management Employee: Employees working in the transportation department have access to areas of transportation only: broadly speaking, this includes buses, ferries, and subways.

CREATE ROLE trans_employee;
GRANT SELECT, INSERT, UPDATE ON transportation, buses, ferries, subways, transport_buses, transport_ferries, transport_subways
TO trans_employee;

Profiler Management Employee: Employees working in the ctOS Profiler department have access to the profiler and central OS databases for tracking and recording people.

CREATE ROLE profiler_employee;
GRANT SELECT, INSERT, UPDATE ON profiler, central_os
TO profiler_employee;

Affiliates are given similar access permissions, but without INSERT and UPDATE, as those actions are reserved for authorized Blume employees only. The example below shows the permissions for a bridge planner affiliate.

CREATE ROLE bridge_affiliate;
GRANT SELECT ON infrastructure, bridges, infrastructure_bridges
TO bridge_affiliate;
Implementation Notes, Known Problems & Future Enhancements

This database is meant to be an information storage system for an operating system that controls and manages an entire city of millions of people. As a result, this database is in its early, simplest form.

- In the game Watch_Dogs, ctOS is capable of tracking any device capable of connecting to the Internet or any local networks existing in the city of Chicago.
- This includes devices such as laptops, cellphones, smart watches, even satellites overlooking the city of Chicago.
- As a result, improvements to this database system would include coverage of all these devices with unique identification as well as supplementary information.

The Profiler is capable of keeping a “relationship network” between every citizen in the city, allowing for advanced uses such as searches for potential criminals or even predicting crime (by tracking conversations and text messages on phones and computers).

- Future implementations would possibly use Social Security Numbers as unique identification, or use it alongside the current person_id.
- Addition of GPS with the use of locational coordinates would allow ctOS to keep track of every person through the use of Internet-enabled devices.

Finally, this operating system and database system is limited to Chicago only (at the moment). The most significant future enhancement would be to apply this system to every major city and eventually every city in the world for a global system. Obviously, there will be ethical and moral considerations to take into account, but that is not the purpose of this proposal.