# Table of Contents

Executive Summary .................................................................................................................. 3
ER Diagram ............................................................................................................................... 4
Tables ...................................................................................................................................... 5
Views ...................................................................................................................................... 21
Reports .................................................................................................................................. 28
Stored Procedures ................................................................................................................... 32
Triggers ................................................................................................................................... 37
Security ................................................................................................................................... 42
Implementation Notes ............................................................................................................. 48
Known Issues ............................................................................................................................. 50
Future Enhancements ............................................................................................................... 52
Executive Summary

In March 2015, hostile extraterrestrial activity was confirmed on Earth. In response, a secret council of nations approved the full activation of the XCOM Project, an international joint military organization tasked with combating the alien threat.

With the aliens confirmed hostile, the organization conducts routine combat operations to thwart alien attacks and missions. Since XCOM is tasked with the protection of the entire planet, it is critical they operate multiple bases and have teams ready to go at all times.

The purpose of this database is to assist XCOM in the assignment and deployment of soldiers towards counter-alien operations. The database tracks all agents of XCOM (including soldiers), strike teams of soldiers, and XCOM bases. It also tracks alien events and the deployment of strike teams to events. This will allow the Commander to quickly assess the situation and rapidly deploy teams with the most information at hand.

This is imperative to the success of the XCOM project as a team can make or break a mission. Providing this information to the Commander will ensure the greatest success of future missions and the survival of humanity as a whole.
Regions

The region table lists all regions used to approximate the location of Bases and alien Events.

Create Statement:
CREATE TABLE Regions (  RID serial UNIQUE NOT NULL,  regionName text,  PRIMARY KEY(RID) );

Functional Dependencies:
RID → regionName
Bases

This table lists all bases currently operated by XCOM.

Create Statement:
CREATE TABLE Bases (  
    BID serial UNIQUE NOT NULL,  
    baseName text,  
    RID integer references Regions(RID) NOT NULL,  
    PRIMARY KEY(BID)  
);

Functional Dependencies:  
BID → baseName, RID

<table>
<thead>
<tr>
<th>bid</th>
<th>basename</th>
<th>rid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Area 51</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Firebase Alpaca</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Jackal Base</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Kennedy Base</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Alps Strikebase</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Asian Coalition Base</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Outback Base</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Pyramid Base</td>
<td>4</td>
</tr>
</tbody>
</table>
ThreatLevels

This table lists the levels of threats used to evaluate alien events

Create Statement:
CREATE TABLE ThreatLevels (threatLevel serial UNIQUE NOT NULL, threatName text, PRIMARY KEY(threatLevel));

Functional Dependencies:
threatLevel → threatName
Events

This table lists the alien events recorded by XCOM which should be responded to.

Create Statement:

```sql
CREATE TABLE Events (
    EID serial UNIQUE NOT NULL,
    codeName text,
    RID integer references Regions(RID),
    threatLevel integer references threatLevels(threatLevel),
    eventDesc text,
    isActive boolean NOT NULL,
    timeDetected timestamp,
    PRIMARY KEY(EID)
);
```

Functional Dependencies:

EID → codeName, RID, threatLevel, eventDesc, isActive, timeDetected

Continued...
## Events

<table>
<thead>
<tr>
<th>eid</th>
<th>codename</th>
<th>rid</th>
<th>threatlevel</th>
<th>eventdesc</th>
<th>isactive</th>
<th>timedetected timestamp without time zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fallen Star</td>
<td>5</td>
<td>3</td>
<td>A UFO touched down in the Nigerian interior</td>
<td>true</td>
<td>2017-04-20 15:10:18</td>
</tr>
<tr>
<td>2</td>
<td>Little Thieves</td>
<td>1</td>
<td>3</td>
<td>Reports of abductions in rural Kansas</td>
<td>false</td>
<td>2017-04-28 02:02:12</td>
</tr>
<tr>
<td>3</td>
<td>Streaked Sky</td>
<td>1</td>
<td>2</td>
<td>Possible UFO spotting in Canada</td>
<td>false</td>
<td>2016-12-15 20:32:08</td>
</tr>
<tr>
<td>4</td>
<td>Vengeful Demon</td>
<td>6</td>
<td>5</td>
<td>Alien attack on Chinese city</td>
<td>false</td>
<td>2017-02-01 10:45:59</td>
</tr>
<tr>
<td>5</td>
<td>Scornful Father</td>
<td>3</td>
<td>1</td>
<td>Signs of alien activity in German forest</td>
<td>false</td>
<td>2017-03-11 04:22:13</td>
</tr>
<tr>
<td>6</td>
<td>Growling Dirt</td>
<td>2</td>
<td>2</td>
<td>Reports of alien scouts in Peruvian outskirts</td>
<td>true</td>
<td>2017-04-20 08:01:02</td>
</tr>
<tr>
<td>7</td>
<td>Big Ocean</td>
<td>1</td>
<td>2</td>
<td>Reports of submerged UFO in Southern Atlantic Ocean</td>
<td>true</td>
<td>2017-04-30 15:13:38</td>
</tr>
</tbody>
</table>
Nations

This table lists all the nations from which XCOM agents are recruited from.

Create Statement:
CREATE TABLE Nations (  
nationCode text UNIQUE NOT NULL,  
nationName text NOT NULL,  
PRIMARY KEY(nationCode)  
);

Functional Dependencies:  
nationCode → nationName

<table>
<thead>
<tr>
<th>nationcode</th>
<th>nationname</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
</tr>
<tr>
<td>MX</td>
<td>Mexico</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
</tr>
<tr>
<td>CL</td>
<td>Chile</td>
</tr>
<tr>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>KR</td>
<td>South Korea</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
</tr>
<tr>
<td>TH</td>
<td>Thailand</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
</tr>
<tr>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
</tr>
<tr>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>ZA</td>
<td>South Africa</td>
</tr>
</tbody>
</table>
AgentStatuses

This table lists all the statuses (such as Active, MIA, KIA) for agents of XCOM.

Create Statement:
CREATE TABLE AgentStatuses (
    statusCode    serial UNIQUE NOT NULL,
    statusName    text,
    PRIMARY KEY(statusCode)
);

Functional Dependencies:
statusCode → statusName

<table>
<thead>
<tr>
<th>statusCode</th>
<th>statusName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KIA</td>
</tr>
<tr>
<td>1</td>
<td>Active</td>
</tr>
<tr>
<td>2</td>
<td>Wounded</td>
</tr>
<tr>
<td>3</td>
<td>Gravely Wounded</td>
</tr>
<tr>
<td>4</td>
<td>MIA</td>
</tr>
<tr>
<td>5</td>
<td>Retired</td>
</tr>
<tr>
<td>6</td>
<td>Removed from Duty</td>
</tr>
</tbody>
</table>
Agents

This table lists all employees of XCOM, which are referred to as “Agents”

Create Statement:

```
CREATE TABLE Agents (  
  AID serial UNIQUE NOT NULL,  
  firstName text,  
  lastName text,  
  DOB date,  
  nationOfOrigin text references Nations(nationCode),  
  statusCode integer references AgentStatuses(statusCode) NOT NULL DEFAULT 1,  
  baseAssignment integer references Bases(BID),  
  PRIMARY KEY(AID)  
);
```

Functional Dependencies:

AID → firstName, lastName, DOB, nationOfOrigin, statusCode, baseAssignment

Continued...
## Agents

<table>
<thead>
<tr>
<th>aid integer</th>
<th>firstname text</th>
<th>lastname text</th>
<th>dob date</th>
<th>nationoforigin text</th>
<th>statusCode integer</th>
<th>baseassign integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
<td>Van Doorn</td>
<td>1987-11-02</td>
<td>US</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Alan</td>
<td>Labouseur</td>
<td>[null]</td>
<td>US</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Maria</td>
<td>Klein</td>
<td>1975-02-23</td>
<td>DE</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Wiktor</td>
<td>Przybylowicz</td>
<td>1992-09-01</td>
<td>PL</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Akio</td>
<td>Takahashi</td>
<td>1993-03-15</td>
<td>JP</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Jung</td>
<td>Kim</td>
<td>1990-01-19</td>
<td>KR</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Kathy</td>
<td>Taylor</td>
<td>1991-07-07</td>
<td>GB</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Tien</td>
<td>Ligrantirhan</td>
<td>1996-05-25</td>
<td>TH</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>David</td>
<td>Windon</td>
<td>1991-11-21</td>
<td>AU</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Juan</td>
<td>Garcia</td>
<td>1989-02-11</td>
<td>MX</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>James</td>
<td>Wells</td>
<td>1985-06-12</td>
<td>GB</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Hanna</td>
<td>Windon</td>
<td>1993-05-27</td>
<td>AU</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>Maria</td>
<td>Perez</td>
<td>1992-11-05</td>
<td>MX</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>Lucas</td>
<td>Qi</td>
<td>1995-02-12</td>
<td>CA</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>Kate</td>
<td>Shepard</td>
<td>1988-11-29</td>
<td>US</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>Hans</td>
<td>Weber</td>
<td>1992-03-11</td>
<td>DE</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
Ranks

This table lists all the ranks given to XCOM soldiers.

Create Statement:
CREATE TABLE Ranks (  
    RID serial UNIQUE NOT NULL,  
    rankName text,  
    PRIMARY KEY(RID)  
);

Functional Dependencies:  
RID → rankName

<table>
<thead>
<tr>
<th>rid integer</th>
<th>rankname text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Squaddie</td>
</tr>
<tr>
<td>2</td>
<td>Lance Corporal</td>
</tr>
<tr>
<td>3</td>
<td>Corporal</td>
</tr>
<tr>
<td>4</td>
<td>Sergeant</td>
</tr>
<tr>
<td>5</td>
<td>Staff Sergeant</td>
</tr>
<tr>
<td>6</td>
<td>Master Sergeant</td>
</tr>
<tr>
<td>7</td>
<td>Lieutenant</td>
</tr>
<tr>
<td>8</td>
<td>Captain</td>
</tr>
<tr>
<td>9</td>
<td>Major</td>
</tr>
<tr>
<td>10</td>
<td>Colonel</td>
</tr>
<tr>
<td>11</td>
<td>Field Commander</td>
</tr>
</tbody>
</table>
Classes

This table lists all the skill classes XCOM soldiers can classified in.

Create Statement:
CREATE TABLE Classes (  
    CID serial UNIQUE NOT NULL,  
    className text,  
    PRIMARY KEY(CID)  
);

Functional Dependencies:
CID → className

<table>
<thead>
<tr>
<th>cid</th>
<th>classname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rookie</td>
</tr>
<tr>
<td>2</td>
<td>Assault</td>
</tr>
<tr>
<td>3</td>
<td>Grenadier</td>
</tr>
<tr>
<td>4</td>
<td>Gunner</td>
</tr>
<tr>
<td>5</td>
<td>Ranger</td>
</tr>
<tr>
<td>6</td>
<td>Sharpshooter</td>
</tr>
<tr>
<td>7</td>
<td>Shinobi</td>
</tr>
<tr>
<td>8</td>
<td>Specialist</td>
</tr>
<tr>
<td>9</td>
<td>Technical</td>
</tr>
<tr>
<td>10</td>
<td>Psi Operative</td>
</tr>
</tbody>
</table>
Striketeams

This table lists the teams XCOM soldiers can be organized into.

Create Statement:
CREATE TABLE Striketeams (  
    TID serial UNIQUE NOT NULL,  
    teamName text,  
    baseOfOperation integer references Bases(BID),  
    PRIMARY KEY(TID)  
)

Functional Dependencies:  
TID → teamName, baseOfOperation
Soldiers

This table lists all XCOM agents which are classified as soldiers for anti-alien operations.

Create Statement:
CREATE TABLE Soldiers (  
  AID integer references Agents(AID) UNIQUE NOT NULL,  
  codeName text,  
  rank integer references Ranks(RID) DEFAULT 1,  
  class integer references Classes(CID) DEFAULT 1,  
  TID integer references Striketeams(TID),  
  PRIMARY KEY(AID)  
);  

Functional Dependencies:  
AID → codeName, rank, class, TID  

Continued...
## Soldiers

<table>
<thead>
<tr>
<th>aid integer</th>
<th>codename</th>
<th>rank integer</th>
<th>class integer</th>
<th>tid integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The General</td>
<td>10</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>The Normalizer</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>1</td>
<td>[null]</td>
</tr>
<tr>
<td>5</td>
<td>Wolf</td>
<td>3</td>
<td>2</td>
<td>[null]</td>
</tr>
<tr>
<td>7</td>
<td>Kat</td>
<td>2</td>
<td>5</td>
<td>[null]</td>
</tr>
<tr>
<td>8</td>
<td>Flourish</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Goat</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>H.G.</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Sparkle</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>Dragontooth</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>Maple</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>Renegade</td>
<td>9</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>Viper</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>
StriketeamDeployments

This table lists the missions striketeams are deployed on.

Create Statement:

```sql
CREATE TABLE StriketeamDeployments (  
    MID serial UNIQUE NOT NULL,  
    TID integer references Striketeams(TID) NOT NULL,  
    EID integer references Events(EID) NOT NULL,  
    timeOfDeployment timestamp,  
    isDeployed boolean NOT NULL,  
    PRIMARY KEY(MID)
);
```

Functional Dependencies:
MID → TID, EID, timeOfDeployment, isDeployed

<table>
<thead>
<tr>
<th>mid</th>
<th>tid</th>
<th>eid</th>
<th>timeofdeployment</th>
<th>isdeployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2017-04-20 08:12:32</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2017-03-11 08:52:22</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2017-02-01 11:01:22</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2016-12-16 01:22:21</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2017-04-28 02:04:03</td>
<td>false</td>
</tr>
</tbody>
</table>
Views
SoldierInfo

Returns all relevant info about each soldier with readability in mind.

Create Statement:
CREATE OR REPLACE VIEW SoldierInfo AS
SELECT
    Soldiers.AID,
    Agents.firstName,
    Agents.lastName,
    Soldiers.CodeName,
    Nations.nationName,
    Ranks.rankName,
    Classes.className,
    Bases.baseName,
    Striketeams.teamName,
    AgentStatuses.statusName,
    Agents.DOB
FROM Soldiers
    INNER JOIN Agents ON Agents.AID = Soldiers.AID
    LEFT JOIN Nations ON Nations.nationCode = Agents.nationOfOrigin
    LEFT JOIN Bases ON Bases.BID = Agents.baseAssignment
    LEFT JOIN Ranks ON Ranks.RID = Soldiers.rank
    LEFT JOIN Classes ON Classes.CID = Soldiers.class
    LEFT JOIN AgentStatuses ON AgentStatuses.statusCode = Agents.statusCode
    LEFT JOIN Striketeams ON Striketeams.TID = Soldiers.TID;

Continued...
# SoldierInfo

<table>
<thead>
<tr>
<th>aid</th>
<th>firstname</th>
<th>lastname</th>
<th>codename</th>
<th>nationname text</th>
<th>rankname text</th>
<th>classname text</th>
<th>basename text</th>
<th>teamname text</th>
<th>statusname text</th>
<th>dob date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
<td>Van Doom</td>
<td>The General</td>
<td>United States of America</td>
<td>Colonel</td>
<td>Ranger</td>
<td>Alps Strikebase</td>
<td>One Man Army</td>
<td>Active</td>
<td>1987-11-02</td>
</tr>
<tr>
<td>2</td>
<td>Alan</td>
<td>Labouseur</td>
<td>The Normalizer</td>
<td>United States of America</td>
<td>Captain</td>
<td>Specialist</td>
<td>Firebase Alpaca</td>
<td>Final Normal Form</td>
<td>Active</td>
<td>[null]</td>
</tr>
<tr>
<td>4</td>
<td>Wiktor</td>
<td>Przybyłowicz</td>
<td>Poland</td>
<td>Squaddie</td>
<td>Rookie</td>
<td>Alps Strikebase</td>
<td>[null]</td>
<td>Active</td>
<td>1992-09-01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Akio</td>
<td>Takahashi</td>
<td>Wolf</td>
<td>Corporal</td>
<td>Assault</td>
<td>Asian Coalition Base</td>
<td>[null]</td>
<td>KIA</td>
<td>1993-03-15</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Kathy</td>
<td>Taylor</td>
<td>Kat</td>
<td>United Kingdom</td>
<td>Lance Corporal</td>
<td>Ranger</td>
<td>Asian Coalition Base</td>
<td>[null]</td>
<td>Wounded</td>
<td>1991-07-07</td>
</tr>
<tr>
<td>8</td>
<td>Tien</td>
<td>Liengtirphan</td>
<td>Flourish</td>
<td>Thailand</td>
<td>Master Sergeant</td>
<td>Shinobi</td>
<td>Firebase Alpaca</td>
<td>Final Normal Form</td>
<td>Active</td>
<td>1995-05-25</td>
</tr>
<tr>
<td>9</td>
<td>David</td>
<td>Windon</td>
<td>Goat</td>
<td>Australia</td>
<td>Sergeant</td>
<td>Gunner</td>
<td>Jackal Base</td>
<td>The Hounds</td>
<td>Active</td>
<td>1991-11-21</td>
</tr>
<tr>
<td>10</td>
<td>Juan</td>
<td>Garcia</td>
<td>Mexico</td>
<td>Corporal</td>
<td>Specialist</td>
<td>Jackal Base</td>
<td>The Hounds</td>
<td>Active</td>
<td>1989-02-11</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>James</td>
<td>Wells</td>
<td>H.G.</td>
<td>United Kingdom</td>
<td>Staff Sergeant</td>
<td>Sharpshooter</td>
<td>Jackal Base</td>
<td>The Hounds</td>
<td>Active</td>
<td>1985-06-12</td>
</tr>
<tr>
<td>12</td>
<td>Hanna</td>
<td>Winson</td>
<td>Sparkle</td>
<td>Australia</td>
<td>Squaddie</td>
<td>Rookie</td>
<td>Outback Base</td>
<td>Wavemakers</td>
<td>Active</td>
<td>1993-05-27</td>
</tr>
<tr>
<td>13</td>
<td>Marla</td>
<td>Perez</td>
<td>Dragontooth</td>
<td>Mexico</td>
<td>Lieutenant</td>
<td>Technical</td>
<td>Outback Base</td>
<td>Wavemakers</td>
<td>Active</td>
<td>1992-11-05</td>
</tr>
<tr>
<td>14</td>
<td>Lucas</td>
<td>Qi</td>
<td>Maple</td>
<td>Canada</td>
<td>Corporal</td>
<td>Gunner</td>
<td>Pyramid Base</td>
<td>Spectres</td>
<td>Active</td>
<td>1995-02-12</td>
</tr>
<tr>
<td>15</td>
<td>Kate</td>
<td>Shepard</td>
<td>Renegade</td>
<td>United States of America</td>
<td>Major</td>
<td>Sharpshooter</td>
<td>Pyramid Base</td>
<td>Spectres</td>
<td>Active</td>
<td>1988-11-29</td>
</tr>
<tr>
<td>16</td>
<td>Hans</td>
<td>Weber</td>
<td>Viper</td>
<td>Germany</td>
<td>Lieutenant</td>
<td>Shinobi</td>
<td>Asian Coalition Base</td>
<td>Snakehead</td>
<td>Active</td>
<td>1992-03-11</td>
</tr>
</tbody>
</table>
UnrespondedEvents

It's important that all alien events are responded to. In order to ensure none are forgotten, the view displays all active events that have not yet had a striketeam dispatched. It also displays how much time has passed since the Event was logged, giving the user a better idea where to focus first.

Create Statement:

```sql
CREATE OR REPLACE VIEW UnrespondedEvents AS
SELECT
    Events.EID,
    Events.CodeName,
    Regions.RegionName,
    ThreatLevels.threatName,
    Events.EventDesc,
    Age(now(), Events.timeDetected) as timeSinceReported
FROM Events
    LEFT JOIN Regions ON Regions.RID = Events.RID
    LEFT JOIN ThreatLevels ON ThreatLevels.ThreatLevel = Events.ThreatLevel
WHERE
    EID NOT IN (SELECT EID FROM StriketeamDeployments)
    AND isActive = True;
```

Continued...
## UnrespondedEvents

<table>
<thead>
<tr>
<th>eid</th>
<th>codename</th>
<th>regionname</th>
<th>threatname</th>
<th>eventdesc</th>
<th>timesincereportedinterval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fallen Star</td>
<td>Africa</td>
<td>Moderate</td>
<td>A UFO touched down in the Nigerian interior</td>
<td>11 days 09:30:51.722134</td>
</tr>
<tr>
<td>7</td>
<td>Big Ocean</td>
<td>North America</td>
<td>Minor</td>
<td>Reports of submerged UFO in Southern Atlantic Ocean</td>
<td>1 day 09:27:31.722134</td>
</tr>
</tbody>
</table>
EventHistory

It is important to review past events, as well as XCOM’s response to them. This view displays every logged Event alongside the records of what team(s) responded to it.

The time it took to for the striketeam to be deployed is calculated and displayed, where applicable.

The data is ordered by Event date, sorted in Descending order.

Create Statement:
CREATE OR REPLACE VIEW EventHistory AS
SELECT
    Events.EID,
    Events.CodeName,
    Regions.RegionName,
    ThreatLevels.threatName,
    Events.EventDesc,
    Events.timeDetected,
    Striketeams.teamName as RespondingTeam,
    Age(StriketeamDeployments.timeOfDeployment, Events.timeDetected) as ResponseTime,
    Events.isActive
FROM Events
    LEFT JOIN Regions ON Regions.RID = Events.RID
    LEFT JOIN ThreatLevels ON ThreatLevels.ThreatLevel = Events.ThreatLevel
    LEFT JOIN StriketeamDeployments ON StriketeamDeployments.EID = Events.EID
    LEFT JOIN Striketeams ON Striketeams.TID = StriketeamDeployments.TID
ORDER BY Events.timeDetected DESC;

Continued...
# EventHistory

<table>
<thead>
<tr>
<th>eid</th>
<th>codename</th>
<th>regionname</th>
<th>threatname</th>
<th>eventdesc</th>
<th>timedetected</th>
<th>respondingteam</th>
<th>responsetimeinterval</th>
<th>inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Big Ocean</td>
<td>North America</td>
<td>Minor</td>
<td>Reports of submerged UFO in Southern Atlantic Oce..</td>
<td>2017-04-30</td>
<td>[null]</td>
<td>[null]</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>Little Thieves</td>
<td>North America</td>
<td>Moderate</td>
<td>Reports of abductions in rural Kansas</td>
<td>2017-04-28</td>
<td>Wavers</td>
<td>00:01:51</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>Fallen Star</td>
<td>Africa</td>
<td>Moderate</td>
<td>A UFO touched down in the Nigerian interior</td>
<td>2017-04-20</td>
<td>[null]</td>
<td>[null]</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>Growling Dirt</td>
<td>South America</td>
<td>Minor</td>
<td>Reports of alien scouts in Peruvian outskirts</td>
<td>2017-04-20</td>
<td>Final Normal Form</td>
<td>00:11:30</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>Scornful Father</td>
<td>Europe</td>
<td>Minimal</td>
<td>Signs of alien activity in German forest</td>
<td>2017-03-11</td>
<td>The Hounds</td>
<td>04:30:09</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>Vengeful Demon</td>
<td>Asia</td>
<td>High</td>
<td>Alien attack on Chinese city</td>
<td>2017-02-01</td>
<td>Snakehead</td>
<td>00:15:23</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>Streaked Sky</td>
<td>North America</td>
<td>Minor</td>
<td>Possible UFO spotting in Canada</td>
<td>2016-12-15</td>
<td>Final Normal Form</td>
<td>04:50:13</td>
<td>false</td>
</tr>
</tbody>
</table>
Reports
Lifetime Events by Region

It is important to know where the most alien activity occurs. This report creates an easy to read table that shows the number of events per region.

Query:

```sql
SELECT Regions.RegionName, count(Events.RID) as LifetimeEvents
FROM Regions
INNER JOIN Events ON Regions.RID = Events.RID
GROUP BY Regions.RegionName
ORDER BY LifetimeEvents DESC;
```
This query retrieves the number of combat ready soldiers for each class. This way XCOM can know what skillsets they are lacking and optimize the training of new troops.

Query:
SELECT Classes.ClassName, count(activeSoldiers.class) as NumSoldiers
FROM Classes
LEFT JOIN (SELECT Soldiers.Class
FROM Soldiers
WHERE Soldiers.AID IN (SELECT Agents.AID
FROM Agents
WHERE Agents.statusCode = 1)) AS ActiveSoldiers
ON ActiveSoldiers.class = Classes.CID
GROUP BY Classes.ClassName
ORDER BY numSoldiers DESC;
Time since last event per Region

Gets time since last event per region. Allows XCOM to see what regions have been hit most recently and can also help see what regions may be "overdue" for hostilities.

The table displays the regions which have had an attack most recently first.

Query:

```sql
SELECT Regions.RegionName,
       Min(Age(Events.TimeDetected)) as TimeSinceLast
FROM Regions
  LEFT JOIN Events ON Regions.RID = Events.RID
GROUP BY Regions.RegionName
ORDER BY TimeSinceLast ASC;
```

<table>
<thead>
<tr>
<th>regionname</th>
<th>timesincelast interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>North America</td>
<td>1 day 08:46:22</td>
</tr>
<tr>
<td>Africa</td>
<td>11 days 08:49:42</td>
</tr>
<tr>
<td>South America</td>
<td>11 days 15:58:58</td>
</tr>
<tr>
<td>Europe</td>
<td>1 mon 21 days 19:37:47</td>
</tr>
<tr>
<td>Asia</td>
<td>3 mons 13:14:01</td>
</tr>
<tr>
<td>Oceania</td>
<td>[null]</td>
</tr>
<tr>
<td>Middle East</td>
<td>[null]</td>
</tr>
</tbody>
</table>
Stored Procedures
rebaseTeam()

Many times a striketeam will be relocated to another base. This can be tedious if updated manually, as each member has to be moved. This has the potential for error. Instead, this function can take a TeamID and a BaseID, and it will relocate each team member to the given base.

The team will then be marked as operating from that base as well.

Create Statement:
CREATE OR REPLACE FUNCTION rebaseTeam(int, int) RETURNS void AS $rebaseTeam$
    DECLARE
        teamID int := $1;
        baseID int := $2;
    BEGIN
        UPDATE Striketeams
        SET baseOfOperation = baseID
        WHERE TID = teamID;
        UPDATE Agents
        SET baseAssignment = baseID
        WHERE AID IN (SELECT AID
                        FROM Soldiers
                        WHERE TID = teamID);
    END; $rebaseTeam$ LANGUAGE plpgsql;
**rebaseTeam() Results**

`rebaseTeam(1, 8)`

<table>
<thead>
<tr>
<th>tid integer</th>
<th>firstname text</th>
<th>lastname text</th>
<th>codename text</th>
<th>rankname text</th>
<th>teamname text</th>
<th>basename text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alan</td>
<td>Labouseur</td>
<td>The Normalizer</td>
<td>Captain</td>
<td>Final Normal Form</td>
<td>Firebase Alpaca</td>
</tr>
<tr>
<td></td>
<td>Tien</td>
<td>Liengtirphan</td>
<td>Flourish</td>
<td>Master Sergeant</td>
<td>Final Normal Form</td>
<td>Firebase Alpaca</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tid integer</th>
<th>firstname text</th>
<th>lastname text</th>
<th>codename text</th>
<th>rankname text</th>
<th>teamname text</th>
<th>basename text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alan</td>
<td>Labouseur</td>
<td>The Normalizer</td>
<td>Captain</td>
<td>Final Normal Form</td>
<td>Pyramid Base</td>
</tr>
<tr>
<td></td>
<td>Tien</td>
<td>Liengtirphan</td>
<td>Flourish</td>
<td>Master Sergeant</td>
<td>Final Normal Form</td>
<td>Pyramid Base</td>
</tr>
</tbody>
</table>
**completeMission()**

When a team returns from a mission, the event should typically be resolved. Using this function, it will set the striketeam’s deployment as no longer deployed, and then mark the event as over. Simply enter the MID for the mission as an argument.

**Create Statement:**

```
CREATE OR REPLACE FUNCTION completeMission(int) RETURNS void AS $completeMission$
DECLARE
    missionID int := $1;
BEGIN
    UPDATE StriketeamDeployments
    SET isDeployed = FALSE
    WHERE MID = missionID;
    UPDATE Events
    SET isActive = FALSE
    WHERE EID in
    (SELECT EID
     FROM StriketeamDeployments
     WHERE MID = missionID);
END
$completeMission$ LANGUAGE plpgsql;
```
# completeMission() Results

## completeMission(1)

<table>
<thead>
<tr>
<th>eid</th>
<th>codename text</th>
<th>rid</th>
<th>threatlevel</th>
<th>eventdesc text</th>
<th>isactive</th>
<th>timedetected timestamp without time zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Growling Dirt</td>
<td>2</td>
<td>2</td>
<td>Reports of alien scouts in Peruvian outskirts</td>
<td>true</td>
<td>2017-04-20 08:01:02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mid</th>
<th>tid</th>
<th>eid</th>
<th>timeofdeployment timestamp without time zone</th>
<th>isdeploy boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2017-04-20 08:12:32</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>eid</th>
<th>codename text</th>
<th>rid</th>
<th>threatlevel</th>
<th>eventdesc text</th>
<th>isactive</th>
<th>timedetected timestamp without time zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Growling Dirt</td>
<td>2</td>
<td>2</td>
<td>Reports of alien scouts in Peruvian outskirts</td>
<td>false</td>
<td>2017-04-20 08:01:02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mid</th>
<th>tid</th>
<th>eid</th>
<th>timeofdeployment timestamp without time zone</th>
<th>isdeploy boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2017-04-20 08:12:32</td>
<td>false</td>
</tr>
</tbody>
</table>
Triggers
A soldier needs to be with their teammates to function as a team. Effectively, a soldier should always be in the same base as the team. This function will automatically relocate a soldier to the proper base when assigned a new team.

Create Statement:
CREATE OR REPLACE FUNCTION moveToTeam()
RETURNS TRIGGER AS
$moveToTeam$
DECLARE
  baseAgent int := NULL;
  baseTeam  int := NULL;
  newTeam   int := NULL;
BEGIN
  SELECT Striketeams.baseOfOperation INTO baseTeam
  FROM Striketeams
  WHERE Striketeams.TID = NEW.TID;
  SELECT Agents.baseAssignment INTO baseAgent
  FROM Agents
  WHERE Agents.AID = NEW.AID;
  IF (baseTeam != baseAgent) THEN
    UPDATE Agents
    SET baseAssignment = baseTeam
    WHERE Agents.AID = NEW.AID;
  END IF;
  RETURN NEW;
END;$moveToTeam$ LANGUAGE plpgsql;

SELECT Striketeams.baseOfOperation INTO baseTeam
FROM Striketeams
WHERE Striketeams.TID = NEW.TID;
SELECT Agents.baseAssignment INTO baseAgent
FROM Agents
WHERE Agents.AID = NEW.AID;
IF (baseTeam != baseAgent) THEN
  UPDATE Agents
  SET baseAssignment = baseTeam
  WHERE Agents.AID = NEW.AID;
END IF;
RETURN NEW;
END;
$moveToTeam$ LANGUAGE plpgsql;

CREATE TRIGGER autoMoveToTeamLocation
BEFORE INSERT OR UPDATE OF TID ON Soldiers
FOR EACH ROW
EXECUTE PROCEDURE moveToTeam();
```sql
UPDATE SOLDIERS
SET TID = 1
WHERE AID = 1;
```

<table>
<thead>
<tr>
<th>aid integer</th>
<th>firstname text</th>
<th>lastname text</th>
<th>basename text</th>
<th>teamname text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
<td>Van Doorn</td>
<td>Alps Strikebase</td>
<td>One Man Army</td>
</tr>
<tr>
<td>2</td>
<td>Alan</td>
<td>Labouseur</td>
<td>Pyramid Base</td>
<td>Final Normal Form</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>aid integer</th>
<th>firstname text</th>
<th>lastname text</th>
<th>basename text</th>
<th>teamname text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
<td>Van Doorn</td>
<td>Pyramid Base</td>
<td>Final Normal Form</td>
</tr>
<tr>
<td>2</td>
<td>Alan</td>
<td>Labouseur</td>
<td>Pyramid Base</td>
<td>Final Normal Form</td>
</tr>
</tbody>
</table>
A striketeam cannot be deployed on a mission when it is already deployed on another mission. This trigger checks to see if the team a user is trying to deploy is already deployed. If so, it raises an exception. Otherwise, it allows the insert to proceed as normal.

Create Statement:
```
CREATE OR REPLACE FUNCTION checkIfActive()
RETURNS TRIGGER AS
$checkIfActive$
BEGIN
  IF (NEW.isDeployed = TRUE) THEN
    IF EXISTS
      (SELECT StriketeamDeployments.isDeployed
       FROM StriketeamDeployments
       WHERE StriketeamDeployments.TID = NEW.TID AND isDeployed = TRUE)
    THEN
      RAISE EXCEPTION 'Cannot deploy a team that is already deployed!';
      RETURN NULL;
    END IF;
  END IF;
  RETURN NEW;
END
$checkIfActive$ LANGUAGE plpgsql;
CREATE TRIGGER check_if_active
BEFORE INSERT ON StriketeamDeployments
FOR EACH ROW
EXECUTE PROCEDURE checkIfActive();
```
check_if_active

```
INSERT INTO StriketeamDeployments (EID, TID, timeOfDeployment, isDeployed) VALUES
(5, 1, '2017-04-20 08:12:32', TRUE);
```

```
mid integer    tid integer    eid integer   timeofdeployment timestamp without time zone   isdeploy boolean
----------    ----------    --------      --------------------------      -------------
        4           1           3   2016-12-16  01:22:21      false
        1           1           6   2017-04-20  08:12:32      true
```

```
ERROR: Cannot deploy a team that is already deployed!
CONTEXT: PL/pgSQL function check_if_active() line 11 at RAISE
********** Error **********

ERROR: Cannot deploy a team that is already deployed!
SQL state: P0001
Context: PL/pgSQL function check_if_active() line 11 at RAISE
```
Security
Admin

The admin role is given to trusted XCOM IT technicians whose main responsibility is overseeing the functionality of the database. This means they have full access to the database in order to maintain it.

Create Statement:

```
CREATE ROLE admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA PUBLIC TO admin;
```
The Commander is the appointed leader of XCOM and oversees ALL of its functions, from recruitment to event response.

The main purpose of this database is to assist him/her in their operation of XCOM. As such, he has almost complete control over the database.

However, there is little reason to give the Commander the ability to DELETE records on most tables, as this should not be required in day-to-day operation. This should only be necessary on Striketeams to remove a team.

Should the need arise, an Admin can be called upon by the Commander.

Create Statement:
CREATE ROLE Commander;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA PUBLIC TO Commander;
GRANT DELETE ON Striketeams TO Commander;
Officer

Officers, or soldiers in combat leadership positions, are allowed to access information about other soldiers for the purposes of putting together striketeams.

However, since the decision is ultimately up to the Commander, they can make no modifications.

Create Statement:

CREATE ROLE Officer;
GRANT SELECT ON
  Agents,
  Nations,
  AgentStatuses,
  Soldiers,
  Striketeams,
  Bases,
  Regions,
  Ranks,
  Classes
TO Officer;
Dispatch

24/7, certain agents are assigned to monitor for alien events. When one is detected, they must be able to log it into the database. This role gives those agents that power.

Create Statement:

```
CREATE ROLE Dispatch;
GRANT SELECT ON
  threatLevels,
  Regions,
  Events
TO Dispatch;
GRANT INSERT, UPDATE
  ON Events
TO Dispatch;
```
The Commander has more important things to do than input the data of all incoming Agents. Therefore, some agents are in charge of the processing of incoming Agents. Thus, they must have the ability to perform the required operations.

Create Statement:
```sql
CREATE ROLE HR;
GRANT SELECT ON
    Nations,
    Agents,
    AgentStatuses,
    Bases,
    Regions,
    Soldiers,
    Classes,
    Ranks
TO HR;
GRANT INSERT, UPDATE
    ON Agents, Soldiers
TO HR;
```
Implementation
Notes
Implementation Notes

-- The primary purpose of this database is the coordination of the military operations of XCOM. Non-combat staff can be placed in the system, but there is no administrative capabilities on them besides their current status.

-- The provided sample data does not include enough non-combat staff to populate all bases. In reality, there should be agents at every base. However, those were omitted in order to provide a clearer picture of the main purpose of the database.
Known Issues
Known Issues

-- In the UnrespondedEvents view, the timeSinceReported values display seconds to an obnoxiously long decimal. This is because the Age() function computes from midnight instead the current time, requiring a calculation instead to avoid, resulting in the decimals.

-- Agents with statuses that should prevent them from serving (such as KIA, retired, etc.) are not prevented from being enrolled in teams.

-- A soldier can be moved to different base from their team, leaving the team fragmented. Ideally, they should be dropped from the team or have the team moved with them.
Future Enhancements
Future Enhancements

-- Map Nations to Regions and use that in place of Regions for Events to better pinpoint location.

-- Add more checks and policies for Soldiers who have been wounded or no longer serving XCOM.

-- Implement more checks on updated/inserted data to avoid erroneous input.

-- Create Table for Skyrangers (Troop Transport Jets) to coordinate Striketeam Deployments. If there are no Skyrangers available at the base, a Striketeam cannot deploy to an event.

-- Expand database to cover the other branches of XCOM staff, such as Engineering and Science departments.