Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU34041 Information Management II
Assignment: MySQL Database

Federal Bureau of Control Relational DB

Daniel Whelan, 19335045

March , 2022

Contents

Section A: Description of Database Application area and ER Model..... 2

A.1 Application Description 2

A.2 Entity Relationship Diagram 3

3

A.3 Mapping to Relationship Schema 4

A.4 Functional Dependency Diagrams.....5

Section B: Explanation of Data and SQL Code 6

B.1 Table Creation and Constraints 6

B.2 Altering Tables 7

B.3 Trigger Operations 7

B.4 Views.....7

B.5 Commands to Populate Tables 8

B.6 Retrieving Information from Database 8

B.7 Security.....9

Section A: Description of Database Application area and ER Model

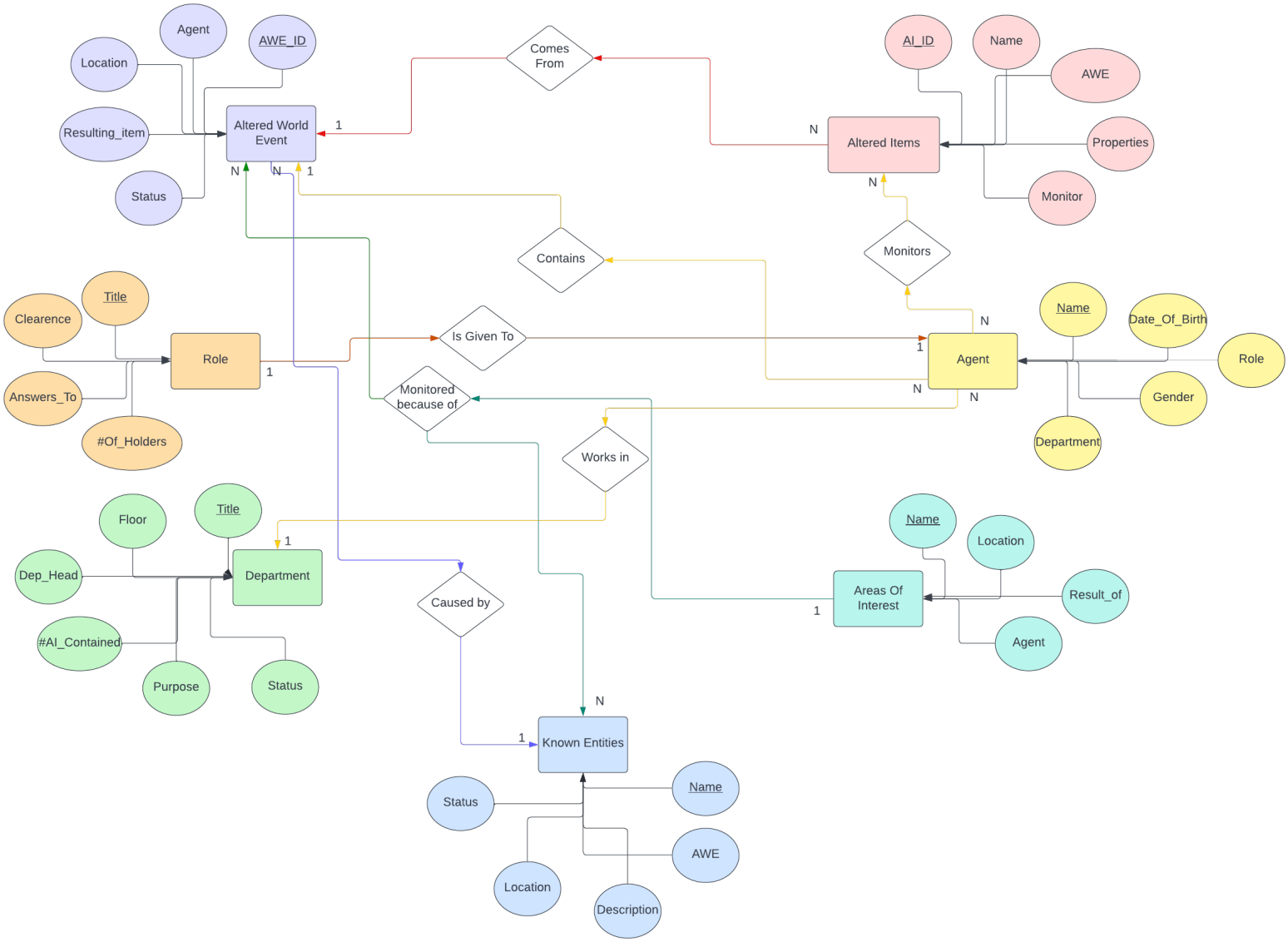
A.1 Application Description

I created this database based on a fictional secret government department known as the Federal Bureau of Control (FBC). This department responds to Altered World Events (AWEs) in which Altered Items (AIs) are created. The FBC has a series of Agents that respond to these events, each of which designated a hierarchical Role in the agency and working within a specified Department within the agency. These AWEs result in Areas of Interest that are watched by a specified agent and are caused by Known Entities which are kept track of.

I will represent the FBC using tables for Altered World Events, Altered Items, Agents, Roles, Departments, Areas of Interest and Known Entities. The attributes of each table will be as follows.

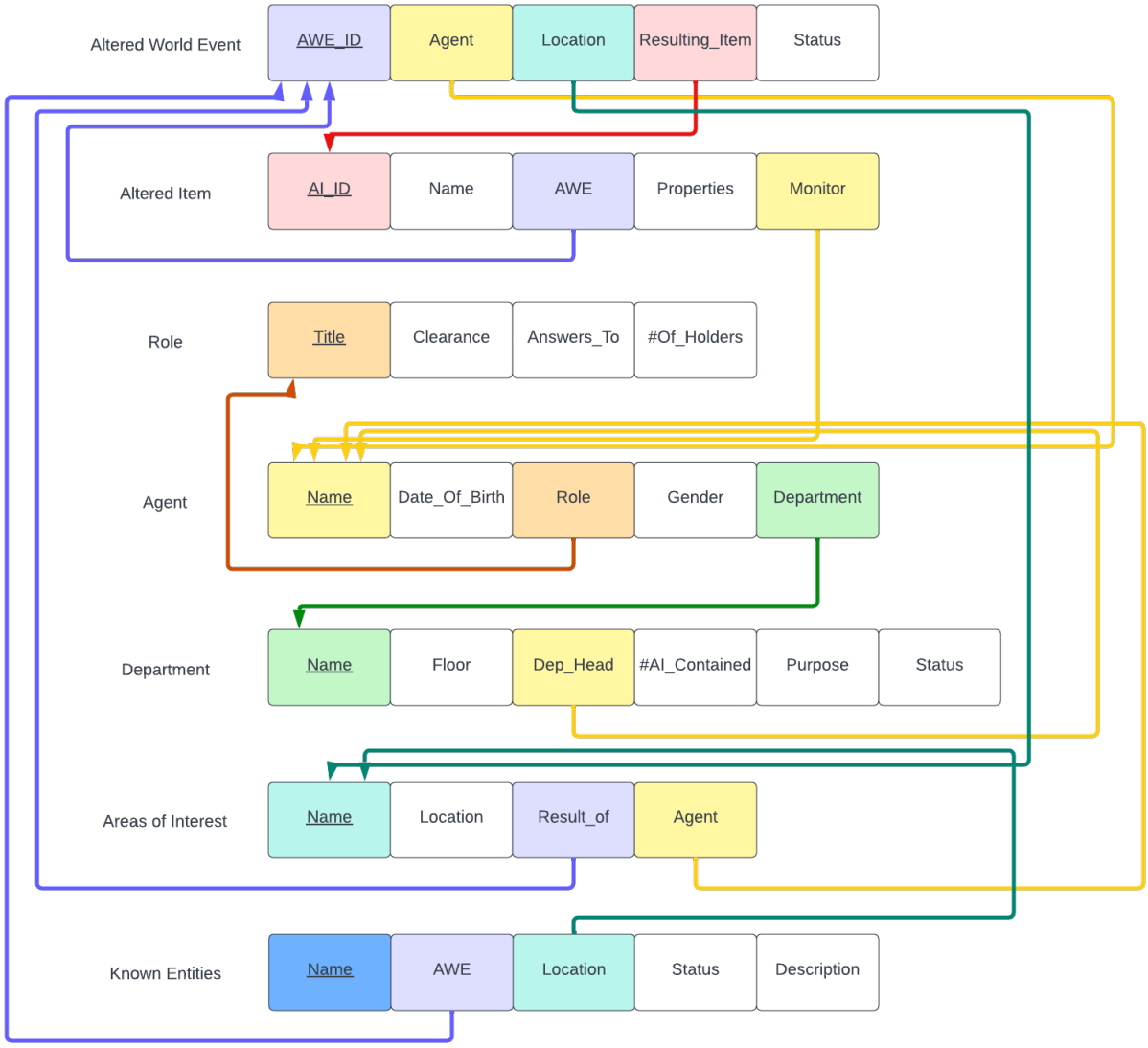
- **Altered World Events (AWE)**
  An AWE has a given ID number (in the form AWE-x), an agent assigned to this AWE, the location in which it occurred, the Altered Item that resulted from the AWE and the current status of the AWE
- **Altered Items (AI)**
  An AI also has a given ID number (in the form AI-x), a name, the AWE it is associated with, any properties of the AI and the agent that is assigned to monitor the AI.
- **Agent**
  Each Agent has a unique name given by the FBC, a date of birth, the role that they have been assigned in the agency, a Gender and the department in which they work.
- **Role**
  Each Role has a unique title that it is associated with, It has a level of clearance, it also has whom it answers to in the hierarchical structure of the FBC and the number of current holders of that specified role.
- **Department**
  Each Department has a unique name associated with it, a floor of the FBC headquarters (The Oldest House) that it is located on, a head of department, the number of AIs contained within that department, a specified purpose alongside the current status of that department.
- **Area of Interest**
  Each area of interest has a unique name and the physical location in which it is based. It also keeps track of the most recent AWE that resulted in the area being of interest and the agent that has been assigned to monitor the area.
- **Known Entities**
  All Entities have a unique name, the AWE that made the FBC aware of the entity (multiple entities can be known from one AWE), the location that the entity currently occupies, the status of the entity and also a brief description of who, or what, the entity is.

A.2 Entity Relationship Diagram

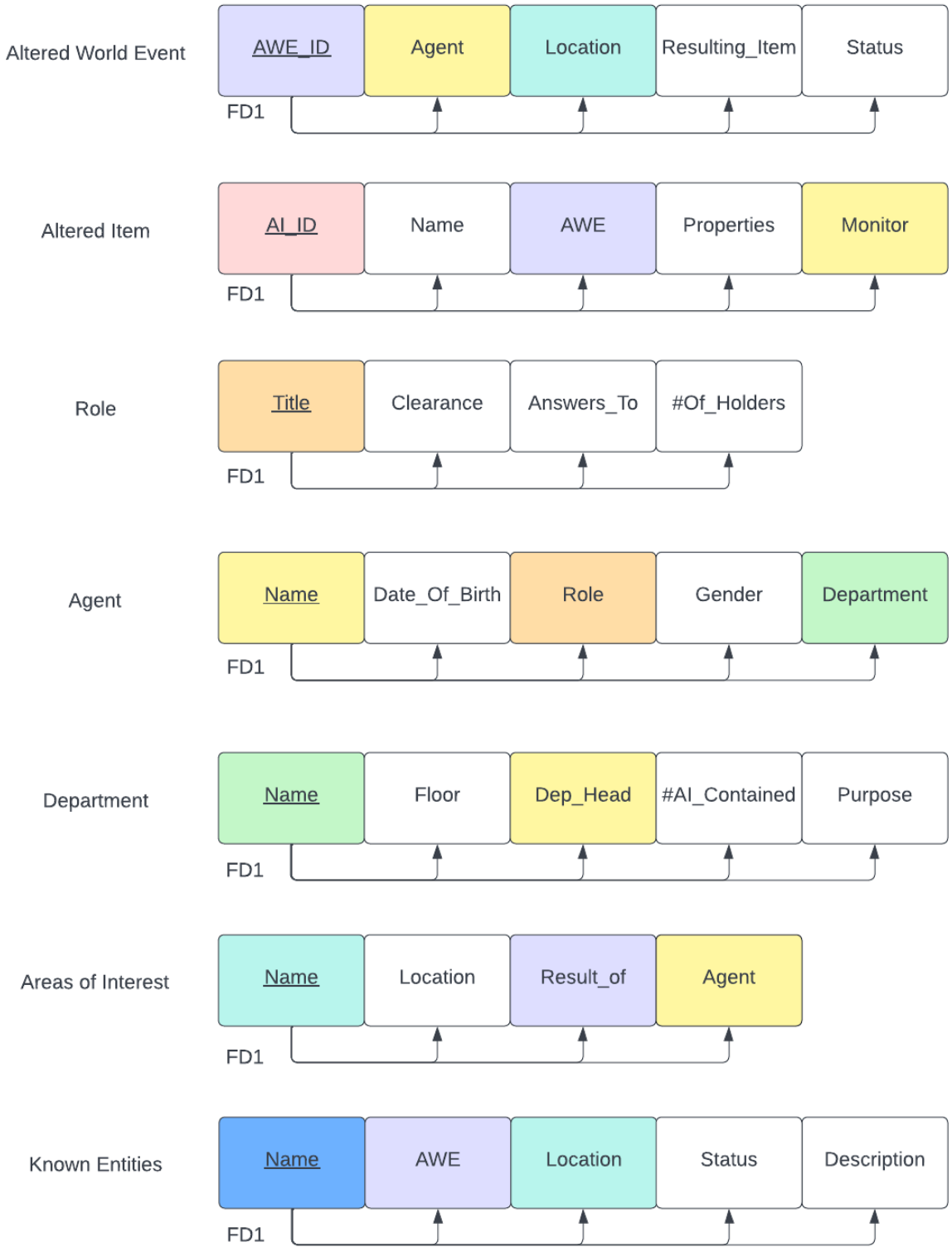


Note: Underlined Attributes highlight a Primary Key
Note: Arrows are colour coded based on Table they stem from

A.3 Mapping to Relationship Schema



A.4 Functional Dependency Diagrams



Note: Underlined Attributes are Primary Keys
Note: Coloured arrows and boxes highlight Primary-Foreign key links

Section B: Explanation of Data and SQL Code

B.1 Table Creation and Constraints

In the creation of the tables I ensured that all attributes were added in and that if the value of the attribute was a variable string input that the length of these using a maximum VARCHAR length. As a result the values that are entered into these attributes are appropriately constrained. All other values are constrained by their data type hence no further action is required with these attributes.

Following this there are also a number of NOT NULL and UNIQUE instructions in order to ensure that primary keys are unique and cannot be null and to prevent foreign keys from referencing elements that are not in the primary key that they reference.

Some constraints unique to this database relate to the status attributes that are stored in the tables created for this database. For example the AWE status attribute is constrained to two options, either the AWE investigation and file is 'Ongoing' or 'Closed'.
The status of Departments are also constrained to three different options, either the department is 'Normal' and it is business as usual, 'Danger' and there is an ongoing threat in the department or 'Compromised' and the department is no longer functioning and operational.
Another status constrain consists with the nature of the Known Entities. Either the Entity is 'Friendly' or 'Dangerous'. If the nature of the Entity cannot be discerned, it is stored in the database as 'Unknown'.

```
CONSTRAINT `ENTITY_STATUS` CHECK ((`STATUS` = 'Friendly') OR
(`STATUS` = 'Dangerous') OR (`STATUS` = 'Unknown'))
```

Status Constraint for Known Entity Table

```
CONSTRAINT `DEP_STATUS` CHECK ((`STATUS` = 'Normal') OR
(`STATUS` = 'Danger') OR (`STATUS` = 'Compromised'))
```

Status Constraint for Department Table

```
CONSTRAINT `AWE_STATUS` CHECK ((`STATUS`= 'Ongoing') OR
(`STATUS` = 'Closed')),
```

Status Constraint for Altered World Event Table

Another semantic constraint to this database is related to the clearance level of the roles that are present in the FBC. This numerical constraint ensures that the value entered into the 'CLEARANCE' attribute of the Role table is less than or equal to the maximum clearance level of 5 and is greater than the minimum clearance level of 1.

```
CONSTRAINT `CLEARANCE_LEVEL` CHECK ((`CLEARANCE` >= 1) AND
(`CLEARANCE` <= 5)),
```

Clearance constraint for the Role Table.

B.2 Altering Tables

The only element of the database that regarded the use of altering tables was the introduction of foreign keys into the database. When declaring the foreign keys I used the ALTER TABLE command. This was done in order to ease the understandability of the SQL code. By excluding the insertion of foreign keys from the main CREATE TABLE command, the code is more readable and it becomes more obvious to see where there are foreign keys and which tables they reference

```
ALTER TABLE `ALTERED_WORLD_EVENT` ADD FOREIGN KEY (`AGENT`) REFERENCES
`AGENT`(`NAME`);
ALTER TABLE `ALTERED_WORLD_EVENT` ADD FOREIGN KEY (`LOCATION`) REFERENCES
`AREAS_OF_INTEREST`(`NAME`);
ALTER TABLE `ALTERED_WORLD_EVENT` ADD FOREIGN KEY (`RESULTING_ITEM`)
REFERENCES `ALTERED_ITEMS`(`AI_ID`);
```

ALTER TABLE commands to highlight foreign keys in the ALTERED WORLD EVENTS table.

B.3 Trigger Operations

In this database I have introduced one Trigger that would be used, see below:

```
DELIMITER //
CREATE TRIGGER new_awe AFTER INSERT ON altered_world_event
FOR EACH ROW
BEGIN
        INSERT INTO altered_item SET ai_id = new.resulting_item;
   IF NOT EXISTS(SELECT * FROM areas_of_interest WHERE(areas_of_interest.name =
NEW.location))
                THEN INSERT INTO areas_of_interest SET name = new.location;
   ELSE
                UPDATE areas_of_interest SET result_of = new.awe_id;
        END IF;
END; //
DELIMITER ;
```

In this particular trigger, whenever a new AWE is logged in the altered_world_event table, the trigger inserts a new ai_id into the altered_item table and, depending on whether the location is new or if has previous AWE's occur there, a new location name is added to the areas_of_interest table or the result_of attribute in the table is updated with the new awe_id.

B.4 Views

A number of views were created for this database in order to provide an ease of access to certain elements of information to database users. The first of these is a view to display the clearance level of all agents, which is based on the role that they have:

```
CREATE OR REPLACE VIEW agent_clearance AS
        SELECT agent.name, role.clearance
        FROM agent, role
        WHERE agent.role = role.title;
```

This may be a useful view for the director to make use of to keep track of the number of employees that have each level of clearance in the agency. Although it may not be the most useful of views, it may prove useful if there is a security breach in the agency.

The second view of the database concerns seeing which Altered Items have been created by which Entity:

```
CREATE OR REPLACE VIEW entities_item AS
  SELECT known_entities.name, known_entities.status,
         altered_item.ai_id, altered_item.name AS item_name
  FROM known_entities, altered_item
  WHERE known_entities.awe = altered_item.awe;
```

A database user could use this information in order to discern, based on the entity that created the Item, the danger level that the item may possess due to the status of the entity. This would be hypothetically useful when developing an approach for researching a specific item.

B.5 Commands to Populate Tables

When Populating my tables instead of including all entries in a single INSERT instruction, a used a number in order to improve the readability of the code. See Below:

```
INSERT INTO agent VALUES('Zachariah Trench','1956-04-28','Director','Male','Executive');
INSERT INTO agent VALUES('Dr. Casper Darling','1972-02-08','Department
Head','Male','Research');
INSERT INTO agent VALUES('Lin Salvador','1964-03-14','Department
Head','Male','Maintenance');
INSERT INTO agent VALUES('Dr. Raya Underhill','1985-06-
06','Supervisor','Female','Research');
INSERT INTO agent VALUES('Robert Nightingale','1981-07-18','Agent','Male','Operations');
INSERT INTO agent VALUES('Helen Marshall','1978-04-22','Department
Head','Female','Operations');
INSERT INTO agent VALUES('Ahti','0000-01-01','Janitor','Non-Binary','Foundation');
INSERT INTO agent VALUES('Frederick Langston','1966-03-03','Department
Head','Male','Panopticon');
```

Above are the commands to populate the 'agent' table. Here it can be seen in each INSERT instruction, the agents name, date of birth, role, gender and department is included, in this order respectively.

B.6 Retrieving Information from Database

There are a number of ways of retrieving information from the database. One of these ways is by using the views described above. By using the command:

```
SELECT * FROM entities_item;
```

All of the elements that relate to the view 'entities_item' will be displayed (see above).

Another way of retrieving information from the database is through the use of general queries. These can be SELECT commands that make use of the JOIN feature of SQL that will allow for the connecting of tables in order to view a number of information points and tuples from different tables together. For example:

```
SELECT name, role, floor_number, no_ai_contained,status,purpose
FROM agent
INNER JOIN department
ON department.dep_head = agent.name
```

This calls upon the agent table and joins it with the department table in order to give information about who runs each department and also the details on each department they run. Another example of using join to gather information would be:

```
SELECT awe_id, location, resulting_item, name, properties
FROM altered_world_event
INNER JOIN altered_item
ON altered_item.ai_id = altered_world_event.resulting_item;
```

This Query joins information from the altered_world_event and altered_item tables based on the ai_id and resulting_item attributes, resulting_item being a foreign key to the altered_item table.

This database also makes use of an age function based off of the date_of_birth attribute in the agent table, allowing users to discern which agents are above/below a certain age and hence gain statistics about the agencies staff as a whole.

```
SELECT *
FROM agent
WHERE Calculate_Age(agent.date_of_birth) > 50;
```

The above returns all agents that are over the age of 50 and hence, may not be fit for field duty in the agency, or may need to be "retired" from the FBC.

B.7 Security

As this is a high security database, every user, depending on clearance level, will require a specific series of abilities within the database. Hence, I created a number of ROLE's based off of the title attribute within the role table.

```
CREATE ROLE 'director','department_head','supervisor','agent';
GRANT ALL ON FBC.* TO 'director';
GRANT INSERT,UPDATE,SELECT ON FBC.* TO 'department_head','supervisor';
GRANT SELECT ON FBC.* TO 'agent';
GRANT INSERT,UPDATE ON FBC.altered_world_event TO 'agent';
```

As can be seen above, all commands are made to the director of the agency, for the entire database. All commands except for DELETE are given to heads of department and supervisors for the database. However, only SELECT commands are given to agents in the FBC, except for in the 'altered_world_event' table where agents may need to submit reports on AWEs they have been working on. This will then, in turn, cause the trigger 'new_awe' in the database and hence other tables will be updated to account for the new AWE. I then created users based on the agents stored in the 'agent' table and granted their respective roles to them based on the role attribute.

```
CREATE USER 'z_trench'@'localhost' IDENTIFIED BY 'DIRECTOR03';
CREATE USER 'c_darling'@'localhost' IDENTIFIED BY 'DHEAD13';
CREATE USER 'r_underhill'@'localhost' IDENTIFIED BY 'SUP35';
CREATE USER 'r_nighingale'@'localhost' IDENTIFIED BY 'AG1010';

GRANT 'director' TO 'z_trench'@'localhost';
GRANT 'department_head' TO 'c_darling'@'localhost';
GRANT 'supervisor' TO 'r_underhill'@'localhost';
GRANT 'agent' TO 'r_nighingale'@'localhost';
```