Databases project

Índice

Conceptual Modeling (Entity-Relationship Model)

Para la base de datos he elegido FROM, una serie de misterio de HBO que he visto hace poco y con probablemente el mejor título posible para un proyecto en SQL Server.

Como resumen, la serie trata sobre un pueblo al que llega la gente desde diferentes partes de Estados Unidos y no pueden salir. Durante la noche ocurren sucesos extraños como la aparición de unos monstruos que sonríen y matan a la gente si les dejan entrar en casa.

Como en cada capítulo mueren la mitad de los personajes, he pensado en un primer momento que la base de datos tiene que reflejar bien los incidentes, por lo que es la tabla central en este sentido, ya que se relaciona con casi todas las demás.

Las entidades que he elegido son:

Characters, Locations, Entities, Episodes, Theories (teorías de los personajes) e Incidents.

Para Characters guardaré información como nombre, estado (vivo o muerto), y la fecha de llegada al pueblo (que no de aparición en la serie).

Locations como tabla que representa los sitios de la serie (lugares, casas, etc.)

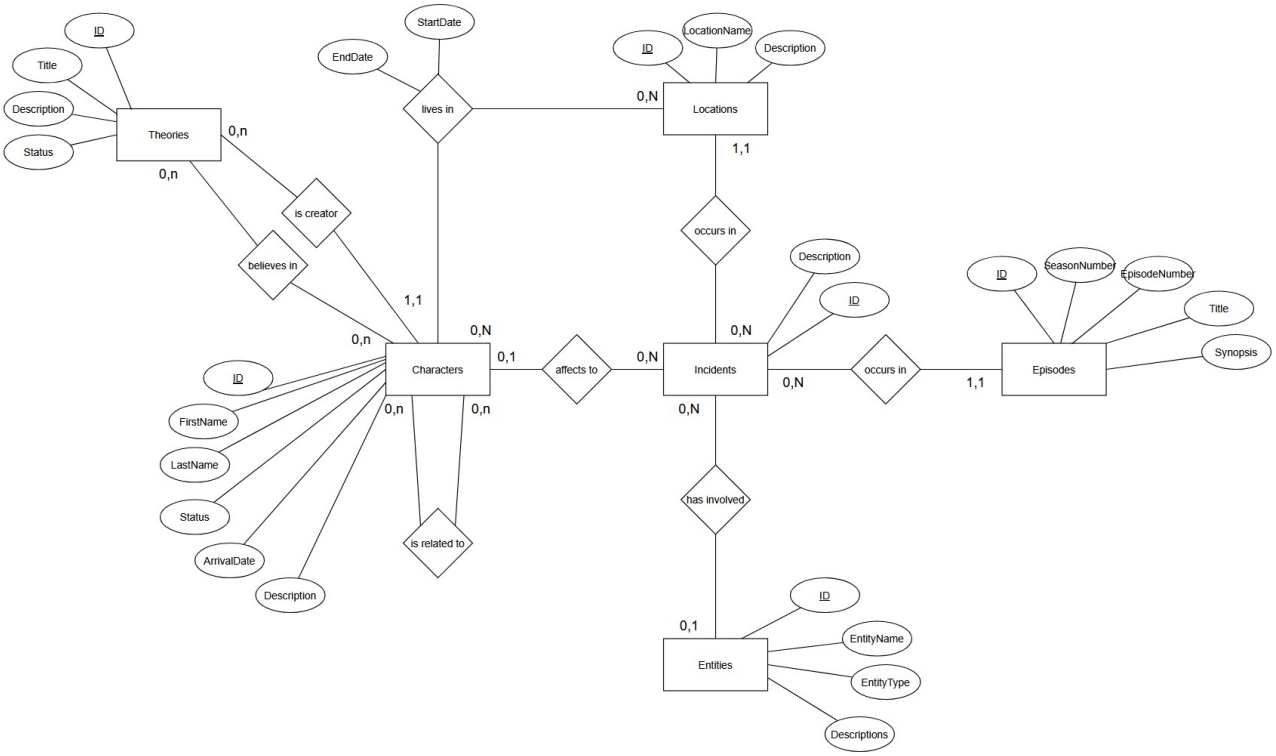
Entities, que hacen referencia a los monstruos o maldiciones de la serie, la palabra entidades se suele usar en el misterio para referirse a cosas que no se conocen, guardaré nombre, descripción y tipo (monstruo o maldición). Representa los peligros de la serie.

Episodes guarda la información sobre los episodios, número, título, etc.

Theories, los personajes desarrollan diferentes teorías sobre lo que está pasando durante la serie. Aquí guardaré información como el creador, una descripción de la teoría, etc.

Incidents, como he dicho, se trata de una tabla central, guarda relación sobre los incidentes como: a quien afecta, qué lo ha provocado, en que capítulo ocurre, etc.

E-R Scheme (Logical Scheme)[1]



1 Subiré el diagrama a parte, por si no se ve bien.

Centrándome en el modelo de entidad-relación, he reflejado las siguientes relaciones:

- **Characters-Theories** son dos relaciones: desde el creador de la teoría y el que cree en ella. Creador solo hay uno y seguidores varios por lo que las relaciones son de 1 a muchos (un personaje puede tener varias teorías pero una teoría puede ser creada por un solo personaje) y de muchos a muchos.
- **Characters-Locations** representa el sitio en el que vive el personaje, es una relación de muchos a muchos ya que puede cambiar de casa a lo largo de la serie, se guarda la información de la estancia y el tiempo.
- **Incidents-Character** relación de cero o uno a muchos. Un incidente puede no tener un personaje claro, ya que se guardan en incidents sucesos que afectan a todo el pueblo como por ejemplo: que llegue un bus lleno de gente, que se pudran los cultivos, etc).
- **Incidents-Entities** de nuevo relación de cero o uno a muchos. Un incidente puede ser causado por un monstruo o no, y un monstruo puede causar varios incidentes.
- **Incidents-Episodes** relación de uno a muchos, en un episodio pueden ocurrir varios incidentes pero un incidente ocurre en un episodio.
- **Incidents-Locations** relación de uno a muchos, igual que en la anterior.

Relational Model

CHARACTERS (CharacterID, FirstName, LastName, ArrivalDate, Status, Description)

LOCATIONS (LocationID, LocationName, Description)

EPISODES (EpisodeID, SeasonNumber, EpisodeNumber, Title, Synopsis)

ENTITIES (EntityID, EntityName, EntityType, Description)

THEORIES (TheoryID, Title, Description, Status, CreatorID → CHARACTERS)

INCIDENTS (IncidentID, Description, EpisodeID → EPISODES, LocationID→LOCATIONS, CharacterID→CHARACTERS, EntityID→ENTITIES)

RESIDENCES (CharacterID→CHARACTERS, LocationID→LOCATIONS, StartDate, EndDate)

RELATIONSHIPS (CharacterID1→CHARACTERS, CharacterID2→CHARACTERS, RelationType)

CHARACTERTHEORIES (CharacterID→CHARACTERS, TheoryID→THEORIES)

Para Characters, Locations, Episodes y Entities la clave primaria es su ID, sin referencias a otras tablas. El caso de Theories igual pero se relaciona con Characters a través de su creador.

Incidents es un poco complicado, tiene 4 foreign keys (EpisodeID, LocationID, CharacterID y EntityID) de las cuales Episode y Location por lo menos son necesarias. El problema es que no pueden ser claves primarias ya que puede haber un incidente que ocurra en el mismo capítulo y lugar. Por este motivo he creado un ID propio para el incidente.

En el caso de Residences, la clave primaria está formada por CharacterID, LocationID y StartDate. Ya que un personaje puede vivir en la misma localización dos veces pero no mudarse en la misma fecha.

Para Relationships, la clave primaria está formada por los dos personajes, no hay problema en este caso ya que no tiene sentido que pueda repetirse dos veces. Seria más lógico modificar el tipo de relación que introducir otra.

Para CharacterTheories igual. Representa seguir una teoría y la clave primaria se forma con ambos ID.

Database Creation

```
CREATE TABLE Characters (
	CharacterID INT IDENTITY PRIMARY KEY,
	FirstName VARCHAR(50) NOT NULL,
	LastName VARCHAR(50) NOT NULL,
	ArrivalDate DATE NULL,
	Status VARCHAR(10) NOT NULL,
	Description VARCHAR(250) NULL,

	CHECK (Status in ('Alive', 'Dead', 'Missing'))
)
```

Clave primaria CharacterID, el resto de atributos not null excepto ArrivalDate y Description. ArrivalDate puede ser null ya que representa la fecha en la que el personaje llegó al pueblo y en la serie aparecen personajes que nunca han estado ahí. En el caso de description, por no ser muy importante.

Para el estado he creado un check que asegura tener el estado en vivo, muerto o desaparecido.

```
CREATE TABLE Locations (
	LocationID INT IDENTITY PRIMARY KEY,
	LocationName VARCHAR(50) NOT NULL UNIQUE,
	Description VARCHAR(250) NULL
)
```

Clave primaria LocationID. Guardo LocationName que es not null y único (podría funcionar también como clave primaria) y descripción.

```
CREATE TABLE Episodes (
	EpisodeID INT IDENTITY PRIMARY KEY,
	SeasonNumber INT NOT NULL,
	EpisodeNumber INT NOT NULL,
	Title VARCHAR(50) NOT NULL,
	Synopsis VARCHAR(250) NOT NULL,

	CHECK (SeasonNumber > 0),
	CHECK (EpisodeNumber > 0)
)
```

Para Episodes he creado SeasonNumber y EpisodeNumber que representan la temporada y episodio, se comprueba que ninguno sea menor que 0. Title y Synopsis como texto.

```
CREATE TABLE Entities (
	EntityID INT IDENTITY PRIMARY KEY,
	EntityName VARCHAR(50) NOT NULL UNIQUE,
	EntityType VARCHAR(10) NOT NULL,
	Description VARCHAR(250) NULL,

	CHECK (EntityType IN ('Monster', 'Curse'))
)
```

Entities, como he dicho antes, representa a los peligros del pueblo y tiene: EntityName, Description que funcionan como en las anteriores tablas y EntityType que puede ser Monster o Curse, ya que todo lo que pasa en el pueblo se puede explicar mediante estas dos opciones.

```
CREATE TABLE Theories (
	TheoryID INT IDENTITY PRIMARY KEY,
	Title VARCHAR(100) NOT NULL UNIQUE,
	Description VARCHAR(250) NOT NULL,
	Status VARCHAR(20),
	CreatorID INT NOT NULL,

	FOREIGN KEY (CreatorID) REFERENCES Characters(CharacterID),
	CHECK (Status IN ('Proposed', 'Refuted', 'Confirmed'))

)
```

Como el resto, un ID como primary key, un título único y una descripción. En este caso not null ya que creo que en el caso de las teorías una descripción tiene más importancia. El creador es una foreign key y el estado puede ser propuesta, refutada o confirmada.

```
CREATE TABLE CharacterTheories (
	CharacterID INT NOT NULL,
	TheoryID INT NOT NULL,

	PRIMARY KEY (CharacterID, TheoryID),
	FOREIGN KEY (CharacterID) REFERENCES Characters(CharacterID),
	FOREIGN KEY (TheoryID) REFERENCES Theories(TheoryID),
)
```

La tabla que relaciona a los personajes con las teorías. A parte de guardar al creador en la base de datos, se guardan sus seguidores en esta tabla. Los dos ID not null ya que son la clave primaria.

```
CREATE TABLE Incidents (
	IncidentID INT IDENTITY PRIMARY KEY,
	EpisodeID INT NOT NULL,
	LocationID INT NOT NULL,
	CharacterID INT NULL,
	EntityID INT NULL,
	Description VARCHAR(250) NOT NULL,

	FOREIGN KEY (EpisodeID) REFERENCES Episodes(EpisodeID),
	FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),
	FOREIGN KEY (CharacterID) REFERENCES Characters(CharacterID),
	FOREIGN KEY (EntityID) REFERENCES Entities(EntityID)
)
```

Como he indicado antes, Incidents se relaciona con muchas tablas, todo son foreign keys menos su propia ID y la descripción. En este caso Description es not null ya que no tiene un título como tal.

```
CREATE TABLE Residences (
	CharacterID INT NOT NULL,
	LocationID INT NOT NULL,
	StartDate DATE NOT NULL,
	EndDate DATE NULL,

	PRIMARY KEY (CharacterID, LocationID, StartDate),
	FOREIGN KEY (CharacterID) REFERENCES Characters(CharacterID),
	FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),

	CHECK (EndDate IS NULL OR EndDate > StartDate)
)
```

Esta tabla almacena la información sobre donde han vivido los personajes, el inicio y el fin. Todo es not null menos la fecha del final ya que puede seguir viviendo en la actualidad ahí. Normalmente este atributo es null, se modifica cuando se muda o muere. También se comprueba que EndDate sea null o mayor a StartDate.

```
CREATE TABLE Relationships (
	CharacterID1 INT NOT NULL,
	CharacterID2 INT NOT NULL,
	RelationType VARCHAR(50) NOT NULL,

	PRIMARY KEY (CharacterID1, CharacterID2),
	FOREIGN KEY (CharacterID1) REFERENCES Characters(CharacterID),
	FOREIGN KEY (CharacterID2) REFERENCES Characters(CharacterID),

	CHECK (CharacterID1 != CharacterID2)
)
```

Esta tabla guarda las relaciones entre los personajes, los ID de ambos son not null ya que forman la clave primaria y el tipo de relación también. No he creado restricciones ya que podría contemplar casos como amistad tensa u otras. Pero sí se comprueba que un personaje no se relacione con sí mismo.

```
CREATE TABLE Logs (
	LogID INT IDENTITY PRIMARY KEY,
	OldRecord VARCHAR(250) NOT NULL,
	NewRecord VARCHAR(250) NOT NULL,
	LogDate DATETIME NOT NULL
)
```

Por último tengo una tabla para logs, que se usará en un ejercicio posterior.

Ejemplo de inserts en Incidents:

```
INSERT INTO Incidents (EpisodeID, LocationID, CharacterID, EntityID, Description)
VALUES
--Episodio 1, El pueblo como localización, Tabitha como la afectada, sin entidad
(1, 1, 2, NULL, 'The Matthews family and Jade Herrera''s group arrive in town after their vehicles crash.'),
--Episodio 2, En la clínica, Sara, Voces de Sara
(2, 7, 8, 6, 'Guided by the voices, Sara Myers kills one of the newcomers in the clinic.'),
--Episodio 2, En la clínica, Sara, voces de Sara
(2, 7, 8, 6, 'Sara Myers intentionally lets monsters into the clinic, resulting in a massacre that killed Kenny''s father.'),
--Episodio 3, En el pueblo, Jade, La maldición principal
(3, 1, 4, 5, 'Jade, believing it all to be some sort of escape room, begins to have visions as he walks around town.'),
--Episodio 6, En el bosque, Jade, La maldición principal
(6, 3, 4, 5, 'Jade is tormented by a terrifying vision of a Civil War-era soldier while helping Jim with his radio.'),
--Episodio 5, En el pueblo, Sara, Voces de Sara
(5, 1, 8, 6, 'Under the influence of voices, Sara Myers attempts to harm Ethan Matthews but accidentally kills her own brother.'),
--Episodio 7, En Colony House, El padre Kathri, los monstruos
(7, 2, 9, 1, 'The Colony House Massacre: Monsters infiltrate a party after being let in, killing many residents, including Father Khatri.'),
--Episodio 8, En el bosque, Boyd, los monstruos
(8, 3, 1, 1, 'In a flashback, Boyd Stevens finds talismans hiding from monsters in a cave.'),
--Episodio 9, En el bosque, Boyd, La maldición princial
(9, 3, 1, 5, 'Searching for answers in the forest, a storm breaks out and Boyd runs into a tree that transports him to some ruins.'),
--Episodio 10, En el pueblo, sin afectado principal, sin entidad
(10, 1, NULL, NULL, 'A bus full of new people arrives, introducing a new group of residents.')
```

Subo a parte comentados el resto de inserts, junto a las consultas, procedures y triggers.

También lo subo a GitHub:

https://github.com/dgarsau/dbProject/tree/main