École Polytechnique Fédérale de Lausanne

Databases Project

First deliverable

Authors :
Tristan Overney
Morgan Bruhin
Valentine Arrieta

22 mars 2015

1 ER model

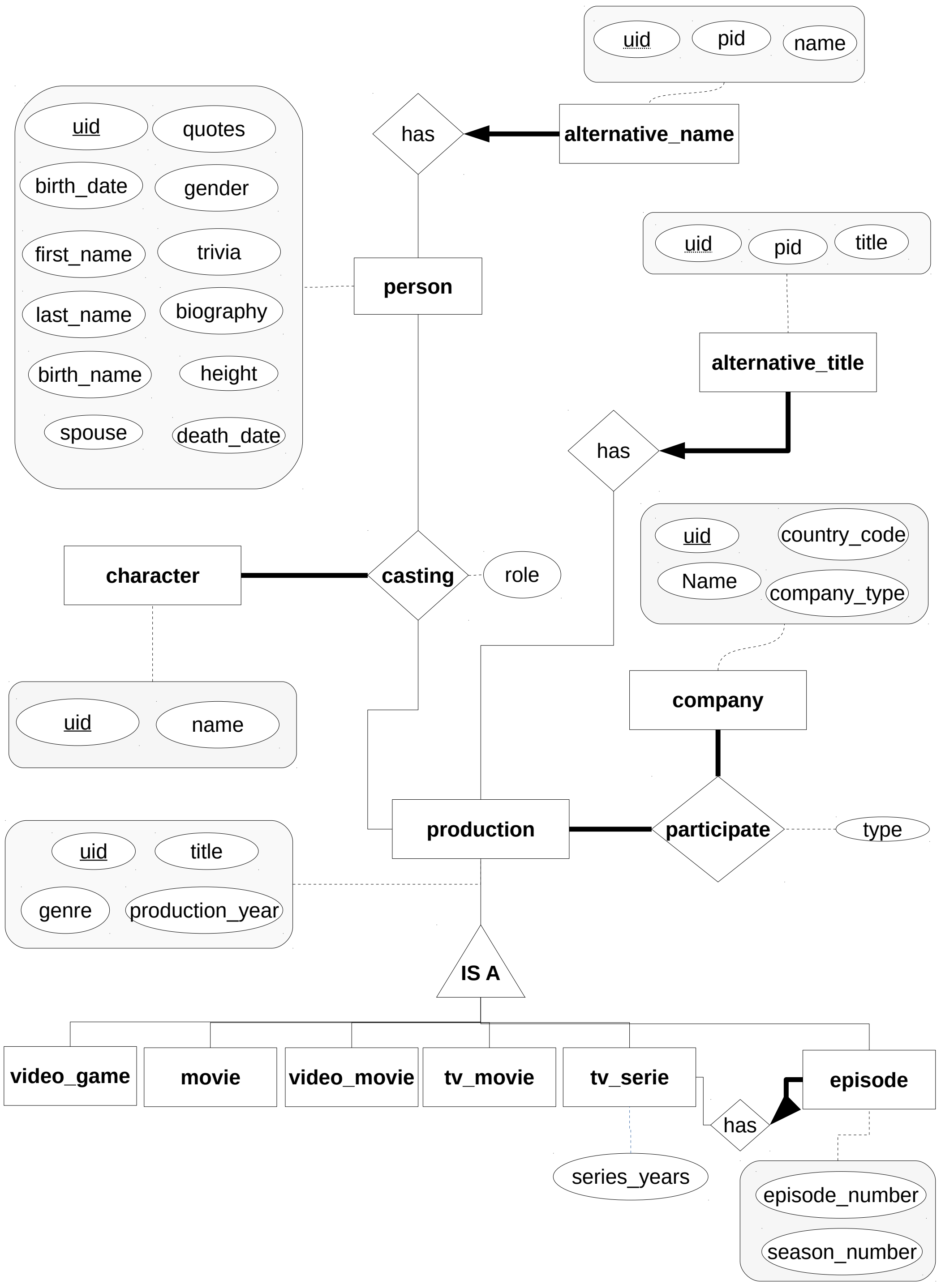

FIGURE 1 – ER model of our DBMS

2 Table creation

We will implement the next deliverables using our own server with PostGreSQL. As such the types used in the following CREATE TABLE possibly doesn't work well in an Oracle DB environnement.

```
CREATE TABLE person
   (uid INTEGER NOT NULL,
    first_name CHAR(75) NOT NULL,
    last_name CHAR(75) NOT NULL,
    gender CHAR(1),
    trivia TEXT,
    quotes VARCHAR(4000),
    birth DATE,
    death DATE,
    biography TEXT,
    spouse CHAR(100),
    height DOUBLE PRECISION,
    primary key (uid));

CREATE TABLE alternative_name
   (uid INTEGER NOT NULL,
    pid INTEGER NOT NULL,
    name CHAR(60) NOT NULL,
    primary key (uid),
    foreign key (pid) references person(uid)
    ON DELETE CASCADE);

CREATE TABLE character
   (uid INTEGER NOT NULL,
    name CHAR(60) NOT NULL,
    primary key (uid));

CREATE TYPE CAST_ROLE AS
    ENUM ('actor', 'actress', 'producer', 'writer', 'cinematographer',
    'composer', 'costume␣designer', 'director', 'editor', 'miscellaneous␣crew',
    'production␣designer');

CREATE TABLE production
   (uid INTEGER NOT NULL,
    title CHAR(80) NOT NULL,
    production_year DATE,
    genre CHAR(20),
    primary key (uid));

CREATE TABLE casting
   (cid INTEGER,
    perid INTEGER NOT NULL,
    prodid INTEGER NOT NULL,
    role CAST_ROLE NOT NULL,
    primary key (cid, perid, prodid, role),
    foreign key (cid) references character (uid),
    foreign key (perid) references person (uid),
    foreign key (prodid) references production (uid));
```

```
CREATE TABLE tv_serie
   (series_years CHAR(11),
    primary key (uid)) INHERITS (Production);

CREATE TABLE episode
   (sid INTEGER NOT NULL,
    season SMALLINT NOT NULL,
    episode SMALLINT NOT NULL,
    primary key (uid),
    foreign key (sid) references tv_serie (uid)
    ON DELETE CASCADE) INHERITS (production);

CREATE TABLE tv_movie
    (primary key (uid)) INHERITS (production);

CREATE TABLE movie
    (primary key (uid)) INHERITS (production);

CREATE TABLE video_movie
    (primary key (uid)) INHERITS (production);

CREATE TABLE video_game
    (primary key (uid)) INHERITS (production);

CREATE TYPE COMPANY_TYPE AS ENUM ('distributors', 'production␣company');

CREATE TABLE company
   (uid INTEGER NOT NULL,
    name CHAR(80) NOT NULL,
    country_code CHAR(6),
    primary key (uid));

CREATE TABLE participate
   (pid INTEGER NOT NULL,
    cid INTEGER NOT NULL,
    type COMPANY_TYPE NOT NULL,
    primary key (pid, cid),
    foreign key (pid) references production(uid),
    foreign key (cid) references company(uid));

CREATE TABLE alternative_title
   (uid INTEGER NOT NULL,
    pid INTEGER NOT NULL,
    title CHAR (80) NOT NULL,
    primary key (uid),
    foreign key (pid) references production (uid)
    ON DELETE CASCADE);
```

3 Discussion about constraint

A person can have one or more alternative name but an alternative name can describe only one person. We thus have a one-to-many relationship set. Also if the person is deleted from the database, his alternative names (if any) will be too. We thus created a weak entity for alternative name. We applied the same design choice for two similar situations : the production and its alternative title(s) and the tv serie and its episode(s).

A production cast make the connection between the production, the person and the character (if any). We thus made a ternary relationship between the three entity person, character and production. We add the role of the person as an attribute of the relationship set. A character exist only if related to at least one person and one production. The person and production entity don't have this constraint since a production can have only a director that won't have any character.

Companies can be involved into production. They can play roles of different type (for example production, distribution), thus we added an attribute to the relationship "participate" called type instead of having a lot of duplicated entries in the "Company" table. Since that design choice, the data set given has been changed and comforted us in our design choice as the "type" attribute has been moved out of the COMPANY.csv file. A production need to have at least one company involved.

We used enums for both the company type and the cast role because their value options are limited. As such we'll avoid repeating a considerable amount of bytes between the many lines sharing the same value for that field.

And last but not least, instead of having a huge melting pot table called production, we made a "parent table" with all the fields shared between the different kinds of production and then we have a table per production kind which inherit from the production table and where some kind have additional info such as the season number in episode. This is the implementation of an IS A relationship in PostGreSQL. This allow us again to avoid having many entries of a table with some null field that they'd never have a value for.