Ingeniería en Sistemas Computacionales

Bases de Datos No Relacionales

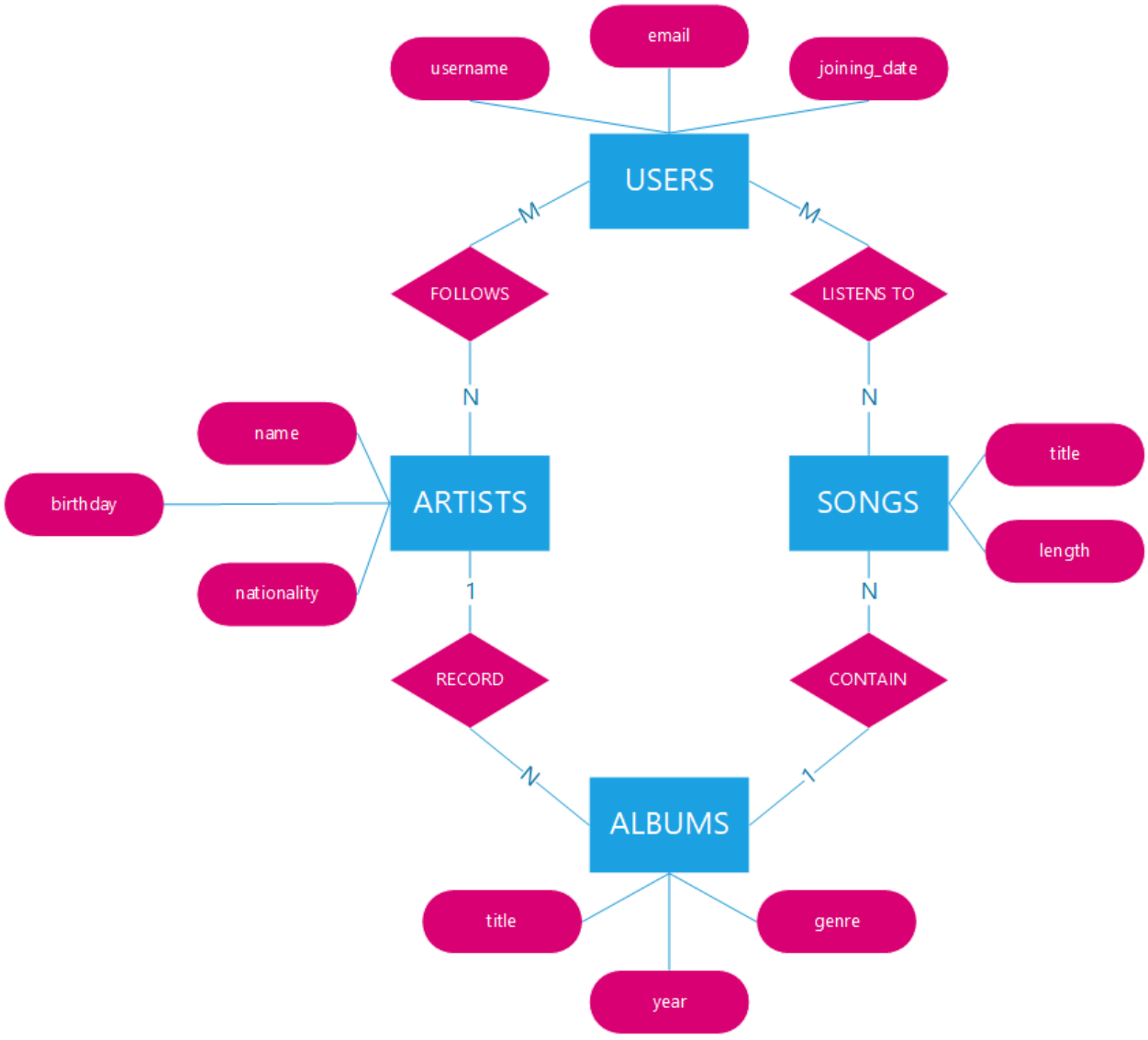
Examen Parcial 1 - CQL

Marco Ricardo Cordero Hernández

Tlaquepaque, Jal., 13 de marzo de 2023

Para el propósito de este reactivo, se ha decidido recrear una parte del funcionamiento de una popular plataforma de streaming de música, a la cual se le denominará como “Espot-if-AI”. Se excluyen algunas restricciones, como bandas, compilatorios, múltiples playlists, y demás elementos similares.

Diagrama de flujo



Consultas

1. El administrador puede ver todos los usuarios de la plataforma y su fecha de subscripción.
2. El usuario puede ver sus canciones escuchadas ordenadas por título en orden descendente.
3. El usuario puede buscar artistas según su nacionalidad.
4. El sistema debe ser capaz de mostrar álbumes según un género determinado.
5. Las canciones pueden ordenarse según su duración.
6. Se puede visualizar la información de un usuario según su correo.
7. Los artistas pueden ser mostrados de más viejo a más joven.
8. Se pueden ver los álbumes lanzados en un año específico.

Tablas en CQL

```
-- 1.   El administrador puede ver todos los usuarios de la plataforma y su
fecha de subscripción.
CREATE TABLE IF NOT EXISTS users (
    username text,
    email text,
    joining_date date,
    PRIMARY KEY ((username), joining_date)
);
-- Se escoge username como llave de partición por relevancia del dato
-- y joining_date porque por esto se filtrarán los datos


-- 2.   El usuario puede ver sus canciones escuchadas ordenadas por título
en orden descendente.
CREATE TABLE IF NOT EXISTS songs_by_user (
    username text,
    title text,
    length varint,
    PRIMARY KEY ((title), title)
);
-- Título determina el identificador del elemento
-- y con el título se ordenarán


-- 3.   El usuario puede buscar artistas según su nacionalidad.
CREATE TABLE IF NOT EXISTS artists_by_nationality (
    name text,
    birthday text,
    nationality date,
    PRIMARY KEY ((name), nationality)
```

```
);
-- Name como identificador primario y nacionalidad para ordenarlos



-- 4.   El sistema debe ser capaz de mostrar álbumes según un género
determinado.
CREATE TABLE IF NOT EXISTS albums_by_genre (
    name_of_artist text,
    title text,
    genre text,
    year varint,
    PRIMARY KEY ((title), genre)
);
-- Se escoge el título del album por lógica de la aplicación
-- y a género para ordenar los resultados


-- 5.   Las canciones pueden ordenarse según su duración.
CREATE TABLE IF NOT EXISTS songs_by_length (
    album_title text,
    title text,
    length varint,
    PRIMARY KEY ((title), length)
);
-- Se opta por particionar por título
-- y se ordena por duración (en segundos)


-- 6.   Se puede visualizar la información de un usuario según su correo.
CREATE TABLE IF NOT EXISTS users (
    username text,
    email text,
    joining_date date,
    PRIMARY KEY ((username), email)
);
-- Se identifica al usuario por su nombre en la plataforma
-- y se le ordena por correo


-- 7.   Los artistas pueden ser mostrados de más viejo a más joven.
CREATE TABLE IF NOT EXISTS artists_by_nationality (
    name text,
    birthday text,
    nationality date,
    PRIMARY KEY ((name), birthday)
```

```
);
-- Se identifica al artista por su nombre completo
-- y se le ordena según su fecha de nacimiento


-- 8.   Se pueden ver los álbumes lanzados en un año específico.
CREATE TABLE IF NOT EXISTS albums_by_genre (
    name_of_artist text,
    title text,
    genre text,
    year varint,
    PRIMARY KEY ((title), year)
);
-- Se selecciona el nombre de album como su identificador
-- y se ordenarán por su año de lanzamiento
```