Schéma : pagila

pagila, relationnel, pattes de corbeau, crowfoot

2024-09-20

⚠ Présentation de la partie `film` du schéma `pagila`, pour l'entrainement à PostGres.

- **L3 MIASHS/Ingémath**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage
- Moodle

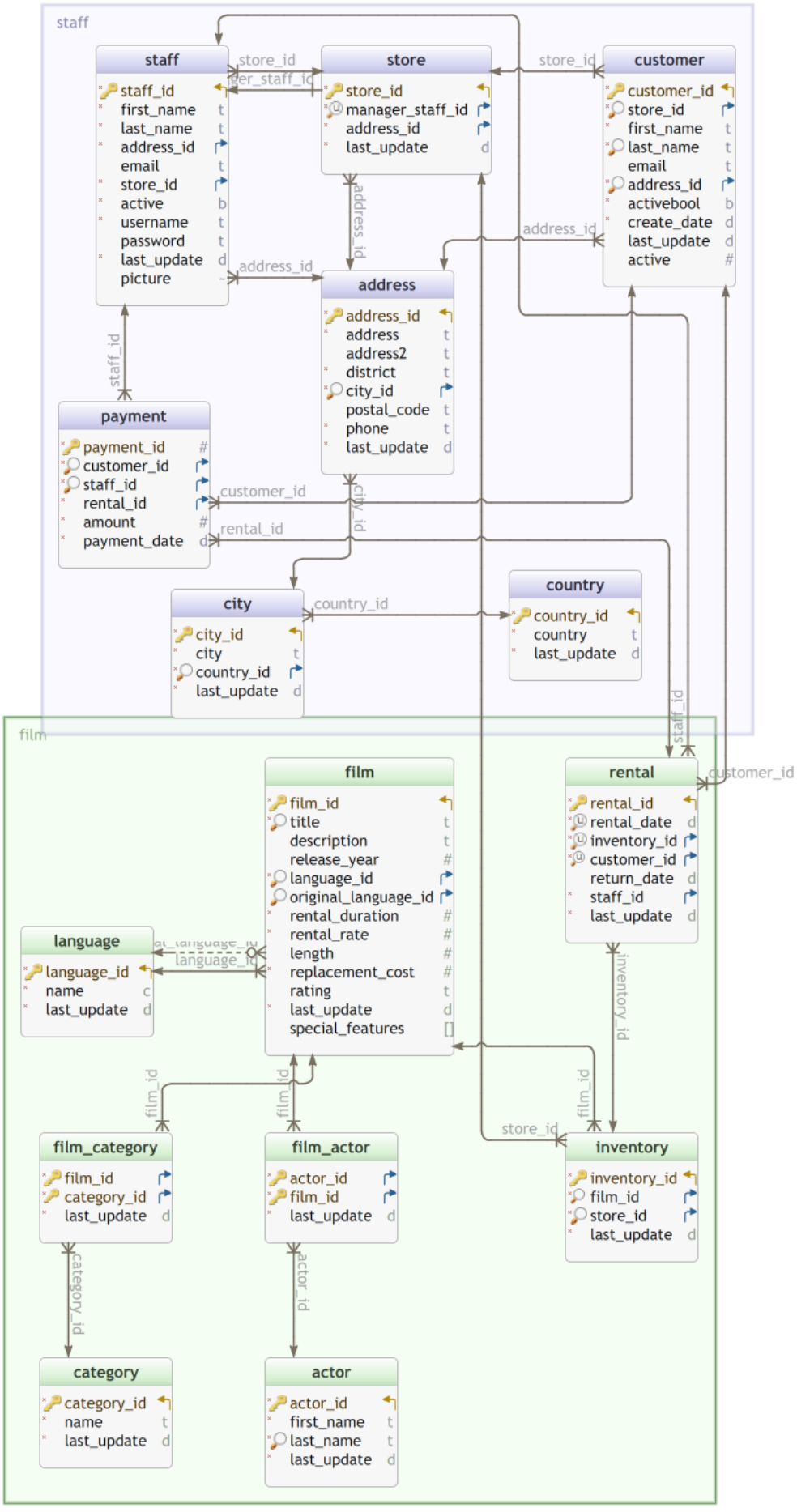

FIG. 1 : Schema pagila créé par DbSchema

Contexte

Description de presque toutes les tables de la base `pagila` ,

- Pagemaison de la base de données `pagila`.
- Pagemaison de la base de données `sakila`
- Détails sur les films
- Détails sur les adresses, clients, magasins, paiements

`pagila` est une base d'entrainement pour PostgreSQL. Cette base est une adaptation de la base `sakila` utilisée pour l'entrainement à MySQL.

La base `pagila` est censée permettre de gérer les prêts et la facturation d'une chaine fictive de magasins de location de DVD (une activité commerciale florissante durant les années 1990 et au début des années 2000).

Le schéma `pagila` est formé de 15 tables. Un groupe de tables concerne les films diponibles au catalogue de la chaîne de magasins : `film`, `actor`, `language`, `category`, `film_actor`, `film_category`.

Les lignes de la table `film` sont les entrées du catalogue de la chaîne. Une ligne est identifiée par l'attribut `film_id` (noter l'attribution automatique des identifiants grâce au mécanisme des séquences).

Un film possède une distribution constituée d'actrices et d'acteurs. Comme une actrice peut jouer dans plusieurs films, l'association plusieurs-à-plusieurs entre films et acteurs/actrices est rerésentée par une *table intémédiaire* `film_actor`.

Un film relève d'une ou plusieurs catégories. Là encore, l'association plusieurs-à-plusieurs entre films et catégories est rerésentée par une *table intémédiaire* `film_category`.

Une entrée au catalogue correspond à un titre de film tourné dans une certaine langue (la langue originale) et à une bande son dans une langue qui peut être différente de la langue originale.

> **i** Notons ici qu'une entrée de la table `film` ne correspond pas tout à fait à l'idée que nous nous faisons d'un film (entrée dans la base IMDB), mais plutôt à une édition (simplifiée) de film en DVD.

Une film disponible au catalogue correspond à un ou plusieurs DVD (objets physiques) qui sont représentés par des lignes de la table `inventory`. Les conditions de location d'un même titre (ligne de `film`) sont identiques pour les DVDs physiques correspondant au même titre.

Un ligne de la table `inventory` peut faire l'objet d'une *location* qui correspond à une ligne de `rental`. La table `rental` est encore un cas de *table intermédiaire* entre les tables `inventory`, `customer` (client) et `staff` (employé). Le fait qu'il s'agisse d'une table intermédiaire destinée à représenter une association ternaire plusieurs-à-plusieurs entre élément d'inventaire, client et employé est en partie caché par le fait que `rental` possède un identifiant propre `rental_id`. Un élément de `rental` devrait pouvoir être identifié grâce aux identifiants des lignes participantes de `inventory`, `customer` et `staff` et à un identifiant relatif comme `rental_date`.

Un client est rattaché à un magasin (`store`). Les employés sont eux aussi attachés à un magasin.

Les clients, employés et magasins sont liés à une adresse (`address`) qui relève d'une ville (`city`) qui elle même relève d'un pays (`country`).

Un paiement relie un employé à un client à propos d'une location.

Table pagila.address

Idx	Name	Data Type
*	address_id	integer DEFAULT nextval('pagila.address_address_id_seq' ::regclass)
*	address	varchar(50)
	address2	varchar(50)
*	district	varchar(20)
*	city_id	smallint
	postal_code	varchar(10)

Idx	Name	Data Type
*	phone	varchar(20)
*	last_update	timestamp DEFAULT now()

Foreign Keys

Type	Name	On
	address_city_id_fkey	(city_id) ref pagila.city (city_id)

👉 Dans les tables `address`, `city`, `country`, `store`, `inventory`, `rental`, `actor`, `film`, `category`, `language`, `payment`, on trouve une colonne nommée `nom de la table_id` qui est la clé primaire de la table. Cette colonne est (en général) renseignée automatiquement : la valeur par défaut est calculée en invoquant la fonction `nextval()` sur la séquence associée à la table. La clé primaire ne comporte d'information intéressante pour l'usager final, elle correspond à un rang d'insertion dans la table.

👉 Les tables de `pagila` comportent une colonne `last_update` (dernière mise-à-jour) de type `timestamp` (instant, nombre de secondes écoulées depuis le premier janvier 1970, origine des temps selon `Unix`). Cette colonne est remplie (renseignée) automatiquement lors des insertions et mises à jour grâce à la valeur par défaut calculée par la fonction `now()`.

🔥 Les colonnes `last_update` des différentes tables de `pagila` n'ont rien à voir les unes avec les autres (vérifier l'absence de contrainte référentielle au sujet de ces colonnes). Il n'est donc pas question (en général) d'utiliser ces colonnes `last_update` pour effectuer des jointures entre les tables de `pagila`.
La présence des colonnes `last_update` nous empêche de faire des jointures naturelles (`NATURAL JOIN`) entre `film`, `film_actor`, `actor`, etc.

```
SELECT count(*)
FROM
  film
JOIN
  film_actor
USING(film_id) ;

 count
-------
  5462
```

alors que

```
SELECT count(*)
FROM
  film
NATURAL JOIN
  film_actor ;

 count
-------
     0
```

Table pagila.city

Idx	Name	Data Type
*	city_id	integer DEFAULT nextval('pagila.city_city_id_seq'::regclass)

Idx	Name	Data Type
*	city	varchar(50)
*	country_id	smallint
*	last_update	timestamp DEFAULT now()

Foreign Keys

Type	Name	On
	city_country_id_fkey	(country_id) ref pagila.country (country_id)

👉 On dit que la contrainte référentielle `city_country_id_fkey` est émise par l'attribut `country_id` de la table `city` vers l'attribut `country_id` de la table `country`. Une ville n'appartient qu'à un seul pays, mais un pays peut comporter plusieurs villes.

Table pagila.country

Idx	Name	Data Type
*	country_id	integer DEFAULT nextval('pagila.country_country_id_seq'::regclass)
*	country	varchar(50)
*	last_update	timestamp DEFAULT now()

Table pagila.customer

Idx	Name	Data Type
*	customer_id	integer DEFAULT nextval('pagila.customer_customer_id_seq'::regclass)
*	store_id	smallint
*	first_name	varchar(45)
*	last_name	varchar(45)
	email	varchar(50)
*	address_id	smallint
*	activebool	boolean DEFAULT true
*	create_date	date DEFAULT ('now'::text)::date
	last_update	timestamp DEFAULT now()
	active	integer

Foreign Keys

Type	Name	On
	customer_store_id_fkey	(store_id) ref pagila.store (store_id)
	customer_address_id_fkey	(address_id) ref pagila.address (address_id)

Table pagila.film

Idx	Name	Data Type
*	film_id	integer DEFAULT nextval('pagila.film_film_id_seq'::regclass)
*	title	varchar(255)
	description	text
	release_year	year
*	language_id	smallint
	original_language_id	smallint
*	rental_duration	smallint DEFAULT 3
*	rental_rate	numeric(4,2) DEFAULT 4.99
	length	smallint
*	replacement_cost	numeric(5,2) DEFAULT 19.99
	rating	mpaa_rating DEFAULT 'G'::pagila.mpaa_rating
*	last_update	timestamp DEFAULT now()
	special_features	text[]

Foreign Keys

Type	Name	On
	film_original_language_id_fkey	(original_language_id) ref pagila.language (language_id)
	film_language_id_fkey	(language_id) ref pagila.language (language_id)

Table pagila.inventory

Idx	Name	Data Type
*	inventory_id	integer DEFAULT nextval('pagila.inventory_inventory_id_seq'::regclass)
*	film_id	smallint
*	store_id	smallint
*	last_update	timestamp DEFAULT now()

Foreign Keys

Type	Name	On
	inventory_store_id_fkey	(store_id) ref pagila.store (store_id)
	inventory_film_id_fkey	(film_id) ref pagila.film (film_id)

Une instance d'`inventory` correspond à un support physique, quelque chose qui peut-être loué. Elle permet au client de voir une instance de `film`, soit un film dans une certaine langue.

Table pagila.payment

Idx	Name	Data Type
*	payment_id	integer DEFAULT nextval('pagila.payment_payment_id_seq'::regclass)
*	customer_id	smallint
*	staff_id	smallint
*	rental_id	integer
*	amount	numeric(5,2)
*	payment_date	timestamp

Foreign Keys

Type	Name	On
	payment_staff_id_fkey	(staff_id) ref pagila.staff (staff_id)
	payment_rental_id_fkey	(rental_id) ref pagila.rental (rental_id)
	payment_customer_id_fkey	(customer_id) ref pagila.customer (customer_id)

Un paiement (une ligne de `payment`) concerne une location (un tuple de `rental`), d'où la contrainte référentielle `payment_rental_id_fkey`. un paiement concerne ausi un client (tuple de `customer`) et un employé (tuple de `staff`).

Table pagila.rental

Idx	Name	Data Type
*	rental_id	integer DEFAULT nextval('pagila.rental_rental_id_seq'::regclass)
*	rental_date	timestamp
*	inventory_id	integer
*	customer_id	smallint
	return_date	timestamp
*	staff_id	smallint
*	last_update	timestamp DEFAULT now()

Foreign Keys

Type	Name	On
	rental_staff_id_fkey	(staff_id) ref pagila.staff (staff_id)
	rental_inventory_id_fkey	(inventory_id) ref pagila.inventory (inventory_id)
	rental_customer_id_fkey	(customer_id) ref pagila.customer (customer_id)

Table pagila.staff

Idx	Name	Data Type
*	staff_id	integer DEFAULT nextval('pagila.staff_staff_id_seq'::regclass)
*	first_name	varchar(45)
*	last_name	varchar(45)
*	address_id	smallint
	email	varchar(50)
*	store_id	smallint
*	active	boolean DEFAULT true
*	username	varchar(16)
	password	varchar(40)
*	last_update	timestamp DEFAULT now()
	picture	bytea

Foreign Keys

Type	Name	On
	staff_store_id_fkey	(store_id) ref pagila.store (store_id)
	staff_address_id_fkey	(address_id) ref pagila.address (address_id)

Table pagila.store

Idx	Name	Data Type
*	store_id	integer DEFAULT nextval('pagila.store_store_id_seq'::regclass)
*	manager_staff_id	smallint
*	address_id	smallint
*	last_update	timestamp DEFAULT now()

Foreign Keys

Type	Name	On
	store_manager_staff_id_fkey	(manager_staff_id) ref pagila.staff (staff_id)
	store_address_id_fkey	(address_id) ref pagila.address (address_id)