Report - DataBase Project - IFEBY140

Table of contents

1 Introduction

Que Faire à Paris ? is a french website that list events happing in Paris.

The event are archived at this address in multiple formats including csv.

In this project we propose a normalized database to store the events based on the csv archive.

Course constraint :

> Code everything in psql.
> Don't use pgsql functions to extract the data.

2 Analyzis

Initial Columns :

```
['ID' 'URL' 'Titre' 'Chapeau' 'Description' 'Date de début' 'Date de fin'
 'Occurrences' 'Description de la date' "URL de l'image"
 "Texte alternatif de l'image" "Crédit de l'image" 'Mots clés'
 'Nom du lieu' 'Adresse du lieu' 'Code postal' 'Ville'
 'Coordonnées géographiques' 'Accès PMR' 'Accès mal voyant'
 'Accès mal entendant' 'Transport' 'Url de contact' 'Téléphone de contact'
 'Email de contact' 'URL Facebook associée' 'URL Twitter associée'
 'Type de prix' 'Détail du prix' "Type d'accès" 'URL de réservation'
 'URL de réservation - Texte' 'Date de mise à jour' 'Image de couverture'
 'Programmes' 'En ligne - address_url' 'En ligne - address_url_text'
 'En ligne - address_text' 'title_event' 'audience' 'childrens' 'group']
```

2.1 Redudant Attributes

We will remove the following attributes : description_de_la_date

2.2 First Normal Form

Non atomic attributs :

- List
  - Occurences : separated by `'_'` (underscore)
  - Tags : separated by `','`
  - Childrens : separated by `','`
  - Transport : separated by `'\n'`
- Multiple attributes
  - Transport : transport_type, transport_line, station, distance
  - Geographic_Coordinates : longitude, latitude

2.3 Functional dependencies :

- id → url
  - For simplicty sake, we consider that we cannot deduce id from url. Perhaps, we will use this property to identify sub_events_id in "childrens"
- id → titre, chapeau, description, date_de_debut, date_de_fin, occurences, url_de_l_image, texte_alternatif_de_l_image, credit_de_l_image, mots_clefs, nom_du_lieu, adresse_du_lieu, code_postale, ville, url_du_contact, telephone_de_contact, email_de_contact, url_facebook_associee, url_twitter_associe, type_de_prix, detail_du_prix, type_d_acces, url_de_reservation, url_de_reservation_texte, date_de_mise_a_jour, image_de_couverture, programmes, en_ligne_address_url, en_ligne_address_url_text, en_ligne_address_text, title_event, audience, childrens, groupe, transport_type, transport_line, station, distance
- date_de_debut, date_de_fin → description_de_la_date
  - Removed attribute
- adresse_du_lieu, ville, code_postale → coordonnees_geographiques
- nom_du_lieu, adresse_du_lieu, ville, code_postale -> acces_pmr, acces_mal_voyant, acces_mal_entendant

2.3.1 Non used functional dependencies

- In 1NF : transport_station → ville
  - It's true, but for simplicity sake, we'll ignore it.
- ville → cp and cp → ville
  - It's neither true in France and `Ile de France`
- (titre, date_de_debut, date_de_fin, nom_du_lieu, adresse_du_lieu, code_postale, ville) → id
  - We cannot use those attributes as a primary key, because they are often `NULL`, thus we'll ignore this functionnal dependency.

2.4 Miscellaneous

We decide to translate everything into English.

2.5 Constraints

- Realistic implementation expectations
  - date_end > date_start
  - NOT NULL :
    - id, url
    - (date_start AND date_end) OR (ocurrences)

        * title

- Unrealistic implementation expections
  - url must finish with id
  - all urls must be valid
  - address must be valid
  - phone_number must be valid
  - every event should have an address, a geographic coordinate and a contact
  - NOT NULL
    * a non null lead text, description, date_start, date_end, title_event, price_type

2.6 Triggers

- event_table.parent_event_id : ON DELETE CASCADE
- transport.event_id : ON DELETE CASCADE

2.7 Functional dependencies after decomposition

- Relation : **geographic_correspondance**
  - address_street, address_zipcode, address_city → lat, lon
- Relation : **address_table**
  - address_name, address_street, address_zipcode, address_city → pmr, blind, deaf
- Relation : **event_table**
  - event_id → event_url, title, lead_text, event_description, date_start, date_end, cover_url, cover_alt, cover_credit, address_name, address_street, address_zipcode, address_city, price_type, price_detail, access_type, access_link, access_link_text, updated_at, image_couverture, programs, title_event, audience, contact_url, contact_phone, contact_mail, contact_facebook, contact_twitter, address_url, address_url_text, address_text, keyword, group_name, parent_event_id
- Relation : **occurence**
  - event_id, occurence_date (no functional dependancy)
- Relation : **tag**
  - event_id, keyword (no functional dependancy)
- Relation : **transport**
  - event_id, transport_type, transport_line, station → distance

2.7.1 Boyce-Codd verification

- 1NF : Atomicity and no list attributs
  - Checked
- 2NF : Attributes cannot depend only on a subpart of the primary key
  - Checked
- 3NF : No transitive dependances
  - Checked
- Boyce-Codd : $X \to Y \implies X$ is a superkey keyword
  - Checked

2.8 Indexes

We often want to identify events using :

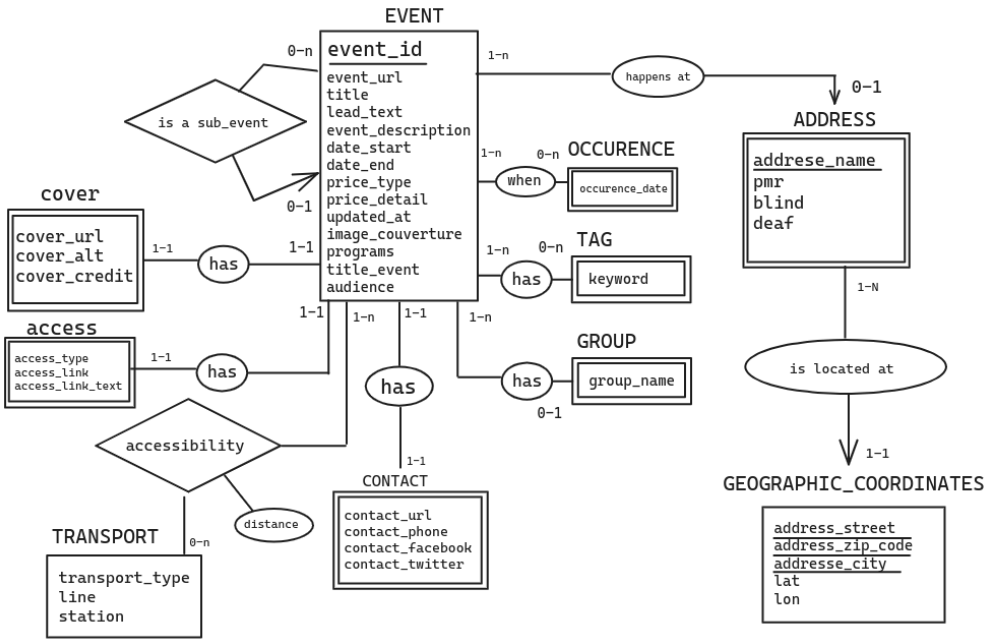
- the title
  - event_table (title)
  - event_table (title_event)
- the period
  - event_table (date_begin)
  - event_table (date_end)
- the arrondissement of Paris
  - event_table (address_zipcode)
- using the url
  - event_table (event_url)
- event_table (parent_event_id)

We also want to do agregations on keywords :

- tag (keyword)

3 Modelization

3.1 ER schema



The following decomposition, satisfies Boyce-codd normal form.

3.2 Tables

geographic_correspondance(address_street, address_zipcode, address_city | lat, lon)

address_table(address_name, address_street, address_zipcode, address_city | pmr, blind, deaf)

event_table (event_id | event_url, title, lead_text, event_description, date_start, date_end, cover_url, cover_alt, cover_credit, address_name, address_street, address_zipcode, address_city, price_type, price_detail, access_type, access_link, access_link_text, updated_at, image_couverture, programs, title_event, audience, contact_url, contact_phone, contact_mail, contact_facebook, contact_twitter, address_url, address_url_text, address_text, keyword, group_name, parent_event_id)

occurence(event_id, occurence_date)

tag(event_id, keyword)

transport(event_id, transport_type, transport_line, station | distance)

4 Implementation

4.1 Execution

Start psql in src folder.

```
cd src
psql -d <database> username
```

Create tables and populate them with :

```
\i XXX_YYY_tables.sql
```

4.2 Code Details

We ignore most redudancy errors using `ON CONFLICT DO NOTHING`.

4.2.1 Populate Transports

Raw transport line :

```
"Métro -> 1 : Bastille (Paris) (304m)
Bus -> 29618791 : Lyon / Daumesnil - Ledru Rollin (Paris) (246m)
Vélib -> Lacuée - Lyon (121.91m)
<a href=""https://www.geovelo.fr/paris/route?to=2.370458543300628,48.849268481958404"">Calcule
```

1. List → Row
   - Split over line jump using : `unnest(string_to_array(transport, E'\n')) AS transport`
2. Remove unnecessary HTML content
   - `WHERE transport NOT LIKE '<%'`
3. Extract (transport_type, transport_line, station + distance)
   - use `split_part` over `->` and `:`
4. `Vélib` case
   - If station is empty, it means it is a `Vélib`, and the station was wrongly splited into transport_line. Then we swawp `station` and `transport_line`

5. Extract distance
    - It can be done using a combination of `substring`, `reverse` and `length` functions.
    - Distance information is always between the last occurence of `(` and `m)`.

4.2.2 Populate Sub-Events

Raw sub event line example:

```
"Rencontre avec Jørn Lier Horst (https://www.paris.fr/evenements/rencontre-avec-jorn-lier-horst
```

1. List → Row
    - Split over `;` delimiters using : `unnest(string_to_array(children, ';'))`
2. Extract id
    - The id is always at the end of the url. It can be extracted knowing it's always between the last `-` and `)` of the string.

5 Request examples

See the interactive preview of the data with :

```
\i XXX_YYY_data.sql
```

5.1 Outputs

5.1.1 Statistics

Number of events : 2816

Number of events with pmr, blind or deaf access : 432

Number of events accessibles with “Vélib” : 264

Number of events with multiple transports : 266

5.1.2 The 10 station with the most events

count	transport_type	transport_line	station
75	Bus	29618791	Lyon / Daumesnil - Ledru Rollin (Paris)
75	Métro	1	Bastille (Paris)
75	Vélib		Lacuée - Lyon
55	Vélib		Boyer - Ménilmontant
30	Bus	2696	Pyrénées - Ménilmontant (Paris)
30	Métro	11	Jourdain (Paris)
29	Bus	397089	Volontaires - Lecourbe (Paris)
29	Vélib		Volontaire - Lecourbe
29	Métro	12	Volontaires (Paris)
25	Métro	3	Gambetta (Paris)

5.1.3 The title of the 10 events with the most occurrences

count	id	title
9970	41529	Un parcours sonore et urbain
4031	37301	La Cité de l’Histoire, l’expérience immersive à travers l’Histoire de France
641	31363	Une plongée immersive avec les baleines à l’Aquarium tropical

count	id	title
634	11871	Revivre, les animaux disparus en réalité augmentée
572	42634	« Notre-Dame de Paris : au cœur du chantier »
552	53785	Montmartre Enchanté insolite : la visite chantée et commentée
511	32056	Mémorial de l'ancienne gare de déportation de Bobigny
379	32135	Exposition " Notre-Dame de Paris. Des bâtisseurs aux restaurateurs."
352	50933	Sauvez le cinéma, l'escape game du Grand Rex
342	31054	« Félins » au Muséum national d'Histoire naturelle

5.1.4 The 10 most used tag

count	keyword
641	Musique
623	Concert
503	Atelier
454	Enfants
393	Loisirs
312	Sport
265	Expo
240	Conférence
234	Théâtre
179	Littérature

5.1.5 The 10 biggest group

count	group_name
1184	Aucun
634	Bibliothèques
602	Agenda
214	Activités DJS
74	Centres d'animations
49	Parcs et jardins
23	Nuit Blanche
22	Associations
14	Musées