Migration of DB from MySQL/MariaDB to PostgreSQL

Author: Elliott Ward Farmer

Contents

Author: Elliott Ward Farmer

1. Why Swap from MySQL to PostgresSQL?

Author: Elliott Ward Farmer

1.1 Why PostgreSQL?

1. Is opensource (free) and has wide support.
2. Can be hosted by my current hosting provider without costly VPS option.
3. Has many more advanced features and extensions than MySQL.
4. Supports inbuilt functions in almost any coding language.
5. Supports storage of JSON data types very well including full text search of JSONB.
6. Beter handling of timestamp fields with timestampTZ referenced to UTC. Automatic time conversion depending on timezone
7. Use of Standard views and Materialized Views can speed up query returns
8. Advanced text search is a feature that I would like to implement by building an advanced search function in my web app using PostgreSQL's powerful tsvector
9. Better handling of Geospatial data types using PostGIS extension
10. More granularity for database security

1.2 phpMyAdmin – BarcelonaTravelhacks.com Database by the Numbers

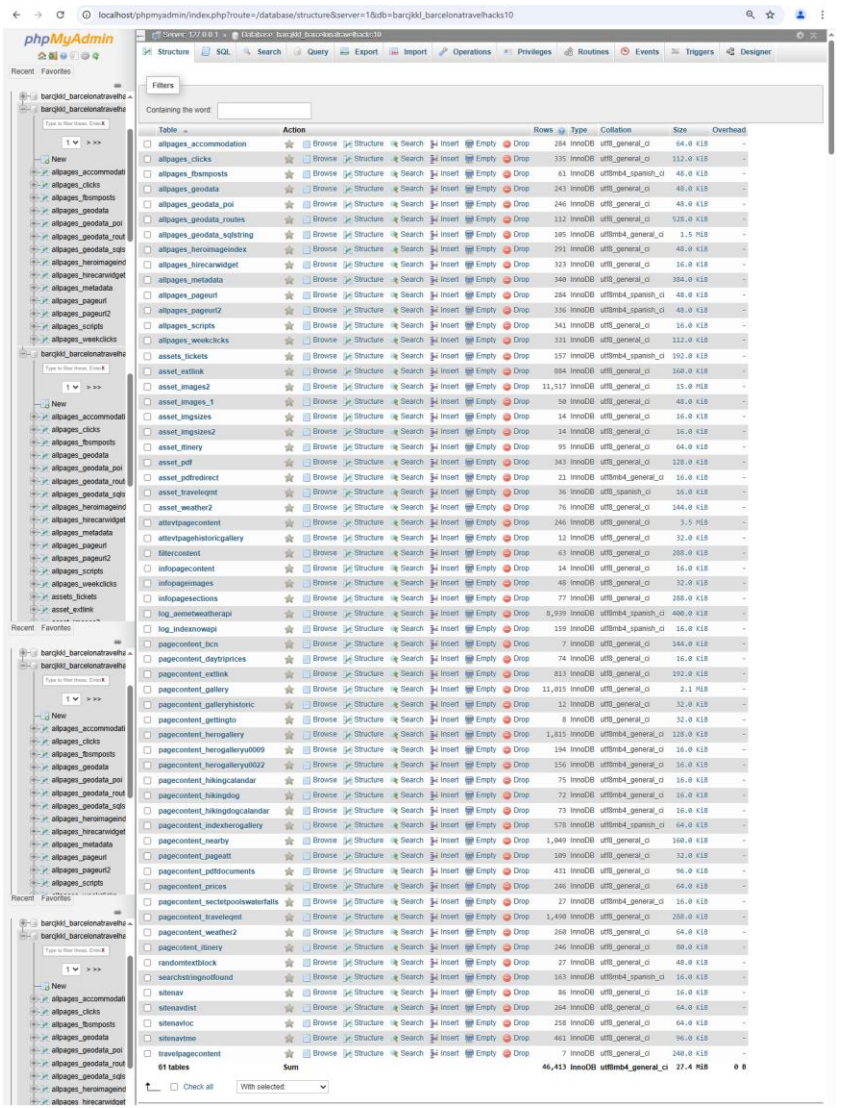
BarcelonaTravelHacks.com

Barcelona Travel hacks is a custom Database that I designed and built 7 years ago and has been continuously upgraded and changed as the design of my website has evolved.

It is not a small database but has 61 tables, the largest containing 11,015 rows. There are complex relations and foreign keys within as well as indexed columns. Database size is a 17 MB SQL file.

The database has some defunct tables and columns as I have evolved around the structure.

The aim of migrating from MySQL to PostgreSQL is to first sanitise the Database, remove obsolete tables and columns and utilise advanced PostgreSQL features to speed up data retrieval for the web app.



Author: Elliott Ward Farmer

1.3 Shattering Database preconceptions

a) MySQL/MariaDB is compatible with PostgreSQL because they are both SQL languages. I can just download the whole database then upload it to PostgreSQL without any issues!

No! MySQL has different datatypes to PostgreSQL so I will need to convert each table from MySQL to PostgreSQL datatypes. The quickest way would be an SQL script for each table.

An example is Boolean. MySQL/MariaDB/phpMyAdmin does not support a Boolean datatype so I have used TINYINT(1) datatype storing a value of 0 or 1 and in other cases a 1 and NULL value to represent Boolean.

MySQL/MariaDB/phpMyAdmin stores datetime or timestamp as `YYYY-MM-DD HH:MM:SS` whereas PostgreSQL has multiple options for date and time datatypes. I have chosen TIMESTAMPTZ to replace datetime because it contains an additional time component of timezone with reference to the UTC clock. So in conclusion a datetime datatype of `YYYY-MM-DD HH:MM:SS` will become `YYYY-MM-DD HH:MM:SS+TZ`

Text based datatypes are compatible. For example I am using VARCHAR() and TEXT in MySQL and they can directly be translated into PostgreSQL datatypes without any need for formatting. Note that in MySQL the query values are in double quotes but in PostgreSQL they are within single quotes so I will need to escape single quotes within any text strings.

1.4 Geospatial Datatype Conversion

MySQL/MariaDB/phpMyAdmin has, in my opinion, poor handling for geospatial data such as point and linestring. For example a point datatype:
Phpmyadmin: `'POINT(2.1749603426429256 41.40390130683638)',0` but shows as `[GEOMETRY - 25 B]`
PostgreSQL: `(2.1749603426429256,41.40390130683638)` and is inserted with `POINT(2.1749603426429256,41.40390130683638)`. To get a point datatype out of MySQL I need to convert from `[GEOMETRY - 25 B]` to `POINT(2.1749603426429256,41.40390130683638)` using `AsText(geoData)` queries.
Line String is an array of points but I still need to decode them from `[GEOMETRY - 749 B]` to get `'LINESTRING(2.174993 41.404071,2.174885 41.403893,....etc)',0` in MySQL. In postgreSQL it is an array point datatype `[(2.174993,41.404071),(2.174885,41.403893),....etc]`.

phpMyAdmin database administrator software is buggy when dealing with geospatial datapoints. You can read more on my stackOverFlow post about what I found and how I have to use pure sql queries to modify these datatypes because of the bug. Further MySQL does not support any geospatial maths such as calculating areas of a polygon, length of a line string etc. These features will be useful as I build out geospatial data and web mapping app.

Author: Elliott Ward Farmer

1.5 Different SQL Syntax

b) The SQL syntax for MySQL/MariaDB is the same as PostgreSQL.

No. I will need a script that selects all rows from a MySQL table then iterates over each row in a loop creating a new INSERT query in PostgreSQL syntax performing any datatype conversion for each column. More on this later when I write the MySQL to PostgreSQL conversion scripts.

MySQL/MariaDB/phpMyAdmin supports camel case for **columnNames** whereas PostgreSQL only accepts lowercase letters and underscores for **column_names**. Take advantage of this opportunity to use better, more logical table names that follow a set nomenclature based on the table content and hierarchy within the database structure.

Author: Elliott Ward Farmer

2. PostgreSQL Database Schemas

Author: Elliott Ward Farmer

2.1 Design of the Database Schema

The following pages contain the Database schemas for all the tables in PostgreSQL format. I carefully analysed all the MySQL/MariaDB/phpMyAdmin tables reviewing data types and compatibility with PostgreSQL to design the schema.

The full schema is too large to fit on one page so I have broken it down into logical blocks.

Author: Elliott Ward Farmer

2.1 Core – Page Handling

t_allpages_scripts
Page_id VARCHAR(6) PRIMARY KEY
page_type HTTP error
google_ads BOOLEAN
get_your_guide BOOLEAN
Stay22 BOOLEAN
app_card_grid_comp_css BOOLEAN
app_video_grid_css BOOLEAN

t_allpages_scripts
Page_id VARCHAR(6) PRIMARY KEY
Hire_car BOOLEAN

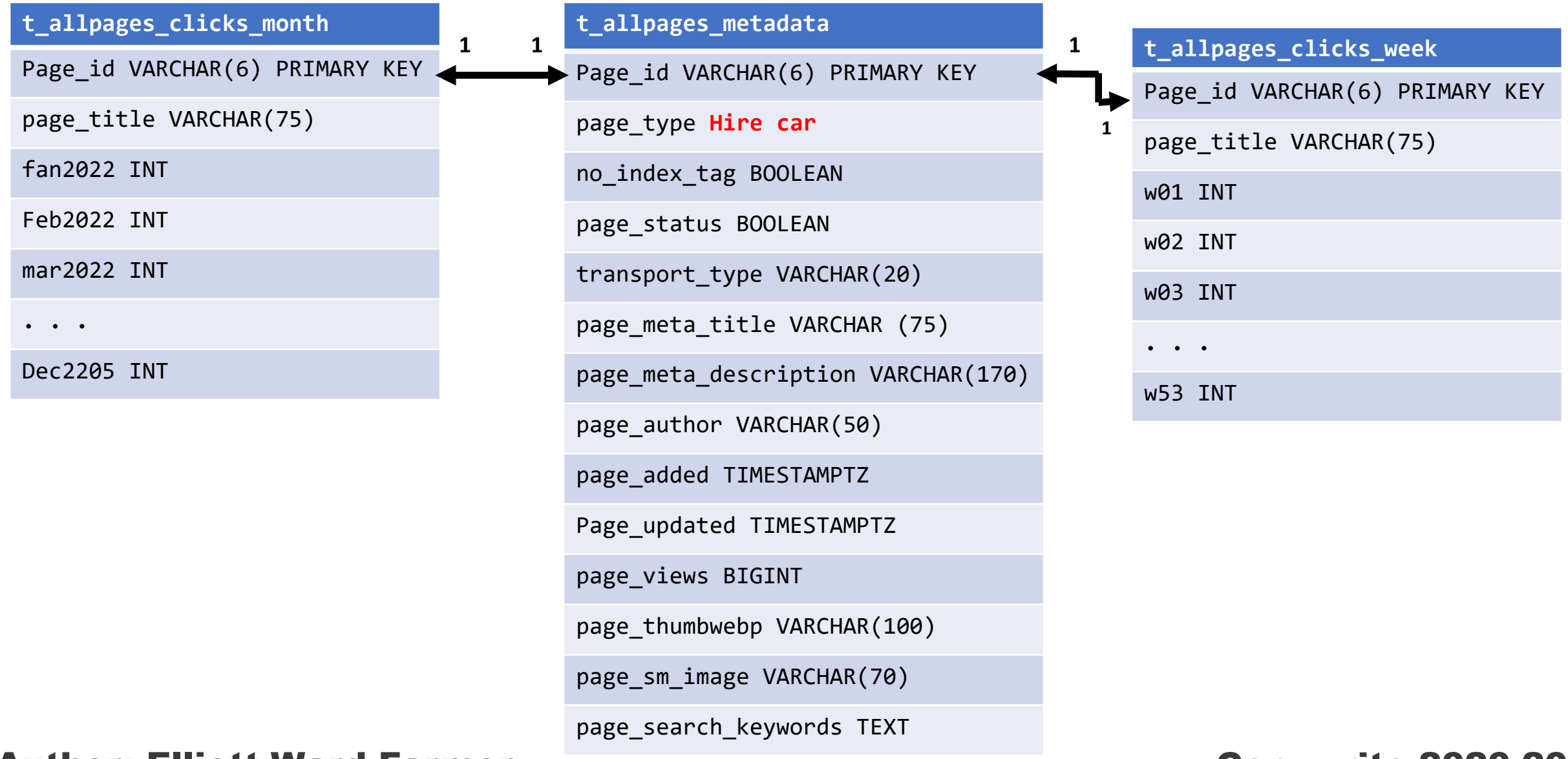
t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type page type
no_index_tag BOOLEAN
page_status BOOLEAN
transport_type VARCHAR(20)
page_meta_title VARCHAR (75)
page_meta_description VARCHAR(170)
page_author VARCHAR(50)
page_added TIMESTAMPTZ
Page_updated TIMESTAMPTZ
page_views BIGINT
page_thumbwebp VARCHAR(100)
page_sm_image VARCHAR(70)
page_search_keywords TEXT

t_allpages_url_handler_old
Page_id VARCHAR(6) PRIMARY KEY
http_error old_url_http_redirect_code
Urlparam0 valid_url_component
Urlparam1 valid_url_component
Urlparam2 valid_url_component

t_allpages_url_handler_new
Page_id VARCHAR(6) PRIMARY KEY
page_type HTTP error
no_index_tag BOOLEAN
page_status BOOLEAN
Urlparam0 valid_url_component
Urlparam1 valid_url_component
Urlparam2 valid_url_component

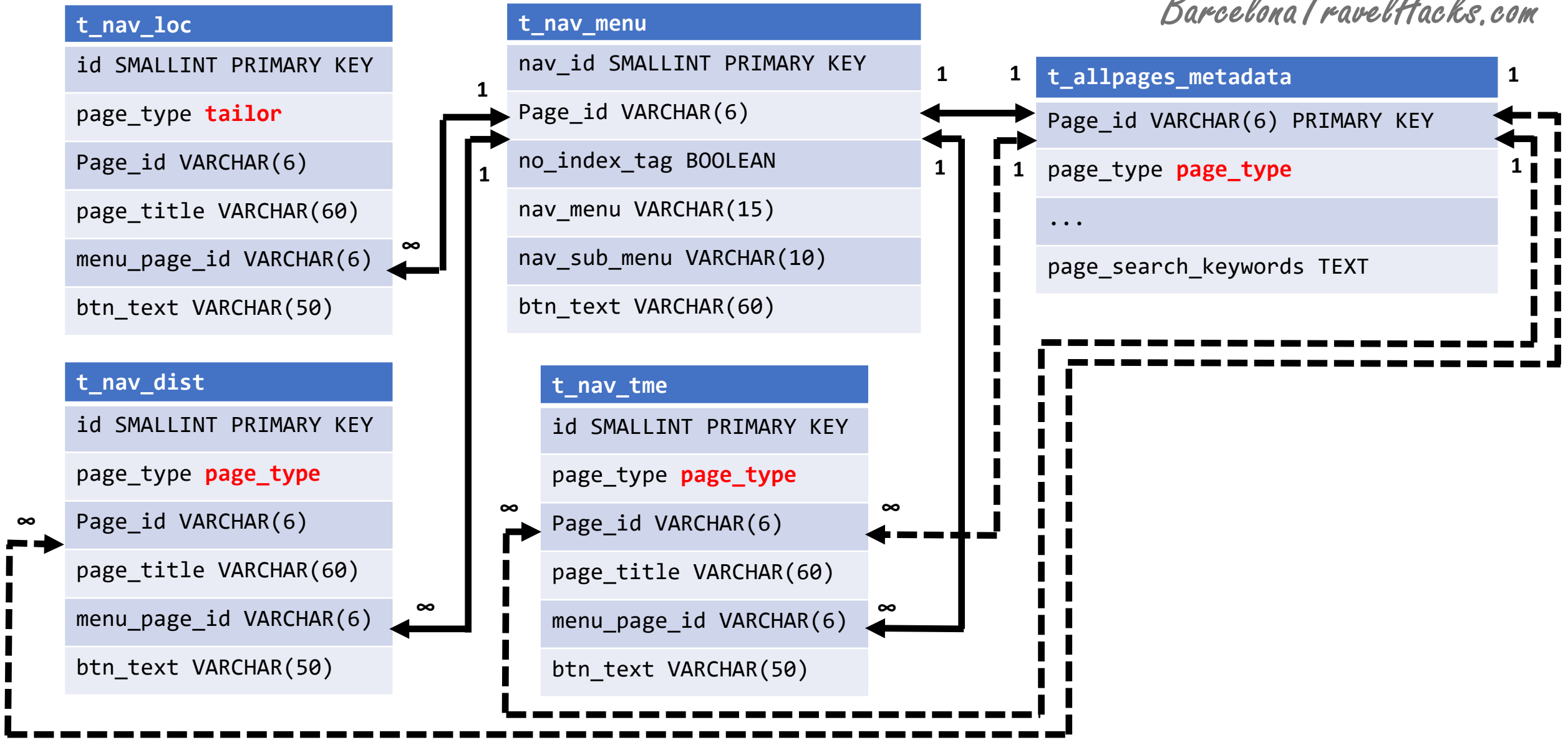
Author: Elliott Ward Farmer

2.2 Core – Page Counters



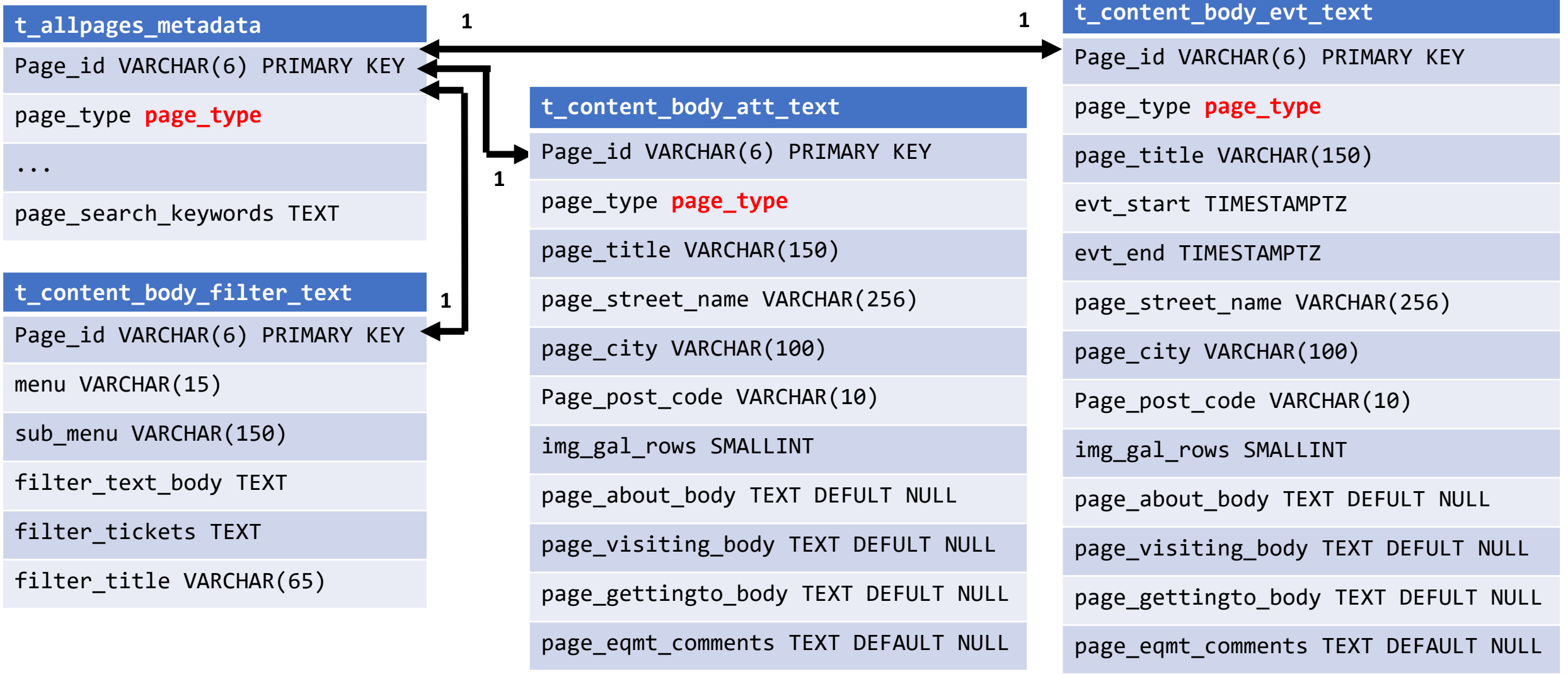
Author: Elliott Ward Farmer

2.3 Core – Site Nav & Header



Author: Elliott Ward Farmer

2.4 Core – Page Text Content



Author: Elliott Ward Farmer

2.5 Core – Page Text Content

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type defat
...
page_search_keywords TEXT

1 1

t_content_body_info_text
Page_id VARCHAR(6) PRIMARY KEY
info_title VARCHAR(150)
Info_tabs VARCHAR[]
info_content_id VARCHAR[]

1

∞

t_content_body_info_sections
section_id VARCHAR(8)
Page_id VARCHAR(6) PRIMARY KEY
section_name VARCHAR(30)
section_title VARCHAR(65)
section_body TEXT

1

∞

t_content_body_info_img
img_id VARCHAR(9) PRIMARY KEY
Page_id VARCHAR(6) NOT NULL
no_index_tag BOOLEAN NOT NULL
img_type SMALLINT NOT NULL
img_author VARCHAR(60) NOT NULL
img_title VARCHAR(60) NOT NULL
source_url VARCHAR(160)
pdf_status BOOLEAN NOT NULL
pdf_path VARCHAR(160)
img_thumb_webp VARCHAR(160)

Author: Elliott Ward Farmer

2.6 Core – Page Text Content

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type page postcode
...
page_search_keywords TEXT

1 1

t_content_body_travel_text
Page_id VARCHAR(6) PRIMARY KEY
page_type defat
page_title VARCHAR(150)
page_city VARCHAR(100)
page_postcode VARCHAR(10)
page_about_body TEXT DEFAULT NULL
page_events_body TEXT DEFAUL NULL
page_food_drink_body TEXT DEFAULT NULL
page_affiliate_links_body TEXT DEFAULT NULL
page_gettingto_body TEXT DEFAULT NULL
page_eqmt_body TEXT DEFAULT NULL
page_prices_body TEXT DEFAULT NULL
page_accomodation_body TEXT DEFAULT NULL

t_content_body_text_block
text_id INT
text_block TEXT

Author: Elliott Ward Farmer

2.7 Core – Page Text Content

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type **text block**
...
page_search_keywords TEXT

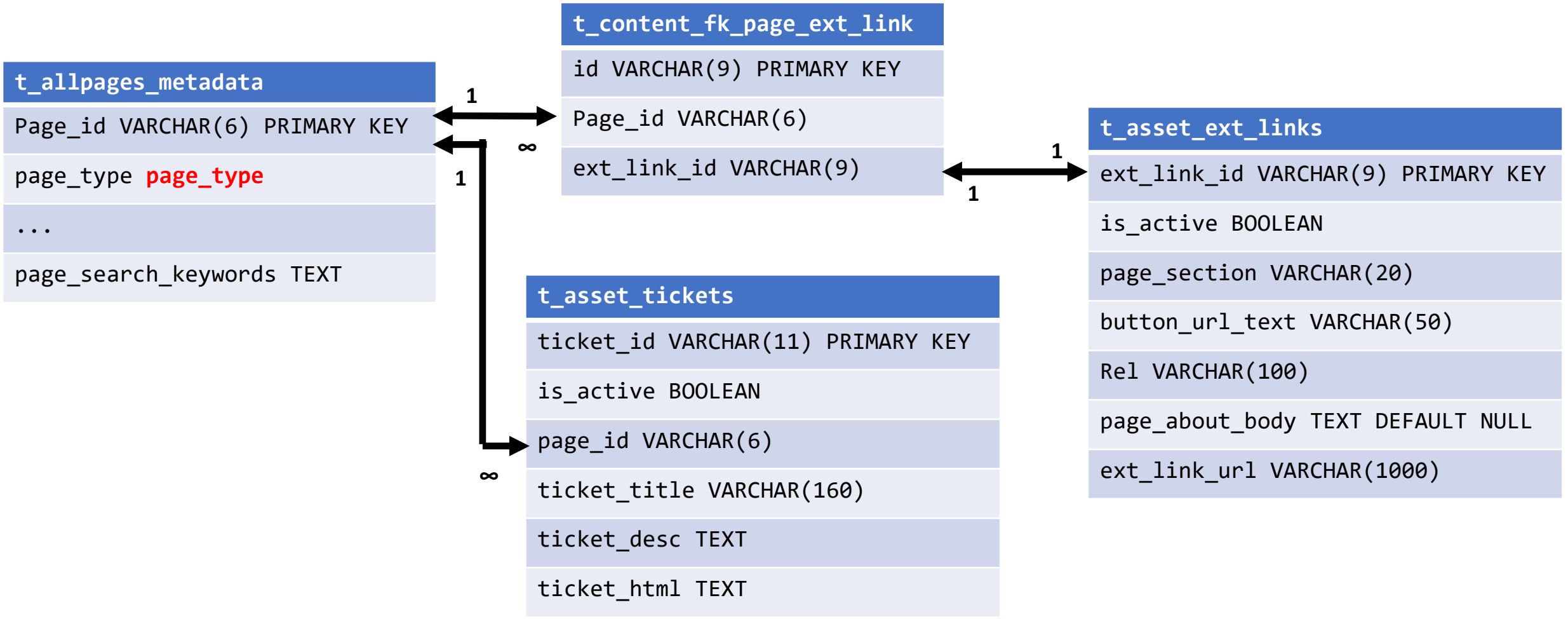
1 1

t_content_body_bcn_text
Page_id VARCHAR(6) PRIMARY KEY
page_type **text block**
page_title VARCHAR(150)
page_city VARCHAR(100)
page_postcode VARCHAR(10)
page_about_body TEXT DEFAULT NULL
page_events_body TEXT DEFAUL NULL
page_food_drink_body TEXT DEFAULT NULL
page_affiliate_links_body TEXT DEFAULT NULL
page_gettingto_body TEXT DEFAULT NULL
page_eqmt_body TEXT DEFAULT NULL
page_prices_body TEXT DEFAULT NULL
page_accomodation_body TEXT DEFAULT NULL

Author: Elliott Ward Farmer

2.8 Assets – Page Content Ext Links & Tickets

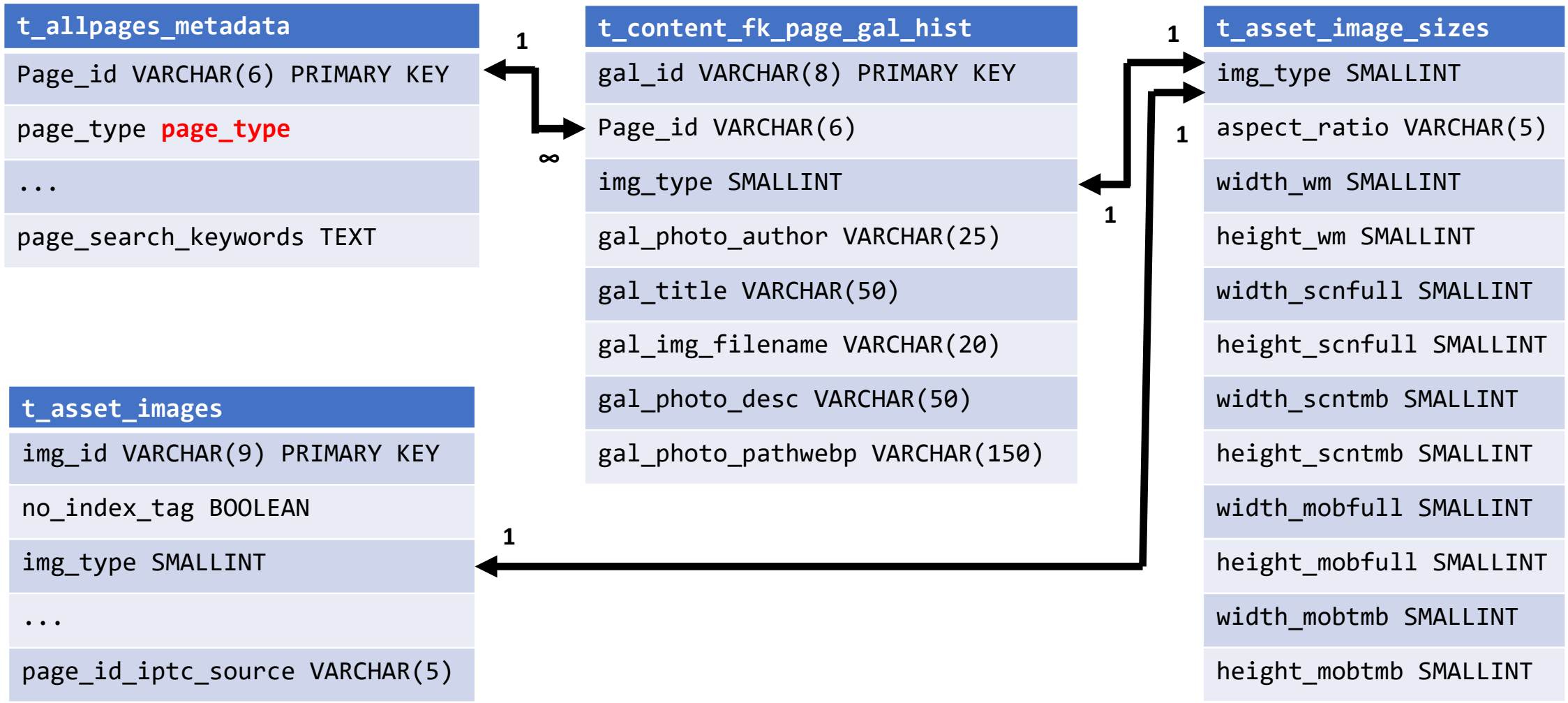
BarcelonaTravelHacks.com



Author: Elliott Ward Farmer

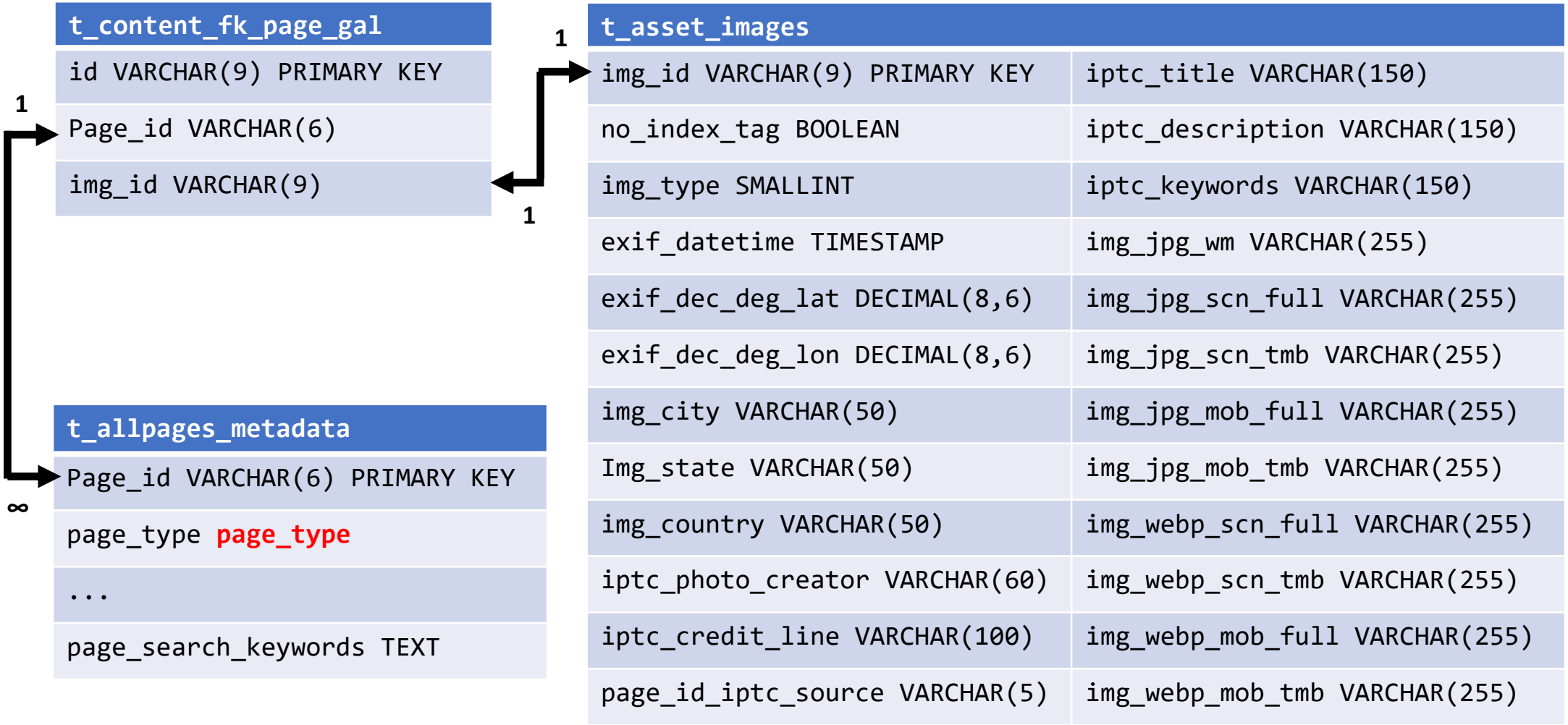
2.9 Assets – Page Content Historic Images

BarcelonaTravelHacks.com



Author: Elliott Ward Farmer

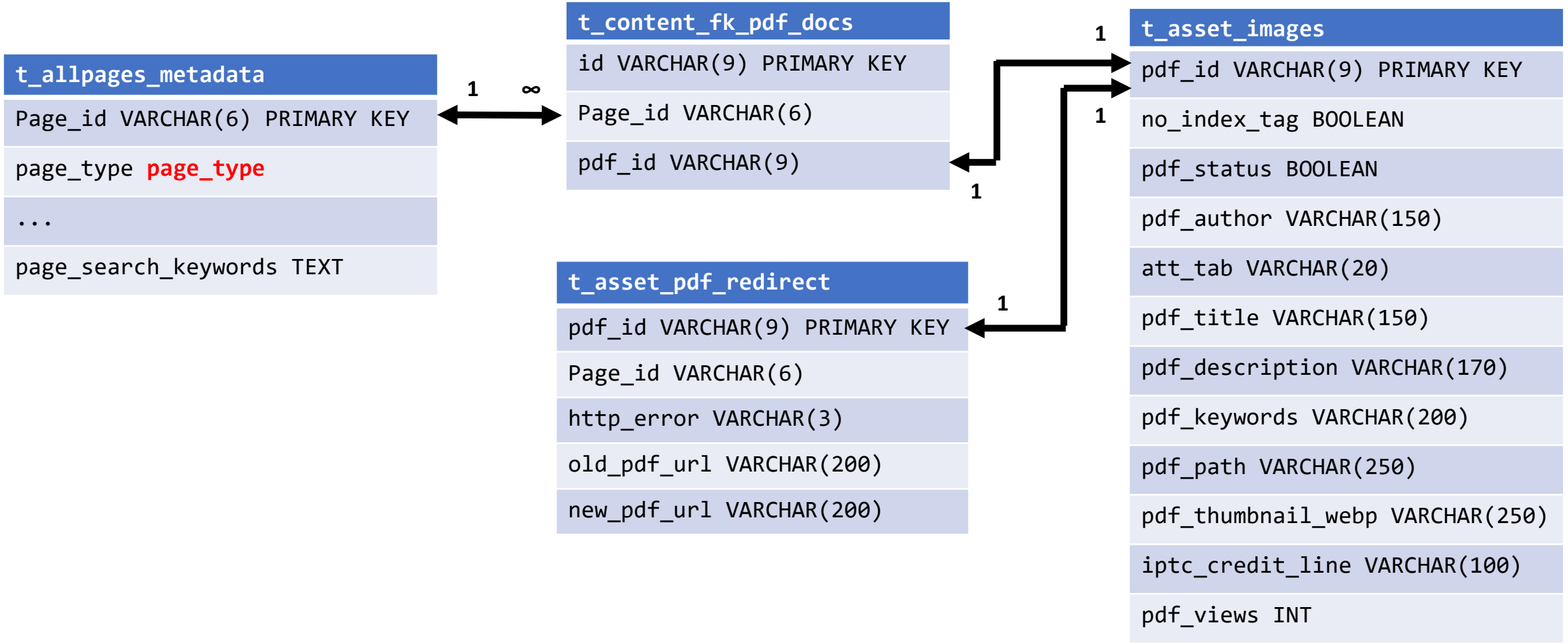
2.10 Assets – Page Content Images



Author: Elliott Ward Farmer

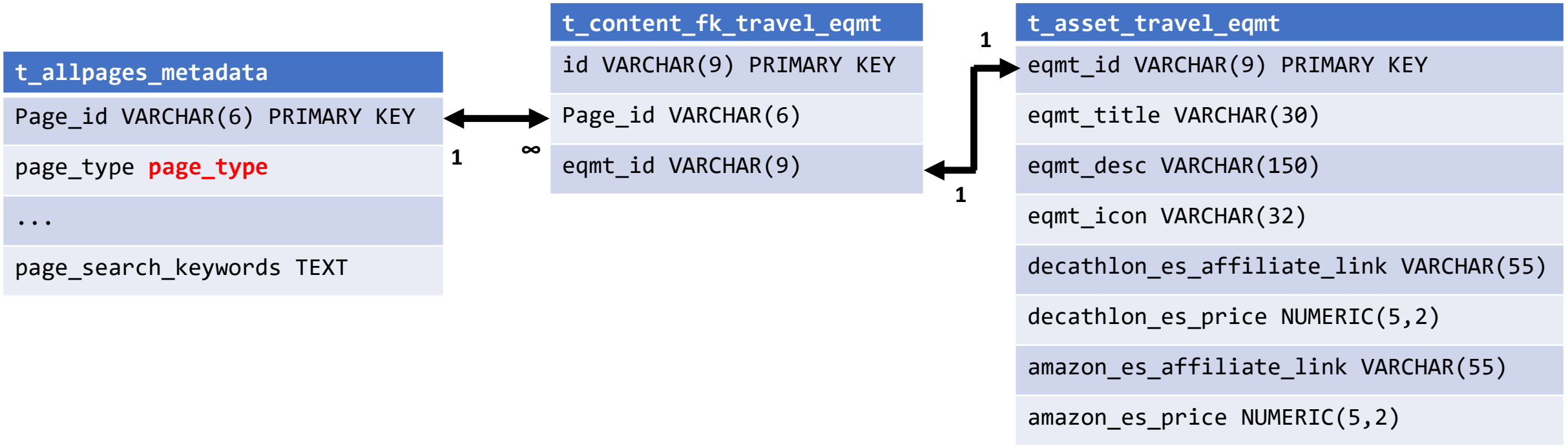
2.11 Assets – Page Content Docs

BarcelonaTravelHacks.com



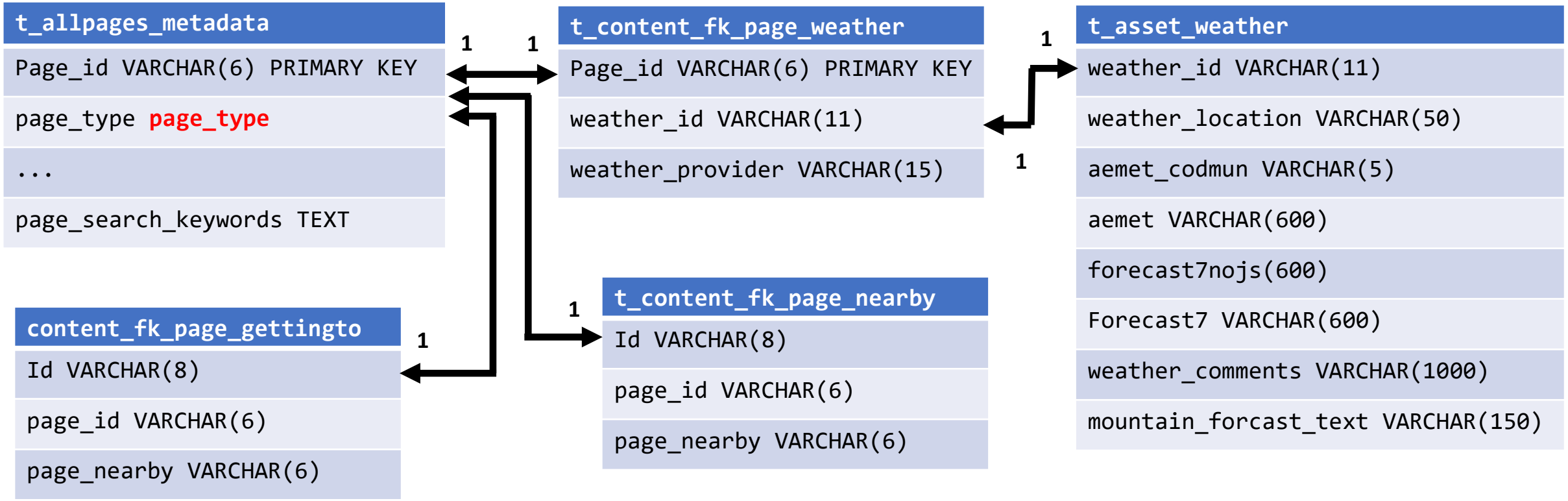
Author: Elliott Ward Farmer

2.12 Assets – Page Content Eqmt



Author: Elliott Ward Farmer

2.13 Assets – Page Content Weather



2.14 Assets – Page Nearby & Getting To

Author: Elliott Ward Farmer

2.15 Assets – Page Content Prices

BarcelonaTravelHacks.com

t_content_fk_page_attevt_prices
page_id VARCHAR(6) PRIMARY KEY
adult_travel_fare_txt VARCHAR(20)
adult_travel_fare_dec NUMERIC(5,2)
child_travel_fare_txt VARCHAR(20)
child_travel_fare_dec NUMERIC(5,2)
adult_ticket_text VARCHAR(20)
adult_ticket_dec DECIMAL(5,2)
adult_ticket_cond VARCHAR(60)
child_ticket_text VARCHAR(20)
child_ticket_dec NUMERIC (5,2)
child_ticket_cond VARCHAR(60)
adult_total NUMERIC(5,2) **GENERATED ALWAYS AS (adult_travel_fare_dec+adult_ticket_dec) STORED**
child_total NUMERIC(5,2) **GENERATED ALWAYS AS (child_travel_fare_dec+child_ticket_dec) STORED**
group_family_ticket VARCHAR(150)
prices_notes VARCHAR(150)
free_days VARCHAR(150)

1 — 1

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type page_type
...
page_search_keywords TEXT

1

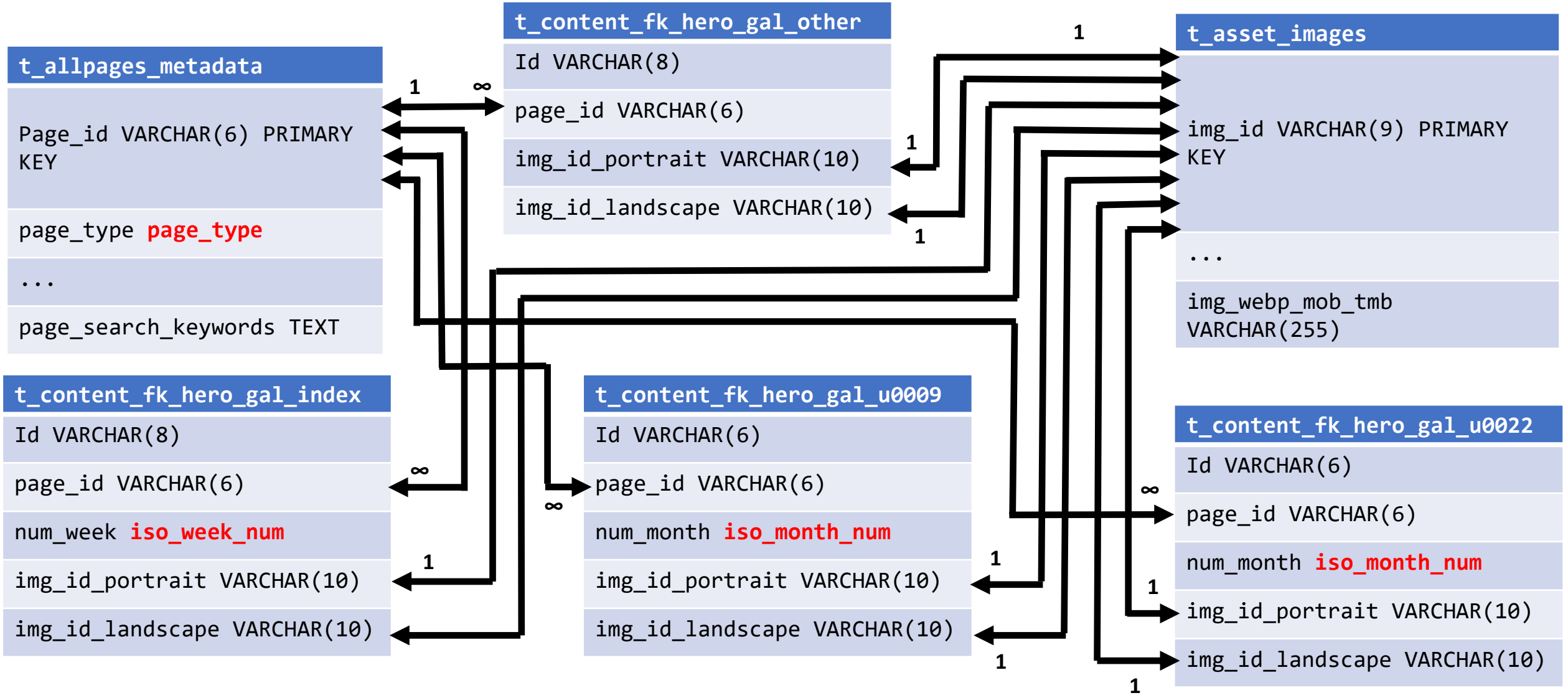
t_content_fk_page_day_trip_prices
id VARCHAR(8) PRIMARY KEY
page_id VARCHAR(6)
Description VARCHAR(260)
adult_ticket_cost NUMERIC(5,2)
adult_ticket_discounts VARCHAR(60)
child_travel_fare_dec NUMERIC(5,2)
child_ticket_cost VARCHAR(60)
child_ticket_discounts DECIMAL(5,2)

1

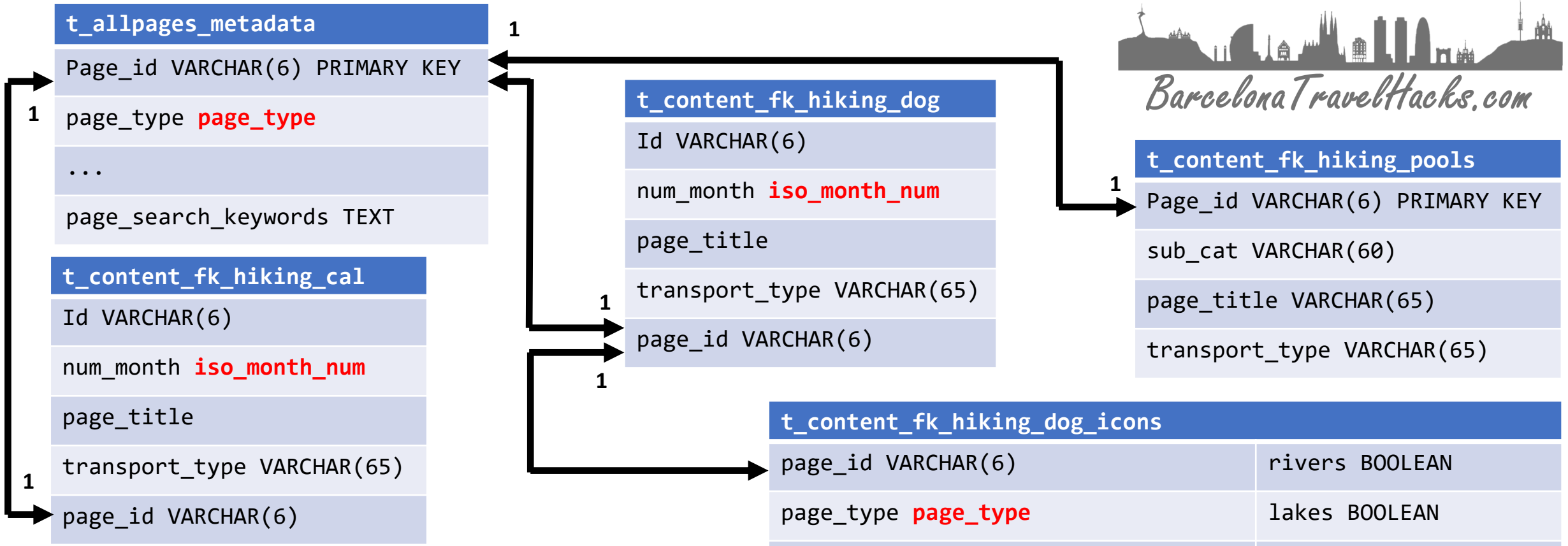
Author: Elliott Ward Farmer

2.16 Assets – Page Content Hero Galleries

BarcelonaTravelHacks.com



Author: Elliott Ward Farmer



BarcelonaTravelHacks.com

t_content_fk_hiking_dog_icons	
page_id VARCHAR(6)	rivers BOOLEAN
page_type page_type	lakes BOOLEAN
page_title VARCHAR(65)	waterfalls BOOLEAN
score dog_hiking_icons_score	ladderschains BOOLEAN
beaches BOOLEAN	walkways BOOLEAN
forest BOOLEAN	Farmanimals BOOLEAN
pines BOOLEAN	Wildanimals BOOLEAN
procession dog_hiking_icons_score	

2.18 Assets – Page Content Hiking

Author: Elliott Ward Farmer

2.19 Spatial Data

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type page_type
no_index_tag BOOLEAN
page_status BOOLEAN
transport_type VARCHAR(20)
page_meta_title VARCHAR (75)
page_meta_description VARCHAR(170)
page_author VARCHAR(50)
page_added TIMESTAMPTZ
Page_updated TIMESTAMPTZ
page_views BIGINT
page_thumbwebp VARCHAR(100)
page_sm_image VARCHAR(70)
page_search_keywords TEXT

1

1

∞

∞

GEOMETRY Datatype is available via the postGIS Extension.

t_geodata_hotels
page_id VARCHAR(6)
geo_title
geo_data GEOMETRY
loc_param

t_geodata_path
path_id VARCHAR(9) PRIMARY KEY
page_id VARCHAR(6)
geo_title
geo_data GEOMETRY
color VARCHAR(10)

t_geodata_poi
poi_id VARCHAR(9) PRIMARY KEY
page_id VARCHAR(6)
geo_title
geo_data GEOMETRY
color VARCHAR(10)

Author: Elliott Ward Farmer

2.20 Domain Data Types

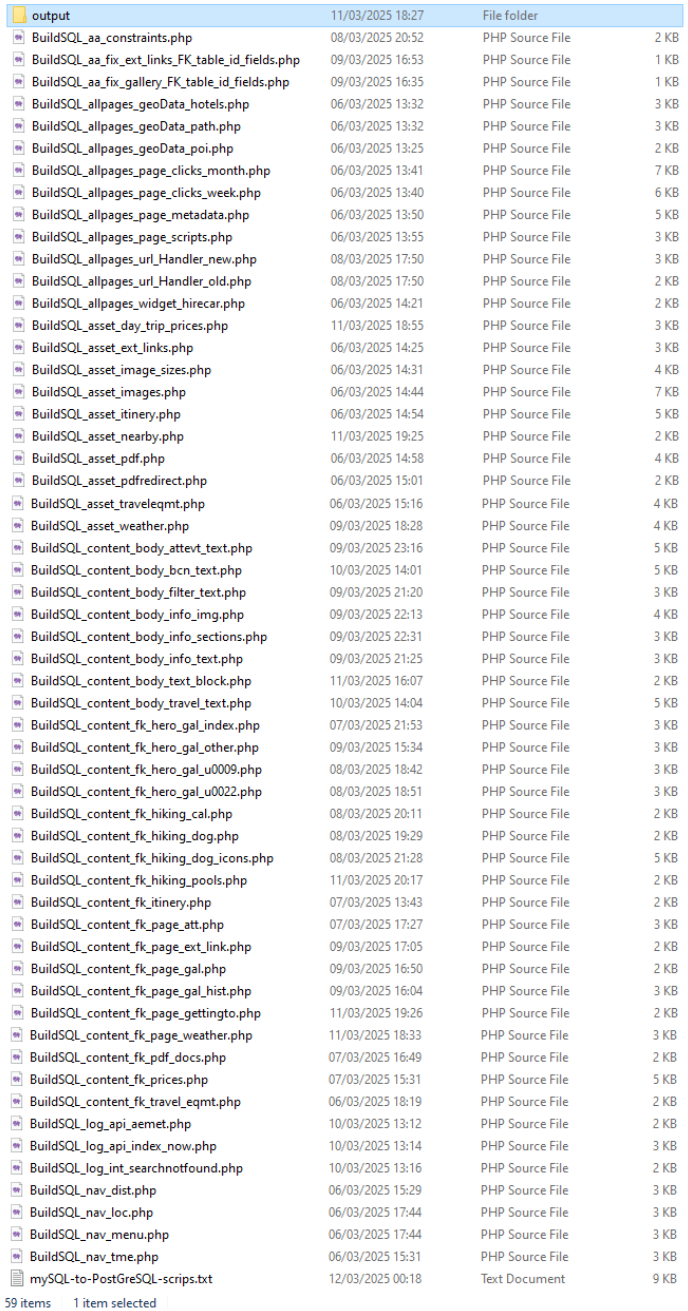

```
// 1. Build iso week Number CHECK DOMAIN
-- ISO WEEK NUMBER 1 to 53 --
CREATE DOMAIN iso_week_num AS SMALLINT
NOT NULL CHECK (VALUE >= 1 AND VALUE <= 53);
----
// 2. Build old_url_http_redirect_code CHECK DOMAIN
-- Build old_url_http_redirect_code CHECK DOMAIN --
CREATE DOMAIN old_url_http_redirect_code AS SMALLINT
CHECK (VALUE IN (301,403,410));
----
// 3. Build urlComponent CHECK DOMAIN
-- Build urlComponent CHECK DOMAIN --
CREATE DOMAIN valid_url_component AS VARCHAR(60)
CHECK (VALUE ~ '^[a-z0-9]([a-z0-9-]*[a-z0-9])?$');
----
// 4. Build page_type DOMAIN CHECK
-- Build page_type DOMAIN CHECK --
CREATE DOMAIN page_type AS VARCHAR(6)
CHECK (VALUE IN ('a','b','e','i','t','u','v','w','x','y','401','404','410','dev-t','index','search'));
----
```

```
// 5. Build iso month Number CHECK DOMAIN
-- ISO MONTH NUMBER 1 to 12 --
CREATE DOMAIN iso_month_num AS SMALLINT
NOT NULL CHECK (VALUE >= 1 AND VALUE <= 12);
----
// 6. Build dog_hiking_icons_score CHECK DOMAIN
-- NUMBER 1 to 5 --
CREATE DOMAIN dog_hiking_icons_score AS SMALLINT
NOT NULL CHECK (VALUE >= 1 AND VALUE <= 5);
----
```

Author: Elliott Ward Farmer

3. Scripts to import data from MySQL to PostgreSQL

Author: Elliott Ward Farmer



output	11/03/2025 18:27	File folder	
BuildSQL_aa_constraints.php	08/03/2025 20:52	PHP Source File	2 KB
BuildSQL_aa_fix_ext_links_FK_table_id_fields.php	09/03/2025 16:53	PHP Source File	1 KB
BuildSQL_aa_fix_gallery_FK_table_id_fields.php	09/03/2025 16:35	PHP Source File	1 KB
BuildSQL_allpages_geoData_hotels.php	06/03/2025 13:32	PHP Source File	3 KB
BuildSQL_allpages_geoData_path.php	06/03/2025 13:32	PHP Source File	3 KB
BuildSQL_allpages_geoData_poi.php	06/03/2025 13:25	PHP Source File	2 KB
BuildSQL_allpages_page_clicks_month.php	06/03/2025 13:41	PHP Source File	7 KB
BuildSQL_allpages_page_clicks_week.php	06/03/2025 13:40	PHP Source File	6 KB
BuildSQL_allpages_page_metadata.php	06/03/2025 13:50	PHP Source File	5 KB
BuildSQL_allpages_page_scripts.php	06/03/2025 13:55	PHP Source File	3 KB
BuildSQL_allpages_url_Handler_new.php	08/03/2025 17:50	PHP Source File	3 KB
BuildSQL_allpages_url_Handler_old.php	08/03/2025 17:50	PHP Source File	2 KB
BuildSQL_allpages_widget_hirecar.php	06/03/2025 14:21	PHP Source File	2 KB
BuildSQL_asset_day_trip_prices.php	11/03/2025 18:55	PHP Source File	3 KB
BuildSQL_asset_ext_links.php	06/03/2025 14:25	PHP Source File	3 KB
BuildSQL_asset_image_sizes.php	06/03/2025 14:31	PHP Source File	4 KB
BuildSQL_asset_images.php	06/03/2025 14:44	PHP Source File	7 KB
BuildSQL_asset_itinery.php	06/03/2025 14:54	PHP Source File	5 KB
BuildSQL_asset_nearby.php	11/03/2025 19:25	PHP Source File	2 KB
BuildSQL_asset_pdf.php	06/03/2025 14:58	PHP Source File	4 KB
BuildSQL_asset_pdfredirect.php	06/03/2025 15:01	PHP Source File	2 KB
BuildSQL_asset_travelеqmt.php	06/03/2025 15:16	PHP Source File	4 KB
BuildSQL_asset_weather.php	09/03/2025 18:28	PHP Source File	4 KB
BuildSQL_content_body_attevt_text.php	09/03/2025 23:16	PHP Source File	5 KB
BuildSQL_content_body_bcn_text.php	10/03/2025 14:01	PHP Source File	5 KB
BuildSQL_content_body_filter_text.php	09/03/2025 21:20	PHP Source File	3 KB
BuildSQL_content_body_info_img.php	09/03/2025 22:13	PHP Source File	4 KB
BuildSQL_content_body_info_sections.php	09/03/2025 22:31	PHP Source File	3 KB
BuildSQL_content_body_info_text.php	09/03/2025 21:25	PHP Source File	3 KB
BuildSQL_content_body_text_block.php	11/03/2025 16:07	PHP Source File	2 KB
BuildSQL_content_body_travel_text.php	10/03/2025 14:04	PHP Source File	5 KB
BuildSQL_content_fk_hero_gal_index.php	07/03/2025 21:53	PHP Source File	3 KB
BuildSQL_content_fk_hero_gal_other.php	09/03/2025 15:34	PHP Source File	3 KB
BuildSQL_content_fk_hero_gal_u0009.php	08/03/2025 18:42	PHP Source File	3 KB
BuildSQL_content_fk_hero_gal_u0022.php	08/03/2025 18:51	PHP Source File	3 KB
BuildSQL_content_fk_hiking_cal.php	08/03/2025 20:11	PHP Source File	2 KB
BuildSQL_content_fk_hiking_dog.php	08/03/2025 19:29	PHP Source File	2 KB
BuildSQL_content_fk_hiking_dog_icons.php	08/03/2025 21:28	PHP Source File	5 KB
BuildSQL_content_fk_hiking_pools.php	11/03/2025 20:17	PHP Source File	2 KB
BuildSQL_content_fk_itinery.php	07/03/2025 13:43	PHP Source File	2 KB
BuildSQL_content_fk_page_att.php	07/03/2025 17:27	PHP Source File	3 KB
BuildSQL_content_fk_page_ext_link.php	09/03/2025 17:05	PHP Source File	2 KB
BuildSQL_content_fk_page_gal.php	09/03/2025 16:50	PHP Source File	2 KB
BuildSQL_content_fk_page_gal_hist.php	09/03/2025 16:04	PHP Source File	3 KB
BuildSQL_content_fk_page_gettingto.php	11/03/2025 19:26	PHP Source File	2 KB
BuildSQL_content_fk_page_weather.php	11/03/2025 18:33	PHP Source File	3 KB
BuildSQL_content_fk_pdf_docs.php	07/03/2025 16:49	PHP Source File	2 KB
BuildSQL_content_fk_prices.php	07/03/2025 15:31	PHP Source File	5 KB
BuildSQL_content_fk_travel_eqmt.php	06/03/2025 18:19	PHP Source File	2 KB
BuildSQL_log_api_aemet.php	10/03/2025 13:12	PHP Source File	2 KB
BuildSQL_log_api_index_now.php	10/03/2025 13:14	PHP Source File	3 KB
BuildSQL_log_int_searchnotfound.php	10/03/2025 13:16	PHP Source File	2 KB
BuildSQL_nav_dist.php	06/03/2025 15:29	PHP Source File	3 KB
BuildSQL_nav_loc.php	06/03/2025 17:44	PHP Source File	3 KB
BuildSQL_nav_menu.php	06/03/2025 17:44	PHP Source File	3 KB
BuildSQL_nav_tme.php	06/03/2025 15:31	PHP Source File	3 KB
mySQL-to-PostGreSQL-scrips.txt	12/03/2025 00:18	Text Document	9 KB

59 items 1 item selected

3.1 MySQL to PostgreSQL Scripts

As a quick overview it took 59 scripts to migrate all the tables and build the domain data types.

The first script is to create the custom domain data types.

There is also a script to fix and harmonise the ID fields of one table.

The rest of the scripts are each dedicated to one table and perform three functions:

1) Build the PostgreSQL table with relevant datatypes, constraints and foreign keys.
2) Query the MySQL table and iterate over each row reformatting the data for PostgreSQL.
3) Performs a PostgreSQL insert query for each row.

Each script produces an SQL output file that I can import into PostgreSQL, and with an imaginary slurpy sound the table is migrated from MySQL to PostgreSQL.

Author: Elliott Ward Farmer

3.2 Handling Text and NULL fields

BarcelonaTravelHacks.com

```
foreach ($sqlArr as $row) {
  $pageId = $row["pageId"];
  $geoTitle = $row["geoTitle"];
  $title = str_replace("'", "''", $geoTitle);
  $geodata = $row["geoData"];
  if ($geodata === NULL) {
    $geo = 'NULL';
  } else {
    $geo = str_replace(" ", ",", $geodata);
  }
  $locParam = $row["locParam"];
  if ($locParam === NULL) {
    $loc = 'NULL';
  } else {
    $loc = "'".$locParam."'";
  }
```

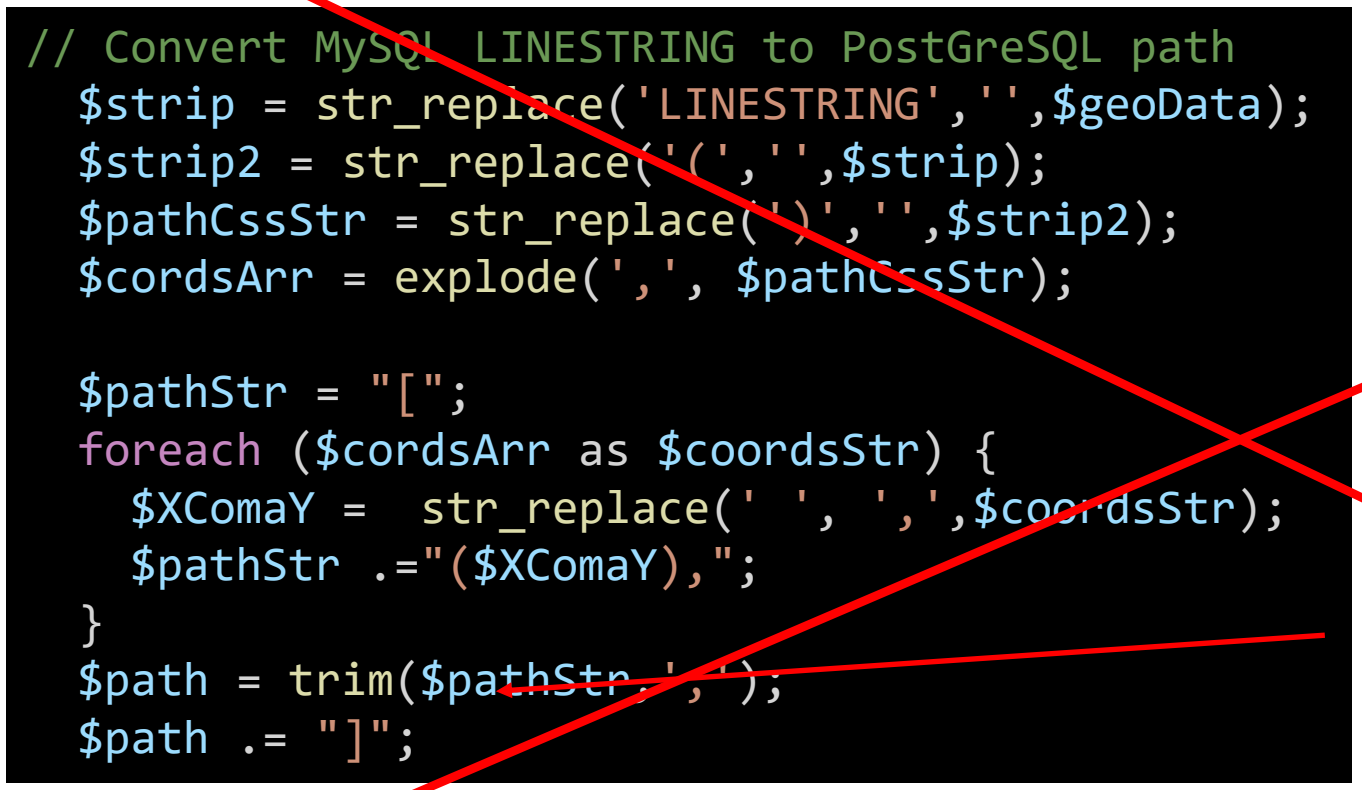
In this code block I am iterating over each row within a foreach loop.

I am **escaping single quotes** in text fields by changing a single quote to two single quotes using a str_replace function.

If a column contains a NULL value then I am writing it as 'NULL' so that in the output sql script it has the correct forming to be interpreted by PostgreSQL. Note that in the SQL output file the NULL appeas without single quotes.

Author: Elliott Ward Farmer

3.3 MySQL LINESTRING to PostgreSQL Path



```
// Convert MySQL LINESTRING to PostGreSQL path
  $strip = str_replace('LINESTRING','',$geoData);
  $strip2 = str_replace('(','',$strip);
  $pathCssStr = str_replace(')','',$strip2);
  $cordsArr = explode(',', $pathCssStr);

  $pathStr = "[";
  foreach ($cordsArr as $coordsStr) {
    $XComaY =  str_replace(' ', ',',$coordsStr);
    $pathStr .="($XComaY),";
  }
  $path = trim($pathStr,",");
  $path .= "]";
```

In this code block I am iterating over a LINESTRING MySQL datatype to convert it to a PostgreSQL datatype.

MySQL outputs the LINESTRING type as:

LINESTRING(x y, x y, etc)

And PostgreSQL takes PATH data type as an array with the format:

[(x,y),(x,y), etc]

NO!!!!!!!

Author: Elliott Ward Farmer

3.4 PostgreSQL Geometric Types (Are Rubbish!)

BarcelonaTravelHacks.com

8.8. Geometric Types

PostgreSQL ships with Geometric Data types already built in.

https://www.postgresql.org/docs/current/datatype-geometric.html

Geometric data types represent two-dimensional spatial objects. Table 8.20 shows the geometric types available in PostgreSQL.

Table 8.20. Geometric Types

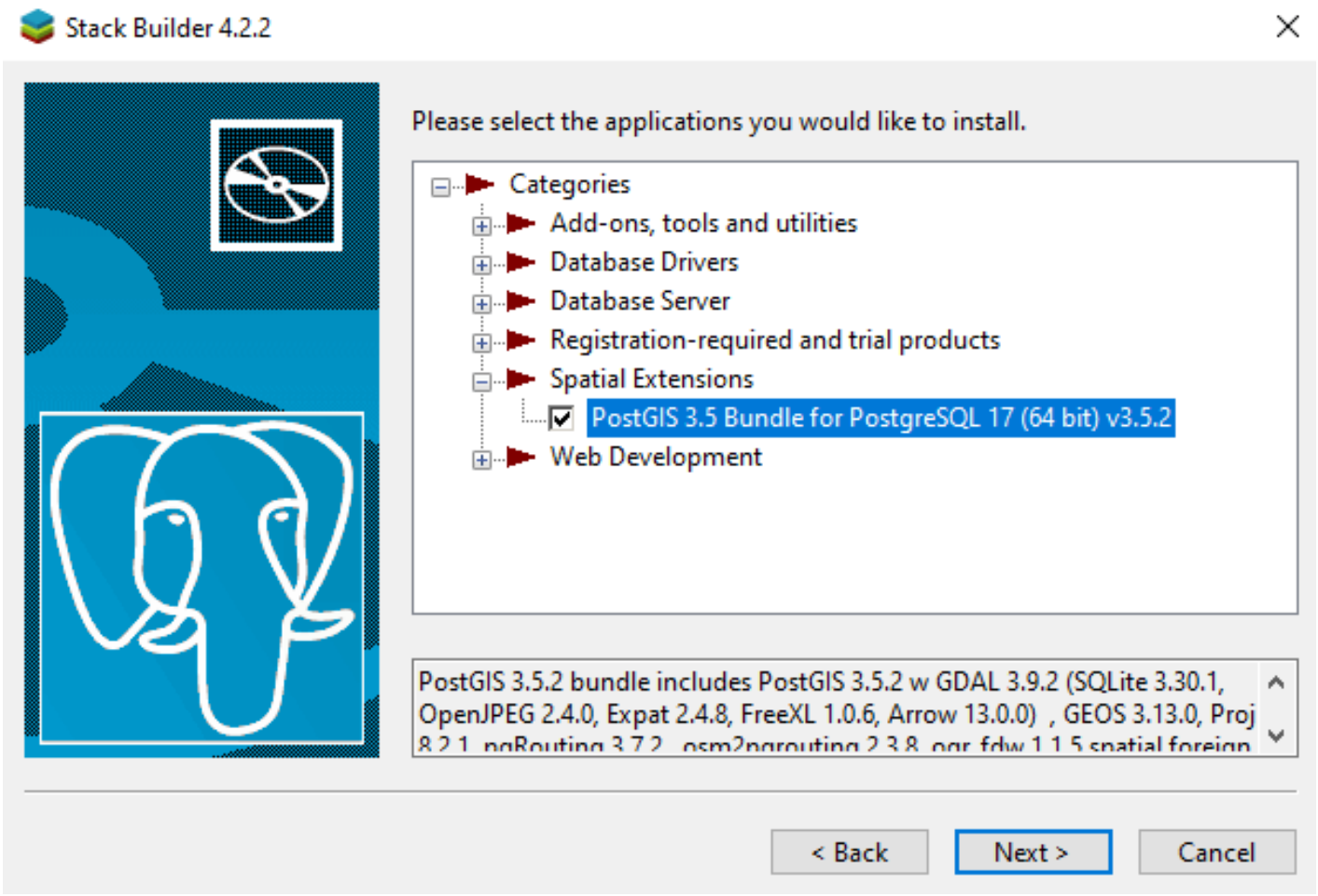
Name	Storage Size	Description	Representation
`point`	16 bytes	Point on a plane	(x,y)
`line`	24 bytes	Infinite line	{A,B,C}
`lseg`	32 bytes	Finite line segment	[(x1,y1),(x2,y2)]
`box`	32 bytes	Rectangular box	(x1,y1),(x2,y2)
`path`	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
`path`	16+16n bytes	Open path	[(x1,y1),...]
`polygon`	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
`circle`	24 bytes	Circle	<(x,y),r> (center point and radius)

However, after initially using these for geospatial data, I found that the format is not the same when building geoJSON feature collections from these data types and I had to write functions with foreach loops to break the data down for point and linestrings to x and y coordinates then rebuild a correctly formatted geometry data type that can be converted to geoJSON.

Author: Elliott Ward Farmer

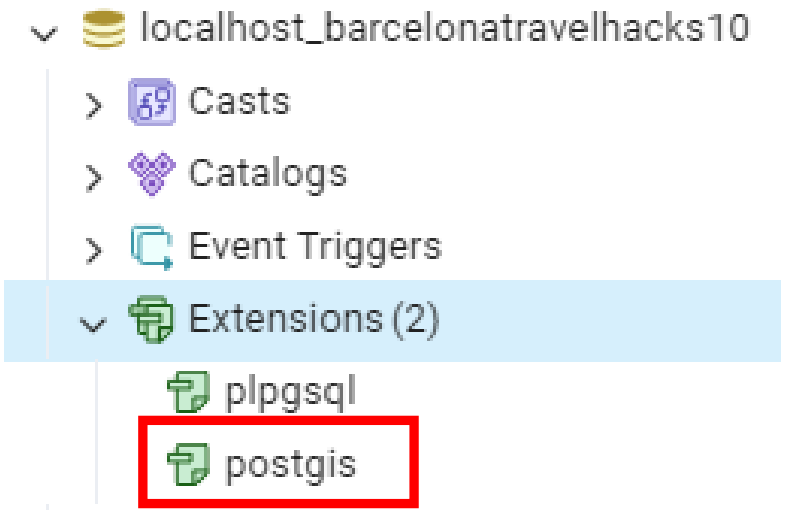
3.5 PostGis extension for PostgreSQL

PostgreSQL native composite Geometric data types are 'lacking' for geospatial data.



Use Stack builder to install postGis bundle of spatial Extensions.

PostGis comes with a ton of built in functions for formatting and performing calculations on geospatial data and working with GeoJSON.



Author: Elliott Ward Farmer

3.5 MySQL to PostgreSQL Geospatial

Type	MySQL / Maria DB / phpMyAdmin	PostgreSQL / PgAdmin4
Point RAW	'POINT(2.1749603426429256 41.40390130683638)',0	'POINT(2.1749603426429256 41.40390130683638)'
Point (stored as)	[GEOMETRY - 25 B]	01010000008FFCAD9B5166014043D6BB09B3B34440
Linestring RAW	'LINESTRING(x y,x y, x y, etc)',0	LINESTRING(x y, x y, x y etc)
Linestring (stored as)	[GEOMETRY - 749 B]	01020000002E0000003F3BE0BA626601408A213999B8B34440D9 ……1408F56B54A3B444402235ED629A690140B4942C27A1B44440
Extracted with	AsText(geoData)	ST_AsText(geo_data)

Spatial data (geometry) in MySQL and PostgreSQL is inserted in a specific RAW format and stored as a compressed HEX data string. To extract the raw geometry data from the HEX use asText function for MySQL and ST_asText for PostgreSQL.
https://dev.mysql.com/doc/refman/8.4/en/fetching-spatial-data.html
https://postgis.net/docs/ST_GeometryType.html
https://stackoverflow.com/questions/56721363/geometry-type-is-read-as-string-from-the-db

Author: Elliott Ward Farmer

3.6 Geospatial conversion script


```
// Convert MySQL POINT to PostGreSQL POINT
$point = "'".rtrim($row["geoData"], ",0")."'";

// Convert MySQL LINESTRING to PostGreSQL LINESTRING
$linestring = rtrim($row["geoData"], ",0");
```

MySQL point and linestring datatypes have an **additional trailing coma and zero** so I have used RTRIM to strip this off of the ST_ASTEXT values to get a postGis compatible data string.

Author: Elliott Ward Farmer

3.7 MySQL faux BOOLEAN

BarcelonaTravelHacks.com

```
foreach ($sqlArr as $row) {
  $pageId = $row["pageId"];

  $pageType = $row["pageType"];

  if ($row["googleAds"] === NULL) {
    $googleAds = "'false'";
  }elseif ($row["googleAds"] === "1") {
    $googleAds = "'true'";
  }

  if ($row["getYourGuide"] === NULL) {
    $getYourGuide = "'false'";
  }elseif ($row["getYourGuide"] === "1") {
    $getYourGuide = "'true'";
  }
```

This MySQL table determines what scripts added to the HTML HEADER markup for each website page.

It uses a value of 1 to represent load this script or CSS file, and a value of NULL for do not load this script or CSS file.

In PostgreSQL I am using a BOOLEAN field type changing from MySQL TINYINT(1).

So each Faux Boolean in MySQL is converted to a true/false BOOLEAN value for PostgreSQL, NOT NULL.

Author: Elliott Ward Farmer

3.8 MySQL css to PostgreSQL Array


```php
if (empty($row["pageIDs"])) {
    $pageIdArrStr2 = 'NULL';
  }
  else {
    $pageIDsTrim = rtrim($pageIDs, ',');
    $pageIdArr = explode(",",$pageIDsTrim);
    $pageIdArrStr = implode("','",$pageIdArr);
    $pageIdArrStr2 = "ARRAY ['".substr($pageIdArrStr, 0, -3)."]";
  }
```

This MySQL table is a log table that records each time I send an API request to the Index now API and it logs the dateTime, and a coma separated string of PageId's for each URL that was sent to the Index Now API. I am not doing any calculations or data manipulation on the coma separated string column. It is purely for periodic manual review. However, looking at the data, I think an Array type would be better than a coma separated string in case I wish to do any future data manipulation so the code block converts the css of 'pageId,pageId,pageId, etc' to ['pageId','pageId,'pageId, etc]

Author: Elliott Ward Farmer

3.9 SQL file output


```
-- Build allpages_geoData_poi SQL Table --
CREATE TABLE IF NOT EXISTS t_allpages_url_Handler_new
(
    page_id VARCHAR(6) PRIMARY KEY,
    page_type Page ID's NOT NULL,
    no_index_tag BOOLEAN NOT NULL DEFAULT FALSE,
    page_status BOOLEAN NOT NULL DEFAULT TRUE,
    urlparam0 valid_url_component NOT NULL,
    urlparam1 valid_url_component NOT NULL,
    urlparam2 valid_url_component
);
```

Each script produces an output SQL file which contains the SQL query to build the PostgreSQL table then below an insert query for each row.

Obviously, these scripts are very long but you get the idea of how they work and what they are made of.

```
-- SQL INSERT QUERIES --
INSERT INTO t_allpages_url_Handler_new
(
page_id,page_type,no_index_tag,page_status,urlparam0,
urlparam1,urlparam2
)
VALUES
(
'a0001','a','false','true','barcelona',
'la-eixampla',
'la-sagrada-familia'
);
```

Author: Elliott Ward Farmer

3.10 PostGreSQL Importing the .SQL


```
\i path/to/file/file_name.sql
```

I used the PostgreSQL terminal to import all the SQL files.

Note that the order of importing is critical.

I imported the sql file to create the domain types first.

I then reviewed the foreign keys importing first the tables that are referenced then the tables that reference other tables.

With a total of 54 tables, This involved some planning so I made a text file of all the SQL file names broken down into sections of batches of files in the correct order that they had to be imported.

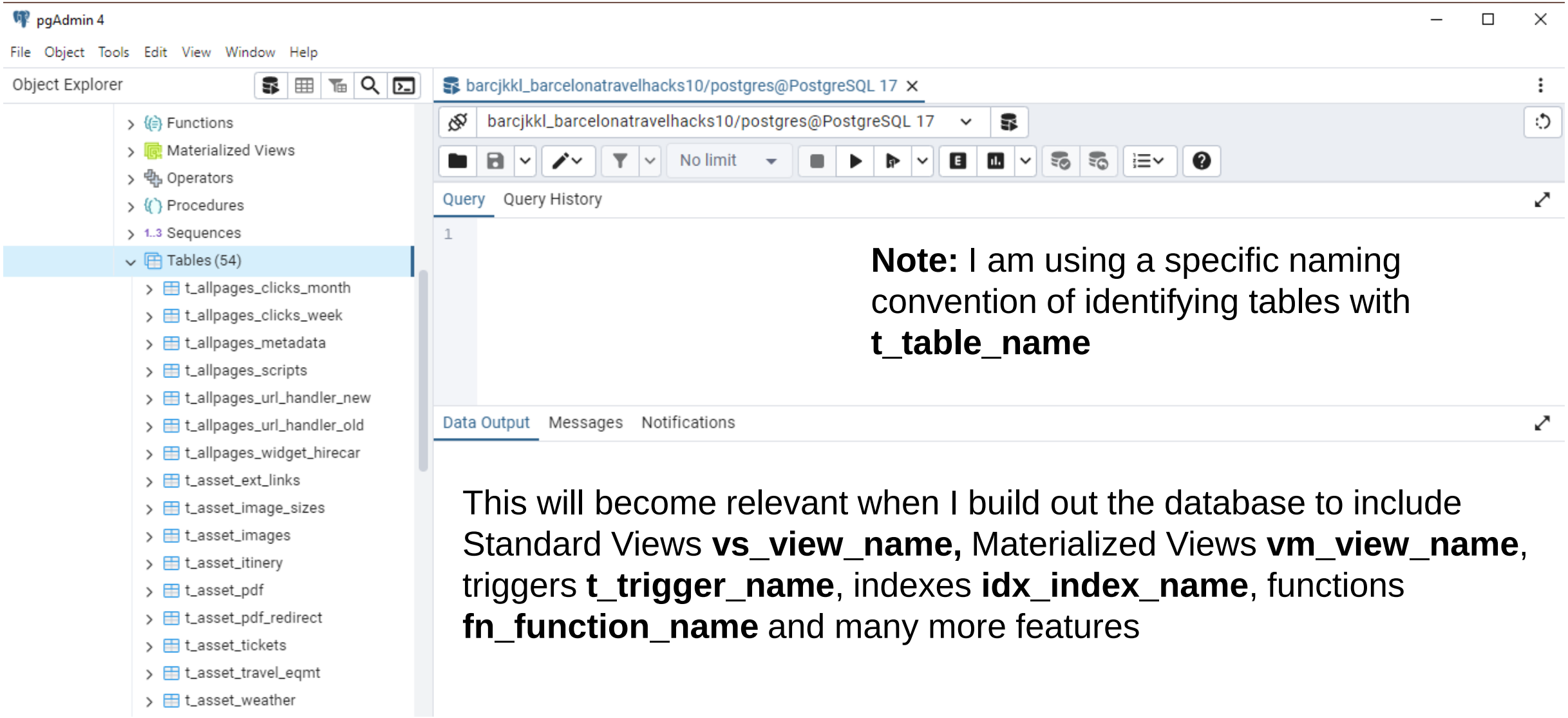
Author: Elliott Ward Farmer

3.11 Table Importing Outcome


```
barcjkkl_barcelonatravelhacks10=# \d
                    List of relations
 Schema |               Name                |   Type   |  Owner
--------+-----------------------------------+----------+----------
 public | t_allpages_clicks_month           | table    | postgres
 public | t_allpages_clicks_week            | table    | postgres
 public | t_allpages_metadata               | table    | postgres
 public | t_allpages_scripts                | table    | postgres
 public | t_allpages_url_handler_new        | table    | postgres
 public | t_allpages_url_handler_old        | table    | postgres
 public | t_allpages_widget_hirecar         | table    | postgres
 public | t_asset_ext_links                 | table    | postgres
 public | t_asset_image_sizes               | table    | postgres
 public | t_asset_images                    | table    | postgres
 public | t_asset_itinery                   | table    | postgres
 public | t_asset_pdf                       | table    | postgres
 public | t_asset_pdf_redirect              | table    | postgres
 public | t_asset_tickets                   | table    | postgres
 public | t_asset_travel_eqmt               | table    | postgres
 public | t_asset_weather                   | table    | postgres
 public | t_content_body_attevt_text        | table    | postgres
 public | t_content_body_bcn_text           | table    | postgres
 public | t_content_body_filter_text        | table    | postgres
 public | t_content_body_info_img           | table    | postgres
 public | t_content_body_info_sections      | table    | postgres
 public | t_content_body_info_text          | table    | postgres
 public | t_content_body_text_block         | table    | postgres
 public | t_content_body_travel_text        | table    | postgres
 public | t_content_fk_hero_gal_index       | table    | postgres
 public | t_content_fk_hero_gal_other       | table    | postgres
 public | t_content_fk_hero_gal_u0009       | table    | postgres
 public | t_content_fk_hero_gal_u0022       | table    | postgres
 public | t_content_fk_hiking_cal           | table    | postgres
 public | t_content_fk_hiking_dog           | table    | postgres
 public | t_content_fk_hiking_dog_icons     | table    | postgres
 public | t_content_fk_hiking_pools         | table    | postgres
 public | t_content_fk_itinery              | table    | postgres
 public | t_content_fk_page_att             | table    | postgres
 public | t_content_fk_page_attevt_prices   | table    | postgres
 public | t_content_fk_page_day_trip_prices | table    | postgres
 public | t_content_fk_page_ext_link        | table    | postgres
 public | t_content_fk_page_gal             | table    | postgres
 public | t_content_fk_page_gal_hist        | table    | postgres
 public | t_content_fk_page_gettingto       | table    | postgres
 public | t_content_fk_page_nearby          | table    | postgres
 public | t_content_fk_page_weather         | table    | postgres
 public | t_content_fk_pdf_docs             | table    | postgres
 public | t_content_fk_travel_eqmt          | table    | postgres
 public | t_geoData_hotels                  | table    | postgres
 public | t_geodata_path                    | table    | postgres
 public | t_geodata_poi                     | table    | postgres
 public | t_log_aemet_api_id_seq            | sequence | postgres
 public | t_log_api_aemet                   | table    | postgres
 public | t_log_api_index_now               | table    | postgres
 public | t_log_index_now_api_id_seq        | sequence | postgres
 public | t_log_int_searchnotfound          | table    | postgres
 public | t_nav_dist                        | table    | postgres
 public | t_nav_loc                         | table    | postgres
 public | t_nav_menu                        | table    | postgres
 public | t_nav_tme                         | table    | postgres
(56 rows)
```

Author: Elliott Ward Farmer

3.13 PgAdmin4 Review of Table Importing



Note: I am using a specific naming convention of identifying tables with **t_table_name**

This will become relevant when I build out the database to include Standard Views **vs_view_name,** Materialized Views **vm_view_name**, triggers **t_trigger_name**, indexes **idx_index_name**, functions **fn_function_name** and many more features

Author: Elliott Ward Farmer

4. Building & Testing the PostgreSQL DB Connection in the MVC code

Author: Elliott Ward Farmer

4.1 Connecting to PostgreSQL


```php
try {
  $con = new PDO("mysql:dbname=barcjkkl_barcelonatravelhacks10; host=localhost", "root", "");
  $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
}
catch (PDOException $e) {
  echo "connection failed: " . $e->getMessage();
}

try {
  $host = "localhost";
  $dbname = "barcjkkl_barcelonatravelhacks10";
  $dbuser = "postgres";
  $userpass = "root";

  $dsn = "pgsql:host=$host;port=5432;dbname=$dbname;user=$dbuser;password=$userpass";

  $conPgSQL = new PDO($dsn);

  $conPgSQL->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

}
catch (PDOException $e) {
  echo "connection failed: " . $e->getMessage();
}
```

This try/catch block handles the PostgreSQL database connection. Note that provisionally, I am using postgres superadmin credentials but later I will define users and security.

Author: Elliott Ward Farmer

4.2 Modifying php.ini file


```
;extension=ftp
extension=fileinfo
extension=gd2
extension=gettext
;extension=gmp
;extension=intl
;extension=imap
;extension=ldap
extension=mbstring
extension=exif      ; Must be after mbstring as it depends on it
extension=mysqli
;extension=oci8_12c  ; Use with Oracle Database 12c Instant Client
;extension=odbc
;extension=openssl
;extension=pdo_firebird
extension=pdo_mysql
;extension=pdo_oci
;extension=pdo_odbc
extension=pdo_pgsql
extension=pdo_sqlite
extension=pgsql
;extension=shmop
```

The php.ini file was modified to un-comment the PDO extensions for PostgreSQL.
https://stackoverflow.com/questions/17251859/using-postgresql-with-php-under-windows-xampp

Author: Elliott Ward Farmer

4.3 Testing PostgreSQL DB connection


```php
//  Configure DB Parameters
$host = "localhost";
$dbname = "masterdb";
$dbuser = "username";
$userpass = "password";

$dsn = "pgsql:host=$host;port=5432;dbname=$dbname;user=$dbuser;password=$userpass";

try{
 // create a PostgreSQL database connection
 $conn = new PDO($dsn);

 // display a message if connected to the PostgreSQL successfully
 if($conn){
 echo "Connected to the $dbname database successfully!";
 echo "\n";
 }
}catch (PDOException $e){
 // report error message
 echo $e->getMessage();
}
```

This test script is to verify the PDO database connection.
https://stackoverflow.com/questions/50237023/php-pdo-postgresql-connection-error-could-not-find-driver

Author: Elliott Ward Farmer

4.4 MVC MySQL Models

Below is an example of a MySQL query model using bound perameters

```
    public function getHeadMetaTags($pageId) : array {
        $sql = "SELECT allpages_metadata.pageId,
                       allpages_metadata.noIndexTag,
                       allpages_metadata.pageMetaTitle,
                       allpages_metadata.pageMetaDescription,
                       allpages_metadata.pageMetaSocialMediaImage,
                       allpages_metadata.pageMetaSearchKeywords,
                       allpages_metadata.pageThumbWebp,
                       allpages_scripts.googleAds,
                       allpages_scripts.getYourGuide,
                       allpages_scripts.stay22,
                       allpages_scripts.AppCardGridCompCss,
                       allpages_scripts.AppVideoGridCSS
                FROM allpages_metadata
                JOIN allpages_scripts ON allpages_scripts.pageId = allpages_metadata.pageId
                WHERE allpages_metadata.pageId = :pageId
                LIMIT 1";
        $stmt = $this->con->prepare($sql);
        $stmt->bindParam(":pageId", $pageId);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
```

BarcelonaTravelHacks.com has hundreds of SQL queries in many models files. Before jumping in and rewriting all the model queries to PostgreSQL format and taking into consideration the new table names and column names lets run a small test to see if I can get data out of the postgeSQL database via a model query.

Author: Elliott Ward Farmer

4.5 MVC PostGreSQL Models

Below is an example of a PostgreSQL query model using bound perameters

```
    public function pgSQL_getHeadMetaTags($pageId) : array {
        $sql = "SELECT
                    md.page_id,
                    md.no_index_tag,
                    md.page_meta_title,
                    md.page_meta_description,
                    md.page_sm_image,
                    md.page_thumbwebp,
                    sc.google_ads,
                    sc.get_your_guide,
                    sc.stay22,
                    sc.app_card_grid_comp_css,
                    sc.app_video_grid_css
                FROM t_allpages_metadata md
                JOIN t_allpages_scripts sc ON sc.page_id = md.page_id
                WHERE md.page_id = :page_id
                LIMIT 1";
        $stmt = $this->conPgSQL->prepare($sql);
        $stmt->bindParam(":page_id", $pageId);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
```

Note that I am also going to take this opportunity to tidy up all the SQL models using better notation such as table aliasing on join queries.

Author: Elliott Ward Farmer

4.6 MVC Views with PostGreSQL

Below is a code snippet from one of the many MVC view files. Note how I have modified the field names to reflect the new PostgreSQL column names. I could alias the columns in the models to keep the same name but I think for the sake of clean code, I will also rewrite all the view files to reflect the new PostgreSQL column names. When I pass them into a variable I will continue using camel case notation rather than underscores. This will serve as a visual reminder of what is a database column and what has been pushed to a variable making the code even more readable.

```
// Social Media Markup
        $index = $pageMetaData['no_index_tag'];
        $metaTitle = $pageMetaData['page_meta_title'];
        $metaDesc = $pageMetaData['page_meta_description'];
        $metaImage = BT_BASE_URL."/".$pageMetaData['page_sm_image'];
        $canonical = (empty($_SERVER['HTTPS']) ? 'http' : 'https') .
"://$_SERVER[HTTP_HOST]$_SERVER[REQUEST_URI]";
```

Author: Elliott Ward Farmer

5. Using Views Rather Than Tables

Author: Elliott Ward Farmer

5.1 Tables vs Views

The previous test worked and now the View and model is drawing data from PostgreSQL rather than MySQL for a small section of the web app.

However, should I take the data from a Table, Standard view or Materialized view?

In short, I have decided to use **Materialized Views** to draw all the data from for web page render. A materialized view is a pre-prepared table that already contains all the data. It is a photo or snapshot of the data at the time the view was created. Materialized views do not automatically refresh and can be considered to have **STALE data**. I will have to refresh the data when I change the information in the underlying data tables using manual refreshing or some kind of cron scheduler script.

This gives me a security benefit in that materialized views are read only and I can create a web app user credential to only allow access to the MVs but not the underlying data tables.

Materialized views can contain data from multiple tables so I can create MVs for each element of the webpage.

Author: Elliott Ward Farmer

5.2 MVs with Dynamic Data

In the web app I have a page views value which changes (counts up) each time the page is viewed. In this model I will use a standard view. **A Standard View** is a pre prepared query that queries the underlying data table to get the **FRESHEST data** which is useful because the page views counter is a value that can change by the minute. For this standard view I will configure it as read only for the web app user credentials to provide additional data security.

NO!!!!! The page views is displayed as K values so unless 1000 people view a page in one day the value will not noticeably change so use an MV and refresh it once or twice daily with a cron scheduler.

5.3 Pro's and Con's of Views

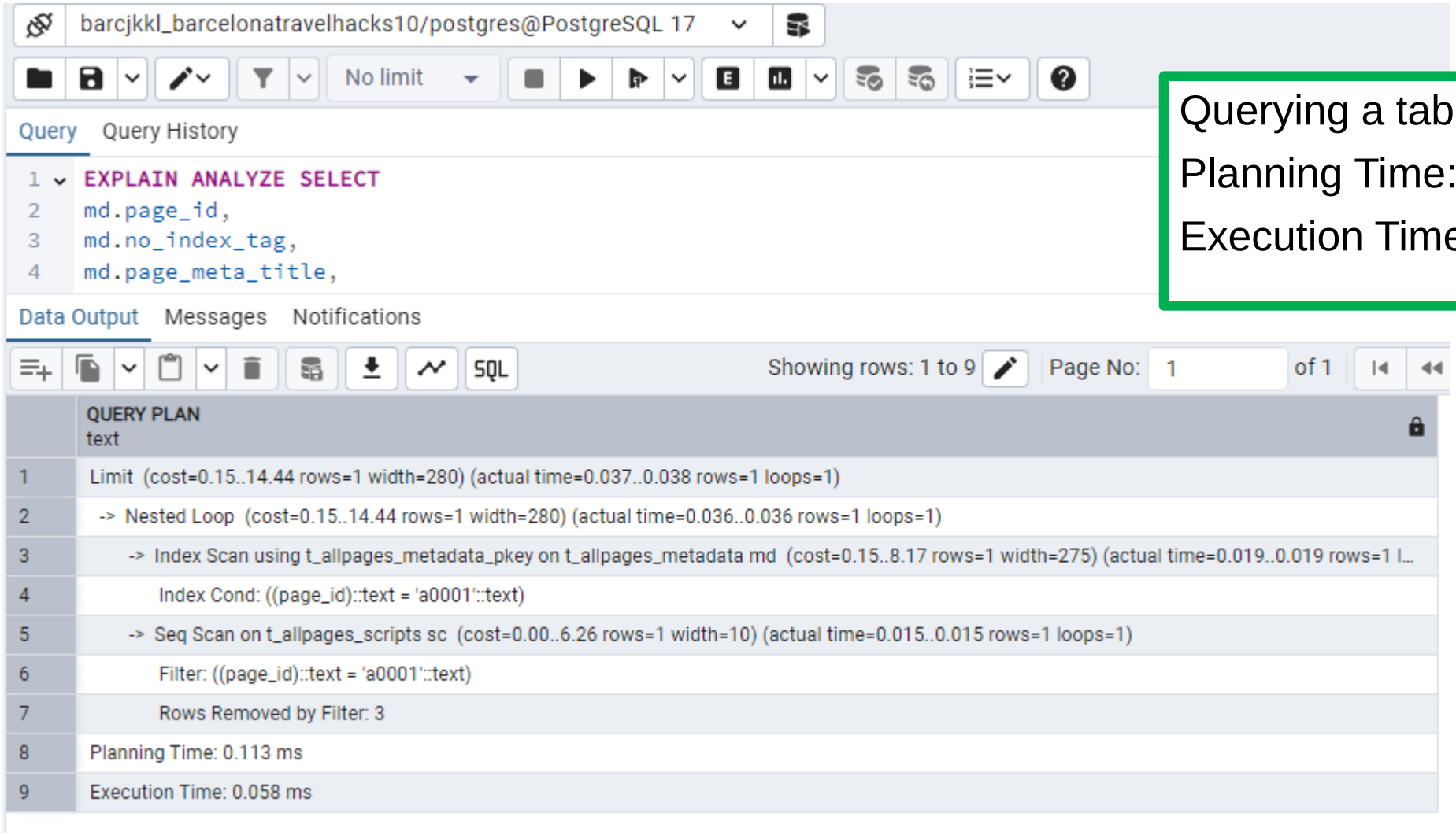
1. CON: If I am not diligent about the refreshing of Materialized Views then I may be serving STALE non updated data.
2. PRO: MVs are faster than a table query because the data is already ready to be served rather than a query that may be using a complex costly query to look in all the joined tables for the relevant data.
3. PRO: added security. The web app can only query the MVs where as the database administrator will adjust the underlying table data.
4. CON: MVs will bloat the database size because the data is stored in two places, the underlying data tables and then the MV's.

In short the choice of views is about query speed versus data freshness. In the case of my web app I need speed rather than freshness. I can always refresh views when I manually update the content.

Author: Elliott Ward Farmer

5.4 Query Analysis - Table

BarcelonaTravelHacks.com



	QUERY PLAN text
1	Limit (cost=0.15..14.44 rows=1 width=280) (actual time=0.037..0.038 rows=1 loops=1)
2	-> Nested Loop (cost=0.15..14.44 rows=1 width=280) (actual time=0.036..0.036 rows=1 loops=1)
3	-> Index Scan using t_allpages_metadata_pkey on t_allpages_metadata md (cost=0.15..8.17 rows=1 width=275) (actual time=0.019..0.019 rows=1 l...
4	Index Cond: ((page_id)::text = 'a0001'::text)
5	-> Seq Scan on t_allpages_scripts sc (cost=0.00..6.26 rows=1 width=10) (actual time=0.015..0.015 rows=1 loops=1)
6	Filter: ((page_id)::text = 'a0001'::text)
7	Rows Removed by Filter: 3
8	Planning Time: 0.113 ms
9	Execution Time: 0.058 ms

Querying a table using a join query:

Planning Time: 0.113 ms

Execution Time: 0.058 ms

Author: Elliott Ward Farmer

5.5 Query Analysis – Standard View

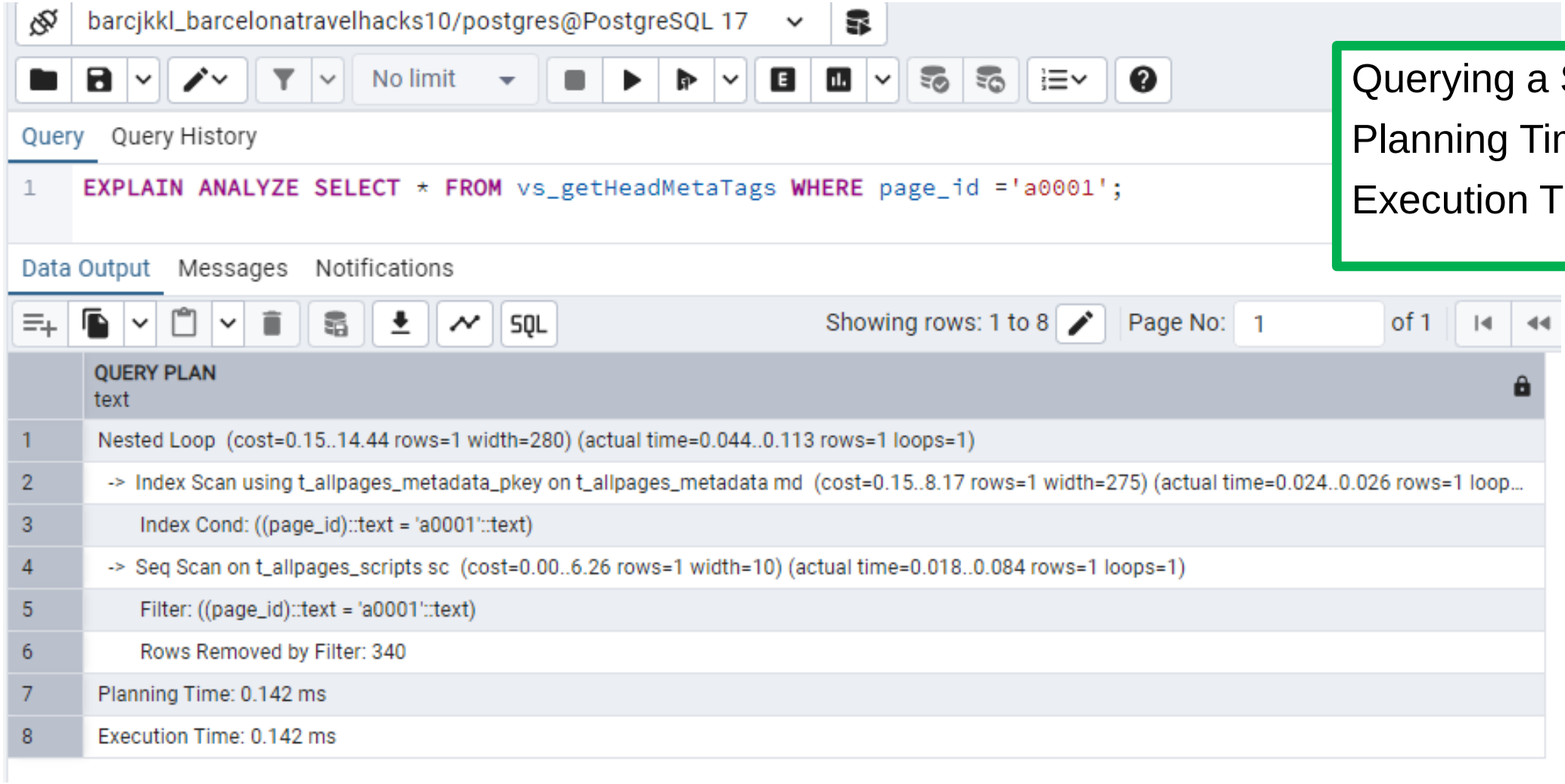

```
CREATE OR REPLACE VIEW vs_getHeadMetaTags AS
SELECT
md.page_id,
md.no_index_tag,
md.page_meta_title,
md.page_meta_description,
md.page_sm_image,
md.page_thumbwebp,
sc.google_ads,
sc.get_your_guide,
sc.stay22,
sc.app_card_grid_comp_css,
sc.app_video_grid_css
FROM t_allpages_metadata md
JOIN t_allpages_scripts sc ON sc.page_id = md.page_id;
CREATE VIEW
Query returned successfully in 63 msec.
```

I have created this test view where each row contains the columns of two tables for each page of my website.

```
SELECT * FROM vs_getHeadMetaTags WHERE page_id ='a0001';
```

I can pull a row of data out of the Standard view but is querying a Standard View faster than querying table?

Author: Elliott Ward Farmer



Querying a Standard View:

Planning Time: 0.142 ms

Execution Time: 0.142 ms

Author: Elliott Ward Farmer

5.6 Query Analysis – Materialized View

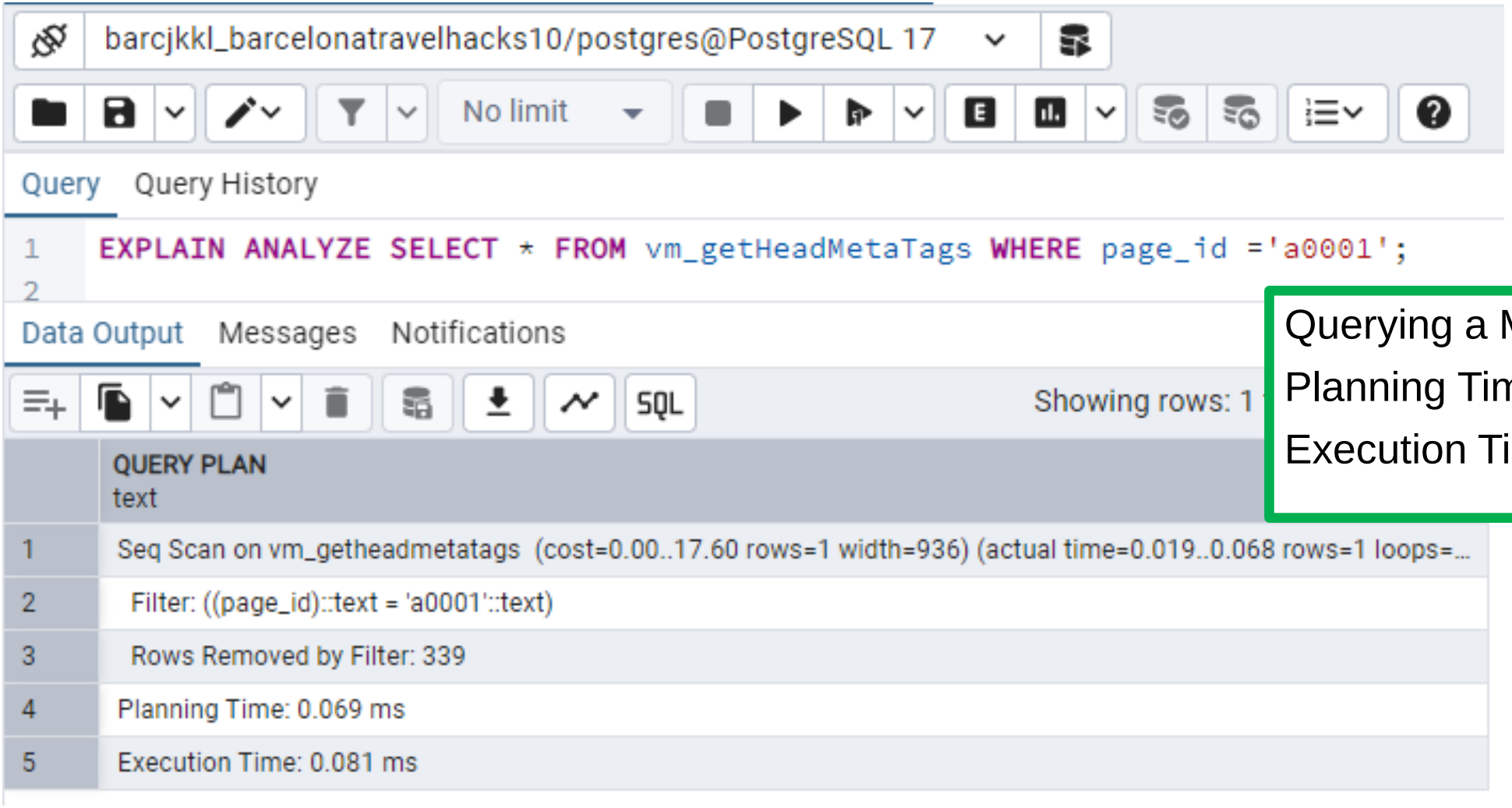

```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_getHeadMetaTags AS
SELECT
md.page_id,
md.no_index_tag,
md.page_meta_title,
md.page_meta_description,
md.page_sm_image,
md.page_thumbwebp,
sc.google_ads,
sc.get_your_guide,
sc.stay22,
sc.app_card_grid_comp_css,
sc.app_video_grid_css
FROM t_allpages_metadata md
JOIN t_allpages_scripts sc ON sc.page_id = md.page_id
WITH DATA;
CREATE VIEW
Query returned successfully in 42 msec.
```

I have created this test view where each row contains the columns of two tables for each page of my website.

```
SELECT * FROM vm_getHeadMetaTags WHERE page_id ='a0001';
```

I can pull a row of data out of the Materialized view but is querying it faster than querying table or a Standard View?

Author: Elliott Ward Farmer



Querying a Materialized View:
Planning Time: 0.069 ms
Execution Time: 0.081 ms

Author: Elliott Ward Farmer

5.7 Table vs MS vs MV query speeds

Querying a table using a join query:	Querying a Standard View:	Querying a Materialized View:
Planning Time: 0.113 ms	Planning Time: 0.142 ms	Planning Time: 0.069 ms
Execution Time: 0.058 ms	Execution Time: 0.142 ms	Execution Time: 0.081 ms

The planning time of a MV query is much faster than a MS or a Table but not by much.

```
CREATE UNIQUE INDEX idx_vm_page_id ON vm_getHeadMetaTags (page_id);
```

What happens If I put an index on the MV?

Author: Elliott Ward Farmer

5.8 MV with an Index

Querying an Indexed Materialized View:
Planning Time: 0.072 ms
Execution Time: 0.034 ms

barcjkkl_barcelonatravelhacks10/postgres@PostgreSQL 17

No limit

Query | Query History

```
1  -- CREATE UNIQUE INDEX idx_vm_getHeadMetaTags_page_id ON vm_getHeadMetaTags (page_id);
2  -- REFRESH MATERIALIZED VIEW CONCURRENTLY vm_getHeadMetaTags;
3  EXPLAIN ANALYZE SELECT * FROM vm_getHeadMetaTags WHERE page_id ='a0001';
4
```

Data Output | Messages | Notifications

Showing rows: 1 to 4 | Page No: 1 of 1

	QUERY PLAN text
1	Index Scan using idx_vm_getheadmetatags_page_id on vm_getheadmetatags (cost=0.15..8.17 rows=1 width=280) (actual time=0.020..0.021 rows=1 loops=...
2	Index Cond: ((page_id)::text = 'a0001'::text)
3	Planning Time: 0.072 ms
4	Execution Time: 0.034 ms

Author: Elliott Ward Farmer

5.7 Using MV in MVC script

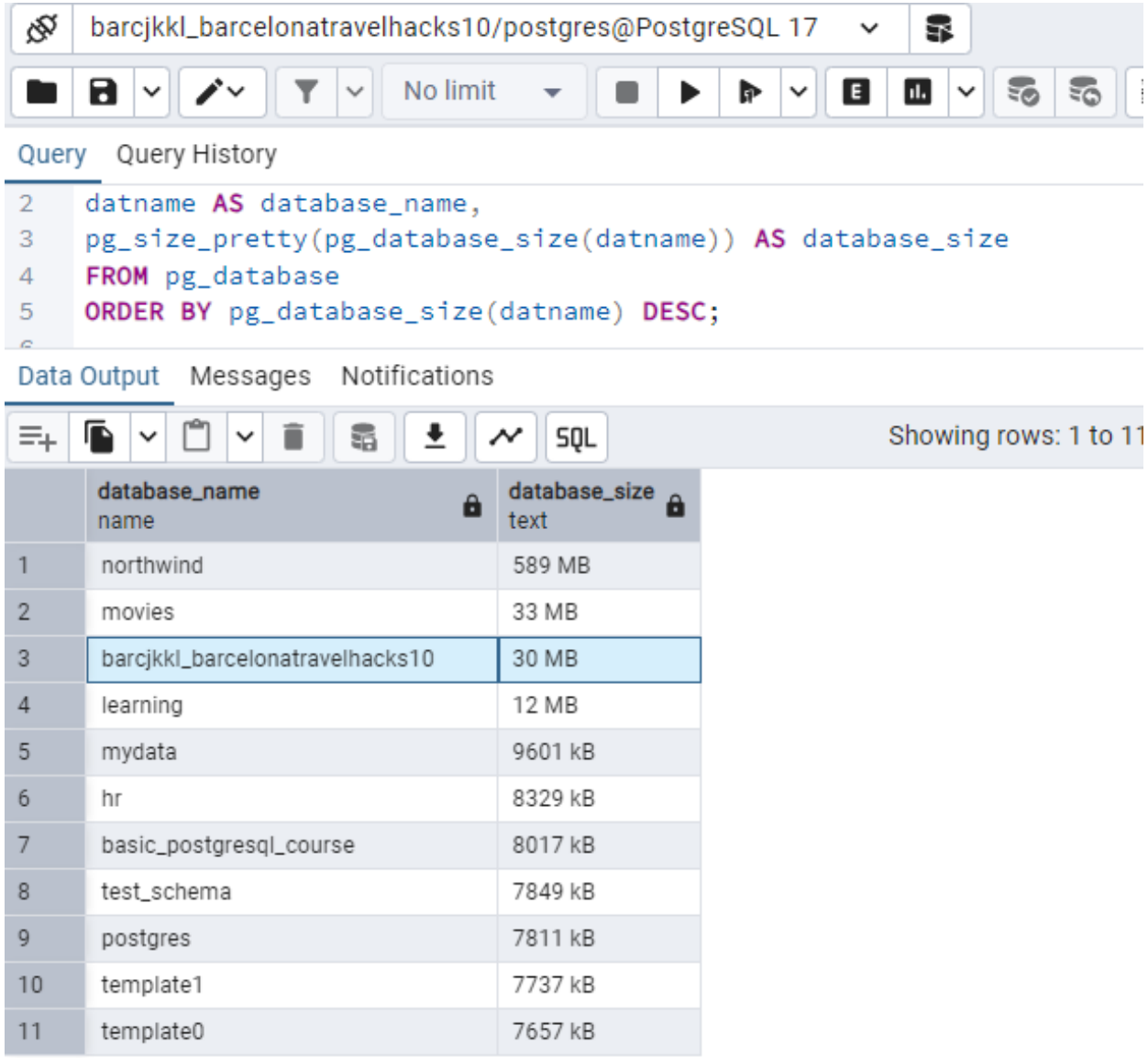

```
// Mar-2025 - pgSQL: MV Query //
public function pgSQL_MV_getHeadMetaTags($pageId) : array {
    $sql = "SELECT *
            FROM vm_getHeadMetaTags
            WHERE page_id = :page_id
            LIMIT 1";
    $stmt = $this->conPgSQL->prepare($sql);
    $stmt->bindParam(":page_id", $pageId);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_ASSOC);
}
```

I have modified the model to use the Materialized view rather than the table query with the join.

The webpage loads with the correct data.

Author: Elliott Ward Farmer

5.9 Conclusions of Views



	database_name name	database_size text
1	northwind	589 MB
2	movies	33 MB
3	barcjkkl_barcelonatravelhacks10	30 MB
4	learning	12 MB
5	mydata	9601 kB
6	hr	8329 kB
7	basic_postgresql_course	8017 kB
8	test_schema	7849 kB
9	postgres	7811 kB
10	template1	7737 kB
11	template0	7657 kB

BarcelonaTravelHacks.com

Using an indexed Materialized view for data retrieval is faster than querying the base tables directly but increases overall database size and the materialized views have to be managed to refresh the data inside to prevent serving STALE DATA

Currently the database is 30MB with the data tables and one Materialized View. I will continue to experiment crating more MVs to see how much it bloats the database size.

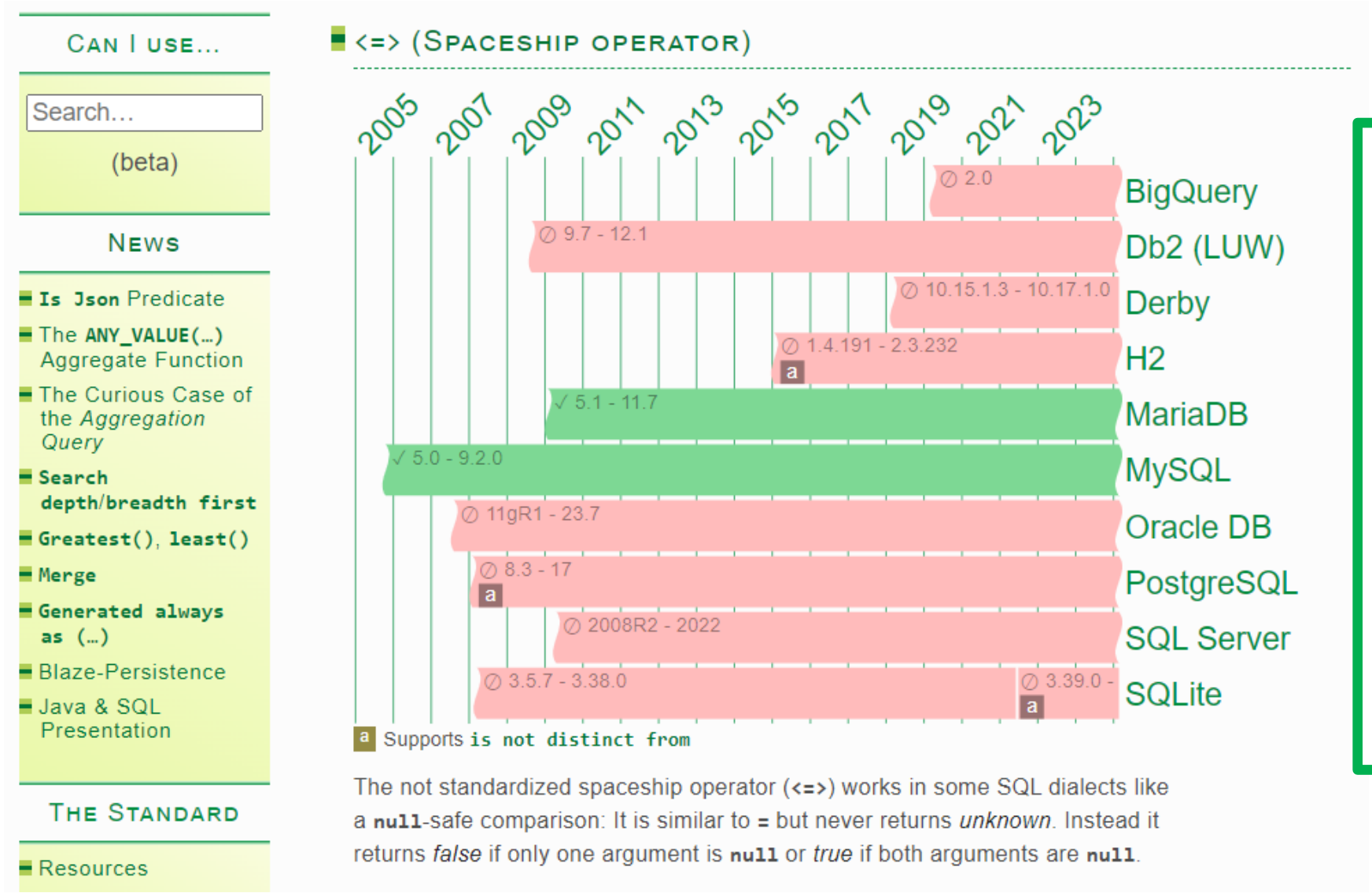
Author: Elliott Ward Farmer

5.10 The importance of an Index

1. An index speeds up querying from tables, Standard Views and Materialized views.
2. The index needs to be placed on the **column(s)** that I am filtering on, i.e. in the where clause find all rows where **column** is equal to value.
3. If I have a materialized view which will return multiple rows, i.e. page gallery which has one row for each image, then the row I am querying on is NOT UNIQUE. The PK is the image ID, and the column I am using in the query is page_id. i.e. fetch all images for page x. Therefore, I will use a unique index on the image_id then page_id. By combining the two columns I now have uniqueness in the index on the where (filter) clause column.

Author: Elliott Ward Farmer

5.11 No Spaceship Operator in PostgreSQL



PostgreSQL does not support the spaceship operator and I am specifically using this in the URL handler script (route file).

I have resolved this by returning the URI as a complete string rather that in parts, thus negating the need for the spaceship operator.

Author: Elliott Ward Farmer

6. Building the MVs

Author: Elliott Ward Farmer

6.1 Static and API loaded website Content

All pages on my website are build in a modular fashion using common coding blocks and table data.

For example the text is rendered on initial page load because it is relatively light. Images and graphical content is loaded dynamically via API calls to my database as the page is scrolled (below the fold loading) for purposes of page speed, which is a Google Ranking factor, and also makes my website faster on mobile devices and low end PCs.

With this in mind, I want an MV that returns a row for each page with all the static text to build a page. At the moment I am using multiple MySQL queries that get the data for each section. This is slowing down my website because I have multiple SQL queries firing just to get the static text content when I can just fire one SQL query to the PostgreSQL database MV and get all the text in one hit.

As for the API loaded data, this is also a messy solution in MySQL. I am currently using a script to query the SQL table (with complex joins) I am then de-structuring the row(s) return and rebuilding it into the appropriate JSON format such as geoJSON. Lets exploit the power of PostgreSQL to build API MVs that store the data already in the correct JSON format and return pure JSON eliminating this time consuming intermediate scripting.

Author: Elliott Ward Farmer

6.2 Page Meta Data MV

```
-- 1. VIEW: vm_getHeadMetaTags
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_get_head_meta_tags AS
SELECT
md.page_id,
md.no_index_tag,
md.page_meta_title,
md.page_meta_description,
md.page_sm_image,
md.page_thumbwebp,
md.page_views,
md.page_author,
md.page_added,
md.page_updated,
sc.google_ads,
sc.get_your_guide,
sc.stay22,
sc.app_card_grid_comp_css,
sc.app_video_grid_css
FROM t_allpages_metadata md
JOIN t_allpages_scripts sc ON sc.page_id = md.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_get_head_meta_tags_page_id ON
vm_get_head_meta_tags (page_id);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_get_head_meta_tags;
-- SEELCT FROM MV
SELECT * FROM vm_get_head_meta_tags;
-- DROP MATERIALIZED VIEW vm_getheadmetatags
```

BarcelonaTravelHacks.com

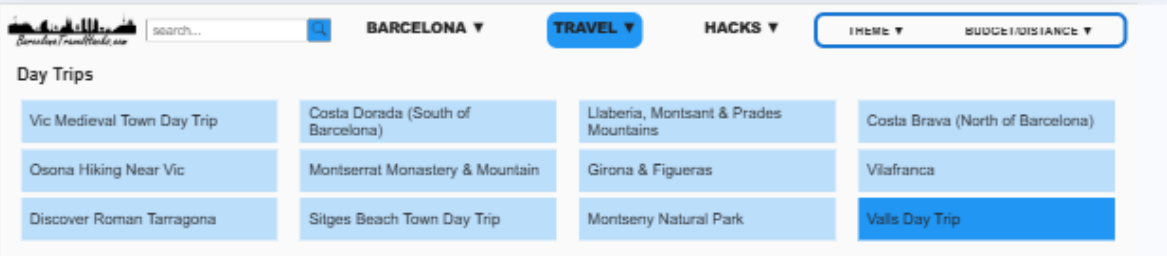
This very simple MV builds a view of all the data I need to render the page metadata that is inside the HTML <head></head> tags.

It contains a mix of VARCHAR and BOOL values. The Bool values determine what JS scripts and CSS stylesheets are loaded for the page content.

It is basically the same as the MySQL query so no complex stuff here.

Author: Elliott Ward Farmer

6.3 Site Nav Header



This is the site nav header menu and drop down menu content which is derived from the t_nav_menu and url_handler table.

```
-- 2. VIEW: vm_getSiteNavMenu
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_get_site_nav_menu AS
SELECT
nm.nav_id,
nm.page_id,
nm.no_index_tag,
nm.nav_menu,
nm.nav_sub_menu,
nm.btn_text,
uhn.urlparam0,
uhn.urlparam1,
uhn.urlparam2
FROM t_nav_menu nm
JOIN t_allpages_url_handler_new uhn ON nm.page_id = uhn.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_get_site_nav_menu_page_id ON
vm_get_site_nav_menu (nav_id,nav_menu,nav_sub_menu);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_get_site_nav_menu;
-- SEELCT FROM MV
SELECT * FROM vm_get_site_nav_menu;
```

This is a perfectly adequate MV for this task but it can be improved. I am using up to three parameters in my URI so I now need to construct the URI from the urlparam0,urlparam1 and urlparam2 fields.

The point of using MVs is to return data in a format that the web app needs it so this MV can be polished.

Author: Elliott Ward Farmer

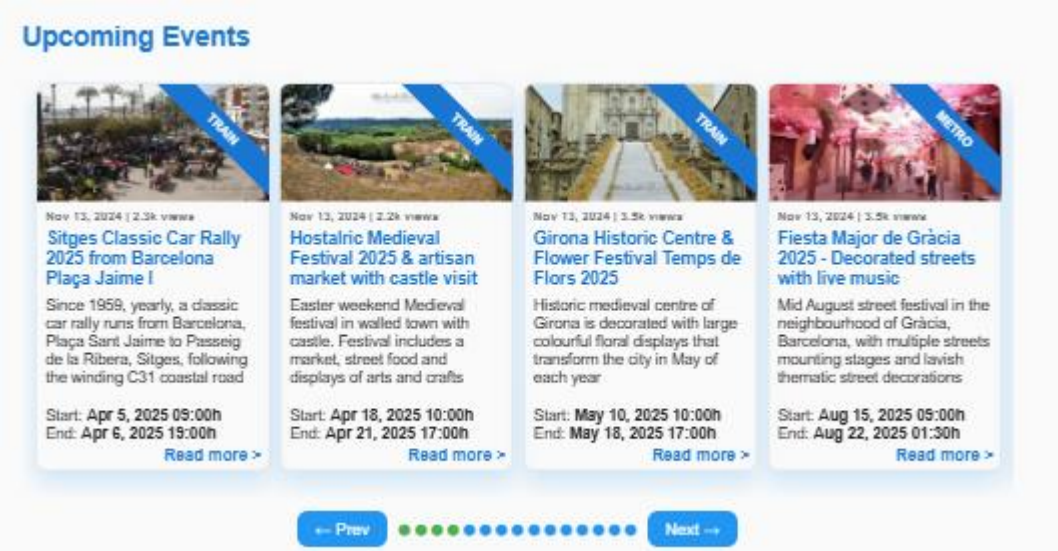
6.4 Site Nav Header Improved MV


```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_2_get_site_nav_menu AS
SELECT
nm.nav_id,
nm.page_id,
nm.no_index_tag,
nm.nav_menu,
nm.nav_sub_menu,
nm.btn_text,
rtrim(coalesce(uhn.urlparam0,'') || '/' || coalesce(uhn.urlparam1,'') || '/' || coalesce(uhn.urlparam2,''),'/') as uri
FROM t_nav_menu nm
JOIN t_allpages_url_handler_new uhn ON nm.page_id = uhn.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_get_site_nav_menu_page_id ON vm_2_get_site_nav_menu (nav_id,nav_menu,nav_sub_menu);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_2_get_site_nav_menu;
-- SEELCT FROM MV
SELECT * FROM vm_2_get_site_nav_menu;
```

I have combined the three URI components into one column which I am aliasing as URI. Note how I am using coalesce to handle NULL values, then TRIM to remove trailing slashes if present. i.e. a URI of 'barcelona/las-ramblas/NULL' becomes URI of 'barcelona/las-ramblas' This also strips out code from my page render functions because now all I have to do is tack on the BASE_URL (domain_name) to the front of the URI rather than rebuilding the URI each time from three database parameters. Faster code.

Author: Elliott Ward Farmer

6.5 Card Components



The whole website runs off of card components. For example attractions and events cards. All the cards have the same basic data of a thumb img, title, updaded_date, total_views, description and URL. Event cards have start and end dates. The cards pull data from multiple tables such as metadata and URL_handler.

Additionally, cards can be filtered by parameters:

location. A card usually has one location except for a few such as cable cars (teleferico del Puerto) that is in Montjuic and Barceloneta.

Theme: museum, historic house, parks and gardens etc. Currently up to 25 themes. Typically, a card will have anywhere from 1 to 5 themes.

Distance: Barcelona Metropolitan, up to 1 hour from Barcelona by train/bus, 1-2 hours by car, 2-3 hours etc. Currently there are 7 distance options with each card typically having one or two.

Budget: Free, 0-10 euros, 10-25 euros etc. 7 budget options in total.

Author: Elliott Ward Farmer

6.6 Card Components MV


```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_3_get_evt_cards AS
SELECT
md.page_id,
md.page_type,
md.page_thumbwebp,
md.transport_type,
md.page_meta_title,
md.page_meta_description,
md.page_added,
md.page_updated,
md.page_views,
cbe.evt_start,
cbe.evt_end,
rtrim(coalesce(uhn.urlparam0,'') || '/' || coalesce(uhn.urlparam1,'') || '/' ||
coalesce(uhn.urlparam2,''),'/') as uri,
array_agg(DISTINCT COALESCE(nl.menu_page_id)) as location,
array_agg(DISTINCT COALESCE(nt.menu_page_id)) as theme,
array_agg(DISTINCT COALESCE(nd.menu_page_id)) as distance,
array_agg(DISTINCT COALESCE(nb.menu_page_id)) as budget
FROM t_allpages_metadata md
JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
JOIN t_content_body_evt_text cbe ON cbe.page_id = md.page_id
JOIN t_nav_loc nl ON nl.page_id = md.page_id
JOIN t_nav_tme nt ON nt.page_id = md.page_id
JOIN t_nav_dist nd ON nd.page_id = md.page_id
JOIN t_nav_bgt nb ON nb.page_id = md.page_id
WHERE md.page_type = 'e'
AND md.page_status = true
GROUP BY md.page_id, uhn.urlparam0,uhn.urlparam1, uhn.urlparam2, cbe.evt_start,
cbe.evt_end
ORDER BY md.page_id
WITH DATA;
```

```
-- CREATE INDEX
CREATE UNIQUE INDEX
idx_vm_3_get_evt_cards_page_id_evt_start_location_theme_distance_budget ON
vm_3_get_evt_cards (page_id,evt_start,location,theme,distance,budget);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_3_get_evt_cards;
-- SEELCT FROM MV
SELECT * FROM vm_3_get_evt_cards;
```

Now the Event cards MV contains all the columns I need to render html plus all the columns I may want to search or filter the cards on. Note that I am putting an index on all the fields that I may want to include in a where clause of a query.

Author: Elliott Ward Farmer

6.6 Card Components MV table

BarcelonaTravelHacks.com

uri text	location character varying[]	theme character varying[]	distance character varying[]	budget character varying[]
barcelona/event/sagrada-familia-christmas-market	{v0001}	{w0011}	{x0001}	{y0001}
barcelona/event/three-kings-float-parade	{b0006,b0007}	{w0014}	{x0001}	{y0001}
barcelona/event/new-years-eve-plaza-espana-fireworks	{b0007}	{w0014}	{x0001}	{y0001}
barcelona/event/fira-santa-llucia-christmas-market	{b0002}	{w0011}	{x0001}	{y0001}
barcelona/event/fiesta-llum-bcn	{NULL}	{NULL}	{x0001}	{y0001}
barcelona/event/christmas-lights-tour	{b0002,v0001}	{NULL}	{x0001}	{y0001}
barcelona/event/fiesta-major-gracia	{v0008}	{w0014}	{x0001}	{y0001}
barcelona/event/gotheborg-sailing-ship-visits-barcelona	{b0006}	{v0017,w0002}	{x0001}	{y0004}
barcelona/event/port-vell-christmas-fair	{b0006}	{w0011,w0014}	{x0001}	{y0002}
travel/event/olesa-de-montserrat-fiesta-de-los-miquelets	{t0003}	{w0013}	{x0003}	{y0002}
travel/event/barcelona-sitges-classic-car-rally	{b0002,t0004}	{v0017}	{x0002}	{y0002}
travel/event/vic-medieval-market-fair	{t0001}	{w0011,w0012,w0014}	{x0002,x0006}	{y0004}

The values I want to search on are stored as arrays {x,y,z} in the columns.

Author: Elliott Ward Farmer

6.7 Simple Model Queries to Filter Cards


```
// Mar-2025 - pgSQL: MV Query //
  public function pgSQL_get_upcomming_events($limit) : array {
    $currentDate = date('Y-m-d 00:00:00', time());
    $sql = "SELECT
              page_id,
              page_thumbwebp,
              transport_type,
              page_meta_title,
              page_meta_description,
              page_updated,
              page_views,
              evt_start,
              evt_end,
              uri
      FROM vm_3_get_evt_cards
      WHERE evt_start >= '$currentDate'
      ORDER BY evt_start ASC
      LIMIT $limit";
    $stmt = $this->conPgSQL->prepare($sql);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  }
```

Get all event cards from the MV table where the event start date is after current date and time.

Get all event cards from the MV table where the location is: Note that if I pass in 'v0001' as a $pageId then I would get all events in 'la-Eixampla' location.

```
// Mar-2025 - pgSQL: MV Query //
  public function pgSQL_get_evt_by_location($PageId) : array {
    $sql = "SELECT
              page_thumbwebp,
              transport_type,
              page_meta_title,
              page_meta_description,
              page_updated,
              page_views,
              evt_start,
              evt_end,
              urlparam0,
              urlparam1,
              urlparam2
            FROM vm_3_get_evt_cards
            WHERE :page_id = ANY(location)";
    $stmt = $this->conPgSQL->prepare($sql);
    $stmt->execute(array(':page_id' => $PageId));
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  }
```

Author: Elliott Ward Farmer

6.8 Pros and Cons of WHERE clause on Array

Query | Query History

```
1  EXPLAIN ANALYZE SELECT
2  page_thumbwebp,
3  transport_type,
4  page_meta_title,
5  page_meta_description,
6  page_updated,
7  page_views,
8  evt_start,
9  evt_end,
10 uri
11 FROM vm_3_get_evt_cards
12 WHERE 'v0001' = ANY(location)
```

Data Output | Messages | Notifications

Showing rc

	QUERY PLAN text
1	Seq Scan on vm_3_get_evt_cards (cost=0.00..2.36 rows=1 width=866) (actual time=0.025..0.032 rows=2 loops=...
2	Filter: ('v0001'::text = ANY ((location)::text[]))
3	Rows Removed by Filter: 14
4	Planning Time: 1.408 ms
5	Execution Time: 0.054 ms

Because the arrays have one or two items in them then the queries are running quick. I can see that it is doing a sequential scan and filtering by column location to match on an element in the array thanks to the previously placed MV indexes.

If I had tens or hundreds of elements in an array then this would be slower and probably not viable, speed wise, for a web app. I would use a different solution.

Author: Elliott Ward Farmer

6.9 fixing unrelated data


```
- 2. Function to insert rows into t_nav_bgt with values based on conditionals
CREATE OR REPLACE FUNCTION fn_insert_t_nav_bgt() RETURNS VOID AS
$$
DECLARE
  rec record;
BEGIN
  FOR rec IN
   SELECT
    md.page_id,
    fkp.adult_total,
    fkp.free_days
  FROM t_allpages_metadata md
  JOIN t_content_fk_prices fkp ON fkp.page_id = md.page_id
  WHERE md.page_type IN('a','e')
    LOOP
      IF rec.adult_total <= 1.14 THEN
        INSERT INTO t_nav_bgt (id,page_id,menu_page_id) VALUES
        (concat(rec.page_id,'-401'),rec.page_id,'y0001');

...
    IF rec.free_days IS NOT NULL THEN
        INSERT INTO t_nav_bgt (id,page_id,menu_page_id) VALUES
        (concat(rec.page_id,'-402'),rec.page_id,'y0003');
      END IF;

    END LOOP;
END;
$$
LANGUAGE plpgsql;
```

When I create the event card MV, I am pulling data to build the budget array from another table called t_nav_bgt

```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_3_get_evt_cards AS
SELECT
md.page_id,
...
array_agg(DISTINCT COALESCE(nb.menu_page_id)) as budget
...
WITH DATA;
```

The t_nav_bgt table is created with a function that calculates the columns based on another table called t_content_fk_prices.

In short, I have a total field in t_content_fk_prices that I am using auto-generation stored to calculate the value based on other column values in that table.

Author: Elliott Ward Farmer

6.10 Using Case to solve 'unrelated' data

t_content_fk_page_attevt_prices
page_id VARCHAR(6) PRIMARY KEY
adult_travel_fare_txt VARCHAR(20)
adult_travel_fare_dec NUMERIC(5,2)
child_travel_fare_txt VARCHAR(20)
child_travel_fare_dec NUMERIC(5,2)
adult_ticket_text VARCHAR(20)
adult_ticket_dec DECIMAL(5,2)
adult_ticket_cond VARCHAR(60)
child_ticket_text VARCHAR(20)
child_ticket_dec NUMERIC (5,2)
child_ticket_cond VARCHAR(60)
adult_total NUMERIC(5,2) **GENERATED ALWAYS AS (adult_travel_fare_dec+adult_ticket_dec) STORED**
child_total NUMERIC(5,2) **GENERATED ALWAYS AS (child_travel_fare_dec+child_ticket_dec) STORED**
group_family_ticket VARCHAR(150)
prices_notes VARCHAR(150)
free_days VARCHAR(150)

From this table I want an array of y values based on the following logic. The array will have one or two y values in it. I can achieve this in a sub query using a case statement.

```
IF adult_total <= 1.14 THEN 'y0001'
ELSIF adult_total > 1.14 AND rec.adult_total <= 10 THEN
'y0002'
ELSIF rec.adult_total > 10.01 AND rec.adult_total <= 25 THEN
'y0004'
ELSIF rec.adult_total > 25.01 AND rec.adult_total <= 40 THEN
'y0005'
ELSIF rec.adult_total > 40.01 AND rec.adult_total <= 60 THEN
'y0006'
ELSIF rec.adult_total > 60.01 AND rec.adult_total <= 80 THEN
'y0007'
END IF
AND
IF rec.free_days IS NOT NULL THEN ADD 'y0003' TO THE ARRAY
END IF
```

Author: Elliott Ward Farmer

6.10 Using Case to solve 'unrelated' data

Casting the return of the sub query to an ARRAY datatype despite being a string return written with PostgreSQL array syntax. If I don't cast this as an array then the column is of type TEXT and I cannot match values within the array in later queries.

```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_3_get_evt_cards AS
SELECT
md.page_id,
md.page_type,
md.page_thumbwebp,
md.transport_type,
md.page_meta_title,
md.page_meta_description,
md.page_added,
md.page_updated,
md.page_views,
cbe.evt_start,
cbe.evt_end,
rtrim(coalesce(uhn.urlparam0,'') || '/' || coalesce(uhn.urlparam1,'') || '/' || coalesce(uhn.urlparam2,''),'/') as uri,
array_agg(DISTINCT COALESCE(nl.menu_page_id)) as location,
array_agg(DISTINCT COALESCE(nt.menu_page_id)) as theme,
array_agg(DISTINCT COALESCE(nd.menu_page_id)) as distance,
(
SELECT budget::TEXT[] FROM
  (
    SELECT fkp.adult_total, fkp.free_days,
    CASE
    WHEN (fkp.adult_total <= 1.14) AND (fkp.free_days IS NULL) THEN '{"y0001"}'
    WHEN (fkp.adult_total <= 1.14) AND (fkp.free_days IS NOT NULL) THEN '{"y0001","y0003"}'
    WHEN (fkp.adult_total > 1.14 AND fkp.adult_total <= 10) AND (fkp.free_days IS NULL) THEN '{"y0002"}'
    WHEN (fkp.adult_total > 1.14 AND fkp.adult_total <= 10) AND (fkp.free_days IS NOT NULL) THEN '{"y0002","y0003"}'
    WHEN (fkp.adult_total > 10.01 AND fkp.adult_total <= 25) AND (fkp.free_days IS NULL) THEN '{"y0004"}'
    WHEN (fkp.adult_total > 10.01 AND fkp.adult_total <= 25) AND (fkp.free_days IS NOT NULL) THEN '{"y0004","y0003"}'
    WHEN (fkp.adult_total > 25.01 AND fkp.adult_total <= 40) AND (fkp.free_days IS NULL) THEN '{"y0005"}'
    WHEN (fkp.adult_total > 25.01 AND fkp.adult_total <= 40) AND (fkp.free_days IS NOT NULL) THEN '{"y0005","y0003"}'
```

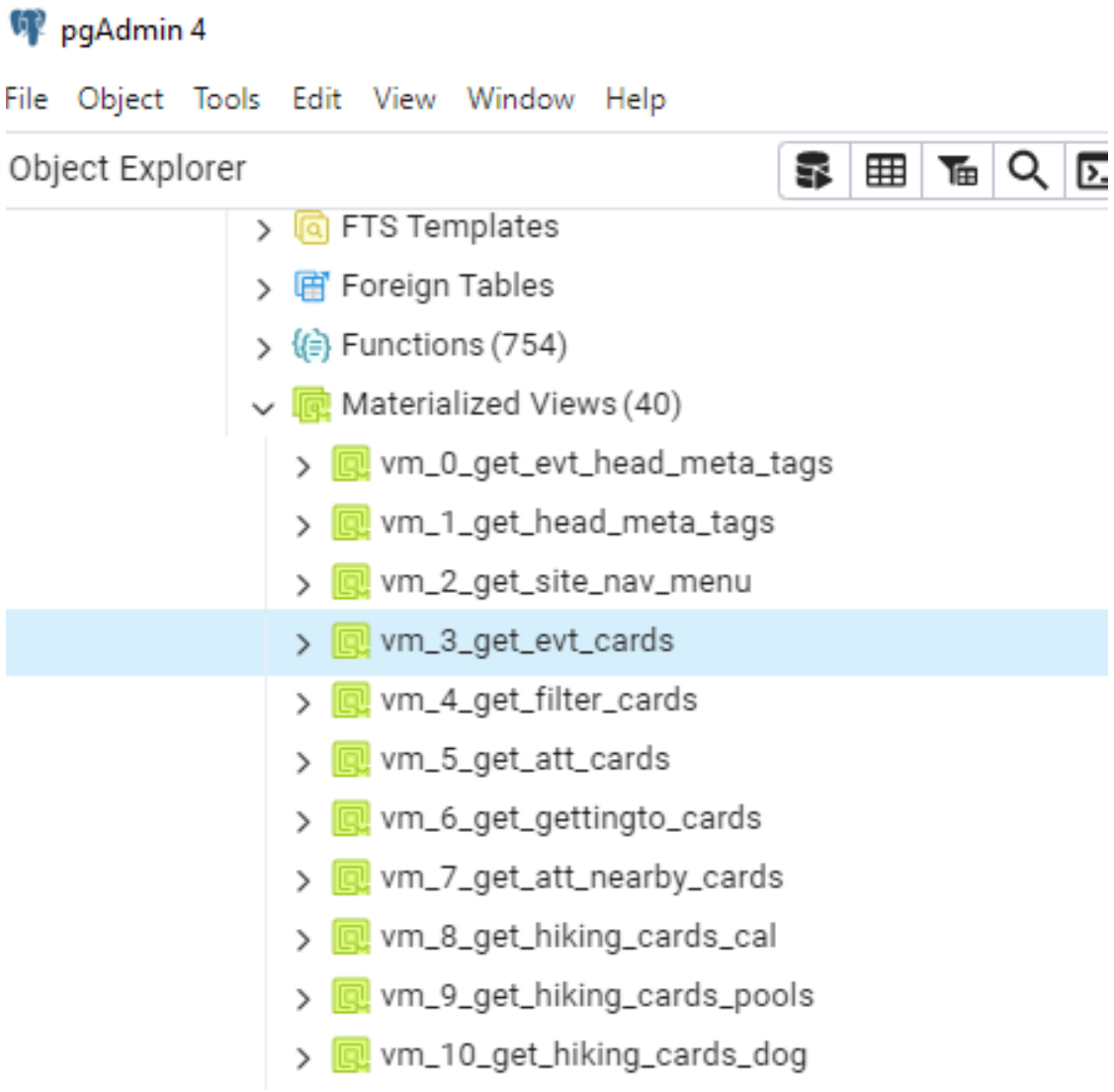
Author: Elliott Ward Farmer

6.10 Using Case to solve 'unrelated' data


```
    WHEN (fkp.adult_total > 40.01 AND fkp.adult_total <= 60) AND (fkp.free_days IS NULL) THEN '{"y0006"}'
    WHEN (fkp.adult_total > 40.01 AND fkp.adult_total <= 60) AND (fkp.free_days IS NOT NULL) THEN '{"y0006","y0003"}'
    WHEN (fkp.adult_total > 60.01 AND fkp.adult_total <= 80) AND (fkp.free_days IS NULL) THEN '{"y0007"}'
    WHEN (fkp.adult_total > 60.01 AND fkp.adult_total <= 80) AND (fkp.free_days IS NOT NULL) THEN '{"y0007","y0003"}'
    END AS budget
    FROM t_content_fk_prices fkp
    WHERE fkp.page_id = md.page_id
  ) as t
)
FROM t_allpages_metadata md
JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
JOIN t_content_body_evt_text cbe ON cbe.page_id = md.page_id
JOIN t_nav_loc nl ON nl.page_id = md.page_id
JOIN t_nav_tme nt ON nt.page_id = md.page_id
JOIN t_nav_dist nd ON nd.page_id = md.page_id
JOIN t_content_fk_prices fkp ON fkp.page_id = md.page_id
WHERE md.page_type = 'e'
AND md.page_status = true
GROUP BY md.page_id, uhn.urlparam0,uhn.urlparam1, uhn.urlparam2, cbe.evt_start, cbe.evt_end,adult_total
ORDER BY md.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_3_get_evt_cards_page_id_evt_start_location_theme_distance_budget ON vm_3_get_evt_cards
(page_id,evt_start,location,theme,distance,budget);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_3_get_evt_cards;
-- SEELCT FROM MV
SELECT * FROM vm_3_get_evt_cards;
```

Author: Elliott Ward Farmer

6.11 Building the rest of the Card MVs



I am using many advanced SQL and PostgreSQL features to build all the MVs for the cards. I tried to bounce these MVs down into less MVs but because I am filtering on a wide range of data, I found that I needed all the MVs.

These MVs are 'costly' to run at creation because they are very complex sql queries but this is not important.

The goal is to have each MV containing the rows I need to build the cards then a few additional columns that I can filter on so that the queries to get the cards are a lot faster than running complex joins with where clauses on multiple columns.

Note that the MVs are fast to refresh and done concurrently to avoid blocking the MV data on refresh.

Author: Elliott Ward Farmer

6.12 Static Text for Each Event Page


```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_11_get_evt_text AS
SELECT
at.page_id,
md.page_type,
at.page_title,
at.page_street_name,
at.page_city,
at.page_postcode,
at.page_about_body,
at.page_visiting_body,
at.page_gettingto_body,
at.page_eqmt_comments,
at.img_gal_rows,
(
  SELECT
    COUNT(id)
  FROM t_content_fk_page_gal fkg
  WHERE at.page_id = fkg.page_id
  GROUP BY fkg.page_id
) AS img_gal_count,
(
  SELECT
    COUNT(id)
  FROM t_content_fk_pdf_docs fkp
  WHERE at.page_id = fkp.page_id
  GROUP BY fkp.page_id
) AS pdf_doc_count,
(
  SELECT
    COUNT(id)
  FROM t_content_fk_travel_eqmt fkt
  WHERE at.page_id = fkt.page_id
  GROUP BY fkt.page_id
) AS eqmt_count,
```

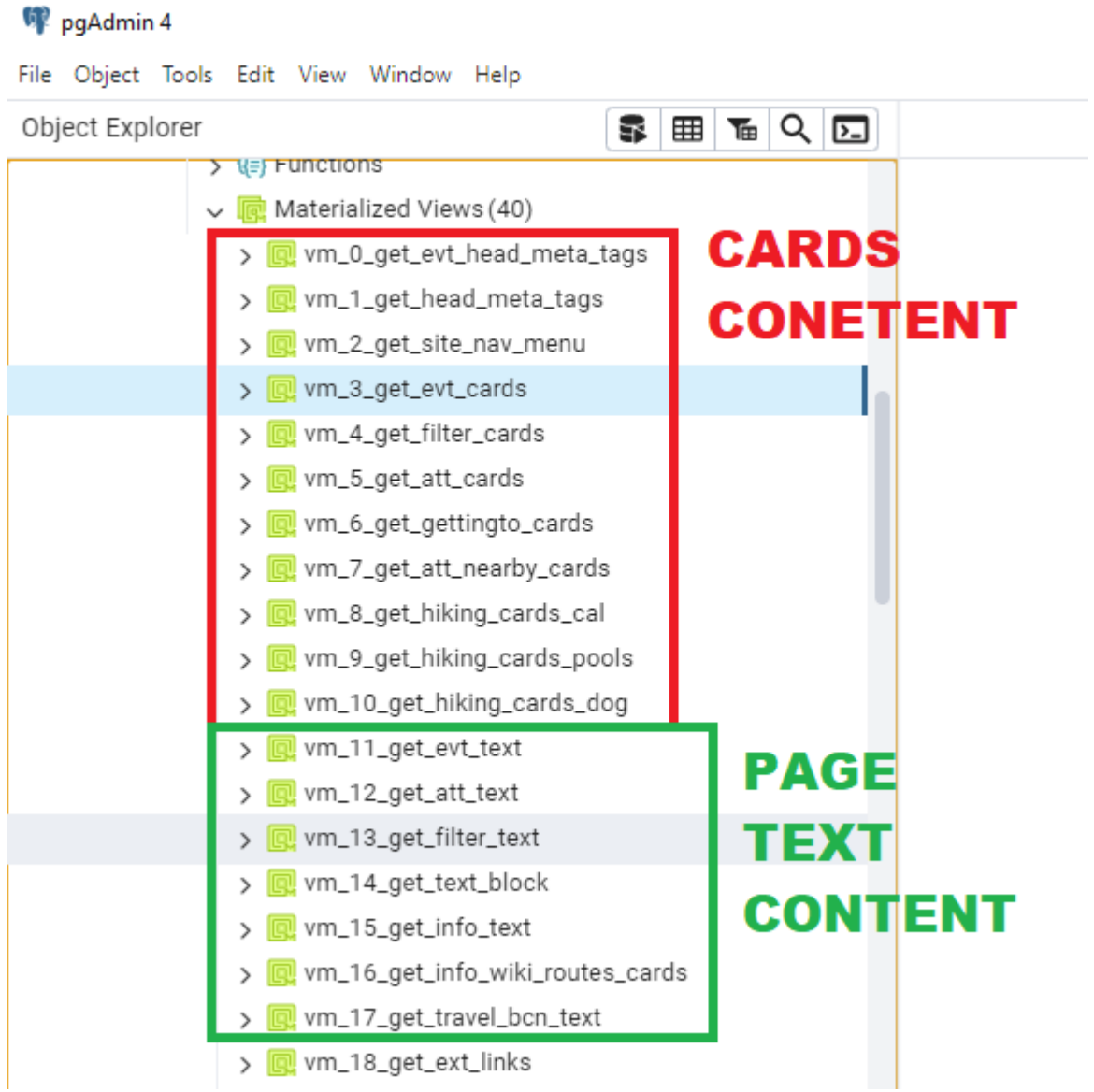
Using Sub queries to get the number of images in each page gallery, the number of PDFs to render and the number of eqmt cards to render.

The thumbnail images, PDF thumbnails and eqmt cards are rendered from a JSON object with below the fold loading.

```
atr.hirecar,
atr.train_bus
FROM t_content_body_evt_text at
JOIN t_allpages_metadata md ON md.page_id = at.page_id
JOIN t_allpages_transport atr ON atr.page_id = md.page_id
WHERE md.page_type = 'e'
AND md.page_status = true
ORDER BY at.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_11_get_evt_text_page_id ON
vm_11_get_evt_text (page_id);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_11_get_evt_text;
-- SEELCT FROM MV
SELECT * FROM vm_11_get_evt_text;
```

Author: Elliott Ward Farmer

6.13 Building all MVs for static text



The website has several different page types each requiring a different MV for the static text content.

Author: Elliott Ward Farmer

7. Building MVs that return JSON built from SQL row data

Author: Elliott Ward Farmer

7.1 Hero Gallery API My SQL Endpoint

Example Script to handle the Hero Gallery API endpoint response with MySQL.

```php
<?php

require_once("../../App/Config/config.php");

class ApiHeroGalleryfetch3 {

  private $con;

  public function __construct($con) {
    $this->con = $con;
  }

  public function getHeroGalleryData($param, $orientation, $qty) {

    $imgPathColumn = $orientation === "portrait" ? "imgWebpMobFull" : "imgWebpScnFull";
    $imgOrientation = $orientation === "portrait" ? "imgId_portrait" : "imgId_landscape";

    // Index page; $param = numeric value for week
    if (is_numeric($param)) {
      $imageArr = $this->getIndexPageImgData($imgPathColumn, $imgOrientation, $param, $qty);
      return $this->buildImgJsonObjWithURL($imageArr);
    }

    // Attraction or Event; $param = pageId
    elseif (!is_numeric($param) && (($param[0] === "a" || $param[0] === "e" ))) {
      $imageArr = $this->getNotIndexPageImgData($imgPathColumn, $imgOrientation, $param, $qty);
      return $this->buildImgJsonObjWithoutURL($imageArr);
    }
```

```php
    // List; $param = pageId
    elseif (!is_numeric($param) && (($param[0] === "t" || $param[0] === "b" || $param[0] === "v" ||
$param[0] === "w" || $param[0] === "x" || $param[0] === "y"))) {
      $imageArr = $this->getNotIndexPageImgData($imgPathColumn, $imgOrientation, $param, $qty);
      return $this->buildImgJsonObjWithURL($imageArr);
      }
    }

    private function getIndexPageImgData($imgPathColumn, $imgOrientation, $param, $qty) {
      $sql ="SELECT
              asset_images2.$imgPathColumn as imgPath,
              asset_images2.iptcTitle,
              allpages_pageurl2.urlparam0,
              allpages_pageurl2.urlparam1,
              allpages_pageurl2.urlparam2
            FROM pagecontent_indexherogallery
            JOIN asset_images2 ON asset_images2.imgId = $imgOrientation
            JOIN allpages_pageurl2 ON allpages_pageurl2.pageid = asset_images2.pageId_iptcSource
            WHERE pagecontent_indexherogallery.NumericWeek = $param";
      $stmt = $this->con->prepare($sql);
      $stmt->execute();
      $imgData = $stmt->fetchAll(PDO::FETCH_ASSOC);
      $imageArr = $qty === "all" ? $imgData : [$imgData[array_rand($imgData)]];
      return $imageArr;
    }
```

Author: Elliott Ward Farmer

7.1 Hero Gallery API My SQL Endpoint


```php
private function getNotIndexPageImgData($imgPathColumn, $imgOrientation, $param, $qty) {
  $sql ="SELECT
          asset_images2.$imgPathColumn as imgPath,
          asset_images2.iptcTitle,
          allpages_pageurl2.urlparam0,
          allpages_pageurl2.urlparam1,
          allpages_pageurl2.urlparam2
        FROM pagecontent_herogallery
        JOIN asset_images2 ON asset_images2.imgId = $imgOrientation
        JOIN allpages_pageurl2 ON allpages_pageurl2.pageid = asset_images2.pageId_iptcSource
        WHERE pagecontent_herogallery.pageId = :pageId";
  $stmt = $this->con->prepare($sql);
  $stmt->execute(array(':pageId'=> $param));
  $imgData = $stmt->fetchAll(PDO::FETCH_ASSOC);
  $imageArr = $qty === "all" ? $imgData : [$imgData[array_rand($imgData)]];
  return $imageArr;
}

private function buildImgJsonObjWithURL($imageArr) {
  $resultsArray = array();
  foreach ($imageArr as $imgDataArr) {
    $urlparam0 = $imgDataArr['urlparam0'];
    $urlparam1 = $imgDataArr['urlparam1'];
    $urlparam2 = $imgDataArr['urlparam2'];

    $imgPath = BT_BASE_URL."/".$imgDataArr['imgPath'];
    $url = BT_BASE_URL."/$urlparam0/$urlparam1/$urlparam2";
    $title = $imgDataArr['iptcTitle'];
```

In essence the script runs a sql query depending on what type of images are needed and what kind of page it is.

Finally the script builds an array of images and returns a json encoded object.

```php
    $jsonResultsArray = json_encode($resultsArray);
    return $jsonResultsArray;
  }

  private function buildImgJsonObjWithoutURL($imageArr) {
    $resultsArray = array();
    foreach ($imageArr as $imgDataArr) {
      $imgPath = BT_BASE_URL."/".$imgDataArr['imgPath'];
      $title = $imgDataArr['iptcTitle'];

      $resultsArray[] = array(
        'path' => $imgPath,
        'title' => $title,
        'url' => "null"
      );
    }
    $jsonResultsArray = json_encode($resultsArray);
    return $jsonResultsArray;
  }
}
?>
```

Author: Elliott Ward Farmer

7.2 MySQL–API script–JS Time Penalties

BarcelonaTravelHacks.com

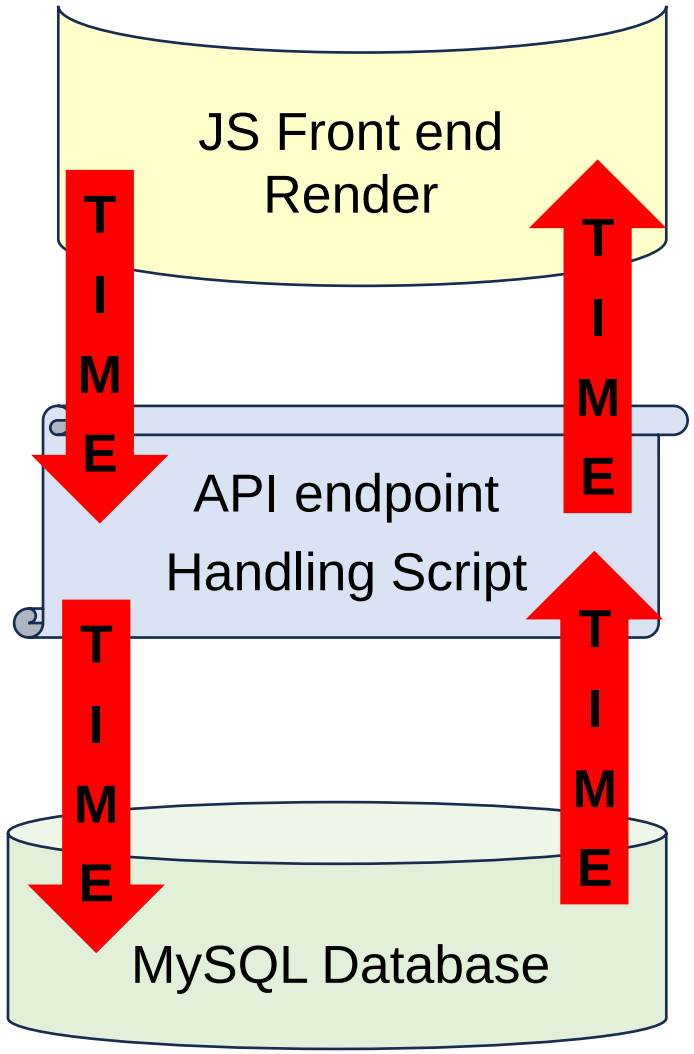
1. Get page_Id, page orientation and page type from data-HTML

2. Call API endpoint with params

3. Await endpoint response

4. Make SQL query based on API params

5. Query the database



8. Process endpoint response and render HTML

7. Return JSON

6. Process SQL rows into JSON array of objects

6. Return Rows

Author: Elliott Ward Farmer

7.3 Dynamically Loaded Hero Gallery


```sql
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_19_get_hero_gallery_index_by_week AS
WITH weeks AS (
    SELECT generate_series(1, 53) AS week_num
)
SELECT
     week_num, -- col1 --
   (
    SELECT json_agg(landscape) AS landscape -- col2 --
    FROM (
      SELECT
          ai.img_webp_scn_full AS path,
          ai.iptc_title AS title,
          RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url
      FROM t_content_fk_hero_gal_index fkhi
      JOIN t_asset_images ai ON ai.img_id = fkhi.img_id_landscape
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = ai.page_id_iptc_source
      WHERE fkhi.num_week = weeks.week_num
    ) AS landscape
  ),
  (
    SELECT json_agg(portrait) AS portrait -- col3 --
    FROM (
      SELECT
          ai.img_webp_mob_full AS path,
          ai.iptc_title AS title,
          RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url
      FROM t_content_fk_hero_gal_index fkhi
      JOIN t_asset_images ai ON ai.img_id = fkhi.img_id_portrait
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = ai.page_id_iptc_source
      WHERE fkhi.num_week = weeks.week_num
    ) AS portrait
  )
FROM weeks
ORDER BY week_num
WITH DATA;
```

The hero gallery is loaded with a JS API call to get an array of image objects. Depending if the screen is a mobile, large landscape images or small portrait images are in the JSON object

```sql
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_19_get_hero_gallery_index_by_week ON
vm_19_get_hero_gallery_index_by_week (week_num);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY
vm_19_get_hero_gallery_index_by_week;
-- SEELCT FROM MV
SELECT * FROM vm_19_get_hero_gallery_index_by_week;
```

The API gets different hero images for each week of the year depending what attractions and events are most viewed. i.e. beaches in summer, museums in winter etc.

Author: Elliott Ward Farmer

7.3 Dynamically Loaded Hero Gallery


```sql
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_19_get_hero_gallery_index_by_week AS
WITH weeks AS (
    SELECT generate_series(1, 53) AS week_num
)
SELECT
     week_num, -- col1 --
   (
    SELECT json_agg(landscape) AS landscape -- col2 --
    FROM (
      SELECT
          ai.img_webp_scn_full AS path,
          ai.iptc_title AS title,
          RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url
      FROM t_content_fk_hero_gal_index fkhi
      JOIN t_asset_images ai ON ai.img_id = fkhi.img_id_landscape
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = ai.page_id_iptc_source
      WHERE fkhi.num_week = weeks.week_num
    ) AS landscape
  ),
  (
    SELECT json_agg(portrait) AS portrait -- col3 --
    FROM (
      SELECT
          ai.img_webp_mob_full AS path,
          ai.iptc_title AS title,
          RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url
      FROM t_content_fk_hero_gal_index fkhi
      JOIN t_asset_images ai ON ai.img_id = fkhi.img_id_portrait
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = ai.page_id_iptc_source
      WHERE fkhi.num_week = weeks.week_num
    ) AS portrait
  )
FROM weeks
ORDER BY week_num
WITH DATA;
```

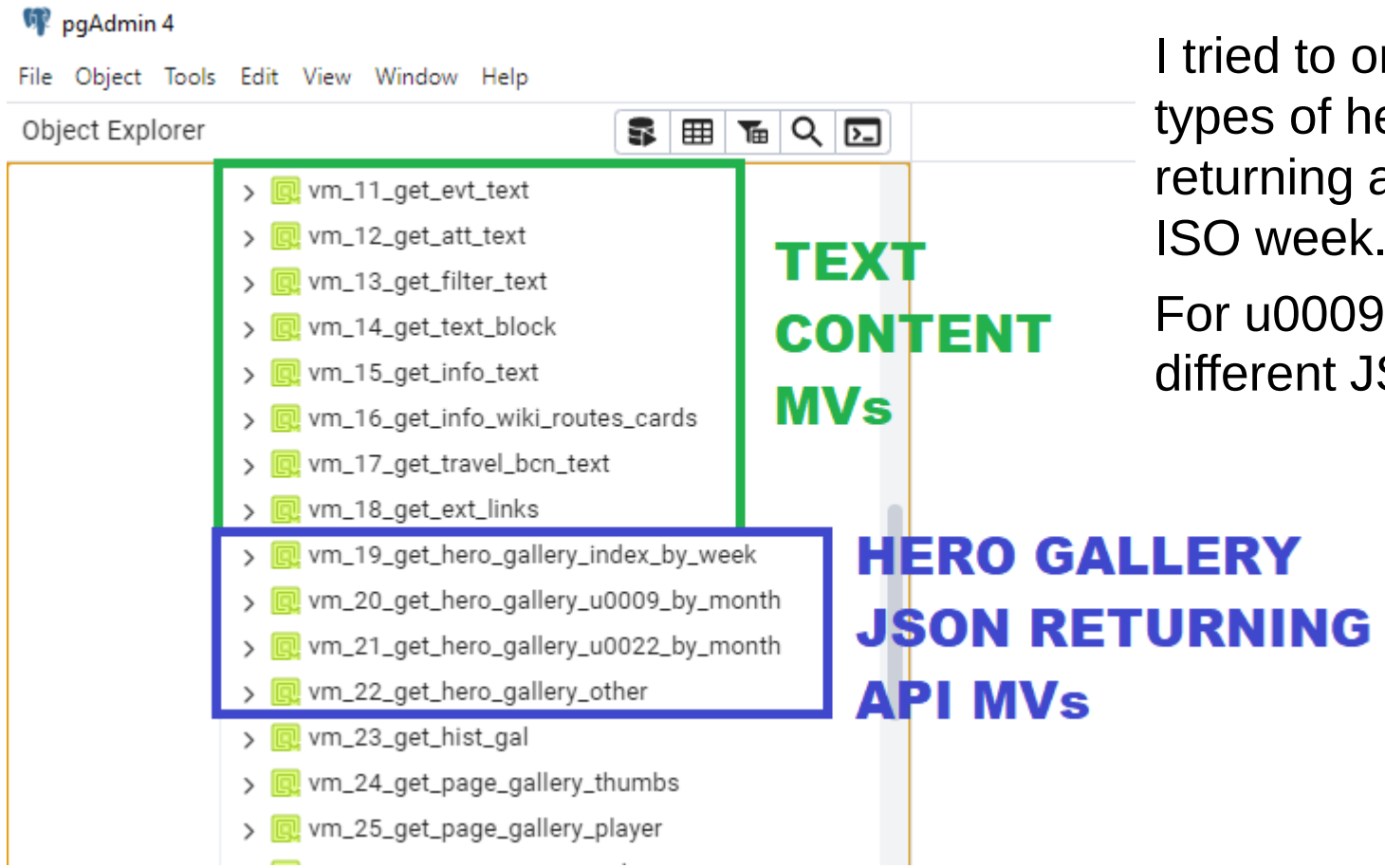
I am using a CTE (common table expression) to generate the ISO week numbers.

Note the use of JSON aggregate to flatten the SQL rows into key value pairs in each JSON object in the JSON array.

Th output of this CTE query is a three column MV where column 1 is the page_id, col2 is the JSON array for landscape and col3 is the JSON array for portrait

Author: Elliott Ward Farmer

7.4 Building all the Hero Gallery MVs



I tried to only have on MV for all the different types of hero gallery but for the index page I am returning a different JSON of images for each ISO week.

For u0009 and u0022 pages I am returning a different JSON for the ISO month.

For all other pages I am returning the same JSON images regardless of the time of the year.

Each MV is three columns of page ID, landscape JSON and portrait JSON

Author: Elliott Ward Farmer

7.5 API endpoint PostgreSQL script


```php
<?php

require_once("../../../App/Config/pgSQL_config.
php");

// Check params are in GET
if( !isset($_GET["p"]) || !isset($_GET["o"]) )
{
    echo "ERROR: No 'p' or 'o' GET data
received";
    exit();
}
else {
  $pageId = $_GET["p"];
  $date = new DateTime();
  $week = $date->format("W");
  $month = $date->format("n");

  if ($_GET["o"] === 's') {
    $orientation = 'landscape';
  }
  elseif ($_GET["o"] === 'm') {
    $orientation = 'portrait';
  }
  else {
    echo "ERROR: No 'o' GET data received";
    exit();
  }
```

```php
  switch ($pageId) {
    case 'index':
      $sql = "SELECT
                $orientation
              FROM
vm_19_get_hero_gallery_index_by_week
              WHERE week_num = $week
              LIMIT 1";
      $stmt = $conPgSQL->prepare($sql);
      $stmt->execute();
      echo $stmt->fetch(PDO::FETCH_COLUMN);
      break;
    case 'u0009':
      $sql = "SELECT
                $orientation
              FROM
vm_20_get_hero_gallery_u0009_by_month
              WHERE month_num = $month
              LIMIT 1";
      $stmt = $conPgSQL->prepare($sql);
      $stmt->execute();
      echo $stmt->fetch(PDO::FETCH_COLUMN);
      break;
```

```php
    case 'u0022':
      $sql = "SELECT
                $orientation
              FROM
vm_21_get_hero_gallery_u0022_by_month
              WHERE month_num = $month
              LIMIT 1";
      $stmt = $conPgSQL->prepare($sql);
      $stmt->execute();
      echo $stmt->fetch(PDO::FETCH_COLUMN);
      break;
    default:
      $sql = "SELECT
                $orientation
              FROM vm_22_get_hero_gallery_other
              WHERE page_id = :pageId
              LIMIT 1";
      $stmt = $conPgSQL->prepare($sql);
      $stmt->execute(array(':pageId' => $pageId));
      echo $stmt->fetch(PDO::FETCH_COLUMN);
  }
 }
?>
```

Now the PostgreSQL API endpoint script is just getting a date component from current date and querying the correct MV to return the JSON. Much simpler and faster API response.

Author: Elliott Ward Farmer

7.6 PDF docs API endpoint MV


```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_34_get_page_pdf_cards AS
WITH pdf AS (
  SELECT DISTINCT
  page_id
  FROM t_content_fk_pdf_docs
  UNION
  SELECT DISTINCT
  page_id
  FROM t_content_body_info_img
  ORDER BY page_id
)
SELECT
  -- col1 --
    page_id,
  -- col2 --
  (
    SELECT json_agg(pdf_cards) AS pdf_cards
    FROM (
      SELECT
      ap.pdf_id AS pdfid,
      ap.pdf_title AS pdftitle,
      ap.pdf_thumbnail_webp AS pdfthumb,
      ap.pdf_author AS pdfauthor,
      ap.pdf_path AS pdfpath
      FROM t_asset_pdf ap
      JOIN t_content_fk_pdf_docs fkpd ON fkpd.pdf_id = ap.pdf_id
      JOIN t_allpages_metadata md ON md.page_id = fkpd.page_id
      WHERE ap.pdf_status = true
      AND fkpd.page_id = pdf.page_id
      UNION
      SELECT
      cbii.img_id AS pdfid,
      cbii.img_title AS pdftitle,
      cbii.img_thumb_webp AS pdfthumb,
      cbii.img_author AS pdfauthor,
      cbii.pdf_path AS pdfpath
      FROM t_content_body_info_img cbii
      WHERE pdf_status = true
      AND cbii.page_id = pdf.page_id
    ) AS pdf_cards
  )
FROM pdf
ORDER BY page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_34_get_page_pdf_cards ON
vm_34_get_page_pdf_cards (page_id);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_34_get_page_pdf_cards;
-- SEELCT FROM MV
SELECT * FROM vm_34_get_page_pdf_cards;
```

Using a union query within a CTE to combine data from two tables.

Union subquery with column aliasing to 'merge' dissimilar data into one table of 'similar' data

This could have been done with two MVs but as the Javascript runs the same code with the same JSON object, I decide to use a union query with aliasing to build the JSON array of PDF objects.

Author: Elliott Ward Farmer

7.7 PDF docs API endpoint Script

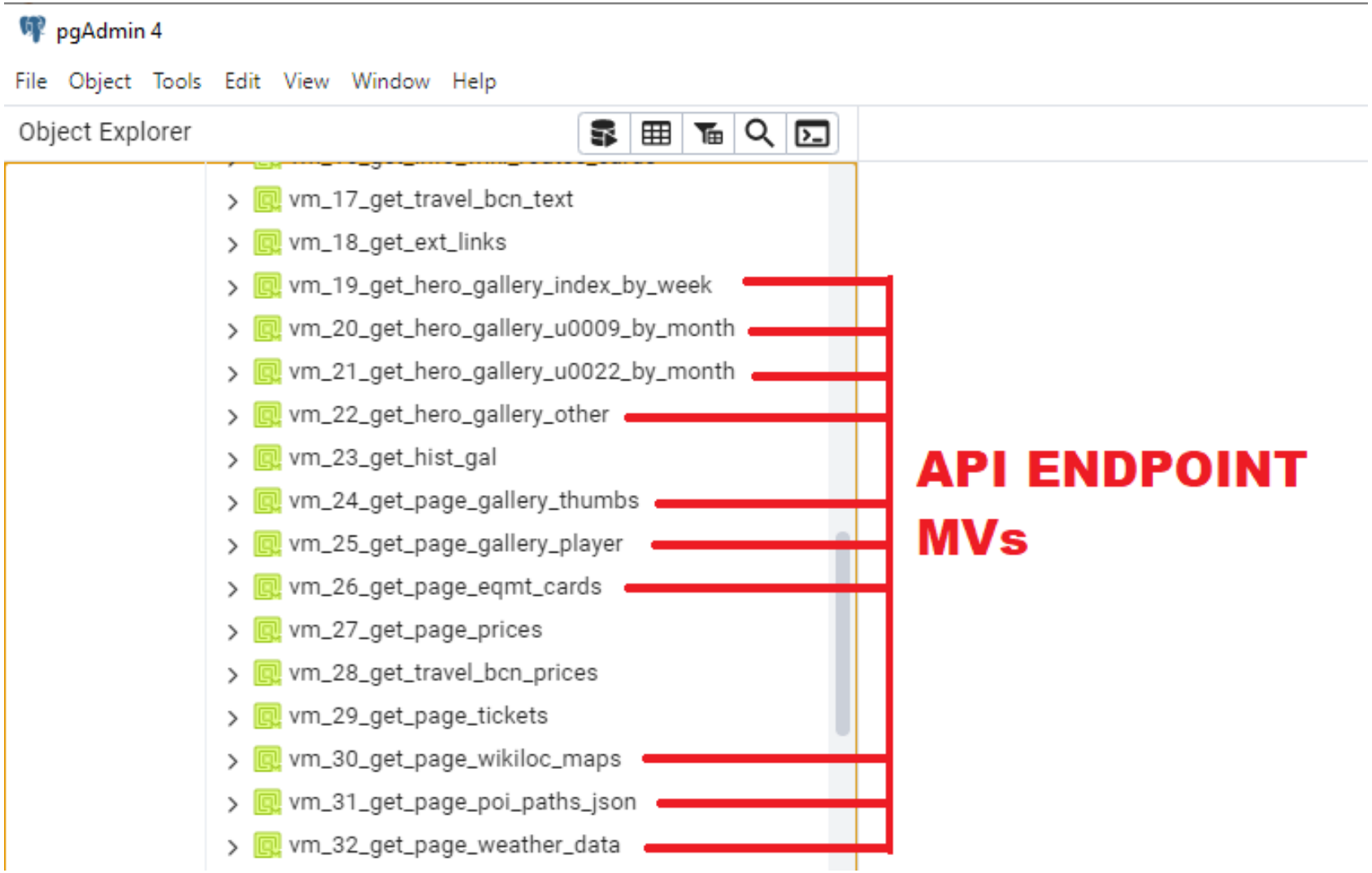

```php
<?php

require_once("../../../App/Config/pgSQL_config.php");

// Check params are in GET
if(!isset($_GET["p"])) {
    echo "ERROR: No url 'p' GET data received PDF API";
    exit();
}
else {
  $pageId = $_GET["p"];

  $sql = "SELECT
            pdf_cards
          FROM vm_34_get_page_pdf_cards
          WHERE page_id = :page_id
          LIMIT 1";
  $stmt = $conPgSQL->prepare($sql);
  $stmt->execute(array(':page_id' => $pageId));
  echo $stmt->fetch(PDO::FETCH_COLUMN);
}

?>
```

The use of the union sub query and CTE union means that the Endpoint API script is simple with just a simple logic: get PDF JSON for page_id.

Again, I am using advanced SQL and PostgreSQL features to reduce the processing time of the API calls making for faster API responses and an overall faster website.

Author: Elliott Ward Farmer

7.8 Building the remaining API MVs



Author: Elliott Ward Farmer

8. The Troublesome geo-JSON MV query

Author: Elliott Ward Farmer

8.1 The troublesome GEO-JSON MV

t_allpages_metadata
Page_id VARCHAR(6) PRIMARY KEY
page_type page_type
no_index_tag BOOLEAN
page_status BOOLEAN
transport_type VARCHAR(20)
page_meta_title VARCHAR (75)
page_meta_description VARCHAR(170)
page_author VARCHAR(50)
page_added TIMESTAMPTZ
Page_updated TIMESTAMPTZ
page_views BIGINT
page_thumbwebp VARCHAR(100)
page_sm_image VARCHAR(70)
page_search_keywords TEXT

1

∞

∞

t_geodata_path
path_id VARCHAR(9) PRIMARY KEY
page_id VARCHAR(6)
geo_title
geo_data GEOMETRY
color VARCHAR(10)

t_geodata_poi
poi_id VARCHAR(9) PRIMARY KEY
page_id VARCHAR(6)
geo_title
geo_data GEOMETRY
color VARCHAR(10)

Objective: Build a GEO-JSON feature collection object consisting of one or more POIs and none or more PATHS from the PATH and POI tables using JOINS to combine with URI data and a pin popup with thumbnail, title.

Author: Elliott Ward Farmer

8.2 GEO-JSON structure


```
{"type":"FeatureCollection",
  "features": [
    {"type": "Feature",
     "geometry": {
        "type": "Point",
        "coordinates": [
           1.811288225876066,
           41.60524059089237]      },
"properties":
  {
  "img": "assets/pub/page-tm/att/a0157-60.webp",
  "url": "travel/montserrat-day-trip/sant-jeroni-from-montserrat-monastery",
  "color": "violet",
  "pageId": "a0157",
  "geoTitle": "Sant Jeroni Summit from Montserrat Monastery"
  }
},
{
  "type": "Feature",
  "geometry": {
    "type": "LineString",
    "coordinates": [
          [
          1.849296,
          41.610028
          ],
          [
          1.848818,
          41.610737
              ]
    ]
  }
}
```


Each page can have multiple point objects and multiple linestring objects in the geo-JSON object.

It has to be built with correct geo-JSON formatting consisting of a feature collection containing an array of features objects. Within the features object are individual features of either point or linestring.

In short, a complex JSON object with nested objects.

Father depending on the page type, the query is different to get all the Point and Linestring geo_data. Point type also has different properties to a linestring type.

Author: Elliott Ward Farmer

8.3 GEO-JSON PostGreSQL MV


```
CREATE MATERIALIZED VIEW IF NOT EXISTS
vm_get_page_poi_paths_json AS
WITH pages AS (
  SELECT page_id, page_type
  FROM t_allpages_metadata
  WHERE page_type IN ('a', 'e', 'b', 't', 'v')
  ORDER BY page_id
),
json_data_poi AS (
  SELECT
  p.page_id,
  p.page_type,
  (
    SELECT jsonb_build_object(
      'type', 'FeatureCollection',
      'features', jsonb_agg(
        jsonb_build_object(
          'type', 'Feature',
          'properties', jsonb_build_object(
            'img', md.page_thumbwebp,
            'geoTitle', gpoi.geo_title,
            'url', RTRIM(uhn.urlparam0 || '/' ||
uhn.urlparam1 || '/' || uhn.urlparam2, '/'),
            'color', gpoi.color
          ),
          'geometry', ST_AsGeoJSON(geo_data)::jsonb
        )
      )
    )
```

```
    FROM t_allpages_metadata md
    JOIN t_allpages_url_handler_new uhn ON uhn.page_id =
md.page_id
    LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id =
md.page_id
    WHERE
      (p.page_type IN ('a', 'e') AND md.page_id =
p.page_id AND md.page_status = TRUE AND md.page_type IN
('a', 'e'))
      OR
      (p.page_type IN ('b', 't') AND md.page_id IN (
          SELECT nvl.page_id
          FROM t_nav_loc nvl
          WHERE nvl.menu_page_id = p.page_id
        )
      )
      OR
      (p.page_type = 'v' AND md.page_id IN (
          SELECT nvl.page_id
          FROM t_nav_loc nvl
          WHERE nvl.menu_page_id = p.page_id
        )
      )
  ) AS json_data
  p.page_type,
  (
```

```
    SELECT jsonb_build_object(
        'type', 'Feature',
        'properties', jsonb_build_object(
          'color', gpath.color
        ),
        'geometry', ST_AsGeoJSON(geo_data)::jsonb
      )

    FROM t_allpages_metadata md
    LEFT JOIN t_geodata_path gpath ON gpath.page_id =
md.page_id
    WHERE
      (p.page_type IN ('a', 'e') AND md.page_id =
p.page_id AND md.page_status = TRUE AND md.page_type IN
('a', 'e'))
      OR
      (p.page_type IN ('b', 't') AND md.page_id IN (
          SELECT nvl.page_id
          FROM t_nav_loc nvl
          WHERE nvl.menu_page_id = p.page_id
        )
      )
```

```
  ) AS json_data
  FROM pages p
)
SELECT
    jm.page_id,
    jm.page_type,
    jsonb_build_object(
        'type', 'FeatureCollection',
        'features',
        COALESCE(
            (jm.json_data->'features') || jp.path_json,
            jm.json_data->'features',
            jp.path_json
        )
    ) AS map_json
FROM json_data_poi jm
LEFT JOIN json_paths jp ON jm.page_id = jp.page_id
ORDER BY jm.page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_get_page_poi_paths_json ON
vm_get_page_poi_paths_json (page_id);
  REFRESH DATA IN MV
FRESH MATERIALIZED VIEW CONCURRENTLY
_get_page_poi_paths_json;
 SEELCT FROM MV
LECT * FROM vm_get_page_poi_paths_json;
```

Does not work as desired. It only pulls one row from the linestring table when it should be pulling more.

Author: Elliott Ward Farmer

8.4 GEO-JSON PHP build geo-json Tbl


```php
<?php

class buildGeoJsonPage {

  private $conPgSQLs;

  public function __construct($conPgSQLs) {
      $this->conPgSQLs = $conPgSQLs;
  }

  public function buildGeoJsonObj($p, $t) {

    # 1. Get GeoJson Data for each page type
    switch ($t) {
      case 'u':
        $json = $this->themePage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      case 'v':
        $json = $this->locationPage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      case 't':
        $json = $this->locationPage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      case 'b':
        $json = $this->locationPage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      case 'a':
        $json = $this->attEvtPage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      case 'e':
        $json = $this->attEvtPage($p, $t);
        return $this->insertIntoGeoJsonTable($p, $json);
        break;
      default:
      echo 'page type ' . $pageType . ' for page' . $pageId . '
cannot be identified';
    }
  }
```

```php
  # 1. Handle Theme Page - Get All POIs for 'u' Theme page
  private function themePage($p, $t) {
    $sql = "SELECT
              md.page_id,
              md.page_thumbwebp AS img,
              gpoi.geo_title,
              RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url,
              ST_AsText(gpoi.geo_data) AS geometry,
              gpoi.color
            FROM t_allpages_metadata md
            JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
            JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
            JOIN t_nav_tme nt ON nt.page_id = md.page_id
            WHERE nt.menu_page_id = :menu_page_id
            AND md.page_status = true
            AND (md.page_type = 'a' OR md.page_type = 'e')
            ORDER BY md.page_views DESC";
    $stmt = $this->conPgSQLs->prepare($sql);
    $stmt->execute(array(':menu_page_id' => $p));
    $PoiGeoDataArr = $stmt->fetchAll(PDO::FETCH_ASSOC);
    return $this->buildThemeGeoJSON($p, $t, $PoiGeoDataArr);
  }
```

After multiple attempts to build this very complex SQL query I decided to try a workaround by building the geo-JSON nested object using PHP then inserting it into a new table.

Author: Elliott Ward Farmer

8.4 GEO-JSON PHP build geo-json Tbl


```php
# 1. Handle Location based Page - Get All POIs for 'b','t' & 'v' page type
private function locationPage($p, $t) {
  $sql = "SELECT
            md.page_id,
            md.page_thumbwebp AS img,
            gpoi.geo_title,
            RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url,
            ST_AsText(gpoi.geo_data) AS geometry,
            gpoi.color
          FROM t_allpages_metadata md
          JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
          JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
          JOIN t_nav_loc nl ON nl.page_id = md.page_id
          WHERE nl.menu_page_id = :menu_page_id
          AND md.page_status = true
          AND (md.page_type = 'a' OR md.page_type = 'e')
          ORDER BY md.page_views DESC";
  $stmt = $this->conPgSQLs->prepare($sql);
  $stmt->execute(array(':menu_page_id' => $p));
  $PoiGeoDataArr = $stmt->fetchAll(PDO::FETCH_ASSOC);
  return $this->buildgeoJSON($p, $t, $PoiGeoDataArr);
}

# 1. Handle single POI based Page - Get All POIs for 'a' & 'e' page type
private function attEvtPage($p, $t) {
  $sql = "SELECT
            md.page_id,
            md.page_thumbwebp AS img,
            gpoi.geo_title,
            RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/') AS url,
            ST_AsText(gpoi.geo_data) AS geometry,
            gpoi.color
          FROM t_allpages_metadata md
          JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
          JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
          WHERE md.Page_id = :page_id
          AND md.page_status = true
          AND (md.page_type = 'a' OR md.page_type = 'e')
          ORDER BY md.page_views DESC";
        $stmt = $this->conPgSQLs->prepare($sql);
        $stmt->execute(array(':page_id' => $p));
        $PoiGeoDataArr = $stmt->fetchAll(PDO::FETCH_ASSOC);
  return $this->buildgeoJSON($p, $t, $PoiGeoDataArr);
}
```

```php
  # 2. Build GeoJSON feature collection array for Theme page type
  private function buildThemeGeoJSON($p, $t, $PoiGeoDataArr) {
    $geoJSON = array(
      'type'      => 'FeatureCollection',
      'features'  => array()
    );

    # Loop through POI rows to build feature arrays
    foreach ($PoiGeoDataArr as $poiArr) {
      $pageId = $poiArr["page_id"];
      $img = $poiArr["img"];
      $geoTitle = str_replace ("'","''",$poiArr["geo_title"]);
      $url = $poiArr["url"];
      $color = $poiArr["color"];
      $poiDataArr = Explode(" ",(str_replace(")","",
(str_replace("POINT(","",$poiArr["geometry"])))));

      $featuresArr = array(
        'type' => 'Feature',

        'properties' => array(
          'pageId' => $pageId,
          'img' => $img,
          'geoTitle' => $geoTitle,
          'url' => $url,
          'color' => $color
        ),
        'geometry' => array(
          'type' => 'Point',
          'coordinates' => array(
            floatval($poiDataArr[0]),
            floatval($poiDataArr[1])
          )
        ),
      );

      # Add feature arrays to feature collection array
      array_push($geoJSON['features'], $featuresArr);
    };
```

Author: Elliott Ward Farmer

8.4 GEO-JSON PHP build geo-json Tbl


```
    # Add linestring features array to feature collection array
    # Get Routes geodata
    $sql = "SELECT DISTINCT
            color,
            ST_AsText(gpath.geo_data) AS geometry
          FROM t_geodata_path gpath
          WHERE page_id = :page_id
          OR page_id IN(SELECT page_id FROM t_nav_tme WHERE menu_page_id = :page_id)";
    $stmt = $this->conPgSQLs->prepare($sql);
    $stmt->execute(array(':page_id' => $p));
    $routesGeoDataArr = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if (count($routesGeoDataArr) > 0) {
      foreach ($routesGeoDataArr as $routeArr) {
        $color = $routeArr["color"];
        $coordsArr = explode(",",(str_replace(")","",(str_replace("LINESTRING(","",$routeArr["geometry"])))));
        $cordinates = [];
        foreach ($coordsArr as $coords) {
          $point = array(
            floatval(explode(' ', $coords)[0]),
            floatval(explode(' ', $coords)[1])
          );
          array_push($cordinates, $point);
        }

        $lineStringArr = array(
          'type' => 'Feature',
          'properties' => array(
            'color' => $color
          ),
          'geometry' => array(
            'type' => 'LineString',
            'coordinates' => $cordinates,
          ),
        );
      array_push($geoJSON['features'], $lineStringArr);
      }
    }
    return json_encode($geoJSON);
```

```
    /*************** TEST CODE **********************/
    // print("<pre>".print_r($routesGeoDataArr,true)."</pre>");
    // echo("<h1>".$_GET["p"]."</h1>");
    //print("<pre>".print_r(json_encode($geoJSON),true)."</pre>");
    /*************** TEST CODE **********************/
  }

  # 2. Build GeoJSON feature collection array for All other page types
  private function buildgeoJSON($p, $t, $PoiGeoDataArr) {
    # Build GeoJSON feature collection array
    $geoJSON = array(
        'type'      => 'FeatureCollection',
        'features'  => array()
    );
```

Author: Elliott Ward Farmer

8.4 GEO-JSON PHP build geo-json Tbl


```php
# Loop through rows to build feature arrays
foreach ($PoiGeoDataArr as $poiArr) {
  $pageId = $poiArr["page_id"];
  $img = $poiArr["img"];
  $geoTitle = str_replace ("'","''",$poiArr["geo_title"]);
  $url = $poiArr["url"];
  $color = $poiArr["color"];
  $poiDataArr = Explode(" ",(str_replace(")","", (str_replace("POINT(","",$poiArr["geometry"])))));

  $featuresArr = array(
    'type' => 'Feature',

    'properties' => array(
      'pageId' => $pageId,
      'img' => $img,
      'geoTitle' => $geoTitle,
      'url' => $url,
      'color' => $color
    ),
    'geometry' => array(
      'type' => 'Point',
      'coordinates' => array(
        floatval($poiDataArr[0]),
        floatval($poiDataArr[1])
      )
    ),
  );

  # Add feature arrays to feature collection array
  array_push($geoJSON['features'], $featuresArr);
};
```


```php
# Loop through rows to build feature arrays
foreach ($PoiGeoDataArr as $poiArr) {
  $pageId = $poiArr["page_id"];
  $img = $poiArr["img"];
  $geoTitle = str_replace ("'","''",$poiArr["geo_title"]);
  $url = $poiArr["url"];
  $color = $poiArr["color"];
  $poiDataArr = Explode(" ",(str_replace(")","", (str_replace("POINT(","",$poiArr["geometry"])))));

  $featuresArr = array(
    'type' => 'Feature',

    'properties' => array(
      'pageId' => $pageId,
      'img' => $img,
      'geoTitle' => $geoTitle,
      'url' => $url,
      'color' => $color
    ),
    'geometry' => array(
      'type' => 'Point',
      'coordinates' => array(
        floatval($poiDataArr[0]),
        floatval($poiDataArr[1])
      )
    ),
  );

  # Add feature arrays to feature collection array
  array_push($geoJSON['features'], $featuresArr);
};
```


Author: Elliott Ward Farmer

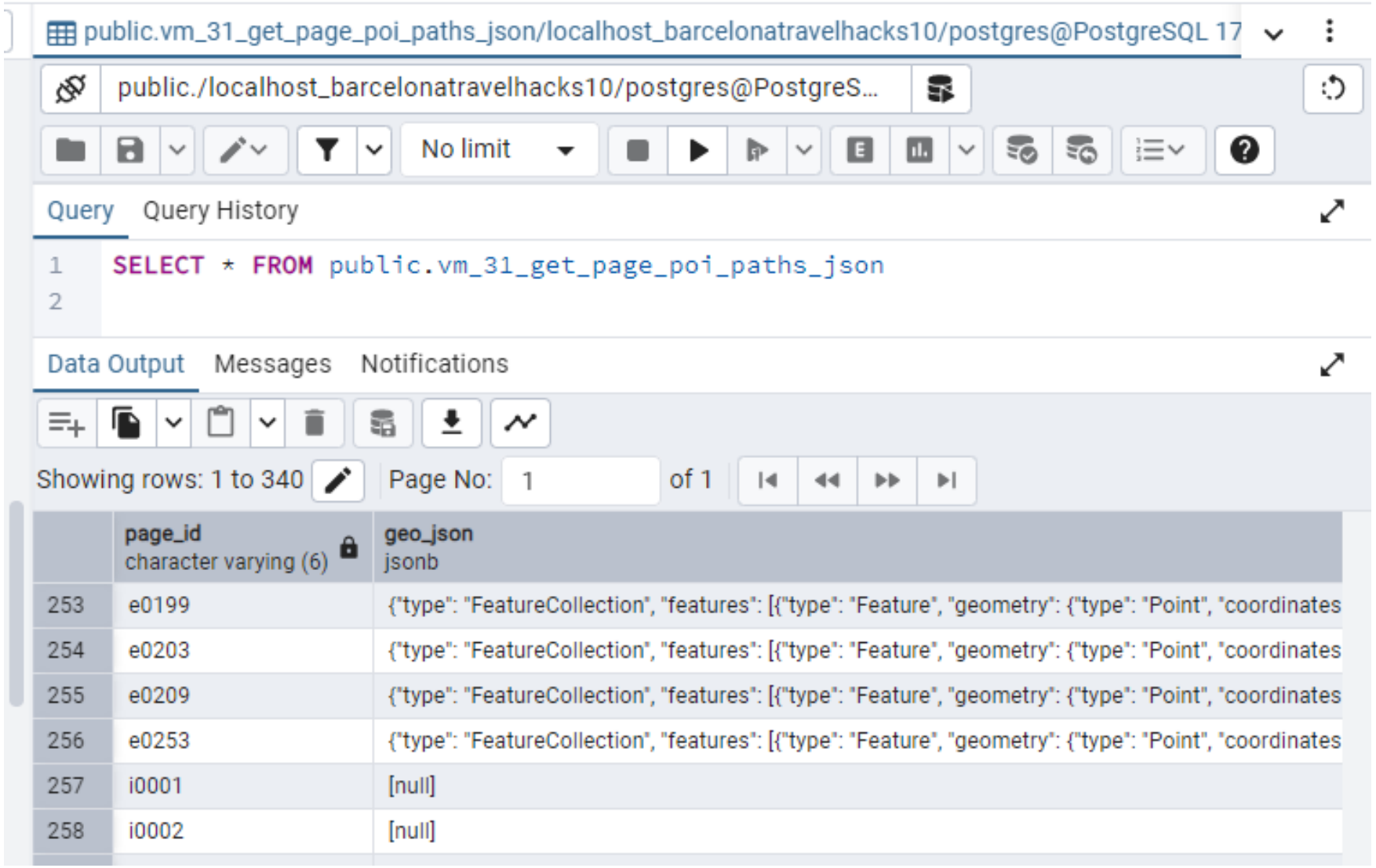
8.4 GEO-JSON PHP build geo-json Tbl


```php
    # Add linestring feature array to feature collection array
    # Get Routes geodata
    $sql = "SELECT DISTINCT
              color,
              ST_AsText(gpath.geo_data) AS geometry
            FROM t_geodata_path gpath
            WHERE page_id = :page_id
            OR page_id IN(SELECT page_id FROM t_nav_loc WHERE menu_page_id = :page_id)";
    $stmt = $this->conPgSQLs->prepare($sql);
    $stmt->execute(array(':page_id' => $p));
    $routesGeoDataArr = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if (count($routesGeoDataArr) > 0) {
      foreach ($routesGeoDataArr as $routeArr) {
        $color = $routeArr["color"];
        $coordsArr = explode(",",(str_replace(")","",(str_replace("LINESTRING(","",$routeArr["geometry"])))));
        $cordinates = [];
        foreach ($coordsArr as $coords) {
          $point = array(
            floatval(explode(' ', $coords)[0]),
            floatval(explode(' ', $coords)[1])
          );
          array_push($cordinates, $point);
        }

        $lineStringArr = array(
          'type' => 'Feature',
          'properties' => array(
            'color' => $color
          ),
          'geometry' => array(
            'type' => 'LineString',
            'coordinates' => $cordinates,
          ),
        );
        array_push($geoJSON['features'], $lineStringArr);
      }
    }

    return json_encode($geoJSON);
```

```php
    /**************** TEST CODE **********************/
    // print("<pre>".print_r($routesGeoDataArr,true)."</pre>");
    // echo("<h1>".$_GET["p"]."</h1>");
    //print("<pre>".print_r(json_encode($geoJSON),true)."</pre>");
    /**************** TEST CODE **********************/
  }

  # 3. Insert into GeoJson Table for each page (row)
  private function insertIntoGeoJsonTable($p, $json) {
    $geoJson = "'".$json."'";
    $pageId = "'".$p."'";

    $sql = "INSERT INTO t_asset_geo_json (\"page_id\", \"geo_json\")
    VALUES ($pageId, $geoJson)
    ON CONFLICT (page_id) DO UPDATE
      SET geo_json = $geoJson
      RETURNING geo_json";
    $stmt = $this->conPgSQLs->prepare($sql);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_COLUMN);
  }
}
?>
```

Author: Elliott Ward Farmer

8.5 GEO-JSON PHP build geo-json Tbl



	page_id character varying (6)	geo_json jsonb
253	e0199	{"type": "FeatureCollection", "features": [{"type": "Feature", "geometry": {"type": "Point", "coordinates
254	e0203	{"type": "FeatureCollection", "features": [{"type": "Feature", "geometry": {"type": "Point", "coordinates
255	e0209	{"type": "FeatureCollection", "features": [{"type": "Feature", "geometry": {"type": "Point", "coordinates
256	e0253	{"type": "FeatureCollection", "features": [{"type": "Feature", "geometry": {"type": "Point", "coordinates
257	i0001	[null]
258	i0002	[null]

BarcelonaTravelHacks.com

The 300-line php script builds the geo-JSON row for each page_id where applicable. I then make an MV from this table.

This is an undesiable solution because if I want to update or add a new row of geo-JSON to the table then I have to run the script and then concurrently refresh the MV

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit


```
-- TEST 1 - Build a union query to combine POI and PATH table data
SELECT
  md.page_thumbwebp as img,
  RTRIM(coalesce(uhn.urlparam0,'') || '/' || coalesce(uhn.urlparam1,'') || '/' ||
coalesce(uhn.urlparam2,''),'/') as uri,
  gpoi.geo_title as title,
  ST_AsGeoJSON(gpoi.geo_data)::jsonb as geometry,
  gpoi.color
  FROM t_allpages_metadata md
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
      LEFT JOIN t_geodata_path gpoi ON gpoi.page_id = md.page_id
      WHERE
        (md.page_type IN ('a', 'e') AND md.page_id = md.page_id AND md.page_status = TRUE AND
md.page_type IN ('a', 'e'))
        OR
        (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
            SELECT nvl.page_id
            FROM t_nav_loc nvl
            WHERE nvl.menu_page_id = md.page_id
          )
        )
        OR
        (md.page_type = 'u' AND md.page_id IN (
            SELECT ntm.page_id
            FROM t_nav_tme ntm
            WHERE ntm.menu_page_id = md.page_id
          )
        )
```

OK. I was not happy about using a php script to build a geo_json table from existing tables then creating an MV from this.

```
  UNION
  SELECT
  NULL as img,
  NULL as uri,
  NULL as title,
  ST_AsGeoJSON(gpath.geo_data)::jsonb as geometry,
  gpath.color
  FROM t_allpages_metadata md
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
      LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
    LEFT JOIN t_geodata_path gpath ON gpath.page_id = md.page_id
      WHERE
        (md.page_type IN ('a', 'e') AND md.page_id = md.page_id AND md.page_status = TRUE AND
md.page_type IN ('a', 'e'))
        OR
        (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
            SELECT nvl.page_id
            FROM t_nav_loc nvl
            WHERE nvl.menu_page_id = md.page_id
          )
        )
        OR
        (md.page_type = 'u' AND md.page_id IN (
            SELECT ntm.page_id
            FROM t_nav_tme ntm
            WHERE ntm.menu_page_id = md.page_id
          )
        )
```

I decided to experiment with a union query on the POI and PATHS geo_data tables.

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit


```
SELECT
  jsonb_build_object(
    'type', 'Feature',
    'properties', jsonb_build_object(
    'img', md.page_thumbwebp,
    'geoTitle', gpoi.geo_title,
    'url', RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/'),
    'color', gpoi.color
    ),
    'geometry', ST_AsGeoJSON(gpoi.geo_data)::jsonb
  )
  FROM t_allpages_metadata md
  JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
  LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
  LEFT JOIN t_geodata_path gpath ON gpath.page_id = md.page_id
  WHERE
    (md.page_type IN ('a', 'e') AND md.page_id = md.page_id AND md.page_status = TRUE AND
md.page_type IN ('a', 'e'))
    OR
    (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
    SELECT nvl.page_id
    FROM t_nav_loc nvl
    WHERE nvl.menu_page_id = md.page_id
    )
    )
    OR
    (md.page_type = 'u' AND md.page_id IN (
    SELECT ntm.page_id
    FROM t_nav_tme ntm
    WHERE ntm.menu_page_id = md.page_id
      )
    }
```

Test 2. using nested jsonb_build_objects to construct the two different type of geo_JSON features

```
UNION
  SELECT
    jsonb_build_object(
    'type', 'Feature',
    'properties', jsonb_build_object(
    'color', gpath.color
    ),
    'geometry', ST_AsGeoJSON(gpath.geo_data)::jsonb
  )
  FROM t_allpages_metadata md
  JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
  LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
  LEFT JOIN t_geodata_path gpath ON gpath.page_id = md.page_id
  WHERE
  (md.page_type IN ('a', 'e') AND md.page_id = md.page_id AND md.page_status = TRUE AND
md.page_type IN ('a', 'e'))
  OR
  (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
  SELECT nvl.page_id
  FROM t_nav_loc nvl
  WHERE nvl.menu_page_id = md.page_id
    )
  )
  OR
  (md.page_type = 'u' AND md.page_id IN (
  SELECT ntm.page_id
  FROM t_nav_tme ntm
  WHERE ntm.menu_page_id = md.page_id
  )
)
```

Author: Elliott Ward Farmer

7.6 GEO-JSON PostGreSQL MV revisit


```
WITH pages AS (
  SELECT page_id, page_type
  FROM t_allpages_metadata
  WHERE page_type IN ('a', 'e', 'b', 't', 'v', 'u')
  ORDER BY page_id
),
json_data AS (
  SELECT
    p.page_id,
    p.page_type,
    (
      SELECT jsonb_build_object(
        'type', 'FeatureCollection',
        'features', (
          SELECT jsonb_agg(features.feature)
          FROM (
            -- POI features
            SELECT jsonb_build_object(
              'type', 'Feature',
              'properties', jsonb_build_object(
                'img', md.page_thumbwebp,
                'geoTitle', gpoi.geo_title,
                'uri', RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/'),
                'color', gpoi.color
              ),
              'geometry', ST_AsGeoJSON(gpoi.geo_data)::jsonb
            ) AS feature
            FROM t_allpages_metadata md
            JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
            LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
      WHERE
                (md.page_type IN ('a', 'e') AND md.page_status = TRUE AND md.page_id = p.page_id)
                OR
                (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
                  SELECT nvl.page_id
                  FROM t_nav_loc nvl
                  WHERE nvl.menu_page_id = md.page_id
                ))
                OR
                (md.page_type = 'u' AND md.page_id IN (
                  SELECT ntm.page_id
                  FROM t_nav_tme ntm
                  WHERE ntm.menu_page_id = md.page_id
                ))

            UNION

            -- Path features
            SELECT jsonb_build_object(
              'type', 'Feature',
              'properties', jsonb_build_object(
                'color', gpath.color
              ),
              'geometry', ST_AsGeoJSON(gpath.geo_data)::jsonb
            ) AS feature
            FROM t_allpages_metadata md
            JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
            LEFT JOIN t_geodata_path gpath ON gpath.page_id = md.page_id
        )
SELECT * FROM json_data
```

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit


```
            WHERE
              (md.page_type IN ('a', 'e') AND md.page_status = TRUE AND md.page_id = p.page_id)
              OR
              (md.page_type IN ('b', 't', 'v') AND md.page_id IN (
                SELECT nvl.page_id
                FROM t_nav_loc nvl
                WHERE nvl.menu_page_id = md.page_id
              ))
              OR
              (md.page_type = 'u' AND md.page_id IN (
                SELECT ntm.page_id
                FROM t_nav_tme ntm
                WHERE ntm.menu_page_id = md.page_id
              ))
          ) AS features
        )
      )
    ) AS json_data
  FROM pages p
)
SELECT * FROM json_data
```

Test 6. (several experiments later!) This gives me the correct output of the combination of POI and PATH features objects within ONE feature collection geoJSON array.

However, if there is a NULL POI or PATH then it still builds a geo_json feature with null (empty) values in the key value pairs.

I experimented with jsonb_remove_nulls() which removes only null key:value pairs from within a jsonb object, not an entirely empty object.

Many more experiments with COALESCE and FILTER did not yield the desired elimination of null objects.

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit


```
CREATE MATERIALIZED VIEW IF NOT EXISTS vm_31_get_page_poi_paths_json AS
WITH pages AS (
  SELECT page_id, page_type
  FROM t_allpages_metadata
  WHERE page_type IN ('a', 'e', 'b', 't', 'v', 'u')
  ORDER BY page_id
),
geo_json AS (
  SELECT
    p.page_id,
    p.page_type,
    (
      SELECT jsonb_build_object(
        'type', 'FeatureCollection',
        'features', (
          SELECT jsonb_agg(features.feature) FILTER (WHERE feature IS NOT NULL)
          FROM (
            -- Path features
      SELECT gpath.color,
      CASE
      WHEN gpath.color IS NOT NULL THEN
        jsonb_build_object(
                'type', 'Feature',
                'properties', jsonb_build_object(
                  'color', gpath.color
                ),
                'geometry', ST_AsGeoJSON(gpath.geo_data)::jsonb
              )
      END
      AS feature
```

```
      FROM t_allpages_metadata md
            JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
            LEFT JOIN t_geodata_path gpath ON gpath.page_id = md.page_id
            WHERE
              (md.page_type IN ('a', 'e', 'b', 't', 'v') AND md.page_status = TRUE AND md.page_id =
p.page_id)
              OR
              (p.page_type IN ('b', 't', 'v') AND md.page_id IN (
                SELECT nvl.page_id
                FROM t_nav_loc nvl
                WHERE nvl.menu_page_id = p.page_id
              ))
              OR
              (p.page_type = 'u' AND md.page_id IN (
                SELECT ntm.page_id
                FROM t_nav_tme ntm
                WHERE ntm.menu_page_id = p.page_id
              ))


      UNION ALL
```

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit


```
 -- POI features
SELECT gpoi.color,
CASE
WHEN gpoi.color IS NOT NULL THEN
  jsonb_build_object(
          'type', 'Feature',
          'properties', jsonb_build_object(
            'pageId', md.page_id,
            'img', md.page_thumbwebp,
            'geoTitle', gpoi.geo_title,
            'uri', RTRIM(uhn.urlparam0 || '/' || uhn.urlparam1 || '/' || uhn.urlparam2, '/'),
            'color', gpoi.color
          ),
          'geometry', ST_AsGeoJSON(gpoi.geo_data)::jsonb
        )
END
AS feature
      FROM t_allpages_metadata md
      JOIN t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
      LEFT JOIN t_geodata_poi gpoi ON gpoi.page_id = md.page_id
      WHERE
        (md.page_type IN ('a', 'e') AND md.page_status = TRUE AND md.page_id = p.page_id)
        OR
        (p.page_type IN ('b', 't', 'v') AND md.page_id IN (
          SELECT nlo.page_id
          FROM t_nav_loc nlo
          WHERE nlo.menu_page_id = p.page_id
        ))
        OR
        (p.page_type = 'u' AND md.page_id IN (
```

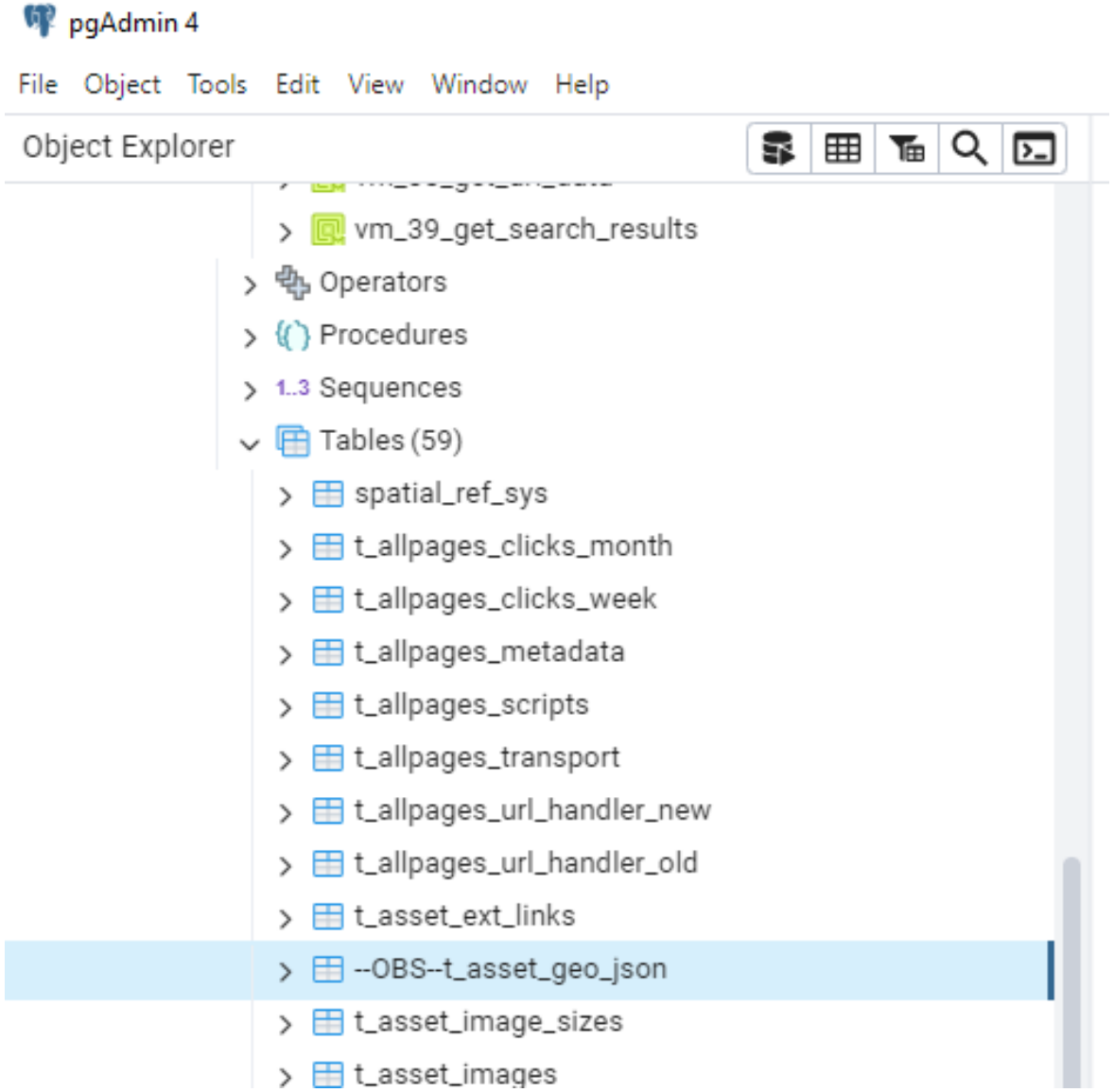
```
                SELECT ntm.page_id
                FROM t_nav_tme ntm
                WHERE ntm.menu_page_id = p.page_id
              ))

          ) AS features
        )
      )
    ) AS geo_json
   FROM pages p

)
SELECT * FROM geo_json
ORDER BY page_id
WITH DATA;
-- CREATE INDEX
CREATE UNIQUE INDEX idx_vm_get_page_poi_paths_json ON vm_31_get_page_poi_paths_json (page_id);
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY vm_31_get_page_poi_paths_json;
-- SEELCT FROM MV
SELECT * FROM vm_31_get_page_poi_paths_json;
```

Test 9. FINAL TESTED WORKING VERSION

I am using a case statement before the creation of the geo-json feature to handle the undesired nulls.

Author: Elliott Ward Farmer

8.6 GEO-JSON PostGreSQL MV revisit



After writing a very complex 100 line SQL query to build the geo-json column in the MV for each page from the raw tables, the workaround table I created earlier now becomes redundant.

AS part of a soft deletion, I have marked it as –OBS– (obsolete) and I will drop it later once I have a week or so working with the new MV query.

Author: Elliott Ward Farmer

9. Updatable Views

Author: Elliott Ward Farmer

9.1 Why Updatable Views?

As a security feature with switching from MySQL to PostgreSQL, I want to separate out the user access from administrator and content creator access by restricting user access (The web app) from the underlying base tables and only permit access to the data via views.

As the click counters are update queries, I will use an updateable view. There is no speed benefit to doing this because a UV is a standard view. In essence the web app will update to the UV which will then cascade the changes to the underlying base table.

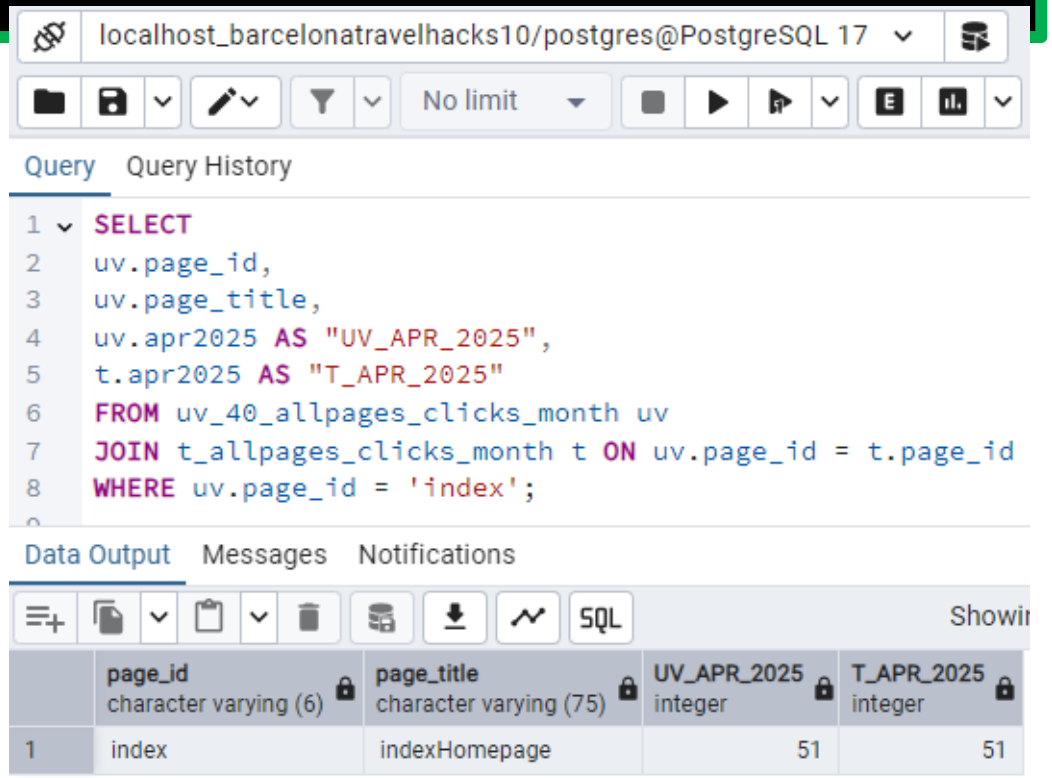
I will later combine the UV with column level security.

9.2 Create a UV for month clicks


```sql
-- 41 - Updatable view to modify MONTH & YEAR clicks
CREATE VIEW uv_40_allpages_clicks_month
AS
SELECT
  page_id,
  page_title,
  jan2025,
  feb2025,
  mar2025,
  apr2025,
  may2025,
  jun2025,
  jul2025,
  aug2025,
  sep2025,
  oct2025,
  nov2025,
  dec2025
FROM
  t_allpages_clicks_month
ORDER BY page_id
-- SEELCT FROM UV
SELECT * FROM uv_40_allpages_clicks_month;
```

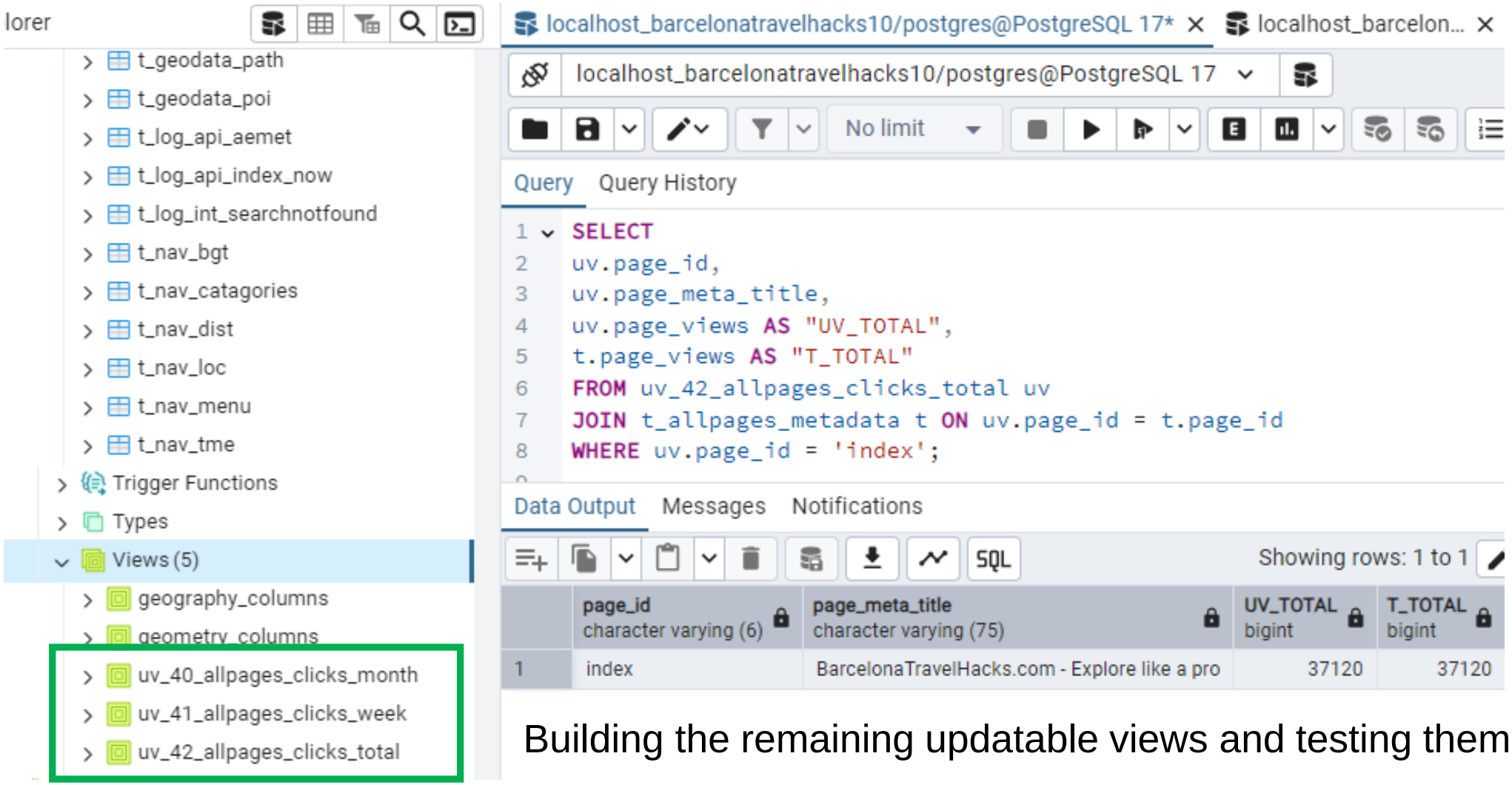
```php
// Mar-2025 - pgSQL: MV Query //
    public function pgSQL_increment_month_year_clicks($pageId) {
        $column = strtolower(date('MY'));
        $sql = "UPDATE uv_40_allpages_clicks_month
                SET $column = $column+1
                 WHERE uv_40_allpages_clicks_month.page_id = :page_id";
        $stmt = $this->conPgSQL->prepare($sql);
        $stmt->execute(array(':page_id' => $pageId));
    }
```

A function to update to the UV and verifying that the base table is also being updated from the UV.



Author: Elliott Ward Farmer

9.3 Create more UV for click counters



Building the remaining updatable views and testing them.

Author: Elliott Ward Farmer

9.4 Page Clicks by timezoneTZ & Lang


```php
<?php

class pgSQL_language {

  public function __construct($conPgSQL, $pageId) {
    $this->conPgSQL = $conPgSQL;
    $this->pageId = $pageId;
  }

  Public function pgSQL_LangHandler() {

    $PrefLang = $_SERVER["HTTP_ACCEPT_LANGUAGE"];

    $langs = [];
    $weightedLocales = [];
      foreach (explode(',', $PrefLang) as $loc) {
        $locParts = explode(';', $loc);
        $lang = substr($locParts[0],0,2);
        $weightedLocale = ['loc' => substr($locParts[0],0,5)];
        if (count($locParts) == 2) {
          $weightParts = explode('=', $locParts[1]);
          $weightedLocale['q'] = $weightParts[1];
        } else {
          $weightedLocale['q'] = 1;
        }
        $weightedLocales[] = $weightedLocale;
        $langs[] = $lang;
        $langs = array_unique($langs);
      }

      if (empty($weightedLocales)) {
        $weightedLocales = [
          'loc' => 'n/a',
          'q' => 1
        ];
      }
```

```php
      array_multisort(array_column($weightedLocales, 'q'), SORT_DESC, $weightedLocales);
      $pageId = "'".$this->pageId."'";
      $lang1 = array_key_exists(0, $langs) ? "'".$langs[0]."'" : 'NULL';
      $lang2 = array_key_exists(1, $langs) ? "'".$langs[1]."'" : 'NULL';
      $lang3 = array_key_exists(2, $langs) ? "'".$langs[2]."'" : 'NULL';
      $lang4 = array_key_exists(3, $langs) ? "'".$langs[3]."'" : 'NULL';
      $lang5 = array_key_exists(4, $langs) ? "'".$langs[4]."'" : 'NULL';
      $json = "'".json_encode($weightedLocales)."'"; ;

      $sql = "INSERT INTO uv_43_allpages_clicks_lang
              (page_id, lang1, lang2, lang3, lang4, lang5, lang_json)
              VALUES
              ($pageId, $lang1, $lang2, $lang3, $lang4, $lang5, $json)";
      $stmt = $this->conPgSQL->prepare($sql);
      $stmt->execute();
    }
  }
?>
```

Future development of the website is to rollout content in multiple languages. To achieve this I want a click counter for pages and desired user languages.

I am leveraging the HTTP_ACCEPT_LANGUAGE header to record what pages are viewed, when and the user desired languages for later analysis.

Author: Elliott Ward Farmer

9.5 base table & MV page clicks & language


```
CREATE TABLE t_allpages_clicks_lang (
id BIGSERIAL PRIMARY KEY,
page_id VARCHAR(6),
datetime timestamptz DEFAULT NOW(),
lang1 VARCHAR(2),
lang2 VARCHAR(2),
lang3 VARCHAR(2),
lang4 VARCHAR(2),
lang5 VARCHAR(2),
lang_json JSONB)
--------
CREATE VIEW uv_43_allpages_clicks_lang
AS
SELECT
id,
page_id,
datetime,
lang1,
lang2,
lang3,
lang4,
lang5,
lang_json
FROM t_allpages_clicks_lang
ORDER BY page_id
-- SEELCT FROM UV
SELECT * FROM uv_43_allpages_clicks_lang;
```

This page click counter will eventually replace the week and yearMonth click counters. I can perform powerful analysis using PostgreSQL windows functions and use partition to break the table down by weeks or months.

This will lead onto using this statistical data to order content on a popularity and seasonal basis which is especially important for the home page.

	id [PK] big	page_id character	datetime timestamp with time zone	lang1 charact	lang2 charact	lang3 charact	lang4 charact	lang5 charact	lang_json jsonb
1	1	index	2025-04-09 21:38:23.242755+02	en	ca	[null]	[null]	[null]	[{"q": 1, "loc": "en-GB"}, {"q": "0.9", "loc": "en"}, {"q": "0.8", "loc": "ca"}]
2	2	index	2025-04-09 21:38:55.530768+02	en	ca	[null]	[null]	[null]	[{"q": 1, "loc": "en-GB"}, {"q": "0.9", "loc": "en"}, {"q": "0.8", "loc": "ca"}]

Author: Elliott Ward Farmer

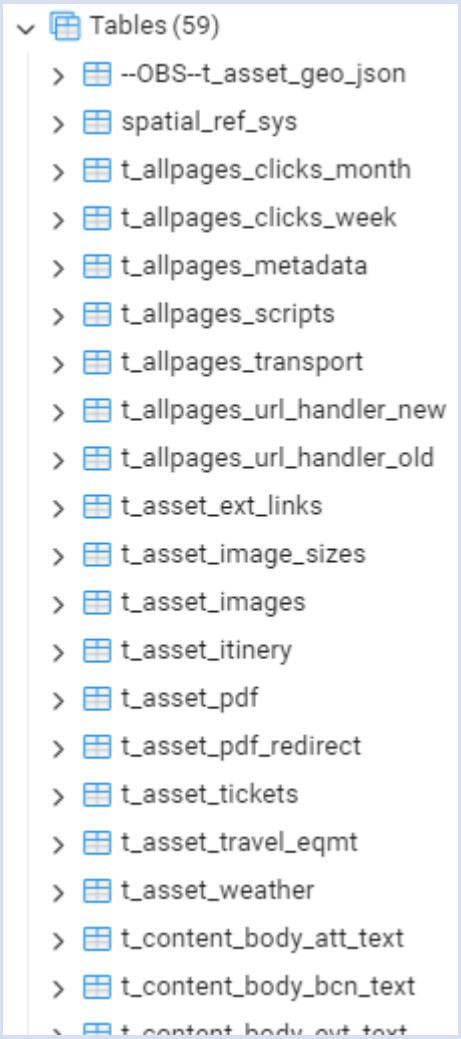
10. PostgreSQL Database Security

Author: Elliott Ward Farmer

10.1 BTH Database security overview

BarcelonaTravelHacks.com

BASE TABLES

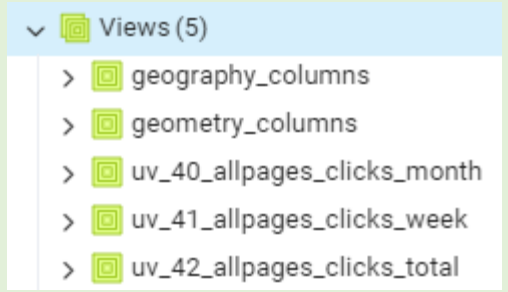


Only DBA's and Content Creators can modify base tables.

FULL CRUD

Web App can only access MVs (read only) and updatable Views.

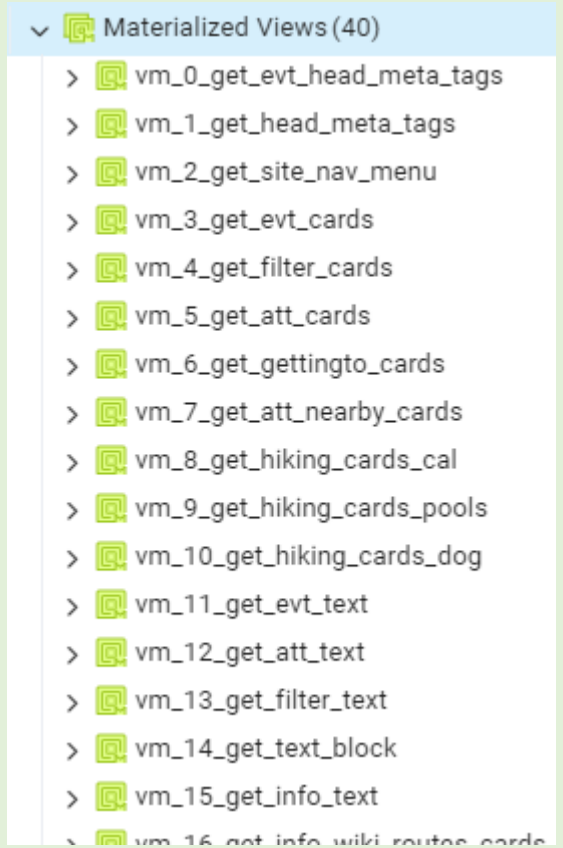
UPDATABLE VIEWS



U ONLY

R ONLY

MATERIALIZED VIEWS



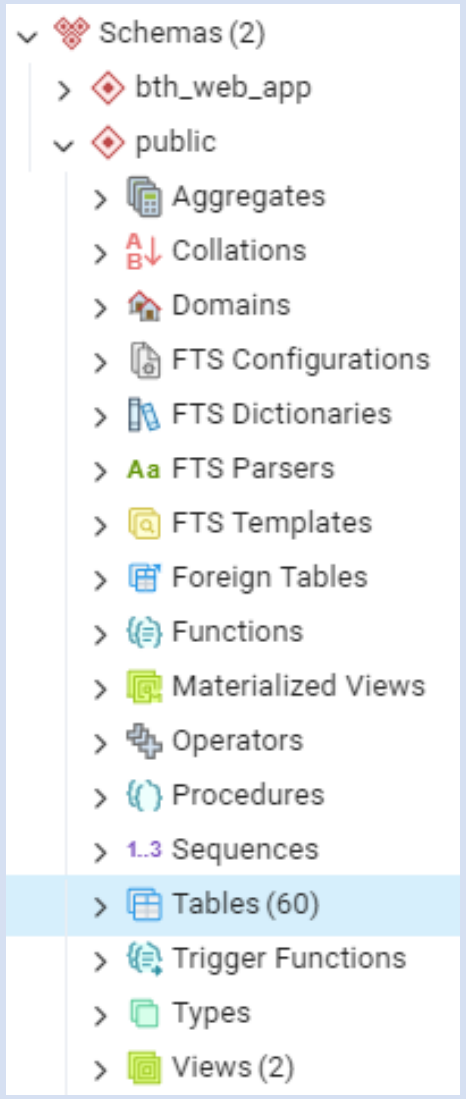
Author: Elliott Ward Farmer

10.2 BTH User Groups & Roles

Postgres & Superusers	Content Creators & Editors	Web App
Default PostgreSQL superuser role	FULL CRUD to all base table rows	R to MVs and U to UVs
Can modify tables & view structure **YES**	Can modify tables & view structure **NO**	Can modify tables & view structure **NO**
Revoke & grant privileges to other roles **YES**	Revoke & grant privileges to other roles **NO**	Revoke & grant privileges to other roles **NO**
Levels: Instance **SUPERUSER, CREATE DB, CREATEROLE, LOGIN, REPLICATION** Database **CREATE**, **CONNECT , TEMP** Schema **CREATE**, **USAGE** Table, **SELECT, INSERT, UPDATE, DELETE, REFERENCE, TRIGGER, TRUNCATE** Column, **SELECT, INSERT,DELETE,REFERENCE** Row, **RLS NO POLICY**	**Levels:** Instance **NO** Database **CONNECT ONLY** Schema **USAGE ONLY** Table, **SELECT, INSERT, UPDATE, DELETE** Column, **SELECT, INSERT,DELETE** Row, **RLS NO POLICY**	**Levels:** Instance **NO** Database **CONNECT ONLY** Schema **USAGE ONLY** Table, **SELECT, UPDATE,** Column, **SELECT, LIMITED UPDATE** Row, **RLS NO POLICY**

Author: Elliott Ward Farmer

10.3 Schemas



SCHEMA: public

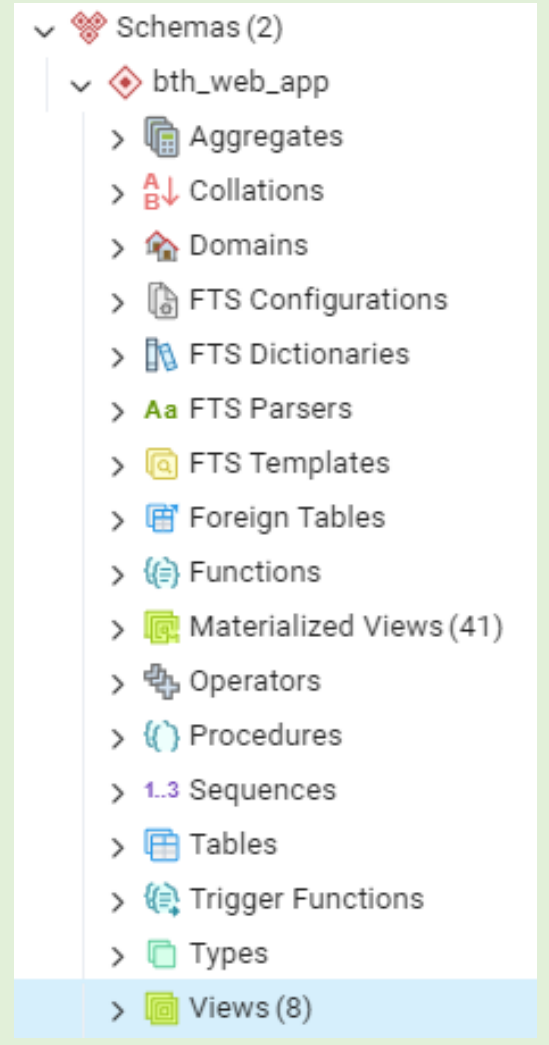
Only DBA's and Content Creators can modify base tables.

The base data tables are in the public schema

SCHEMA: bth_web_app

Web App can only access MVs (read only) and updatable Views.

This schema only contains MVs and UVs



Author: Elliott Ward Farmer

10.4 Setting Up bth_tools role


```
-- 5. Create user bth_tools within content editor role
CREATE ROLE bth_tools WITH
LOGIN PASSWORD 'pa55word'
NOSUPERUSER
NOREPLICATION
NOBYPASSRLS
CONNECTION LIMIT -1;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM bth_tools;
------------------------------------------------------------------
------------------------------------------------------------------
-- 7. grant permissions and privileges for bth_tools role
GRANT CONNECT ON DATABASE localhost_barcelonatravelhacks10 TO bth_tools;
GRANT USAGE ON SCHEMA "public" TO bth_tools;
-- ALTER ROLE content editor SET search path TO '$user', 'public',
'bth_web_app';
GRANT SELECT ON ALL TABLES IN SCHEMA public TO bth_tools;
GRANT UPDATE ON ALL TABLES IN SCHEMA public TO bth_tools;
GRANT INSERT ON ALL TABLES IN SCHEMA public TO bth_tools;
GRANT DELETE ON ALL TABLES IN SCHEMA public TO bth_tools;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO bth_tools;
```

Bth_tools is an administrator account that is used to alter base table data.

Author: Elliott Ward Farmer

10.5 Setting Up bth_web_app role


```
-- 2. Create role for web app
CREATE ROLE bth_webapp WITH
LOGIN PASSWORD 'testpw'
NOSUPERUSER
NOREPLICATION
NOBYPASSRLS
CONNECTION LIMIT -1;
GRANT CONNECT ON DATABASE localhost_barcelonatravelhacks10 TO bth_webapp;
GRANT USAGE ON SCHEMA "bth_web_app" TO bth_webapp;
ALTER ROLE bth_webapp SET search_path TO bth_web_app, '$user';
------------------------------------------------------------------------
------------------------------------------------------------------------
-- 3. revoke all privileges on the public schema
REVOKE ALL ON ALL TABLES IN SCHEMA bth_web_app FROM bth_webapp;
------------------------------------------------------------------------
------------------------------------------------------------------------
-- 4. Grant relevant privileges for MVs and UVs for web app role
GRANT SELECT ON bth_web_app.vm_0_get_evt_head_meta_tags TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_1_get_head_meta_tags TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_2_get_site_nav_menu TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_3_get_evt_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_4_get_filter_cards TO bth_webapp;
```

bth_webapp is the account that is used by the web app to access the MVs and UVs in the bth_web_app schema.

Author: Elliott Ward Farmer

10.5 Setting Up bth_web_app role


```
GRANT SELECT ON bth_web_app.vm_5_get_att_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_6_get_gettingto_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_7_get_att_nearby_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_8_get_hiking_cards_cal TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_9_get_hiking_cards_pools TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_10_get_hiking_cards_dog TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_11_get_evt_text TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_12_get_att_text TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_13_get_filter_text TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_14_get_text_block TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_15_get_info_text TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_16_get_info_wiki_routes_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_17_get_travel_bcn_text TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_18_get_ext_links TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_19_get_hero_gallery_index_by_week TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_20_get_hero_gallery_u0009_by_month TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_21_get_hero_gallery_u0022_by_month TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_22_get_hero_gallery_other TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_23_get_hist_gal TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_24_get_page_gallery_thumbs TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_25_get_page_gallery_player TO bth_webapp;
```

Author: Elliott Ward Farmer

10.5 Setting Up bth_web_app role


```
GRANT SELECT ON bth_web_app.vm_26_get_page_eqmt_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_27_get_page_prices TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_28_get_travel_bcn_prices TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_29_get_page_tickets TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_30_get_page_wikiloc_maps TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_31_get_page_poi_paths_json TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_32_get_page_weather_data TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_33_get_page_hotels TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_34_get_page_pdf_cards TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_35_get_search_pdf_redirects TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_36_get_pdf_docs TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_37_get_page_tags TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_38_get_uri_data TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_39_get_search_results TO bth_webapp;
GRANT UPDATE ON bth_web_app.uv_40_allpages_clicks_month TO bth_webapp;
GRANT UPDATE ON bth_web_app.uv_41_allpages_clicks_week TO bth_webapp;
GRANT UPDATE ON bth_web_app.uv_42_allpages_clicks_total TO bth_webapp;
GRANT INSERT ON bth_web_app.uv_43_allpages_clicks_lang TO bth_webapp;
GRANT USAGE, SELECT ON SEQUENCE t_allpages_clicks_lang_id_seq TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_40_allpages_clicks_month TO bth_webapp;
```

Author: Elliott Ward Farmer

10.5 Setting Up bth_web_app role


```
GRANT UPDATE ON bth_web_app.uv_41_allpages_clicks_week TO bth_webapp;
GRANT UPDATE ON bth_web_app.uv_42_allpages_clicks_total TO bth_webapp;
GRANT INSERT ON bth_web_app.uv_43_allpages_clicks_lang TO bth_webapp;
GRANT USAGE, SELECT ON SEQUENCE t_allpages_clicks_lang_id_seq TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_40_allpages_clicks_month TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_41_allpages_clicks_week TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_42_allpages_clicks_total TO bth_webapp;
GRANT INSERT ON bth_web_app.uv_44_nav_search_notfound TO bth_webapp;
GRANT USAGE, SELECT ON SEQUENCE t_nav_search_notfound_id_seq TO bth_webapp;
GRANT INSERT ON bth_web_app.uv_45_log_api_aemet TO bth_webapp;
GRANT USAGE, SELECT ON SEQUENCE t_log_aemet_api_id_seq TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_46_log_api_index_now TO bth_webapp;
GRANT INSERT ON bth_web_app.uv_46_log_api_index_now TO bth_webapp;
GRANT USAGE, SELECT ON SEQUENCE t_log_index_now_api_id_seq TO bth_webapp;
GRANT SELECT ON bth_web_app.uv_47_index_now_api_data TO bth_webapp;
GRANT SELECT ON bth_web_app.vm_48_get_forecast7_weather TO bth_webapp;
```

Note: Base tables that have auto-incrementing (sequence) id columns, for the bth_web_app UVs to be able to write (insert) a new column, it must have usage privileges on these sequences.

Author: Elliott Ward Farmer

10.6 Proving the Security


```
Server [localhost]:
Database [postgres]: localhost_barcelonatravelhacks10
Port [5432]:
Username [postgres]: bth_webapp
Password for user bth_webapp:
psql (17.2)

localhost_barcelonatravelhacks10=> select * from public.t_allpages_metadata;
ERROR:  permission denied for table t_allpages_metadata
localhost_barcelonatravelhacks10=> select * from t_allpages_metadata;
ERROR:  relation "t_allpages_metadata" does not exist
LINE 1: select * from t_allpages_metadata;
                      ^
localhost_barcelonatravelhacks10=> \d
                              List of relations
   Schema    |                  Name                   |       Type        |  Owner
-------------+-----------------------------------------+-------------------+----------
 bth_web_app | uv_40_allpages_clicks_month             | view              | postgres
 bth_web_app | uv_41_allpages_clicks_week              | view              | postgres
 bth_web_app | uv_42_allpages_clicks_total             | view              | postgres
 bth_web_app | uv_43_allpages_clicks_lang              | view              | postgres
-- More  --
```

Using a psql terminal, I have logged in as the webApp user.

I have verified that The webApp user can not access an of the base data tables and can only see the views as per the security design.

Author: Elliott Ward Farmer

11. PostgreSQL DB Migration Preparation

Author: Elliott Ward Farmer

11.1 Code to revert PostgreSQL

BarcelonaTravelHacks.com

index.php

```php
<?php

$activeDb = "PostgreSQL";
// $activeDb = "mySQL";

if ($activeDb === "PostgreSQL") {
    loadWebAppFromPgSQL();
}
if ($activeDb = "mySQL") {
    loadWebAppFromMySQL();
}

function loadWebAppFromPgSQL() {…
}

function loadWebAppFromMySQL() {…
}

require_once("App/Config/footer2.php");
?>
```

config.js

```js
// dev server
export const BASE_URL = "https://barcelonaTravelHacks.local";

// PgSQL
export const BASE_API_PATH = "https://barcelonaTravelHacks.local/App/Api/PgSQL/";
// MySQL
// export const BASE_API_PATH = "https://barcelonaTravelHacks.local/App/Api/MySQL/";

export const BASE_ICO_PATH = "https://barcelonaTravelHacks.local/assets/pvt/icons/";
export const BASE_ICO_PATH_SITE_NAV = "https://barcelonaTravelHacks.local/assets/pvt/icons/svgSiteNav/";
export const BASE_ICO_PATH_PAGE_NAV = "https://barcelonaTravelHacks.local/assets/pvt/icons/svgPageNav/";
export const BASE_ICO_PATH_INFO_NAV = "https://barcelonaTravelHacks.local/assets/pvt/icons/svgInfoNav/";
export const BASE_ICO_PATH_LIST_NAV = "https://barcelonaTravelHacks.local/assets/pvt/icons/svgListNav/";
export const BASE_ICO_PATH_TRVL_NAV = "https://barcelonaTravelHacks.local/assets/pvt/icons/svgTravelNav/";
let scnwidth = window.screen.width >= 704 ? "landscape" : "portrait";
export let ORIENTATION = scnwidth;

// production
// export const BASE_URL = "https://www.barcelonaTravelHacks.com";
// PgSQL
```

These controller files allow me to switch between PostgreSQL and MySQL databases by un commenting the $activeDb.

Author: Elliott Ward Farmer

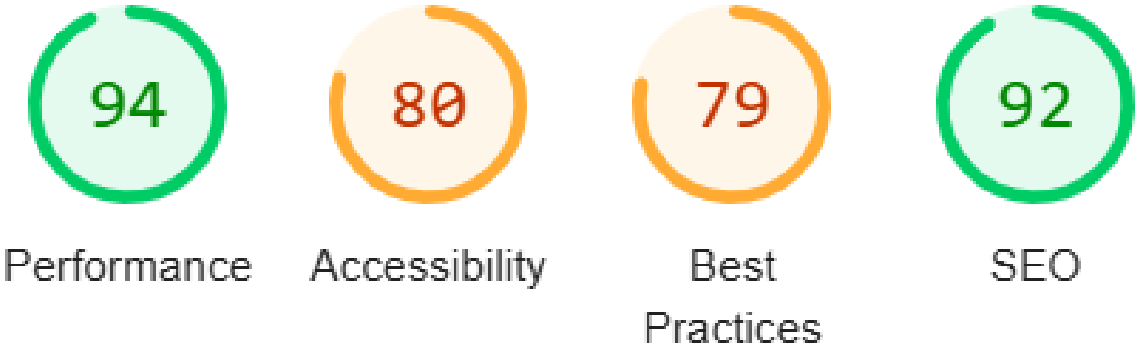
11.1 Testing PostgreSQL vs MySQL

BarcelonaTravelHacks.com

PostgreSQL

https://barcelonatravelhacks.local/travel/roman-tarragona-day-trip

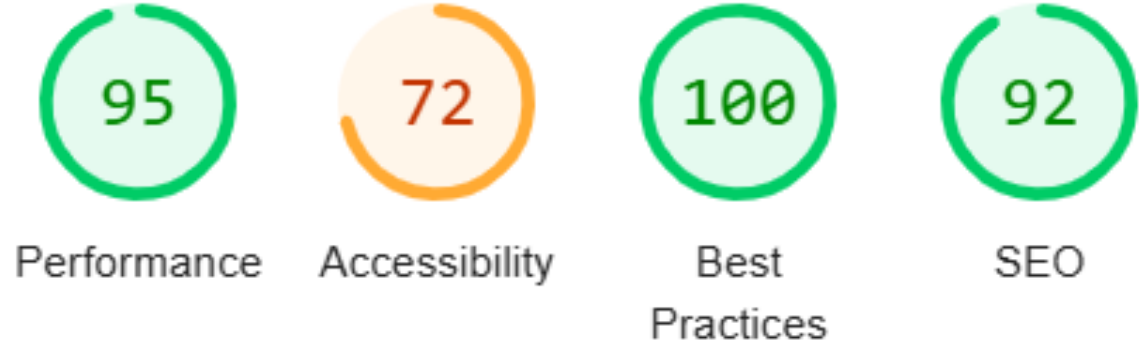
https://barcelonatravelhacks.local/



MySQL

https://barcelonatravelhacks.local/travel/roman-tarragona-day-trip

https://barcelonatravelhacks.local/



Author: Elliott Ward Farmer

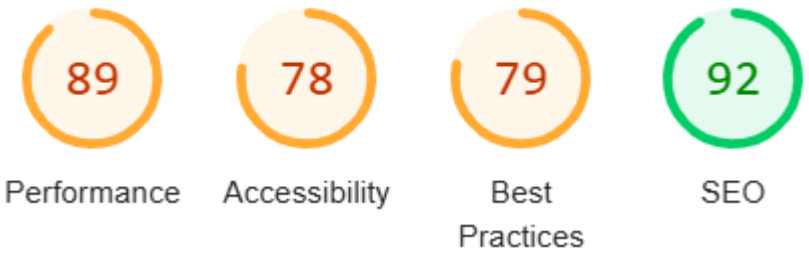
11.2 Testing PostgreSQL vs MySQL

BarcelonaTravelHacks.com

PostgreSQL

https://barcelonatravelhacks.local/travel/costa-dorada/tarragona-to-tamarit-and-altafulla-coastal-walk

https://barcelonatravelhacks.local/travel/pyrenees-bergueda



MySQL

https://barcelonatravelhacks.local/travel/costa-dorada/tarragona-to-tamarit-and-altafulla-coastal-walk

https://barcelonatravelhacks.local/travel/pyrenees-bergueda

I have picked several pages that I know have a lot of content and performed comparative lighthouse page loading speed tests for mobile pages. PostgreSQL is a fraction slower than MySQL but the advanced functionality of PostgreSQL makes it worth accepting this small performance degradation.

Author: Elliott Ward Farmer

11.3 Hosting Provider Research

Hello Namecheap support,

In Q2 or early Q3 of 2025 I would like to Migrate my website database from MySQL/phpMyAdmin to PostgreSQL/pgAdmin4. The new PostgreSQL database is built in postgreSQLv17, and at final testing stage. I am completing the final refinements as well as speed optimisations and it is almost ready to deploy.

I am reviewing my current hosting with Namecheap and have some technical questions about the database migration.

1. I am currently on the Stellar plan but would need to upgrade to the Stellar plus to have PostgreSQL in the cPanel but is PostgreSQL also available on a supersonic CDN plan? Note that for technical reasons I am ruling out a VPS solution at this stage.
https://www.namecheap.com/hosting/shared/
https://www.namecheap.com/supersonic-cdn/

2. I only have the option of PostgreSQL: 10.23?
https://www.namecheap.com/support/knowledgebase/article.aspx/129/22/what-version-of-the-software-is-used-on-your-servers/#:~:text=PostgreSQL%3A%2010.23%20(available%20for%20old%20Professional/Ultimate/Business%20SSD%20and%20new%20Stellar%20Plus/Stellar%20Business%20plans)

I ask because PostgreSQL: 10.23 is the final release of version 10 which is now end of life and my database is built on version 17, which for sure, will throw compatibility errors.
https://www.postgresql.org/about/news/postgresql-151-146-139-1213-1118-and-1023-released-2543/#:~:text=PostgreSQL%2010%20is%20EOL

3. Where are the knowledgebase documents and articles for PostgreSQL support? All I can find is this.
https://www.namecheap.com/support/knowledgebase/article.aspx/9797/29/cpanel-control-panel-overview/#Databases1

An example of the kind of documentation that I am looking for is this:
https://www.bluehost.com/help/article/postgresql-guide#CreatePostgreSQLUser

4. I would like to maintain a MySQL database and PostgreSQL database in parallel for about 3 to 6 months after the migration so that I can revert back to MySQL in the case that PostgreSQL is a catastrophic disaster or not meeting performance goals. Is this possible?

5. Are there any restrictions or barriers to PostgreSQL extensions with name cheap shared hosting? For example, I used ‘Application stack builder 4.2.2’ to add ‘PostGIS 3.5.2 bundle’ into the PostgreSQL system folder which is essential for my database.

Author: Elliott Ward Farmer

11.3 Hosting Provider Research

Thank you for contacting Namecheap Support!

We will answer the questions one-by-one:

1) Supersonic is a CDN service, so it does not provide database features. They are available with the hosting plan.
2) Yes, PostgreSQL **10.23 is the only available software version on the Shared hosting servers**.
3) Regretfully, we do not currently have documentation on how to utilize the PostgreSQL databases. We suggest referring to the official resources or to the cPanel knowledgebase**: https://docs.cpanel.net/**
4) As long as the databases are present in the MySQL Databases and the PostgreSQL Databases menus, it will be possible to connect the applications with them without issues.
5) From our check, there are **no restrictions on the PostgreSQL modules in the Shared hosting environment**. All PostgreSQL modules are allowed.

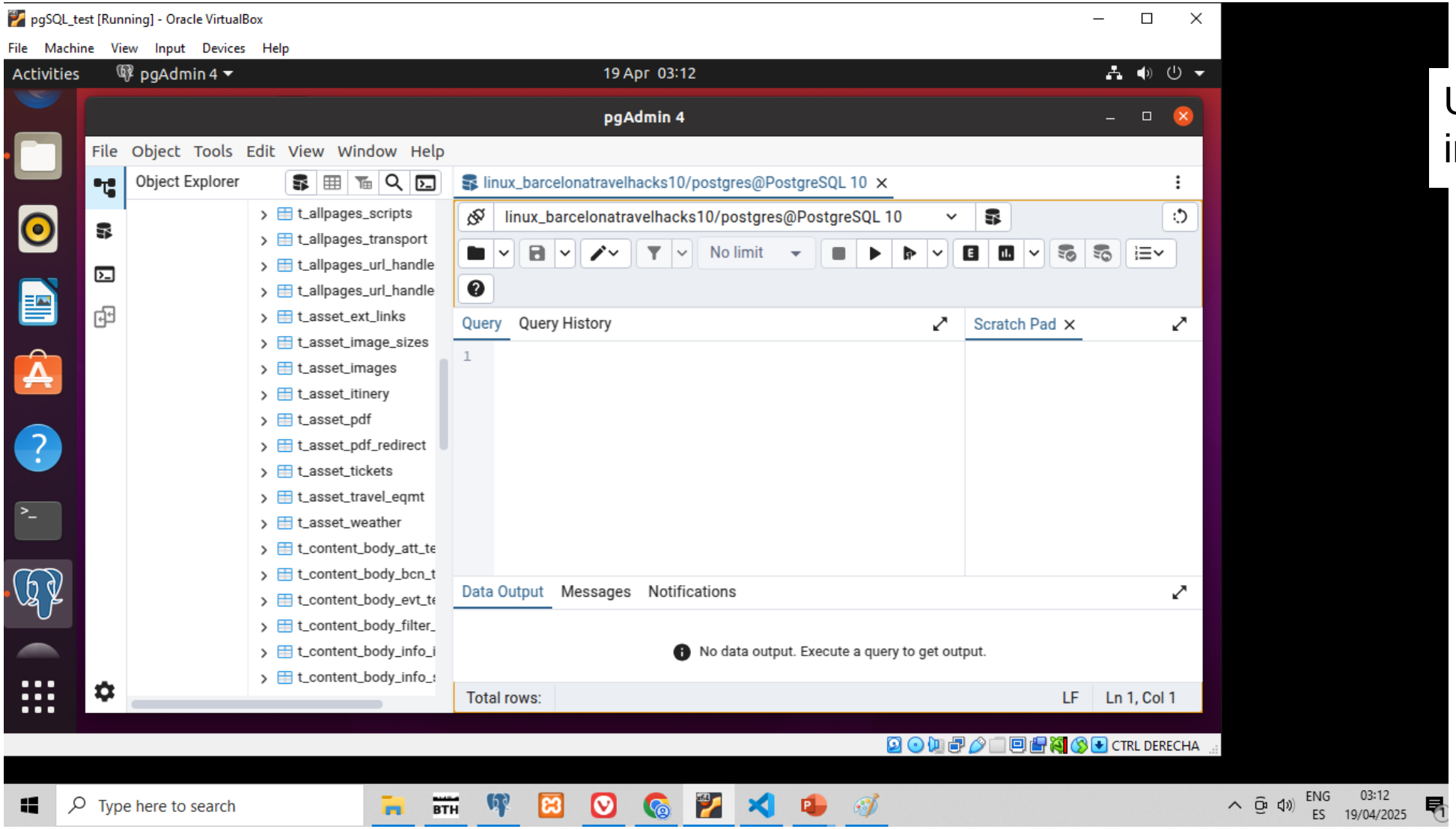
Please check it from your side and let us know if we can be of further assistance.

Looking forward to hearing from you.

Best regards,
Namecheap Team

Author: Elliott Ward Farmer

11.3 Testing PostgresSQL 10.23 in VM



Ubuntu 20.04 running in VirtualBox VM.

Author: Elliott Ward Farmer

11.4 Installing PostgresSQL in Linux

1. install postgres version 10.53: https://howtodojo.com/install-postgresql-10-ubuntu-20-04/

2. install pgadmin

2a. Download the repo from: https://www.pgadmin.org/download/pgadmin-4-apt/

2b. Install the public key for the repository (if not done previously): **`curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --dearmor -o /usr/share/keyrings/packages-pgadmin-org.gpg`**

2c. Create the repository configuration file: **`sudo sh -c 'echo "deb [signed-by=/usr/share/keyrings/packages-pgadmin-org.gpg] https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" > /etc/apt/sources.list.d/pgadmin4.list && apt update'`**

2d. Install for desktop mode only: **`sudo apt install pgadmin4-desktop`**

2e. Install for web mode only: **`sudo apt install pgadmin4-web`**

2f. Configure the webserver, if you installed pgadmin4-web: **`sudo /usr/pgadmin4/bin/setup-web.sh`**

Author: Elliott Ward Farmer

11.5 PostgresSQL in Linux dependencies

3. install postGis

3a. download ver10 package: https://download.osgeo.org/postgis/windows/pg10/

OR

3b. install direct from a repo: https://trac.osgeo.org/postgis/wiki/UsersWikiPostGIS3UbuntuPGSQLApt

```
sudo apt install postgresql-10-postgis-3
sudo apt install postgis
```

3c: Create extension: `CREATE EXTENSION postgis;`

4. create database

4a. Connect to PostgreSQL using psql: `sudo su - postgres`

4b. Create Database:

```
CREATE DATABASE linux_barcelonatravelhacks10 WITH OWNER "postgres" ENCODING 'UTF8'
LC_COLLATE = 'en_GB.utf8' LC_CTYPE = 'en_GB.utf8' TEMPLATE template0;
```

11.7 Migrating the Database

4. dump the windows pgsql database: https://www.a2hosting.com/kb/developer-corner/postgresql/import-and-export-a-postgresql-database/4

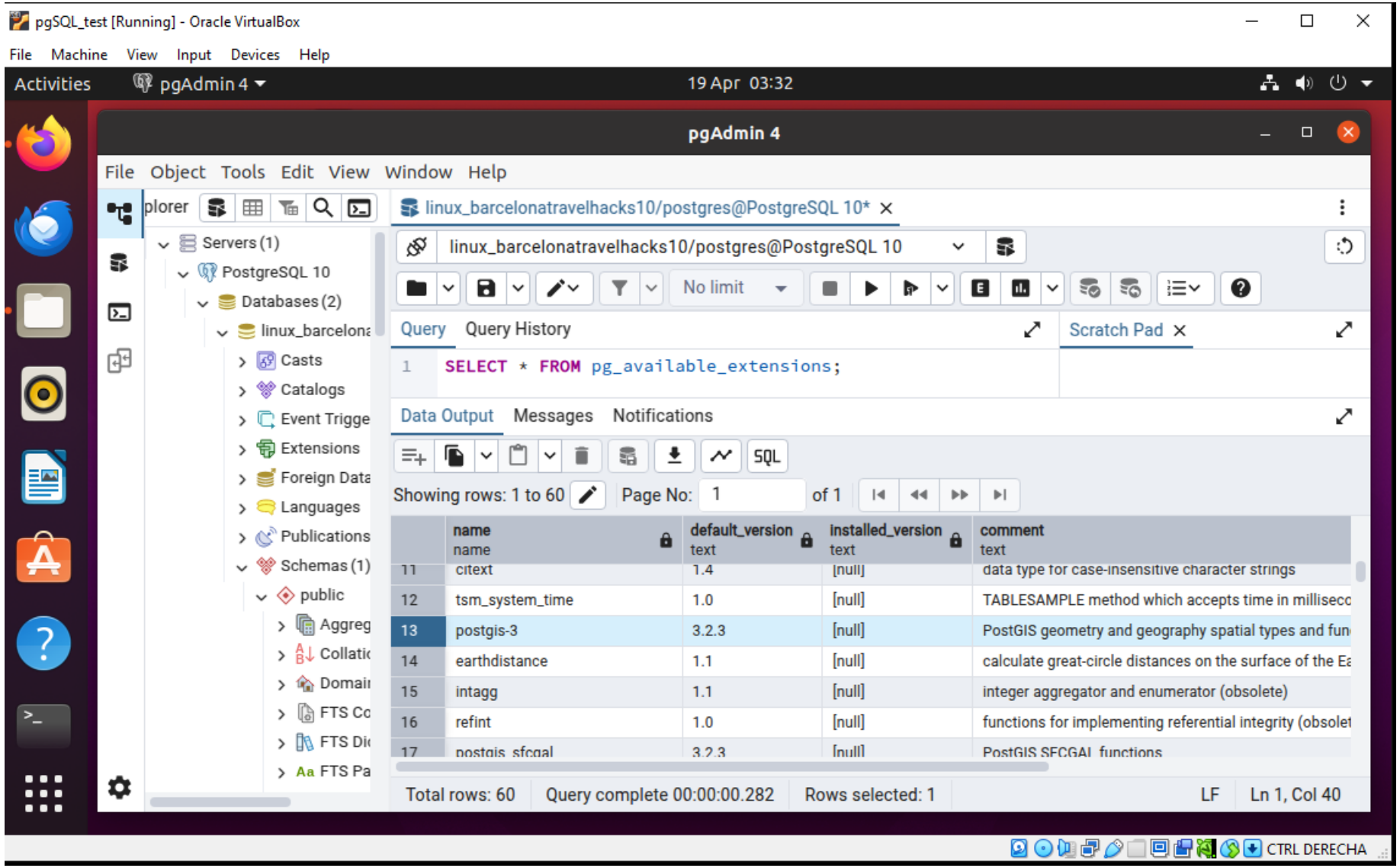
```
pg_dump -h localhost -U postgres -Fc localhost_barcelonatravelhacks10 >
C:/Users/ellio/Downloads/public_dbexport.sql
```

5. Import the windows created DB instance to the linux DB: https://www.a2hosting.com/kb/developer-corner/postgresql/import-and-export-a-postgresql-database/

```
pg_restore -h localhost -U postgres --clean -d linux_barcelonatravelhacks10
/home/ef/Desktop/public_dbexport.sql
```

Author: Elliott Ward Farmer

11.8 Verify Extensions and DB in Linux



Verify that postgis extension is correctly installed.

Do I need Hstore extension? No, technically not for this database because I am not storing any key:value pairs in the fields but I may need it in future iterations as the PostgreSQL DB design evolves.

For now I will omit hstore extension.

Author: Elliott Ward Farmer

11.9 Snagging the DB migration

Was the pg_dump and pg_restore process smooth?

No!

A number of issues arose:

1.PostgreSQL version 17 supports generated stored columns and this feature was introduced in PostgreSQL version 12. As my hosting provider only supports version 10, I cannot use generated columns. https://www.postgresql.org/docs/12/ddl-generated-columns.html

2. The pg_restore throws an error for the materialized views. This is because the schema does not exist so I will have to manually create the bth_web_app schema **before** performing the pg_restore operation.

```
pg_restore: error: could not execute query: ERROR:  schema "bth_web_app" does not exist
Command was: DROP VIEW bth_web_app.uv_46_log_api_index_now;
```

3. The **users and roles** also need to be manually created **before the pg_restore**. Once the restore operation has been performed, The revocation and granting of privileges needs to be performed.

11.10 Rollback of Generated Stored Columns


```
CREATE MATERIALIZED VIEW IF NOT EXISTS bth_web_app.vm_3_get_evt_cards AS
SELECT
md.page_id,
md.page_type,
md.page_thumbwebp,
md.transport_type,
md.page_meta_title,
md.page_meta_description,
md.page_added,
md.page_updated,
md.page_views,
cbe.evt_start,
cbe.evt_end,
rtrim(coalesce(uhn.urlparam0,'') || '/' || coalesce(uhn.urlparam1,'') || '/' || coalesce(uhn.urlparam2,''),'/') as uri,
array_agg(DISTINCT COALESCE(nl.menu_page_id)) as location,
array_agg(DISTINCT COALESCE(nt.menu_page_id)) as theme,
array_agg(DISTINCT COALESCE(nd.menu_page_id)) as distance,
(
SELECT budget::TEXT[] FROM
  (
    SELECT fkp.adult_travel_fare_dec, fkp.adult_ticket_dec, fkp.overnight, fkp.free_days,
    CASE
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight <= 1.14) AND (fkp.free_days IS NULL) THEN '{"y0001"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight <= 1.14) AND (fkp.free_days IS NOT NULL) THEN '{"y0001","y0003"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 1.14 AND 10) AND (fkp.free_days IS NULL) THEN '{"y0002"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 1.14 AND 10) AND (fkp.free_days IS NOT NULL) THEN '{"y0002","y0003"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 10.01 AND 25) AND (fkp.free_days IS NULL) THEN '{"y0004"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 10.01 AND 25) AND (fkp.free_days IS NOT NULL) THEN '{"y0004","y0003"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 25.01 AND 40) AND (fkp.free_days IS NULL) THEN '{"y0005"}'
```

Previously I was using the generated stored adult_total table column but I can deprecate this column in the table and just use the columns to calculate in the materialized view creation.

Author: Elliott Ward Farmer

11.10 Rollback of Generated Stored Columns


```
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 25.01 AND 40) AND (fkp.free_days IS NULL) THEN '{"y0005"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 25.01 AND 40) AND (fkp.free_days IS NOT NULL) THEN '{"y0005","y0003"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 40.01 AND 60) AND (fkp.free_days IS NULL) THEN '{"y0006"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 40.01 AND 60) AND (fkp.free_days IS NOT NULL) THEN '{"y0006","y0003"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 60.01 AND 80) AND (fkp.free_days IS NULL) THEN '{"y0007"}'
    WHEN (fkp.adult_travel_fare_dec+fkp.adult_ticket_dec+fkp.overnight BETWEEN 60.01 AND 80) AND (fkp.free_days IS NOT NULL) THEN '{"y0007","y0003"}'
    END AS budget
    FROM public.t_content_fk_prices fkp
    WHERE fkp.page_id = md.page_id
  ) as t
)
FROM public.t_allpages_metadata md
JOIN public.t_allpages_url_handler_new uhn ON uhn.page_id = md.page_id
JOIN public.t_content_body_evt_text cbe ON cbe.page_id = md.page_id
JOIN public.t_nav_loc nl ON nl.page_id = md.page_id
JOIN public.t_nav_tme nt ON nt.page_id = md.page_id
JOIN public.t_nav_dist nd ON nd.page_id = md.page_id
JOIN public.t_content_fk_prices fkp ON fkp.page_id = md.page_id
WHERE md.page_type = 'e'
AND md.page_status = true
GROUP BY md.page_id, uhn.urlparam0,uhn.urlparam1, uhn.urlparam2, cbe.evt_start, cbe.evt_end
ORDER BY md.page_id
-- REFRESH DATA IN MV
REFRESH MATERIALIZED VIEW CONCURRENTLY bth_web_app.vm_3_get_evt_cards;
-- SEELCT FROM MV
-- SELECT * FROM bth_web_app.vm_3_get_evt_cards;
```

Author: Elliott Ward Farmer

11.10 Rollback of Generated Stored Columns

t_content_fk_page_attevt_prices
page_id VARCHAR(6) PRIMARY KEY
adult_travel_fare_txt VARCHAR(20)
adult_travel_fare_dec NUMERIC(5,2)
child_travel_fare_txt VARCHAR(20)
child_travel_fare_dec NUMERIC(5,2)
adult_ticket_text VARCHAR(20)
adult_ticket_dec DECIMAL(5,2)
adult_ticket_cond VARCHAR(60)
child_ticket_text VARCHAR(20)
child_ticket_dec NUMERIC (5,2)
child_ticket_cond VARCHAR(60)
adult_total NUMERIC(5,2) GENERATED ALWAYS AS (adult_travel_fare_dec+adult_ticket_dec) STORED
child_total NUMERIC(5,2) GENERATED ALWAYS AS (child_travel_fare_dec+child_ticket_dec) STORED
group_family_ticket VARCHAR(150)
prices_notes VARCHAR(150)
free_days VARCHAR(150)

Generated Stored columns not supported in PostgreSQL 10. These columns are dropped and the values are calculated in the materialized views instead. The two below MVs were modified.

bth_web_app.vm_3_get_evt_cards

bth_web_app.vm_5_get_att_cards

Author: Elliott Ward Farmer

11.11 Testing PostgresSQL 10 in the VM

1. The Virtualbox VM has a bridged IP and is reachable from the windows machine.

```
ping 192.168.1.46
Pinging 192.168.1.46 with 32 bytes of data:
Reply from 192.168.1.46: bytes=32 time=1ms TTL=64
Reply from 192.168.1.46: bytes=32 time<1ms TTL=64
Reply from 192.168.1.46: bytes=32 time<1ms TTL=64
Reply from 192.168.1.46: bytes=32 time<1ms TTL=64

connection failed: SQLSTATE[08006] [7] could not connect to server: Connection refused
(0x0000274D/10061)
Is the server running on host "192.168.1.46" and accepting TCP/IP connections on port 5432?

PS C:\Users\ellio> Test-NetConnection 192.168.1.46 -p 5432
WARNING: TCP connect to (192.168.1.46 : 5432) failed
ComputerName            : 192.168.1.46
RemoteAddress           : 192.168.1.46
RemotePort              : 5432
InterfaceAlias          : Ethernet
SourceAddress           : 192.168.1.52
PingSucceeded           : True
PingReplyDetails (RTT) : 0 ms
TcpTestSucceeded        : False
```

Author: Elliott Ward Farmer

11.11 Testing PostgresSQL 10 in the VM


```
ef@pgSQLtest:~$ sudo netstat -naptu | grep 5432
[sudo] password for ef:
sudo: netstat: command not found
ef@pgSQLtest:~$ sudo apt install net-tools
...
ef@pgSQLtest:~$ sudo netstat -naptu | grep 5432
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      756/postgres
tcp        0      0 127.0.0.1:5432          127.0.0.1:46510         ESTABLISHED 3725/postgres: 10/m
tcp        0      0 127.0.0.1:46510         127.0.0.1:5432          ESTABLISHED 2906/python3
tcp        0      0 127.0.0.1:39070         127.0.0.1:5432          ESTABLISHED 2906/python3
tcp        0      0 127.0.0.1:5432          127.0.0.1:39060         ESTABLISHED 2993/postgres: 10/m
tcp        0      0 127.0.0.1:5432          127.0.0.1:39070         ESTABLISHED 2994/postgres: 10/m
tcp        0      0 127.0.0.1:39060         127.0.0.1:5432          ESTABLISHED 2906/python3

sudo ss -atnp | grep 5432
sudo ss -atp | grep postgres
ps -aux | grep postgres
```

The PostgreSQL server is running correctly but for some reason it is not allowing remote connections.

Author: Elliott Ward Farmer

11.12 Configuring PostgreSQL conns

By default PostgreSQL only allows connections from the local host so it needs to be configured to allow remote connections.
https://www.configserverfirewall.com/postgresql/postgresql-allow-remote-connections/

1. Find the path to the hba_file

`SHOW hba_file;` → `:/etc/postgresql/10/main/`

2. Navigate to this folder and open in nano text editor:

```
ef@pgSQLtest:/etc/postgresql/10/main$ sudo nano pg_hba.conf
```

Insert line **host all baffle 0.0.0.0/0 md5** to allow connections from all IPs. Q and save.

3. Open the postgresql.conf in nano

```
ef@pgSQLtest:/etc/postgresql/10/main$ sudo nano postgresql.conf
```

Change listning address from `#listen_addresses = 'localhost'` to **`listen_addresses = '*'`** Q and save.

```
ef@pgSQLtest:/etc/postgresql/10/main$ cd ../
ef@pgSQLtest:/etc/postgresql/10$ cd ../
ef@pgSQLtest:/etc/postgresql$ cd ../
ef@pgSQLtest:/etc$ cd ../
```

4. Restart postgreSQL

```
ef@pgSQLtest:/$ systemctl restart postgresql
```

Author: Elliott Ward Farmer

11.13 Modify php DB connection


```
# DB user connection
try {
  // $host = "localhost";
  $host = "192.168.1.46";
  // $dbname = "localhost_barcelonatravelhacks10";
  $dbname = "linux_barcelonatravelhacks10";
  $dbuser = "bth_webapp";
  $userpass = "testpw";

  $users = "pgsql:host=$host;port=5432;dbname=$dbname;user=$dbuser;password=$userpass";

  $conPgSQL = new PDO($users);
  $conPgSQL->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
```

Now, the webApp is connected to the PostgreSQL version 10 running in the virtualbox VM and page content has loaded correctly.

Conclusions:

1. Successfully proved that the database design can run on PostgreSQL version 10 with only slight design modifications.
2. Have a process and order of operations for migrating the database from the local dev environment to the production environment.
3. Benchmark speed tests do not show a noticeable speed difference between v10 and v17.

Next steps: **Migration and go live**.

Author: Elliott Ward Farmer