Montgomery Transit Accessibility to New Residential Areas

Standard Operating Procedures



By [Krishna Chaitanya Mummadi](mailto:mt68291@umbc.edu)

**Standard Operating Procedures**

**Introduction**

The purpose of this document is to cover the full lifecycle of the data tables in the transit\_access\_project database, detailing how they will be created, updated, validated and retired. The primary focus of the document will be procedures for updating the database during its active life.

# **Role-Based Access Control**

Role-based access control is utilized to ensure proper data usage and data security. Unified access control between the development and production environments ensures that only authorized users have access to the data they need, while reducing the risk of breaches.

Who Should Interact & How?

**Dataset Owners:** This individual will have ultimate authority to approve the creation, modification, and deletion of all schemas and tables in the database.

**Administrators:** Will be responsible for adding new tables, removing old ones, and adding/ removing fields in existing tables. Each table must be assigned an administrator of the same name. One person may be responsible for each administrator role.

**Editors:** Will be responsible for populating and updating the data in the tables contained within the database. A single Editor role can be created and given permissions across all tables within all schemas.

**Readers (users):** Will be using the data. These individuals will query the data to answer questions and perform institutional tasks (for example, creating a list of transit stations near to the newly developed areas). A single Reader role can be created and given ‘Select on’ permissions across all tables within all schemas.

**Stakeholder:** These individuals will be requesting summarized information and insights from the Readers (users). They will not have access to the database.

# Plan and Implement

**Actors**: Data Owner, Administrator, Editor, Reader (User)

**Description**: This section describes the process for creating a relational database for the Montgomery real estate and transit data.

**Steps**:

* The GES Team (administrator, editor, reader), will plan, how to create and implement a PostgreSQL database to best support the project goals.
* Listed research questions to solve the objectives
* Once I understand the subject matter information, I will work on:
  + Firstly carried out research on the counties where there is increase of population and development in maryland. After analyzing references, The team choose to work on Montgomery County, due to its proximity to the capital city, Washington D.C, and observed patterns of new development.
  + Identify state-government-maintained tax assessment data, which can be normalized to better answer questions. This task involves:
    - Identifying governing bodies that maintain information related to land and structure ownership, including tax assessment information. This information is maintained by the Maryland State Department of Taxation and Assessment (SDAT), and is distributed by the Maryland Department of Planning via the Maryland Open Data Portal:
      * [Montgomery County](https://opendata.maryland.gov/Business-and-Economy/Montgomery-County-Real-Property-Assessments-Hidden/kb22-is2w/about_data)
      * County boundaries will be included in the database for the display of summary statistics. These will be sourced from [Maryland iMAP](https://data.imap.maryland.gov/datasets/4c172f80b626490ea2cff7b699febedb_1/about).
      * Transit station datasets from the following sources:
        + [MARC train stations](https://opendata.maryland.gov/Transportation/MD-iMAP-Maryland-Transit-MARC-Train-Stops/qmkd-vkf9/about_data) (Maryland Transit Administration)
        + [Amtrak stations](https://opendata.maryland.gov/Transportation/MD-iMAP-Maryland-Transit-Amtrak-Rail-Stops/yiyd-dnxa/about_data) (Federal Railroad Administration)
        + [WMATA Metro stations](https://opendata.maryland.gov/Transportation/MD-iMAP-Maryland-Transit-WMATA-Metro-Stops/5ttq-ykbc/about_data) (Washington Metropolitan Area Transit Authority)
        + [MTA Bus stops](https://opendata.maryland.gov/Transportation/MD-iMAP-Maryland-Transit-MTA-Bus-Stops/j2zf-ej96/about_data) (Maryland Transit Administration)
        + [Montgomery County Ride On bus stops](https://data.imap.maryland.gov/datasets/maryland::maryland-local-transit-montgomery-county-ride-on-routes/about) (MCDOT via GTFS)
    - Ensuring data has been updated recently, to confirm that insights derived from these sources are relevant and accurate.
    - Proposing and implementing extract, transform, load (ETL) workflows which can be used to ingest data into the final database product.
      * An ETL logical model will be developed in conjunction with the implementation of initial ETL workflows. Please review the finalized version of the [ETL Logical Model](https://docs.google.com/document/d/1EEOONWZlpRRa67cm7Dln5GdU_pkXy5rGWUQzWgvzAAk/edit?tab=t.evygnoxyk6e2).
      * [ETL workflows](https://docs.google.com/document/d/1XAMeXJTOGz01fRmB_0e7TNRkC8Hy2aWjkjplLhBce9U/edit?tab=t.nf0oksv59b71) will be used to perform additional data cleaning and normalization, such as capitalization and date-time conversion, before insertion into the database.
    - Create a [data model](https://docs.google.com/spreadsheets/d/1Ewg8PTCsSB5PEE6FlmcX6Muojs2XkK3W/edit?gid=668135550#gid=668135550) to be shared with the Data Owner, to ensure that all relevant data is implemented before services are published, to avoid future downtime for schema updates. This task involves:
      * Identifying [fields](https://docs.google.com/spreadsheets/d/1Ewg8PTCsSB5PEE6FlmcX6Muojs2XkK3W/edit?gid=668135550#gid=668135550) that can be excluded based on existing data structures and client needs.
      * Proposing [domains](https://docs.google.com/spreadsheets/d/1Ewg8PTCsSB5PEE6FlmcX6Muojs2XkK3W/edit?gid=2064439153#gid=2064439153) to enforce data integrity and improve query accuracy.
      * Identifying relational constraints, including [primary-foreign key relationships](https://docs.google.com/spreadsheets/d/1Ewg8PTCsSB5PEE6FlmcX6Muojs2XkK3W/edit?gid=1252747421#gid=1252747421) and creating an entity-relationship diagram to ensure that data remains consistent across entities upon data update.
      * Identifying database users for each role which accesses the data, and assigning requisite privileges to each, to ensure that privileges are made available only to individuals that understand the database structure.
* Once the data model was finalized, it was implemented by the database administrator in a local PostgreSQL/PostGIS environment. See the **Scripts Appendix** for the SQL and R code used to complete the following tasks:
  + Using the arcgislayers R package to acquire raw spatial datasets from ArcGIS Feature Servers and Maryland’s Open Data Portal
  + Using SQL to create normalized tables and enforce spatial and logical constraints based on the data model
  + Using R (with sf, dplyr, and DBI) to clean, transform, and load datasets into their respective schemas (acplan, bndy, trans)
  + Creating domains and constraints to validate values such as land use codes, county names, and flags for new development
  + Assigning schema ownership to dedicated roles (e.g., acplan, trans, bndy) and granting appropriate privileges to ges\_editor and ges\_reader roles
* After the tables were populated, editor and reader users were granted access through local database credentials. This setup supports secure, role-based querying and allows editors to modify content while maintaining overall data integrity and access control.

**Manage (Routine/Periodic Tasks)  
Actors:** Administrator, Editor  
**Description:** This section outlines when and by whom data will be updated or maintained across schemas, particularly for parcels and transit-related tables.  
**Timeline:** Data updates should occur annually, after new parcel records have been released by the State Department of Assessments and Taxation (SDAT), and after updates to transit service layers have been published by the Maryland Transit Administration (MTA), WMATA, or local GTFS providers.

**Steps:**

* Either the Administrator or Editor may perform ETL updates, depending on team responsibilities and access.
* Updated datasets must be retrieved from authoritative sources, such as Maryland’s Open Data Portal and ArcGIS Feature Servers, identified during the Planning & Implementation phase.
* The Editor will process and insert the cleaned data into the appropriate tables in schemas such as acplan (parcels), trans (transit), and bndy (boundaries).
* All spatial datasets must be checked to ensure consistent SRID (EPSG:26985) before loading.
* If technical issues occur (e.g., database crash or connection loss), the Administrator will notify stakeholders, including the Data Owner.
* Backups must be performed immediately before and after each major data update and stored securely by the Administrator to preserve data integrity

**Update  
Actors:** Data Owner, Administrator, Editor  
**Description:** This section outlines the process for modifying the structure of the database, including the addition or removal of tables or schemas, and changes to fields within existing tables.

**Steps:**

* Future updates to the database structure may be necessary due to:
  + New data becoming available (requiring the addition of fields or tables).
  + Data becoming obsolete or deprecated (requiring the removal of fields or tables). See theDepreciation & Archivalsection before proceeding with deletions.
* When structural changes are needed, the responsible party will update the *Schema* Modification tab in the data model workbook.
* The database administrator and editor will update the *Physical*, *Relationships*, and Domains tabs in the Data Model to reflect all schema changes.
* A copy of the revised Data Model and a formal change log will be submitted to the Data Owner for review and approval.
* After receiving sign-off from the Data Owner, the Administrator will implement the updates as documented in the change log and Data Model.
* Following implementation, access will be restored to editors and readers via remote connections or updated backups.
* A copy of the change log will be archived in the project folder as part of the documentation.
* If changes are expected to interrupt service or require downtime, the GES team will coordinate a change control process, scheduling updates outside of business hours.

**Use & Sharing  
Actors:** Readers, Stakeholders  
**Description:** This section describes the process of providing analytics and spatial insights to stakeholders.  
**Stakeholder:** Montgomery County Planning Department

**Steps:**

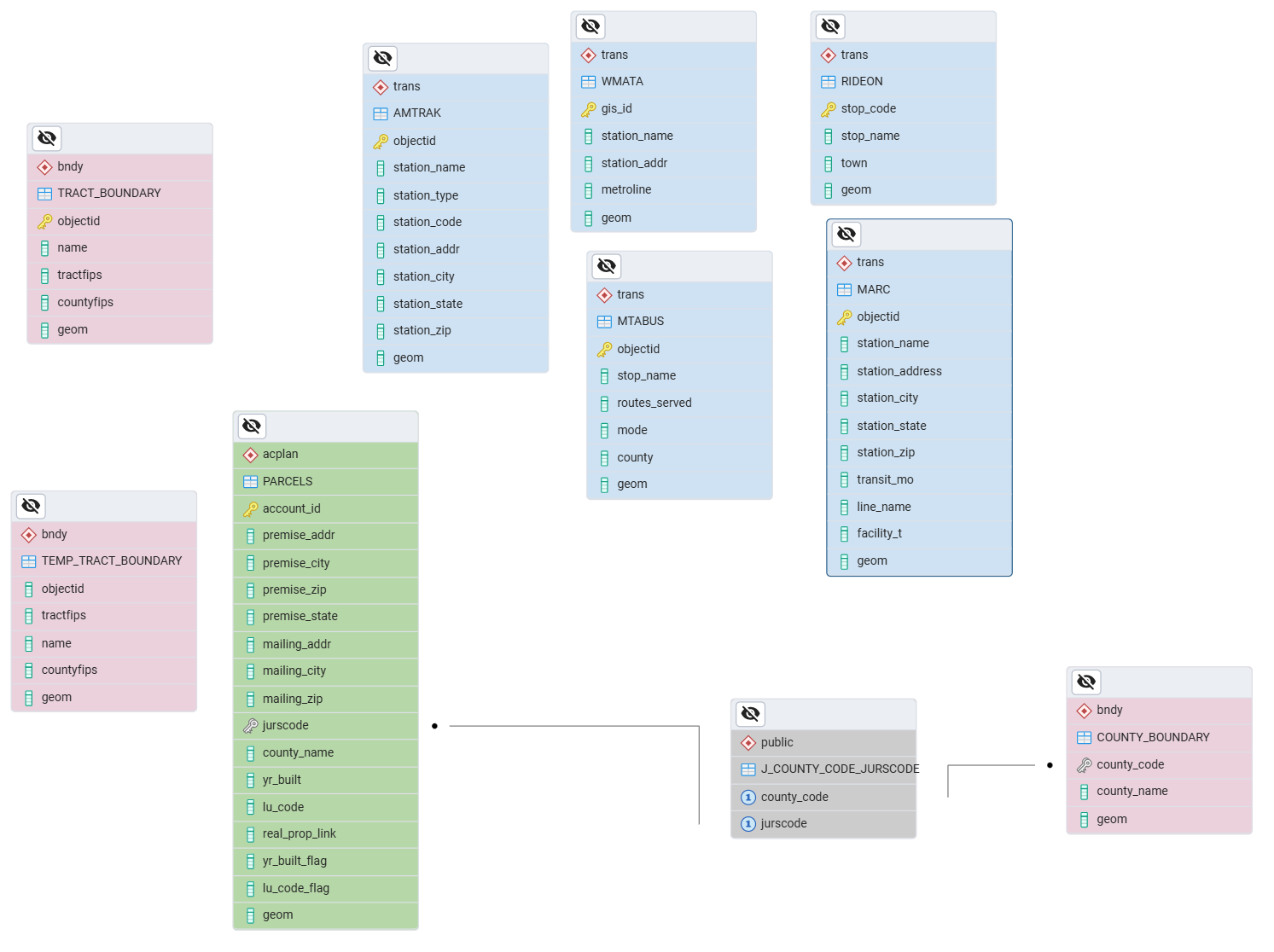
* On an annual and/or ad-hoc basis, stakeholders will be provided information and analysis compiled by the Reader(s).
* The following is a comprehensive list of information that should be provided (if Montgomery County Planning requests revisions, this list should be updated accordingly):
  + Count of residential parcels built in or after 2020 (based on yr\_built\_flag = 1)
  + Number of these parcels within 500 meters of a bus stop (MTA Bus, RideOn)
  + Number of these parcels within 1000 meters of a transit station (MARC, Amtrak, WMATA)
  + County-level breakdowns of transit accessibility for newly developed residential parcels
  + Identification of parcels with no access to nearby public transit within defined thresholds
* These outputs will be accompanied by visual maps and summary tables generated through spatial SQL queries and visualized via GIS software.
* Planning staff may request a data extract (.csv, .xlsx, or spatial file formats like GeoPackage) or a full database backup by contacting the Data Administrator.

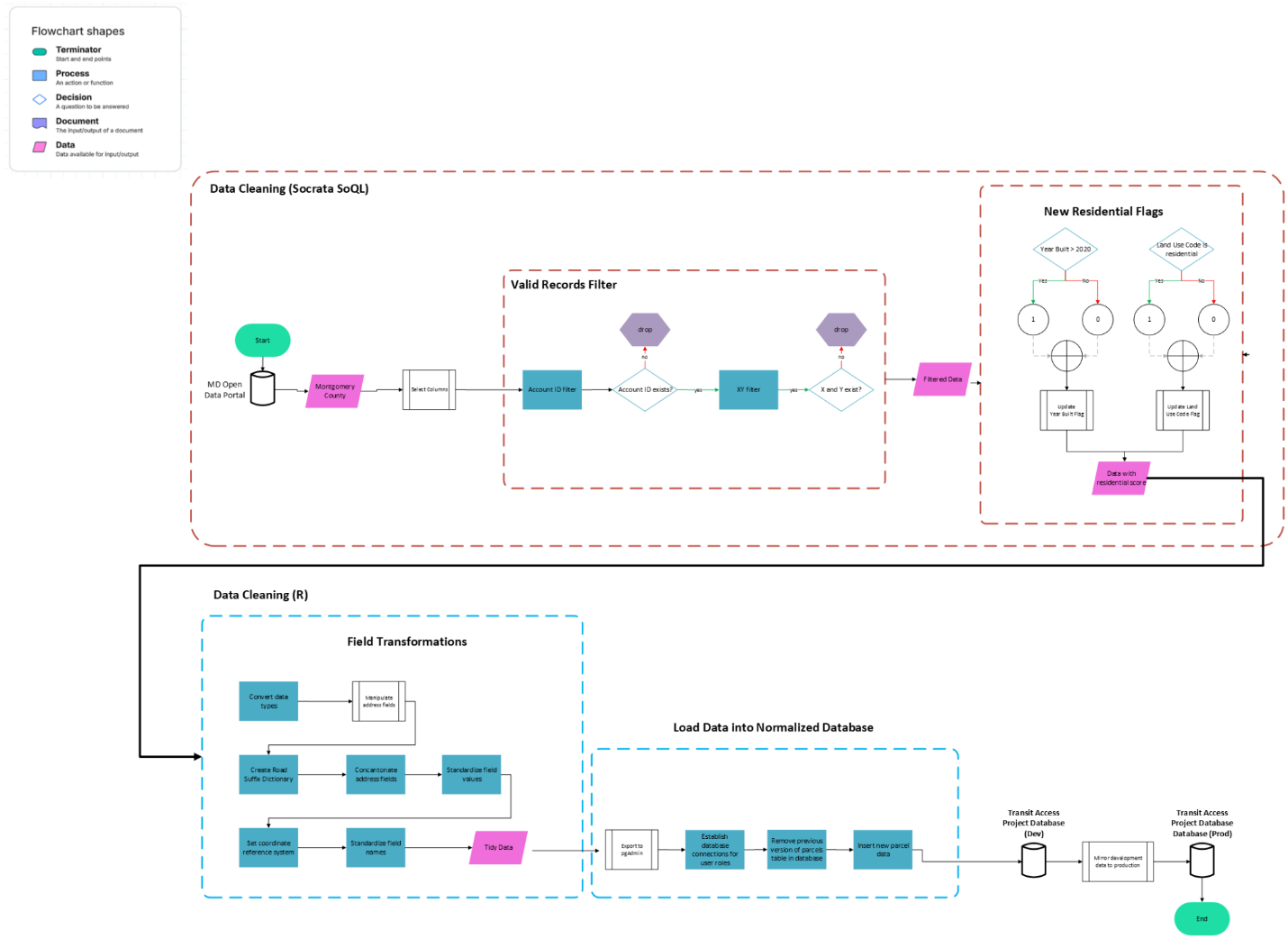
**Depreciation & Archival  
Actors:** Data Owner, Administrator  
**Description:** This section describes how and when a dataset or schema may be retired. The *Manage* section outlines conditions under which individual tables can be deprecated. The same steps apply if the entire database is to be archived.

**Steps:**

* Follow the steps described in the Update section for table or schema removal, with additional archival procedures listed below.
* Gather the following documentation and place it in an archive folder named with the deprecation date (serving as the effective end date of the data version):
  + A copy of the current data model, including all schemas (e.g., acplan, trans, bndy)
  + A full change log detailing ETL updates, schema changes, and administrative actions from project start to deprecation
  + [If retiring the entire database] A brief report explaining the rationale for deprecation, how to access archived data, and (if relevant) where and how to access its replacement
* All spatial and tabular data must be saved to a secure backup server or cloud storage approved by the Data Owner.
* Archived datasets must be retained for a minimum of **ten years**, or a duration defined by county/state policy, to ensure stakeholders can access historical parcel and transit data for future reference or analysis.

**ERD diagram**

****

**ETL Logic Model**

# **Scripts Appendix**

# **ETL Process**

Note that these are *template* scripts, for future reference on best practices. The scripts used by the GES team to create and administer the database can be found in the [db\_administrative\_scripts](https://drive.google.com/drive/u/0/folders/1d6ame6Hz_Fr9SMCSLtdMKrifR6helK8c) folder.

## Extract and Transform

* Actors: Database Administrator, Editor
* Description: Connect to source data API, and load in based on the desired columns and filters. Refer to the ETL scripts here.
* Steps:
  + Establish a connection to the API
  + Specify filters and desired columns
  + Create new columns
* R Packages & Key Functions:

### [arcgislayers](https://r.esri.com/arcgislayers/)

* + - simplifies accessing and managing data the ArcGIS Ecosystem

#### arc\_open()

* + - * Takes a URL to create a reference to a remote ArcGIS layer, server, or table.

#### arc\_select()

* + - * To query the feature layer object and return an sf object.
  + [base](https://search.r-project.org/R/refmans/base/html/00Index.html) R library

#### [as.character()](https://search.r-project.org/R/refmans/base/html/character.html)

* + - * Create or test for objects of type "character".

#### [as.Date()](https://search.r-project.org/R/refmans/base/html/as.Date.html)

* + - * Functions to convert between character representations and objects of class "Date" representing calendar dates.

#### [c()](https://search.r-project.org/R/refmans/base/html/c.html)

* + - * This is a generic function which combines its arguments.

#### [grepl()](https://search.r-project.org/R/refmans/base/html/grep.html)

* + - * Search for matches in argument pattern within each element of a character vector.

#### [is.na()](https://search.r-project.org/R/refmans/base/html/NA.html)

* + - * The generic function is.na indicates which elements are missing.

#### [read.csv()](https://www.rdocumentation.org/packages/COVID19/versions/2.0.3/topics/read.csv)

* + - * Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

#### [trimws()](https://search.r-project.org/R/refmans/base/html/trimws.html)

* + - * Remove leading and/or trailing whitespace from character strings.

#### [DBI](https://dbi.r-dbi.org/) to allow commands to be sent to database

#### [dbConnect()](https://dbi.r-dbi.org/reference/dbConnect.html)

* + - * Connect to a DBMS going through the appropriate authentication procedure.

#### [dbExecute()](https://www.rdocumentation.org/packages/DBI/versions/0.5-1/topics/dbExecute)

* + - * Call a stored procedure that performs data manipulation or other actions that do not return a result set.

#### [Id()](https://dbi.r-dbi.org/articles/spec.html?q=Id" \l "quote-identifiers)

* + - * The method can quote column names that contain special characters such as a space, a dot, a comma, or quotes used to mark strings or identifiers, if the database supports this.
  + [dplyr](https://dplyr.tidyverse.org/index.html) for advanced data cleaning

#### [across()](https://dplyr.tidyverse.org/news/index.html?q=across" \l "across-1-0-0)

* + - * To apply a function (or a set of functions) to a selection of columns.
    - [bind\_rows()](https://dplyr.tidyverse.org/reference/bind_rows.html)
      * Bind any number of data frames by row, making a longer result.

#### [case\_when()](https://dplyr.tidyverse.org/reference/case_when.html?q=case" \l "null)

* + - * This function allows you to vectorise multiple [if\_else()](https://dplyr.tidyverse.org/reference/if_else.html) statements. Each case is evaluated sequentially and the first match for each element determines the corresponding value in the output vector. If no cases match, the .default is used as a final "else" statement.

#### [mutate()](https://bookdown.org/yih_huynh/Guide-to-R-Book/mutate.html)

* + - * Adds new columns or modifies current variables in the dataset.

#### [relocate()](https://dplyr.tidyverse.org/articles/base.html?q=rename" \l "relocate-change-column-order)

* + - * Makes it easy to move a set of columns to a new position (by default, the front).

#### [rename()](https://dplyr.tidyverse.org/articles/base.html?q=rename" \l "rename-rename-variables-by-name)

* + - * Renames a column/variable.

#### [select()](https://bookdown.org/yih_huynh/Guide-to-R-Book/select.html)

#### Select only the columns (variables) that you want to see. Gets rid of all other columns. You can refer to the columns by the column position (first column) or by name. The order in which you list the column names/positions is the order that the columns will be displayed.

#### RPostgres to establish postgres driver

#### [RPostgres()](https://rpostgres.r-dbi.org/reference/Postgres.html)

* + - * Set drv = Postgres() to connect to a PostgreSQL(-ish) database.
  + [stringr](https://stringr.tidyverse.org/) for handling string data types

#### [str\_replace\_all()](https://stackoverflow.com/questions/74205866/str-replace-all-changes-all-of-the-string)

* + - * Replaces all matches with new text.

#### [sf](https://r-spatial.github.io/sf/) for spatial data operation

#### [st\_as\_sf()](https://r-spatial.github.io/sf/articles/sf2.html?q=st_as_sf" \l "dataset-and-layer-reading-or-creation-options)

* + - * Convert a foreign object to an sf object.

#### [st\_transform()](https://r-spatial.github.io/sf/reference/st_transform.html?q=st_Transform" \l "null)

* + - * Transform or convert coordinates of a simple feature.

#### [st\_write()](https://r-spatial.github.io/sf/reference/st_write.html)

* + - * Write simple features object to file or database.

# 

# **Database Design**

Note that these are *template* scripts, for future reference on best practices. The scripts used by the GES team to create and administer the database can be found in the [db\_administrative\_scripts](https://drive.google.com/drive/folders/1h5TzF8Wn6-rKmf3LQ8YDNbrgJGT8y7LZ?usp=drive_link) folder of the documentation.

## Database Set Up

* Actors: Database Owner, Project Manager
* Description: Set up databases for development and for production processes.
* Steps:
  + Establish a secure server connection.
  + Create database “dev” for data acquisition, updates, statistics, and spatial analysis.
  + Create database “prod” for stakeholder web experience application.
  + **\*Follow database naming conventions: names must be lower case to ensure proper database connections between Postgres and ArcPro.**
* SQL Commands:

-- Create a database

CREATE DATABASE database\_name;



## Spatially Enable Database

* Actors: Database Owner
* Description: Create an extension for postgis to enable spatial data types, indices, and functions in the database.
* Steps:
  + Check that the postgis supporting files are installed.
  + Create the extension and establish the database owner role.
  + **\*Ensure there are no duplicate extension names.**
  + **\*Must be called before creating tables.**
* Documentation:
  + <https://www.postgresql.org/docs/current/sql-createextension.html>
* SQL Commands:

-- Load an extension

-- (DO NOT ALTER THIS CODE)

CREATE EXTENSION postgis;

## Define Schemas

* Actors: Database Owner
* Description: Define schemas for data usage into spatial and non-spatial tables according to appropriate ISO categories, which were decided during the Planning phase.
* Steps:
  + Create schemas for all categories.
  + Use comments within SQL script to make notes.
* SQL Commands:

-- Create a schema, repeat for all schemas

-- (Replace schema\_name with appropriate ISO schema name)

CREATE SCHEMA schema\_name;



## Define User Roles And Privileges

* Actors: Database Owner, Project Manager
* Description: Define who has access to each database schema and how users can use the database according to permissions set by the database owner, determined by the database owner and project manager.
* Steps:
  + - Create administrator roles specific to each schema with password security.
      * Follow a user naming convention
    - **\*In ArcPro, Admin Users = Schema Owners**
    - **\*Ensure only the database owner is a “superuser”.**
  + Grant all privileges to each role.
  + Create editor and reader roles.
    - Follow a user naming convention.
  + Grant limited privileges to editor and reader roles.
    - Editors can insert, update and delete but not drop schemas.
    - Readers can only read or select data.
  + Set default permissions for future tables to ensure seamless permission granting for new database users (e.g. new employees).
  + When applicable, update SOP RBAC and [Shared Documentation](https://docs.google.com/document/d/1XAMeXJTOGz01fRmB_0e7TNRkC8Hy2aWjkjplLhBce9U/edit?tab=t.0#heading=h.sjby16ts2pkg) to reflect naming conventions and database changes.
* SQL Commands:
* Create database administrator role and assign all privileges

-- Create an Admin role, repeat for all schemas

-- (Replace 'admin' with unqiue schema admin name and 'password' with secure password)

CREATE ROLE admin WITH LOGIN PASSWORD 'password' NOSUPERUSER;

-- Grant Admin permissions, repeat for all schemas

-- (Replace 'schema\_name' with appropriate schema name and 'admin' with admin name)

GRANT ALL PRIVILEGES ON SCHEMA schema\_name TO admin;



* Create editor role and assign schema modification privileges

-- Create an Editor role

-- (Replace 'editor' with unique editor name and 'password' with secure password)

CREATE ROLE editor WITH LOGIN PASSWORD 'password' NOSUPERUSER;

-- Grant Editor permissions

-- (Replace 'editor' with appropriate user name amd 'schema\_names')

GRANT USAGE, CREATE ON SCHEMA schema\_name1,schema\_name2,schema\_name3 TO editor;

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA schema\_name1,schema\_name2,schema\_name3 TO editor;

-- Set default permissions for future tables

-- (Replace 'editor' with appropriate user name)

ALTER DEFAULT PRIVILEGES IN SCHEMA schema\_name1,schema\_name2,schema\_name3

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO editor;



* Create reader role and assign read-only access

-- Create a Reader role

-- (Replace 'reader' with appropriate user name and 'password' with secure password)

CREATE ROLE reader WITH LOGIN PASSWORD 'ReaderPass123' NOSUPERUSER;

-- Grant Reader permissions

-- (Replace 'reader' with appropriate user name and 'schema\_names')

GRANT USAGE OF SCHEMA schema\_name1,schema\_name2,schema\_name3 TO reader;

GRANT SELECT ON ALL TABLES IN SCHEMA schema\_name1,schema\_name2,schema\_name3 TO reader;

-- Set default permissions for future tables

-- (Replace 'reader' with appropriate user name and 'schema\_names')

ALTER DEFAULT PRIVILEGES IN SCHEMA schema\_name1,schema\_name2,schema\_name3

GRANT SELECT ON TABLES TO reader;



## Define Domains

* Actors: Database Owner
* Description: Using ‘Domains’ tab of the Data Model, create domains to ensure data integrity and normalization.
* SQL Commands:

-- Create domains to ensure data integrity and normalization

CREATE DOMAIN dDomainName VARCHAR(10)

CHECK (VALUE IN (

'CATEGORY1',

'CATEGORY2',

'CATEGORY3',

'CATEGORY4'

));



## Create A Temporary Table

* + Look up appropriate [data types for fields](https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/manage-postgresql/enterprise-geodatabase-limits.htm)

-- Create a temporary table to store raw data from a CSV

-- (Replace 'temp\_table', 'table\_id', fields, and DATATYPES with appropriate names)

CREATE TABLE temp\_table(table\_id DATATYPE PRIMARY KEY, field\_1 DATATYPE, field\_2 DATATYPE, field\_3 DATATYPE);

-- Import CSV file and populate temporary table

-- (Determine file path to csv and replace file\_path; replace comma delimiter with appropriate delimiter symbol if applicable)

COPY temp\_table FROM 'file\_path.csv' DELIMITER ',' CSV HEADER;



## Transfer Data To New Schema

* + Utilize a unique identifier to transfer unnormalized data from a temporary table into a new table with 1NF.

-- Create an empty table with no constraints

-- (Replace 'schema\_name', 'table1', 'table\_id', 'field\_names', and DATATYPES to reflect Physical Data Model)

CREATE TABLE schema\_name.table1 (table\_id DATATYPE PRIMARY KEY, field\_name1 DATATYPE, field\_name2 DATATYPE, field\_name3 DATATYPE);

-- Populate empty table with data from a temporary table

-- Make field names follow the same order between INSERT INTO and SELECT lines.

-- (Replace 'schema\_name', 'table1', 'table\_id', 'field\_names', and DATATYPES to reflect new table)

INSERT INTO schema\_name.table1 (table\_id, field\_name1, field\_name2, field\_name3)

SELECT temp\_table\_id, temp\_table\_field1, temp\_table\_field2, temp\_table\_field3

FROM temp\_table

## Create A Relational Table

* Establish referential integrity between tables using a foreign key

CREATE TABLE schema\_name.table2 (

table2\_id DATATYPE PRIMARY KEY,

field\_name1 DATATYPE NOT NULL, -- constraint: makes field mandatory

FOREIGN KEY (field\_name1) REFERENCES schema\_name.table1(table1\_id) ON DELETE CASCADE -- constraint: If a table1\_id is deleted, then delete associated records in table2

field\_name2 DATATYPE,

field\_name3 DATATYPE);

# ****

# **Data Exploration**

## Non-Spatial Queries

* Non-spatial queries used to establish data integrity.

-- Examine schema metadata

-- (Replace 'your\_table\_name' with appropriate table name)

SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = 'your\_table\_name';

-- View sample of data table; limit query results to first 10 records

-- (\* astrisk represents all fields)

-- (Replace 'table\_name' with appropriate table name)

SELECT \* FROM table\_name LIMIT 10;

-- Count the number of records in a table

-- (\* astrisk represents all fields)

-- (Replace 'table\_name' with appropriate table name)

SELECT COUNT(\*) FROM table\_name;

-- Identify missing data

-- IS NULL represents matching the record condition [NULL]

-- (Replace 'table\_id', 'table\_name', and 'data\_field' with appropriate names)

SELECT table\_id FROM table\_name WHERE data\_field IS NULL;

-- Identify all unique values in a given field

-- (Replace 'field\_name' and 'table\_name' with appropriate names)

SELECT DISTINCT field\_name FROM table\_name;

-- Identify duplicate field values

-- (Replace 'field\_name' and 'table\_name' with appropriate names)

SELECT field\_name, COUNT(\*) FROM table\_name GROUP BY field\_name HAVING COUNT(\*) >1;

-- Rank records according by a field from largest to smallest value

-- Limit query results if only interested in the largest values

-- Limits improve execution time

-- (Replace 'table\_id', 'table\_name', and 'data\_field' with appropriate names)

SELECT table\_id, data\_field FROM table\_name ORDER BY data\_field ORDER BY data\_field DESC LIMIT 10;

# **Database Management**

## Update Or Alter Data

* Actors:Administrator, Editor
* Description: Make alterations to a table to reflect metadata changes to existing features.
* SQL Commands:

-- Rename a column/constraint/table

ALTER TABLE table\_name ADD COLUMN column\_name DATATYPE;

-- Reassign table to a different schema

ALTER TABLE [IF EXISTS] table\_name

SET SCHEMA new\_schema;

-- Alter a column datatype and constraint

ALTER TABLE table\_name MODIFY field1 DATATYPE NOT NULL;

ALTER TABLE table\_name ALTER COLUMN field2 TYPE DATATYPE;

-- Add a new data constraint

ALTER TABLE table\_name ADD CHECK (field1 <> '');

ALTER TABLE table\_name ADD CONSTRAINT field2 UNIQUE (table\_id);

ALTER TABLE table\_name ADD FOREIGN KEY (table2\_id) REFERENCES table2;

-- Remove a constraint on a specific field

ALTER TABLE table\_name DROP CONSTRAINT field1;

ALTER TABLE table\_name ALTER COLUMN field2 DROP NOT NULL;

-- Find records that match a field condition and modify to match a new condition

UPDATE table\_name SET field1 = 'new\_condition' WHERE field1 = 'old\_condition'

-- Fine records that match a field condition and modify multiple fields to match new conditions

UPDATE table\_name SET field1 = #, field2 = #, field3 = # WHERE field1 > #;



## 

## Append Data

* Actors: Administrator, Editor
* Description: Add new records or fields to a table while maintaining a dataset.
* SQL Commands:

-- Add a new column to a table

ALTER TABLE table\_name ADD [COLUMN] column\_name;

-- Add a new row to a table

ALTER TABLE table\_name ADD

## Delete Data

* Actors: Editor
* Description: Removing records of existing features in a table
* SQL Commands:

-- Remove a column

ALTER TABLE table\_name DROP COLUMN column\_name;

-- Remove a column and all relationships referenced by a foreign key

ALTER TABLE table\_name DROP COLUMN column\_name CASCADE;

-- Remove a row

DELETE FROM table\_name WHERE field = 'some\_condition'

-- Remove a table

DROP TABLE table\_name;

-- Empty a table but keep columns

DELETE FROM table\_name;

-- Remove a schema

DROP SCHEMA schema\_name;

