



# Informe Técnico: Consolidación y Actualización de Edades de Construcción



## Contenido

<b>1. Objetivos.....</b>	<b>3</b>
Objetivo general .....	3
Objetivos específicos.....	3
<b>2. Metodología.....</b>	<b>4</b>
2.1. Importación de archivos Parquet a la base de datos .....	4
2.2. Estandarización y consolidación de edades .....	6
2.3. Clasificación de escenarios de emparejamiento .....	8
2.4. Integración de construcciones y unidades .....	8
2.6. Actualización de campos nulos en cr_unidadconstruccion .....	9
<b>3. Conclusiones.....</b>	<b>10</b>

## Índice de tablas

Tabla 1: Unión de todos los parquets asignandoles un nivel de calidad .....	7
Tabla 2: Descripción de Iso niveles de calidad .....	7
Tabla 3: Clasificación de escenarios de emparejamiento .....	8

## Índice de Ilustraciones

Ilustración 1: Creación de nueva conexión de base de datos DuckDB .....	4
Ilustración 2: Prámetros de DuckDB .....	5
Ilustración 3: Inserción de parámetros de conexión PostgreSQL a DuckDB.....	6
Ilustración 4: Creación de tabals en Postgres desde DuckDB por medio de parquets .....	6
Ilustración 5: Resultado final de la actualización de años de construcción .....	9

## 1. Objetivos

### Objetivo general

Consolidar la información de años de construcción provenientes de diferentes históricos a través de parquet, y actualizar los registros de la tabla cr\_unidadconstruccion donde este campo es nulo, con base en las edades estimadas de otras fuentes relacionadas.

### Objetivos específicos

- ✓ Importar los archivos históricos en formato Parquet a la base de datos PostgreSQL, asegurando la correcta estructura y correspondencia de campos.
- ✓ Unificar y estandarizar los datos geográficos y alfanuméricos de terrenos rurales y urbanos, consolidándolos en una estructura única, coherente y sin duplicidades.
- ✓ Identificar y clasificar los distintos escenarios de emparejamiento entre geometrías de terreno y predios, asignando etiquetas según el tipo de coincidencia (por ejemplo: cruce directo, sin matriz, sin predio relacionado, duplicados).
- ✓ Integrar construcciones y unidades aplicando reglas de emparejamiento progresivo, que consideren múltiples criterios como código predial, área, identificador, número de piso y relación espacial de geometría.
- ✓ Aplicar validaciones de calidad para detectar, documentar y corregir inconsistencias estructurales tales como códigos erróneos, geometrías inválidas, objetos sin identificador o sin vinculación predial.
- ✓ Consolidar la información migrada en las tablas finales cr\_terreno y cr\_unidadconstruccion, asegurando trazabilidad de origen y registro de observaciones que permitan su análisis y auditoría posterior.

## 2. Metodología

### 2.1. Importación de archivos Parquet a la base de datos

**Parquet** es un formato columnar muy eficiente para grandes volúmenes de datos [DuckDB](#), y existen múltiples vías para llevar ese contenido a PostgreSQL:

- **DuckDB** ofrece una ruta nativa con su extensión postgres y la función `read_parquet`.
- **Python/Pandas** permite un flujo muy directo mediante `pandas.read_parquet` y `DataFrame.to_sql`.
- **FDW** (Foreign Data Wrapper) integra Parquet como tablas foráneas dentro de PostgreSQL, ya sea con el plugin oficial o `pg_duckdb`.

Para este caso y por mayor facilidad se escoge la primera opción **DuckDB**, para eso el primer paso es configurar la base de datos.

#### 2.1.1 Conexión a Duckdb

En primero lugar se agrega la base de datos eligiendo la opción DuckDB

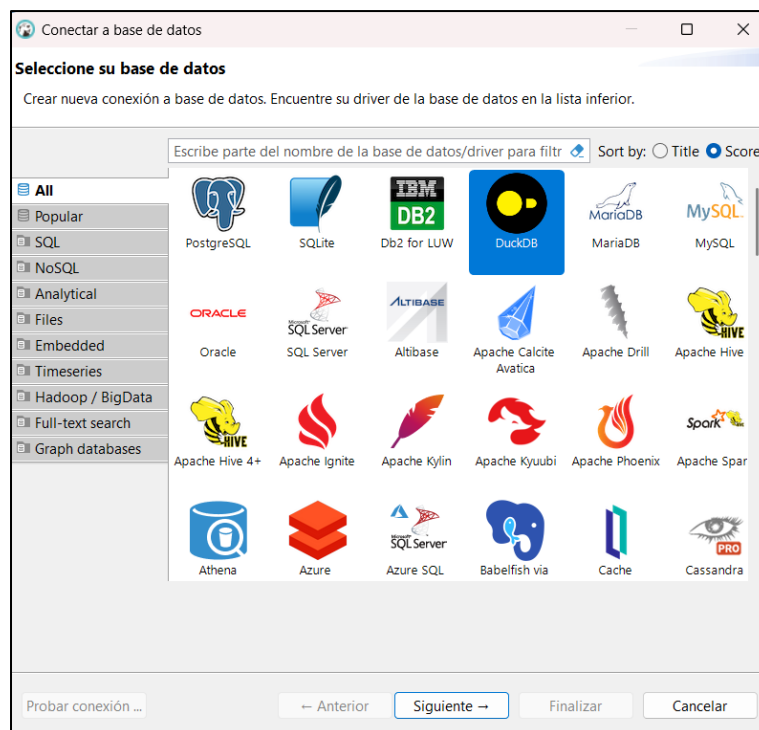
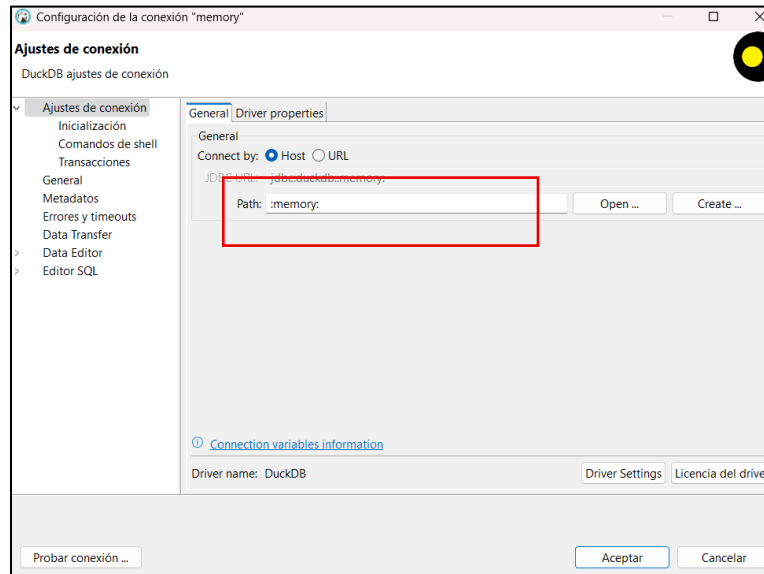


Ilustración 1: Creación de nueva conexión de base de datos DuckDB

DuckDB monta la base en memoria por defecto, lo que significa que arranca más rápido al evitar I/O en disco, es ideal para consultas ad-hoc o análisis temporales, y cuando se cierra la conexión todo lo

almacenado en `:memory:` desaparece. Si se necesita que los datos persistan entre sesiones se deb reemplazar `:memory:` por la ruta de un archivo `.duckdb`.



**Ilustración 2: Prámetros de DuckDB**

### 2.1.2 Instalación de extensiones en DuckDB

DuckDB dispone de un repositorio oficial de extensiones que se pueden instalar y cargar en cualquier momento durante la sesión. El flujo básico es:

#### Instalar

Ejecutar en SQL:

**INSTALL postgres;**

Esto descarga e instala la extensión desde el repositorio oficial de DuckDB.

#### Cargar

Una vez instalada, se debe cargar para poder usarla:

**LOAD postgres;**

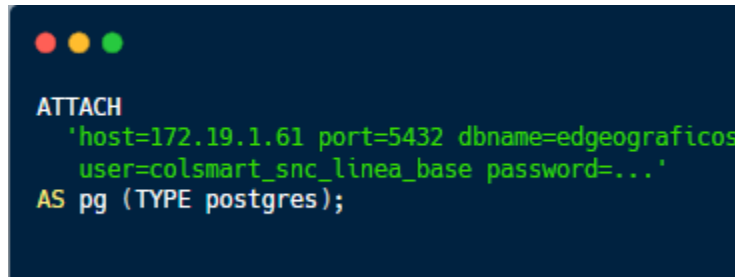
Hasta que no se cargue, las funciones y objetos de la extensión no estarán disponibles.

### 2.1.3 Conectar Postgres a DuckDB

El comando `ATTACH ... AS pg (TYPE postgres);` permite establecer una conexión desde un entorno SQL que soporta múltiples fuentes (como DuckDB) hacia una base de datos PostgreSQL remota. Esta instrucción expone la base externa como una fuente de datos adicional dentro del contexto actual de ejecución SQL, habilitando operaciones de lectura y escritura sobre esa fuente remota como si fuera local.

Esto es necesario cuando los datos de origen (por ejemplo, archivos Parquet) se encuentran en el sistema de archivos local, pero el destino es una base de datos PostgreSQL. La conexión ATTACH actúa como un puente que permite que los resultados de funciones como `read_parquet()` se puedan persistir directamente en tablas del motor remoto.

El bloque completo:

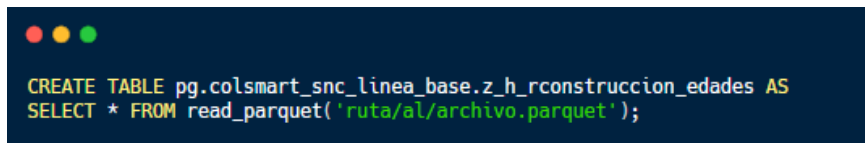


```
ATTACH
  'host=172.19.1.61 port=5432 dbname=edgeograficos
  user=colsmart_snc_linea_base password=...'
AS pg (TYPE postgres);
```

Ilustración 3: Inserción de parámetros de conexión PostgreSQL a DuckDB

- ✓ **host=172.19.1.61**: IP del servidor PostgreSQL.
- ✓ **dbname=edgeograficos**: nombre de la base de datos remota.
- ✓ **user, password**: credenciales de conexión.
- ✓ **AS pg**: nombre con el que vas a referenciar esa conexión (alias).
- ✓ **(TYPE postgres)**: especifica el tipo de fuente de datos.

Después de establecer la conexión mediante ATTACH, se utiliza la instrucción:



```
CREATE TABLE pg.colsmart_snc_linea_base.z_h_rconstruccion_edades AS
SELECT * FROM read_parquet('ruta/al/archivo.parquet');
```

Ilustración 4: Creación de tablas en Postgres desde DuckDB por medio de parquets

Esta sentencia crea una nueva tabla en la base de datos PostgreSQL remota (especificada por el alias `pg`) dentro del esquema `colsmart_snc_linea_base`. La tabla `z_h_rconstruccion_edades` se genera con la estructura y contenido que retorna la función `read_parquet()`.

La función `read_parquet()` lee el contenido de un archivo `.parquet` ubicado en el sistema de archivos local y lo presenta como una tabla temporal en memoria. Luego, el comando `CREATE TABLE ... AS SELECT` toma ese resultado y lo inserta directamente en la tabla remota, creando la tabla con las columnas inferidas del archivo Parquet y copiando todos los registros.

Este patrón se utiliza para transferir datos estructurados desde archivos Parquet a PostgreSQL sin intervención de herramientas intermedias, aprovechando la capacidad del motor SQL para operar sobre múltiples fuentes en una misma sesión.

## 2.2. Estandarización y consolidación de edades

Una vez insertados los archivos `.parquet` en la base de datos, se procede a **estandarizar las tablas de entrada**, unificando su estructura y homogeneizando los campos clave necesarios para el análisis.

Posteriormente, se construye la tabla **z\_h\_edades** mediante una operación UNION ALL, a partir de múltiples tablas intermedias que contienen **edades estimadas** provenientes de distintos componentes del modelo (unidades, construcciones y terrenos, tanto formales como informales, urbanos y rurales).

La tabla resultante **z\_h\_edades** incluye los siguientes datos:

Tabla 1: Unión de todos los parquets asignandoles un nivel de calidad

Componente	Tabla origen	Fuente (fuente)	Nivel
Unidad Rural	z_h_runidad_edades	unidad_r	1
Unidad Informal Rural	z_h_runidad_informal_edades	unidad_r_i	1
Unidad Urbana	z_h_uunidad_edades	unidad_u	1
Unidad Informal Urbana	z_h_uunidad_informal_edades	unidad_u_i	1
Construcción Rural	z_h_rconstruccion_edades	construccion_r	2
Construcción Urbana	z_h_uconstruccion_edades	construccion_u	2
Construcción Informal U	z_h_uconstruccion_informal_edades	construccion_u_i	2
Construcción Informal R	z_h_rconstruccion_informal_edades	construccion_r_i	2
Terreno Urbano	z_h_uterreno_edades, z_h_rterreno_edades	terreno_u	3
Terreno Informal Urbano	z_h_uterreno_informal_edades (aparece 2 veces)	terreno_u_i	3

La columna nivel tiene un propósito **técnico y jerárquico: priorizar** las fuentes de información según su confiabilidad o proximidad lógica al dato que queremos consolidar (el anio\_construccion). No es arbitrario; cada número representa un nivel de confianza o cercanía al objeto físico (unidad, construcción, terreno).

Tabla 2: Descripción de Iso niveles de calidad

Nivel	Significado	¿Por qué se le da ese número?
1	<b>Unidad</b> (urbana/rural)	Es el objeto principal donde reside o se usa la construcción. Se espera que su año esté más cerca al uso real.
2	<b>Construcción</b>	Representa el objeto físico construido. Si se conoce su año, se considera el más representativo del estado físico.
3	<b>Terreno</b>	Es el dato menos específico en términos de edad construida. Puede tener uso anterior o sin construcción aún.

### 2.3. Clasificación de escenarios de emparejamiento

Durante el proceso de consolidación de edades, es fundamental identificar cómo se comportan los registros en función de la disponibilidad de datos provenientes de cada nivel jerárquico (unidad, construcción, terreno). Para ello, se clasifica cada predio (numero\_predial) en un escenario específico, de acuerdo con la **combinación de presencia o ausencia** de edades por nivel.

Esto se realiza sobre la tabla z\_h\_edades\_anio, construida previamente mediante agrupación y selección jerárquica de fechas mínimas por nivel.

Los escenarios de emparejamiento se identifican utilizando sentencias WHERE que permiten verificar qué columnas tienen valores nulos y cuáles no. A continuación, se describen los principales casos detectados:

Tabla 3: Clasificación de escenarios de emparejamiento

Escenario	Descripción	Condición lógica SQL
Terreno sin edad superior	El predio solo tiene edad estimada del terreno, sin datos de unidad ni construcción.	anio_unidad IS NULL AND anio_construccion IS NULL
Solo construcción disponible	Se tiene edad para la construcción, pero no para la unidad.	anio_unidad IS NULL AND anio_construccion IS NOT NULL
Unidad y construcción presentes	Ambas fechas están disponibles. Puede usarse la construcción como prioridad.	anio_unidad IS NOT NULL AND anio_construccion IS NOT NULL
Solo unidad disponible	Hay edad para la unidad, pero no para la construcción.	anio_unidad IS NOT NULL AND anio_construccion IS NULL
Edad igual a 2024	Se identifican registros con año estimado igual a 2024, útil para control de datos recientes.	anio_unidad = 2024 (combinado con otras condiciones)

numero_predial	anio_unidad	anio_construccion	anio_terreno	anio_final
138360101000003180001000010000	2.017	[NULL]	[NULL]	2.017
138360101000003180001000020000	2.017	[NULL]	[NULL]	2.017
134300102000001810011000000000	2.024	2.011	2.010	2.011
138360101000003400003000000000	2.024	2.017	2.010	2.017
134300102000001490005500000001	[NULL]	2.011	[NULL]	2.011
134300102000001520002000000000	[NULL]	[NULL]	2.010	2.010

### 2.4. Integración de construcciones y unidades

Una vez consolidada la información de edades por predio en la tabla z\_h\_edades\_anio, se procede a actualizar el campo anio\_construccion en la tabla cr\_unidadconstruccion, únicamente para aquellos registros que actualmente no tienen esta información (es decir, su valor es NULL).

Para ello, se implementa un proceso de cruce alfanumérico que consiste en:



1. **Identificar las unidades sin año de construcción:**

Se extraen todos los registros de `cr_unidadconstruccion` donde el campo `anio_construccion` es nulo. Esta subconsulta se denomina `unidad_sinanio`.

2. **Cruzar con la tabla de edades estimadas:**

A continuación, se realiza un INNER JOIN entre las unidades sin año y la tabla `z_h_edades_anio`, utilizando como clave de emparejamiento el campo `codigo` (en `cr_unidadconstruccion`) y `numero_predial` (en `z_h_edades_anio`). Este cruce permite asociar cada unidad sin año con el año consolidado previamente (`anio_final`), en caso de existir coincidencia.

3. **Crear una tabla auxiliar para la actualización:**

El resultado del cruce se almacena en una nueva tabla llamada `z_h_edades_anio_cruce`, que contiene tres columnas: `codigo`, `globalid` y `anio_final`. Esta tabla servirá como insumo directo para ejecutar la actualización de los datos.

## 2.6. Actualización de campos nulos en `cr_unidadconstruccion`

Con base en la tabla auxiliar `z_h_edades_anio_cruce`, que contiene los años consolidados (`anio_final`) para unidades sin dato previo, se procede a realizar la actualización del campo `anio_construccion` en la tabla oficial `cr_unidadconstruccion`.

Este paso se realiza mediante una sentencia UPDATE con JOIN, utilizando como clave el campo `globalid`, que garantiza la correspondencia exacta entre los registros.

Como resultado final del proceso, se obtiene una tabla **`cr_unidadconstruccion` enriquecida con nuevos valores en el campo `anio_construccion`**, los cuales fueron estimados a partir de múltiples fuentes históricas, estandarizadas y jerarquizadas previamente.

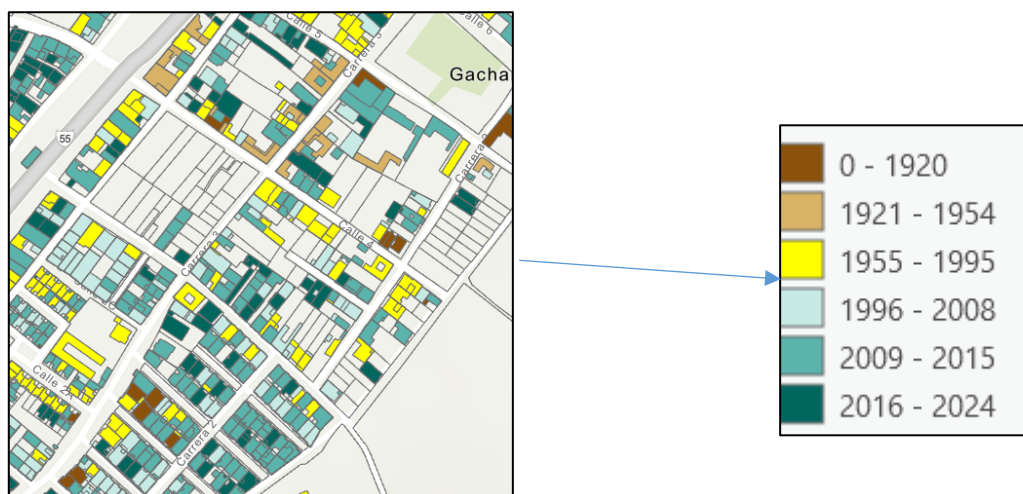


Ilustración 5: Resultado final de la actualización de años de construcción

### 3. Conclusiones

Como resultado del proceso de consolidación, se logró integrar exitosamente los años de construcción provenientes de múltiples fuentes históricas almacenadas en archivos .parquet, estandarizando su estructura y clasificándolos según su nivel jerárquico (unidad, construcción o terreno). A partir de esta integración, se generó una tabla consolidada que permite estimar un año representativo (anio\_final) para cada predio, priorizando siempre la fuente más específica disponible. Esta estrategia permitió recuperar información faltante, identificar escenarios de emparejamiento según la disponibilidad de datos, y garantizar trazabilidad en cada registro. Finalmente, se enriqueció la tabla oficial cr\_unidadconstruccion con nuevos valores en el campo anio\_construccion, mejorando sustancialmente la completitud y calidad de la base de datos, lo que habilita análisis posteriores más confiables y estructurado