En PostgreSQL, existen **dos catálogos principales** para gestionar y consultar los metadatos de la base de datos:

1. `pg_catalog`: Es el catálogo interno de PostgreSQL, que contiene información técnica sobre las tablas, columnas, índices, restricciones, y más.
2. `information_schema`: Es un catálogo estandarizado según el estándar **SQL ISO/IEC 9075**, diseñado para ofrecer una forma portable y amigable de consultar metadatos, especialmente útil si se trabaja con múltiples sistemas de bases de datos.

A continuación, te explico por qué existen ambos, cómo usarlos y ejemplos con las tablas más comunes de cada uno.

¿Por qué hay dos catálogos en PostgreSQL?

1. `pg_catalog`:
    - Es específico de PostgreSQL.
    - Contiene información detallada y de bajo nivel sobre la estructura interna de la base de datos.
    - Es usado internamente por PostgreSQL para su funcionamiento y es accesible para consultas avanzadas.
2. `information_schema`:
    - Es un estándar SQL que proporciona una interfaz común para consultar metadatos en cualquier sistema de bases de datos compatible.
    - Es más sencillo y portable para desarrolladores que necesitan trabajar con sistemas de bases de datos diferentes.

Resumen:

- `pg_catalog`: Completo, detallado y específico para PostgreSQL.
- `information_schema`: Simple, portable y estandarizado.

Cómo Usar Ambos Catálogos

1. Consultar Tablas en la Base de Datos

Usando `pg_catalog`:

```
SELECT relname AS nombre_tabla, relkind AS tipo
FROM pg_class
WHERE relkind IN ('r', 'v') -- 'r' para tablas, 'v' para vistas
```

```
  AND relnamespace IN (SELECT oid FROM pg_namespace WHERE nspname = 'public');
```

- **`pg_class`**: Contiene información sobre tablas, vistas, índices, etc.
- **`relnamespace`**: Relación entre las tablas y el esquema al que pertenecen.

Resultado: Lista de tablas y vistas en el esquema `public`.

Usando `information_schema`:

```
SELECT table_name, table_type
FROM information_schema.tables
WHERE table_schema = 'public';
```

- **Más sencillo** y cumple con el estándar SQL.

---

2. Consultar Columnas de una Tabla

Usando `pg_catalog`:

```
SELECT attname AS nombre_columna, atttypid::regtype AS tipo_dato, attnotnull AS
no_nulo
FROM pg_attribute
WHERE attrelid = 'empleados'::regclass
  AND attnum > 0 AND NOT attisdropped;
```

- **`pg_attribute`**: Contiene detalles sobre las columnas de una tabla.
- **`atttypid`**: Identificador del tipo de dato.
- **`attnotnull`**: Indica si la columna tiene la restricción `NOT NULL`.

Usando `information_schema`:

```
SELECT column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_name = 'empleados';
```

- **Más legible**, pero menos detallado.

---

3. Consultar Claves Primarias

Usando `pg_catalog`:

```
SELECT conname AS nombre_restriccion, a.attname AS columna
FROM pg_constraint c
JOIN pg_class t ON c.conrelid = t.oid
JOIN pg_attribute a ON a.attnum = ANY (c.conkey) AND a.attrelid = t.oid
WHERE c.contype = 'p' AND t.relname = 'empleados';
```

- **`pg_constraint`**: Almacena todas las restricciones (claves primarias, foráneas, etc.).
- **`contype`**: `p` indica una clave primaria.

Usando `information_schema`:

```
SELECT tc.constraint_name, kcu.column_name
FROM information_schema.table_constraints tc
JOIN information_schema.key_column_usage kcu
  ON tc.constraint_name = kcu.constraint_name
WHERE tc.table_name = 'empleados' AND tc.constraint_type = 'PRIMARY KEY';
```

- Más simple, pero menos flexible.

---

4. Consultar Restricciones de Claves Foráneas

Usando `pg_catalog`:

```
SELECT conname AS nombre_restriccion, a.attname AS columna, t2.relname AS
tabla_referenciada
FROM pg_constraint c
JOIN pg_class t1 ON c.conrelid = t1.oid
JOIN pg_class t2 ON c.confrelid = t2.oid
JOIN pg_attribute a ON a.attnum = ANY (c.conkey) AND a.attrelid = t1.oid
WHERE c.contype = 'f' AND t1.relname = 'empleados';
```

- Muestra las claves foráneas de la tabla `empleados`, incluyendo las tablas referenciadas.

Usando `information_schema`:

```
SELECT tc.constraint_name, kcu.column_name, ccu.table_name AS tabla_referenciada,
ccu.column_name AS columna_referenciada
FROM information_schema.table_constraints tc
JOIN information_schema.key_column_usage kcu ON tc.constraint_name =
kcu.constraint_name
JOIN information_schema.constraint_column_usage ccu ON ccu.constraint_name =
tc.constraint_name
```

WHERE tc.table_name = 'empleados' AND tc.constraint_type = 'FOREIGN KEY';

Tablas Más Comunes y Sus Usos

Tablas de `pg_catalog`

Tabla	Propósito
`pg_class`	Información sobre tablas, vistas, índices, secuencias y otros objetos.
`pg_attribute`	Información sobre columnas de tablas, vistas y otros objetos.
`pg_namespace`	Esquemas definidos en la base de datos.
`pg_constraint`	Restricciones definidas en tablas (claves primarias, foráneas, UNIQUE, CHECK).
`pg_type`	Tipos de datos definidos por el usuario o predefinidos.
`pg_index`	Información sobre índices en la base de datos.

Tablas de `information_schema`

Tabla	Propósito
`tables`	Lista de tablas y vistas de un esquema.
`columns`	Información sobre las columnas de las tablas (nombre, tipo, restricciones).
`key_column_usage`	Detalles sobre las columnas utilizadas en claves primarias y foráneas.
`table_constraints`	Lista de restricciones (`PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `CHECK`).
`referential_constraints`	Detalles sobre claves foráneas (tabla y columnas relacionadas).

¿Cuál Usar y Cuándo?

1. Usa **`pg_catalog`**:

    - Para consultas avanzadas que necesiten detalles técnicos.
    - Cuando estés desarrollando algo muy específico para PostgreSQL.
2. Usa `information_schema`:

    - Para consultas estándar y portables entre bases de datos.
    - Cuando necesitas legibilidad o trabajar con herramientas genéricas.

Ambos catálogos son útiles dependiendo del nivel de detalle que necesites y el contexto en el que trabajes.