Base de Datos

DC. FCEyN
2025-04-30 teórica 5

Emilio Platzer
tute@dc.uba.ar

∃ ∄ ⟶ ⟹ ∀

Planes de Ejecución

JOIN

Se juntan datos de dos (o más tablas). Caso típico, traer datos de otra tabla a través de un código.

```
SELECT ar.nro_lu, ar.nota, al.apellido, al.nombre
  FROM acta_renglon ar INNER JOIN alumnos al USING (nro_lu)
  WHERE ar.año = 2023 AND ...
```

Planes de Ejecución - JOIN

Existen 3 estrategias básicas

- Block Nested Loop Join (BNLJ)
- Index Nested Loop Join (INDLJ)
- Sort Merge Join (SMJ)
- Hash Join (HJ)

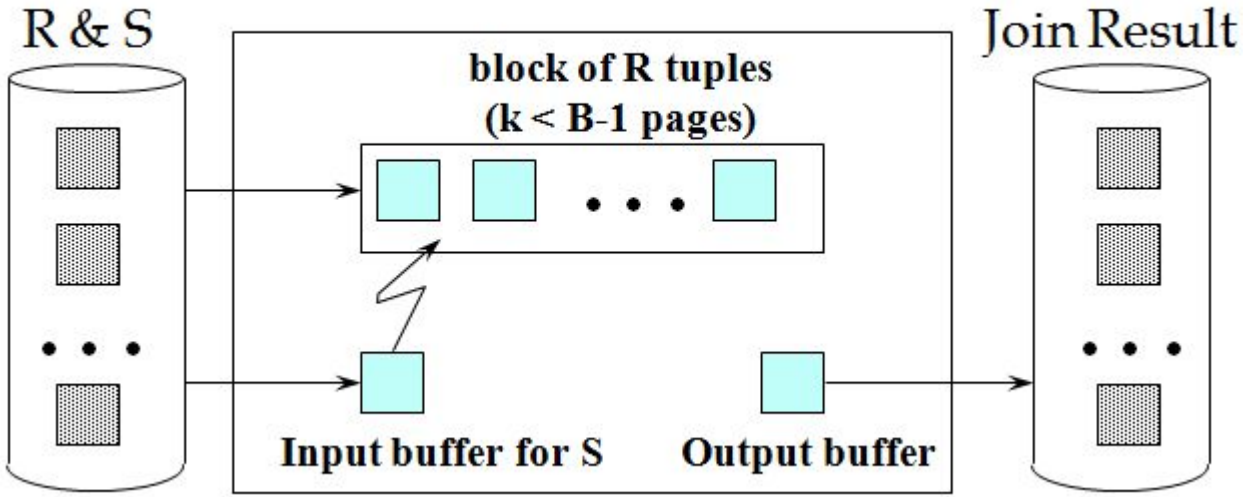
Planes de Ejecución - JOIN

¿Qué cosas se pueden hacer a mano?

Depende de las primitivas:

- Acceder al registro (tabla, #bloque, #registro) ⟶ tupla
- Siguiente registro (tabla, #bloque, #registro) ⟶ (#bloque, #registro)
- Primer registro (tabla) ⟶ (#bloque, #registro)
- Obtener la clave (tupla) ⟶ clave
- Buscar en índice (tabla, valor de la clave clave) ⟶ (#bloque, #registro)
- Siguiente en índice (tabla, valor de la clave clave) ⟶ (#bloque, #registro)
- Primero en índice (tabla) ⟶(bloque, registro)

Planes de Ejecución - BNLJ



imágenes de https://cs186.fandom.com/wiki/Joins

¿Qué pasa cuando ejecutamos un sql?

- Se valida sintáctica y semánticamente
  - Controla la sintaxis
  - Se verifica que existan las tablas y los campos
  - Se valida que los tipos de datos estén correctamente utilizados (que no se comparen campos fecha contra números, etc.)
- El optimizador genera el plan de ejecución
- El motor de ejecución se encarga de ejecutarlo

optimizador

- Recibe la consulta
- Escribe el AR correspondiente
- Genera el árbol canónico
- Arma distintos planes de ejecución alternativos
- Mide el tiempo estimado (en base a heurísticas, estadísticas, etc)
- Selecciona el mejor

Equivalencias algebraicas

con el árbol canónico se pueden realizar transformaciones que tienen la misma semántica.

(Ojo con los NULL, short circuit, etc…)

https://www.postgresql.org/message-id/attachment/32495/equivalenceRules.pdf

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections; *cascade of* σ.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Selection operations are commutative:

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Only the final operations in a sequence of projection operations is needed, the others can be omitted; *cascade of* Π

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) = \Pi_{L_1}(E)$$

4. Selections can be combined with Cartesian products and theta joins:

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$
$$\sigma_{\theta_1}(E_1 \bowtie_{\sigma_{\theta_2}} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$$

5. Theta join operations are commutative:

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

6. Natural-join operations are associative:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

heurísticas

- Resolver las selecciones y proyecciones lo más cerca posible de las hojas
- Convertir los productos cartesianos en joins
- Resolver primero los selects más selectivos
- Tomar en cuenta los índices existentes a la hora de hacer los cambios

Estadísticas

- Supongamos que tenemos una tabla EMPLEADOS
- $T_{EMPLEADOS} = 8.000$
- SUELDO $\in$ [12000, 400.000]
- Como se imaginan que es esta curva?
- Cual es el T’ de SUELDO >= 150.000?

Estadísticas - Histogramas

- de igual ancho
- de igual alto
- de muestras

Los índice de hoy en día

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] name ] ON [ ONLY ] table_name [ USING method ]
    ( { column_name | ( expression ) } [ COLLATE collation ]
      [ opclass [ ( opclass_parameter = value [, ... ] ) ] ]
      [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...]
    )
    [ INCLUDE ( column_name [, ...] ) ]
    [ NULLS [ NOT ] DISTINCT ]
    [ WITH ( storage_parameter [= value] [, ... ] ) ]
    [ TABLESPACE tablespace_name ]
    [ WHERE predicate ]
```

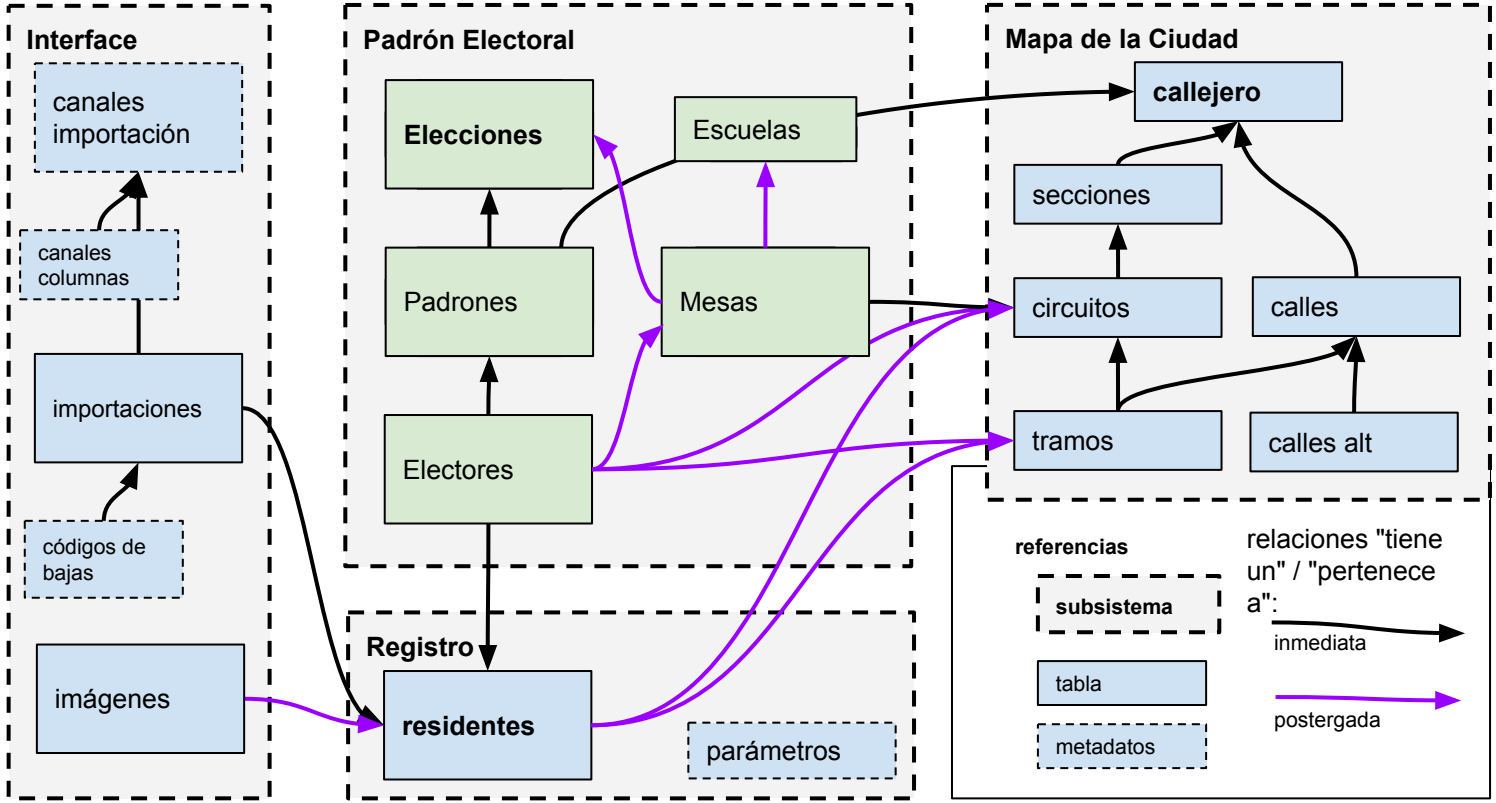
method

The name of the index method to be used. Choices are `btree`, `hash`, `gist`, `spgist`, `gin`, `brin`, or user-installed access methods like bloom. The default method is `btree`.

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name ON <object>
    ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WHERE <filter_predicate> ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { partition_scheme_name ( column_name )
         | filegroup_name
         | default
    } ]
    [ FILESTREAM_ON { filestream_filegroup_name | partition_scheme_name | "NULL" } ]
```

Ejemplos de planes de ejecución

Padrón Electoral - Ciudad A - Diagrama de Tablas y Relaciones



Ejemplos de planes de ejecución - PostgreSQL

```
select ejemplar, count(*)
  from residentes
  group by cube (ejemplar)
  order by ejemplar;
```

ejemplar text	count bigint
A	164434
B	148658
C	59969
D	17485
E	4447
F	1115
[null]	396108

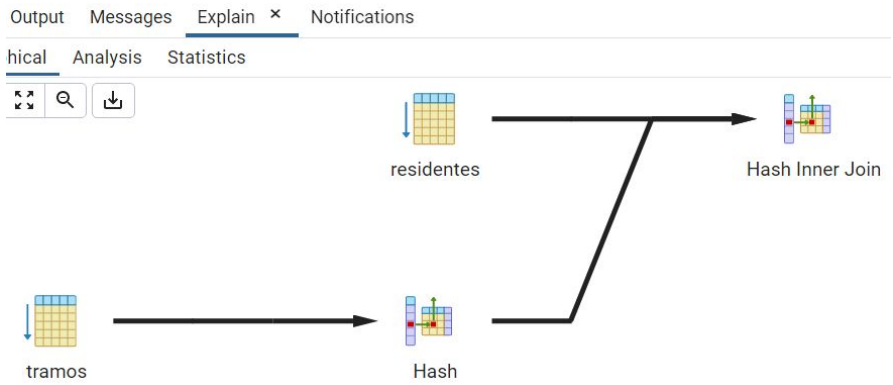
```
select tipo, count(*)
  from tramos
  group by cube (tipo)
  order by tipo;
```

tipo text	count bigint
B	102
I	205
O	183
TCA	5260
TSA	111
[null]	5861

Ejemplos de planes de ejecución

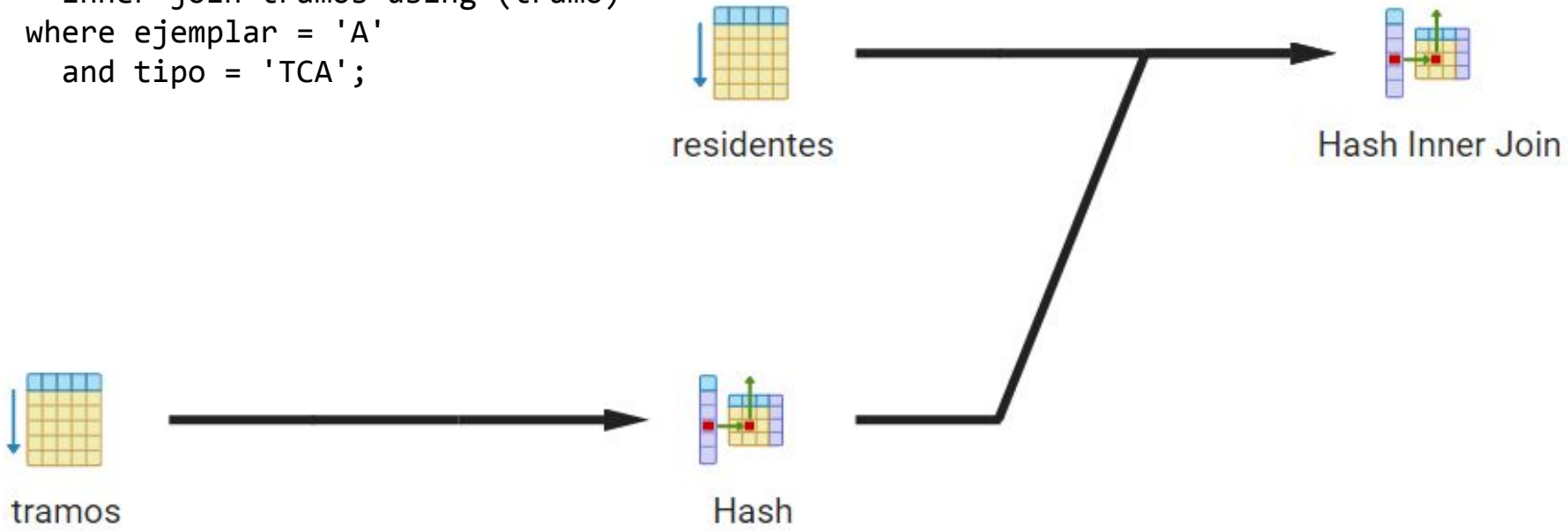
```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'A'
    and tipo = 'TCA';
```

tipo text	tramo text	precision numeric
TCA	GRAL J G DE ARTIGAS: 101-599 IMP...	0.9
TCA	VIRREY CEVALLOS: 801-1200	0.9
TCA	F J STA M DE ORO: 2001-2500	0.9
TCA	LARRAYA: 1501-1600	1
TCA	MAGARIÑOS CERVANTES: 3201-3800	0.9
TCA	ALBARRACIN: 1401-2100	0.9
TCA	M ACOSTA: 2301-3700	0.88
TCA	AV. SCALABRINI ORTIZ: 1102-1900 P...	0.88



Ejemplos de planes de ejecución

```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'A'
    and tipo = 'TCA';
```



Ejemplos de planes de ejecución

```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'A'
    and tipo = 'TCA';
```

#	Node	Rows Actual
1.	→ Hash Inner Join (rows=124035 loops=1) **Hash Cond**: (residentes.tramo = tramos.tramo)	124035
2.	→ Seq Scan on residentes as residentes (rows=164434 loops=1) **Filter**: (ejemplar = 'A'::text) **Rows Removed by Filter**: 232140	164434
3.	→ Hash (rows=5260 loops=1) **Buckets**: 8192 **Batches**: 1 **Memory Usage**: 364 kB	5260
4.	→ Seq Scan on tramos as tramos (rows=5260 loops=1) **Filter**: (tipo = 'TCA'::text) **Rows Removed by Filter**: 601	5260

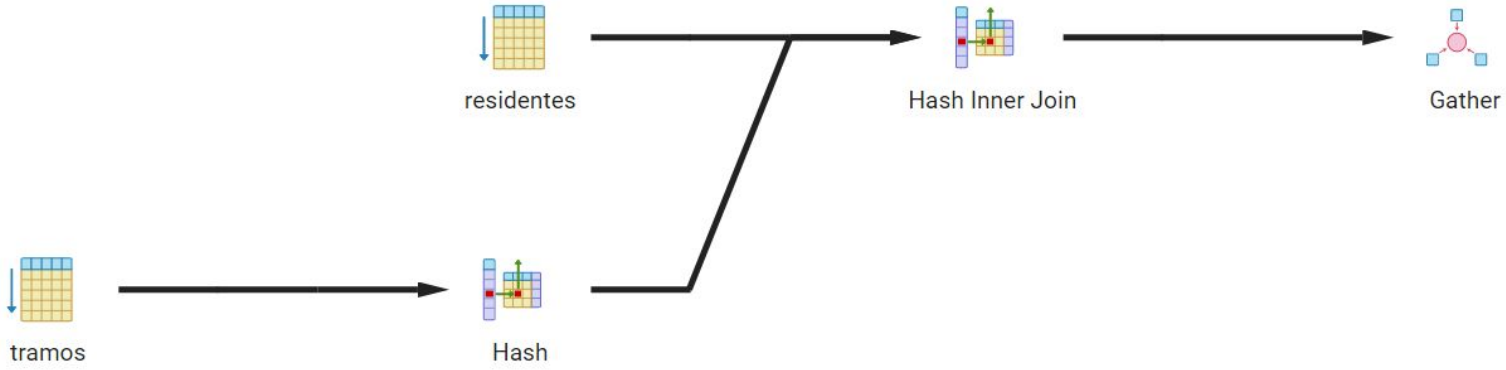
Ejemplos de planes de ejecución

```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'C'
    and tipo = 'I';
```

tipo text	tramo text	precision numeric
I	RIESTRA Y M ACOSTA; ...	0.67
I	RIESTRA Y M ACOSTA; ...	0.67
I	E BONORINO Y CASTA...	0.649999999
I	VARELA Y P MORENO; ...	0.67
I	VARELA Y P MORENO; ...	0.67
I	MONTEAGUDO/O CRUZ...	0.8

Ejemplos de planes de ejecución

```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'C'
    and tipo = 'I';
```



Node Type	Gather
Parallel Aware	false
Async Capable	false
Actual Rows	200
Actual Loops	1
Workers Planned	2
Workers Launched	2
Single Copy	false
loops	1

Ejemplos de planes de ejecución

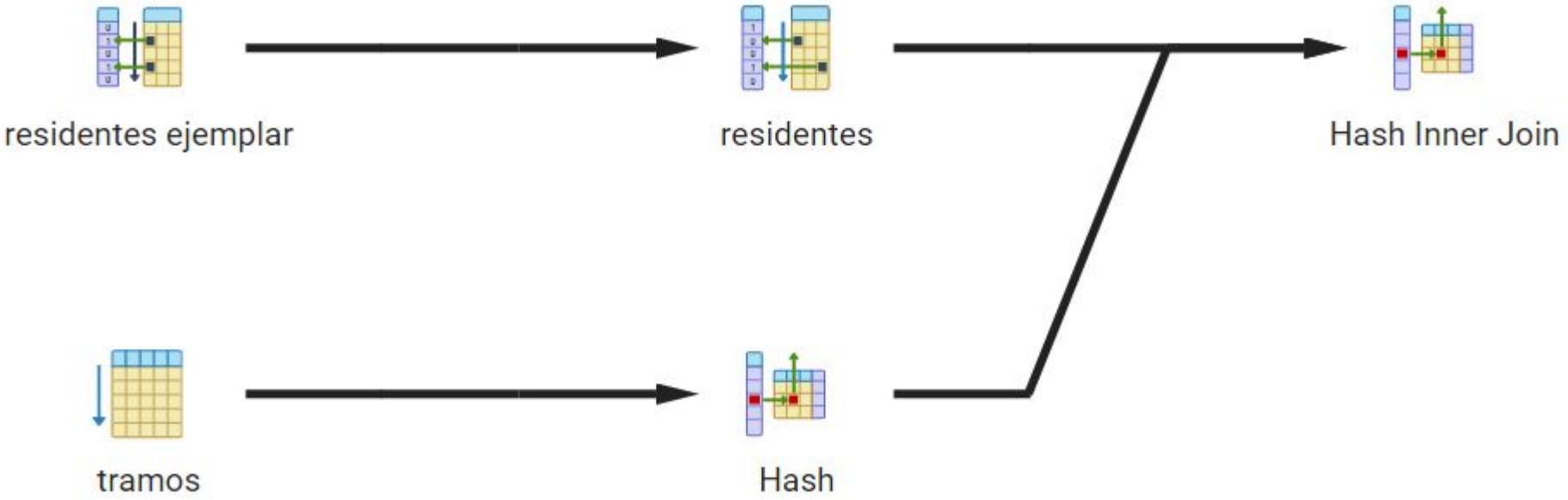
```
select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'A'
    and tipo = 'TCA';
```

#	Node	Rows
		Actual
1.	→ Gather (rows=200 loops=1)	200
2.	→ Hash Inner Join (rows=67 loops=3) **Hash Cond**: (residentes.tramo = tramos.tramo)	67
3.	→ Seq Scan on residentes as residentes (rows=19990 loops=3) **Filter**: (ejemplar = 'C'::text) **Rows Removed by Filter**: 112202	19990
4.	→ Hash (rows=205 loops=3) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 21 kB	205
5.	→ Seq Scan on tramos as tramos (rows=205 loops=3) **Filter**: (tipo = 'I'::text) **Rows Removed by Filter**: 5656	205

Ejemplos de planes de ejecución

```
create index "residentes ejemplar" on residentes (ejemplar);

select tipo, tramo, precision
  from residentes inner join tramos using (tramo)
  where ejemplar = 'C' and tipo = 'I';
```



Ejemplos de planes de ejecución

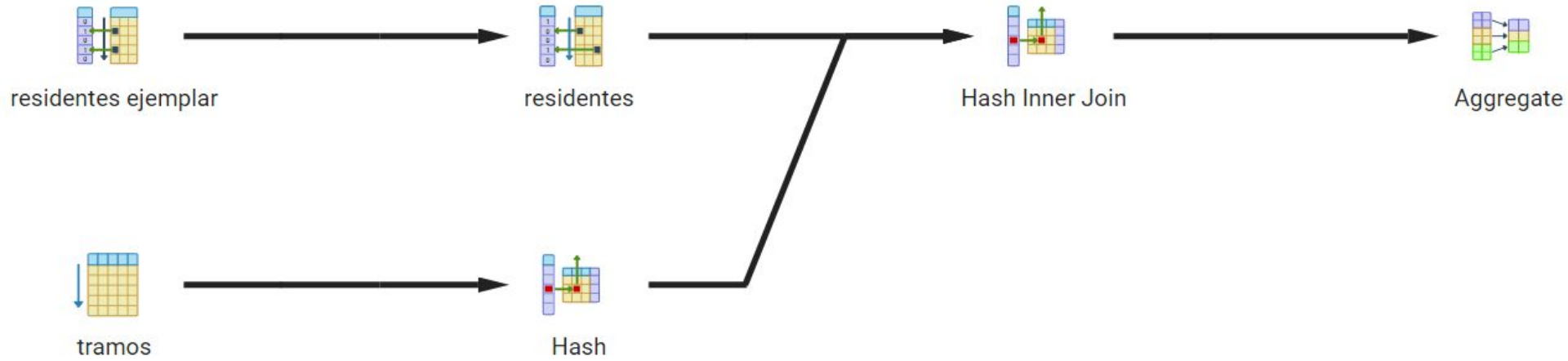
```
create index "residentes ejemplar" on residentes (ejemplar);

select tipo, tramo, precision
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'C'
    and tipo = 'I';
```

1.	→ Hash Inner Join (rows=200 loops=1) **Hash Cond**: (residentes.tramo = tramos.tramo)
2.	→ Bitmap Heap Scan on residentes as residentes (rows=59969 loops=1) **Recheck Cond**: (ejemplar = 'C'::text) **Heap Blocks**: exact=12055
3.	→ Bitmap Index Scan using residentes ejemplar (rows=59969 loops=1) **Index Cond**: (ejemplar = 'C'::text)
4.	→ Hash (rows=205 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 21 kB
5.	→ Seq Scan on tramos as tramos (rows=205 loops=1) **Filter**: (tipo = 'I'::text) **Rows Removed by Filter**: 5656

Ejemplos de planes de ejecución

```
select tipo, count(*)
  from residentes
    inner join tramos using (tramo)
  where ejemplar = 'C'
     and tipo = 'I'
  group by tipo
```



Ejemplos de planes de ejecución SQL server

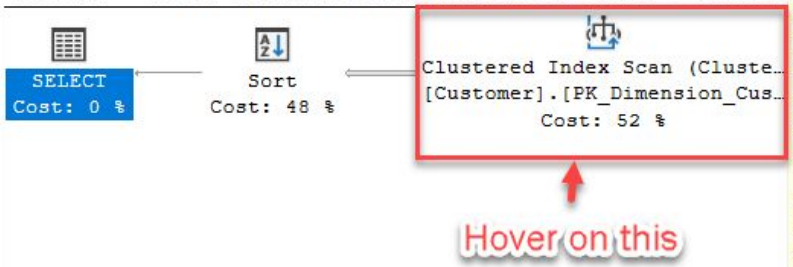
SQLQuery3.sql - G...RGROUP\ADas (72))* | SQLQuery2.sql - G...RGROUP\ADas (54))*

```
1  USE WideWorldImportersDW
2  GO
3
4  SELECT * FROM [Dimension].[Customer]
5  WHERE [Postal Code] = '90761'
6  ORDER BY [Customer]
7  GO
```

110 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [Dimension].[Customer] WHERE [Postal



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	403
Actual Number of Rows	3
Actual Number of Batches	0
Estimated I/O Cost	0.0120139
Estimated Operator Cost	0.0126142 (52%)
Estimated CPU Cost	0.0006003
Estimated Subtree Cost	0.0126142
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	3
Estimated Number of Rows to be Read	403
Estimated Row Size	409 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	1

Predicate
[WideWorldImportersDW].[Dimension].[Customer].[Postal Code] =N'90761'

Object
[WideWorldImportersDW].[Dimension].[Customer].[PK_Dimension_Customer]

Output List
[WideWorldImportersDW].[Dimension].[Customer].Customer Key,