Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: N/A | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|
| #1 HashAggr... | | | |
| #2 └ Nested L... | | | |
| #3 ├ Seq Sc... | | | |
| #4 └ Memo... | | | |
| #5 └ Ind... | | | |

## #1 HashAggregate

by `departments.department_name`

*HashAggregate Node* groups records together based on a GROUP BY or aggregate function (like sum()). Hash Aggregate uses a hash to first organize the records by a key.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.284ms | 28%
- **Rows:** 11 (Planned: 200) | ↑ over estimated by 18 ×
- **Cost:** 5.7 (Total: 45.4)

## #2 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.454ms | 45%
- **Rows:** 741 (Planned: 741)
- **Cost:** 21.1 (Total: 39.7)

## #3 Seq Scan

on `employees`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.233ms | 23%
- **Rows:** 741 (Planned: 741)
- **Cost:** 18.4 (Total: 18.4)

## #4 Memoize

*Memoize Node* is used to cache the results of the inner side of a nested loop. It avoids executing underlying nodes when the results for the current parameters are already in the cache.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0ms | 0%
- **Rows:** ~741 (Planned: ~741)
- **Cost:** 0.01 (Total: 0.24)
- **Loops:** 741

## #5 Index Scan

on `departments`
using `departments_pkey`

*Index Scan Node* finds relevant records based on an **Index**. Index Scans perform 2 read operations: one to read the index and another to read the actual value from the table.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.033ms | 3%
- **Rows:** ~11 (Planned: ~11)
- **Cost:** 0.23 (Total: 0.23)
- **Loops:** 11

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: 0.228ms | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|

#1 HashAggr...
#2 └ Hash Join
#3 ├ Seq Sc...
#4 └ Hash
#5 └ Seq...

## #1 HashAggregate

by `departments.department_name`

*HashAggregate Node* groups records together based on a GROUP BY or aggregate function (like sum()). Hash Aggregate uses a hash to first organize the records by a key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.311ms | 33%
**Rows:** 11 (Planned: 11)
**Cost:** 3.81 (Total: 26.2)

## #2 Hash Join

on `employees.department_id = departments.department_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.452ms | 48%
**Rows:** 741 (Planned: 741)
**Cost:** 2.83 (Total: 22.4)

## #3 Seq Scan

on `employees`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.169ms | 18%
**Rows:** 741 (Planned: 741)
**Cost:** 18.4 (Total: 18.4)

## #4 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.007ms | 1%
**Rows:** 11 (Planned: 11)

## #5 Seq Scan

on `departments`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.009ms | 1%
**Rows:** 11 (Planned: 11)
**Cost:** 1.11 (Total: 1.11)

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: 0.787ms | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|

#1 HashAggr...
#2 └ Hash Join
#3 ├ Seq Sc...
#4 └ Hash
#5 └ Seq...

## #1 HashAggregate

by locations.city

*HashAggregate Node* groups records together based on a GROUP BY or aggregate function (like sum()). Hash Aggregate uses a hash to first organize the records by a key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.011ms | 16%
**Rows:** 7 (Planned: 11)
**Cost:** 0.16 (Total: 14.8)

## #2 Hash Join

on locations.location_id = departments.location_id

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.015ms | 22%
**Rows:** 11 (Planned: 11)
**Cost:** 1.15 (Total: 14.7)

## #3 Seq Scan

on locations

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.007ms | 10%
**Rows:** 7 (Planned: 240) | ↑ over estimated by **34** ×
**Cost:** **12.4** (Total: 12.4)

## #4 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.009ms | 13%
**Rows:** 11 (Planned: 11)

## #5 Seq Scan

on departments

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.026ms | 38%
**Rows:** 11 (Planned: 11)
**Cost:** 1.11 (Total: 1.11)

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: 0.589ms | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|
| #1 HashAggr... | | | |
| #2 └ Hash Join | | | |
| #3 ├ Seq Sc... | | | |
| #4 └ Hash | | | |
| #5 └ Seq... | | | |

### #1 HashAggregate
by `locations.city`

*HashAggregate Node* groups records together based on a GROUP BY or aggregate function (like sum()). Hash Aggregate uses a hash to first organize the records by a key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.01ms | 9%
**Rows:** 7 (Planned: 7)
**Cost:** 0.12 (Total: 2.43)

### #2 Hash Join
on `departments.location_id = locations.location_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.051ms | 48%
**Rows:** 11 (Planned: 11)
**Cost:** 0.13 (Total: 2.31)

### #3 Seq Scan
on `departments`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.004ms | 4%
**Rows:** 11 (Planned: 11)
**Cost:** 1.11 (Total: 1.11)

### #4 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.006ms | 6%
**Rows:** 7 (Planned: 7)

### #5 Seq Scan
on `locations`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.035ms | 33%
**Rows:** 7 (Planned: 7)
**Cost:** 1.07 (Total: 1.07)

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: N/A | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|
| #1 Sort | | | |
| #2 └ Nested L... | | | |
| #3 └ Hash J... | | | |
| #4 └ Nes... | | | |
| #5 ├ S... | | | |
| #6 └ ... | | | |

## #1 Sort

by jobs.job_title

*Sort Node* sorts a record set based on the specified sort key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.544ms | 30%
**Rows:** 740 (Planned: 741)
**Cost:** 37.2 (Total: 107)

## #2 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.443ms | 24%
**Rows:** 740 (Planned: 741)
**Cost:** 24.5 (Total: 70)

## #3 Hash Join

on employees.department_id = departments.department_id

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.258ms | 14%
**Rows:** 741 (Planned: 741)
**Cost:** 3.94 (Total: 45.5)

## #4 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.507ms | 28%
**Rows:** 741 (Planned: 741)
**Cost:** 22.9 (Total: 41.5)

## #5 Seq Scan

on employees

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.089ms | 5%
**Rows:** 741 (Planned: 741)
**Cost:** 18.4 (Total: 18.4)

## #6 Memoize

*Memoize Node* is used to cache the results of the inner side of a nested loop. It avoids executing underlying nodes when the results for the current parameters are already in the cache.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0ms | 0%
**Rows:** ~741 (Planned: ~741)
**Cost:** 0.24 (Total: 0.24)
**Loops:** 741

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: N/A | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|

#1 Sort
#2 └ Nested L...
#3 └ Hash J...
#4 └ Has...
#5 ├ S...
#6 └ H...

## #1 Sort

by `jobs.job_title`

*Sort Node* sorts a record set based on the specified sort key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.524ms | 32%
**Rows:** 740 (Planned: 741)
**Cost:** 37.2 (Total: 87.8)

## #2 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.416ms | 26%
**Rows:** 740 (Planned: 741)
**Cost:** 24.5 (Total: 50.6)

## #3 Hash Join

on `employees.department_id = departments.department_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.275ms | 17%
**Rows:** 741 (Planned: 741)
**Cost:** 3.93 (Total: 26.1)

## #4 Hash Join

on `employees.job_id = jobs.job_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.298ms | 18%
**Rows:** 741 (Planned: 741)
**Cost:** 2.62 (Total: 22.2)

## #5 Seq Scan

on `employees`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.102ms | 6%
**Rows:** 741 (Planned: 741)
**Cost:** 18.4 (Total: 18.4)

## #6 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.01ms | 1%
**Rows:** 19 (Planned: 19)
**Cost:** 1.19 (Total: 1.19)

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: N/A | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|
| #1 Sort | | | |
| #2 └ Nested L... | | | |
| #3 └ Hash J... | | | |
| #4 └ Has... | | | |
| #5 ├ S... | | | |
| #6 └ H... | | | |

## #1 Sort

by `jobs.job_title`

*Sort Node* sorts a record set based on the specified sort key.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.491ms | 30%
- **Rows:** 740 (Planned: 741)
- **Cost:** 37.2 (Total: 87.8)

## #2 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.426ms | 26%
- **Rows:** 740 (Planned: 741)
- **Cost:** 24.5 (Total: 50.6)

## #3 Hash Join

on `employees.department_id = departments.department_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.272ms | 17%
- **Rows:** 741 (Planned: 741)
- **Cost:** 3.93 (Total: 26.1)

## #4 Hash Join

on `employees.job_id = jobs.job_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.316ms | 19%
- **Rows:** 741 (Planned: 741)
- **Cost:** 2.62 (Total: 22.2)

## #5 Seq Scan

on `employees`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.105ms | 6%
- **Rows:** 741 (Planned: 741)
- **Cost:** 18.4 (Total: 18.4)

## #6 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

- **Timing:** 0.02ms | 1%
- **Rows:** 19 (Planned: 19)
- **Cost:** 1.19 (Total: 1.19)

Plan Raw Query Stats 1.5.0

Execution time: N/A | Planning time: N/A | Triggers: N/A | Settings

| time | rows | estimation | c |
|---|---|---|---|
| #1 Sort | | | |
| #2 └ Nested L... | | | |
| #3 └ Hash J... | | | |
| #4 └ Has... | | | |
| #5 ├ S... | | | |
| #6 └ H... | | | |

## #1 Sort

by `jobs.job_title`

*Sort Node* sorts a record set based on the specified sort key.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.558ms | 30%
**Rows:** 740 (Planned: 741)
**Cost:** 37.2 (Total: 86.7)

## #2 Nested Loop

*Nested Loop Node* merges two record sets by looping through every record in the first set and trying to find a match in the second set. All matching records are returned.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.552ms | 29%
**Rows:** 740 (Planned: 741)
**Cost:** 24.4 (Total: 49.5)

## #3 Hash Join

on `employees.department_id = departments.department_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.243ms | 13%
**Rows:** 741 (Planned: 741)
**Cost:** 3.93 (Total: 25.1)

## #4 Hash Join

on `employees.job_id = jobs.job_id`

*Hash Join Node* joins two record sets by hashing one of them (using a **Hash Scan**).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.422ms | 22%
**Rows:** 741 (Planned: 741)
**Cost:** 2.62 (Total: 21.2)

## #5 Seq Scan

on `employees`

*Seq Scan Node* finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read).

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.106ms | 6%
**Rows:** 741 (Planned: 741)
**Cost:** 17.4 (Total: 17.4)

## #6 Hash

*Hash Node* generates a hash table from the records in the input recordset. Hash is used by **Hash Join**.

General | IO & Buffers | Output | Workers | Misc

**Timing:** 0.008ms | 0%
**Rows:** 19 (Planned: 19)
**Cost:** 1.19 (Total: 1.19)