



CEFET/RJ



ANÁLISE DE PLANOS DE EXECUÇÃO DE CONSULTAS NOS SBGD

Eduardo Ogasawara
eogasawara@ieee.org
<https://eic.cefet-rj.br/~eogasawara>

Esquema de análise

- DEPARTAMENTO
 - PRIMARY KEY (DNUMERO)
- EMPREGADO
 - PRIMARY KEY (CPF)
 - FOREIGN KEY (GERENTECPF) REFERENCES EMPREGADO (CPF),
 - FOREIGN KEY (DNO) REFERENCES DEPARTAMENTO (DNUMERO)
- DEPT_LOCALIZACOES
 - PRIMARY KEY (DNUMERO,DLOCALIZACAO),
 - FOREIGN KEY (DNUMERO) REFERENCES DEPARTAMENTO (DNUMERO)
- PROJETO
 - PRIMARY KEY (PNUMERO)
 - FOREIGN KEY (DNUM) REFERENCES DEPARTAMENTO (DNUMERO)
- TRABALHA_EM
 - PRIMARY KEY (ECPF,PNO)
 - FOREIGN KEY (ECPF) REFERENCES EMPREGADO (CPF),
 - FOREIGN KEY (PNO) REFERENCES PROJETO (PNUMERO)
- DEPENDENTE
 - PRIMARY KEY (ECPF, NOME_DEPENDENTE),
 - FOREIGN KEY (ECPF) REFERENCES EMPREGADO (CPF)

PostgreSQL

Consultas pela chave primária (empregado a partir do cpf)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view of database objects, with 'projetos' expanded. The main area is the 'Query Editor', which contains the following SQL query:

```
1 SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'
```

Below the query editor, the 'Data Output' tab is active, showing the 'Explain' view of the query plan. The plan consists of a single node:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..1.35 rows=1 width=78) Filter: ((cpf)::bpchar = '999887777'::bpchar)	1

SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'

Consulta por valor (empregado a partir do primeiro nome)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' database selected. The main window shows the Query Editor with the following SQL query:

```
1 SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

The Query Editor also shows the execution plan for the query. The plan is displayed in the 'Analysis' tab, showing a single node:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..1.35 rows=1 width=78) Filter: ((pnome)::text = 'Alicia'::text)	1

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

Sub-consulta na projeção

(lista de departamentos com seus respectivos gerentes)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view of the database structure, with 'projetos' selected. The main area displays a SQL query in the 'Query Editor' and its execution plan in the 'Explain' tab.

```
1 SELECT D.*,
2     (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME
3 FROM DEPARTAMENTO D
4
5
```

The execution plan shows two nodes:

#	Node	Rows
1.	→ Seq Scan on departamento as d (cost=0..5.08 rows=3 width=117)	3
2.	→ Seq Scan on empregado as e (cost=0..1.35 rows=1 width=8) Filter: ((cpf)::bpchar = (d.gercpf)::bpchar)	1

```
SELECT D.*,
       (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME
FROM DEPARTAMENTO D
```

Consulta equivalente em junção (lista de departamentos com seus respectivos gerentes)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view with 'projetos' expanded. The main area displays a SQL query in the 'Query Editor' and its execution plan in the 'Data Output' pane.

```
1 SELECT D.*, E.PNOME
2 FROM DEPARTAMENTO D, EMPREGADO E
3 WHERE E.CPF = D.GERCPF
4
5
```

The execution plan table is as follows:

#	Node	Rows
1.	→ Hash Inner Join (cost=1.07..2.48 rows=3 width=47) Hash Cond: ((e.cpf)::bpchar = (d.gercpf)::bpchar)	3
2.	→ Seq Scan on empregado as e (cost=0..1.28 rows=28 width=20)	28
3.	→ Hash (cost=1.03..1.03 rows=3 width=39)	3
4.	→ Seq Scan on departamento as d (cost=0..1.03 rows=3 width=39)	3

```
SELECT D.*, E.PNOME
FROM DEPARTAMENTO D, EMPREGADO E
WHERE E.CPF = D.GERCPF
```

Árvore de consulta equivalente em junção (lista de departamentos com seus respectivos gerentes)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view with 'projetos' selected. The 'Query Editor' pane contains the following SQL query:

```
1 SELECT D.*, E.PNOME
2 FROM DEPARTAMENTO D, EMPREGADO E
3 WHERE E.CPF = D.GERCPF
4
5
```

Below the query, the 'Explain' tab is active, showing a graphical execution plan. The plan consists of three main components:

- departamento**: A table icon representing the source of the 'D' table.
- empregado**: A table icon representing the source of the 'E' table.
- Hash**: A table icon representing the intermediate hash table created for the join.
- Hash Inner Join**: A table icon representing the final join operation.

Arrows indicate the flow of data: from 'departamento' to 'Hash', from 'empregado' to 'Hash Inner Join', and from 'Hash' to 'Hash Inner Join'.

```
SELECT D.*, E.PNOME
FROM DEPARTAMENTO D, EMPREGADO E
WHERE E.CPF = D.GERCPF
```

Subconsulta não correlacionada com in (empregados com dependentes)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' schema selected. The main window shows the Query Editor with the following SQL query:

```
1 SELECT *
2 FROM EMPREGADO E
3 WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
4
5
```

Below the query editor, the 'Explain' tab is active, showing the execution plan. The plan consists of five nodes:

#	Node	Rows
1.	→ Hash Inner Join (cost=8.61..10.21 rows=23 width=78) Hash Cond: ((e.cpf)::bpchar = (d.ecpf)::bpchar)	23
2.	→ Seq Scan on empregado as e (cost=0..1.28 rows=28 width=78)	28
3.	→ Hash (cost=8.32..8.32 rows=23 width=12)	23
4.	→ Aggregate (cost=8.09..8.32 rows=23 width=12)	23
5.	→ Seq Scan on dependente as d (cost=0..7.07 rows=407 width=12)	407

```
SELECT *
FROM EMPREGADO E
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Subconsulta correlacionada com in (empregados com dependentes do mesmo sexo)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with the 'projetos' schema selected. The main window shows the 'Query Editor' with the following SQL query:

```
1 SELECT *
2 FROM EMPREGADO E
3 WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D WHERE D.SEXO = E.SEXO)
4
5
```

Below the query editor, the 'Data Output' tab is active, showing the 'Analysis' view of the execution plan. The plan consists of two nodes:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..114.72 rows=14 width=78) Filter: (SubPlan 1)	14
2.	→ Seq Scan on dependente as d (cost=0..8.09 rows=4 width=12) Filter: ((sexo)::text = (e.sexo)::text)	4

```
SELECT *
FROM EMPREGADO E
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D WHERE D.SEXO = E.SEXO)
```

Subconsulta correlacionada com exists (empregados com dependentes do mesmo sexo)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with 'projetos' expanded to show various objects like 'Casts', 'Catalogs', etc. The main window is in 'Query Editor' mode, displaying the following SQL query:

```
1 SELECT *
2 FROM EMPREGADO E
3 WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
4
5
```

Below the query editor, the 'Explain' tab is active, showing the execution plan. The plan consists of five nodes:

#	Node	Rows
1.	→ Hash Inner Join (cost=10.13..11.64 rows=7 width=78) Hash Cond: (((e.cpf)::bpchar = (d.ecpf)::bpchar) AND ((e.sexo)::text = (d.sexo)::text))	7
2.	→ Seq Scan on empregado as e (cost=0..1.28 rows=28 width=78)	28
3.	→ Hash (cost=9.52..9.52 rows=41 width=14)	41
4.	→ Aggregate (cost=9.11..9.52 rows=41 width=14)	41
5.	→ Seq Scan on dependente as d (cost=0..7.07 rows=407 width=14)	407

```
SELECT *
FROM EMPREGADO E
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
```

Consulta com agregação

(quantidade de empregados que trabalham em mais de um projeto)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view of the database structure, with 'projetos' expanded. The main area displays a SQL query in the 'Query Editor' and its execution plan in the 'Data Output' pane.

```
1 SELECT E.PNOME, COUNT(*) AS QTD
2 FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
3 GROUP BY E.PNOME HAVING COUNT(*) > 1
4
5
```

The execution plan table is as follows:

#	Node	ROWS
1.	→ Aggregate (cost=2.92..3.12 rows=5 width=16) Filter: (count(*) > 1)	5
2.	→ Hash Inner Join (cost=1.63..2.84 rows=16 width=8) Hash Cond: ((te.ecpf)::bpchar = (e.cpf)::bpchar)	16
3.	→ Seq Scan on trabalha_em as te (cost=0..1.16 rows=16 width=12)	16
4.	→ Hash (cost=1.28..1.28 rows=28 width=20)	28
5.	→ Seq Scan on empregado as e (cost=0..1.28 rows=28 width=20)	28

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Criação de índice

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows a tree view of database objects, with the 'projetos' database selected. The main area is divided into a 'Query Editor' and a 'Messages' pane. The Query Editor contains the SQL command: `CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)`. The Messages pane shows the execution result: `CREATE INDEX` and `Query returned successfully in 120 msec.`. A 'Maintenance' notification box is visible in the bottom right, indicating a VACUUM operation on the 'projetos' database, which took 13.88 seconds and was successfully completed.

CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)

Índice nem sempre é usado (busca pelo empregado a partir do primeiro nome)

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane shows a tree view of database objects, with 'projetos' expanded. The main area is the 'Query Editor', which contains the following SQL query:

```
1 SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

Below the query editor, the 'Data Output' tab is active, showing the 'Analysis' view of the execution plan. The plan consists of a single node:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..1.35 rows=1 width=78) Filter: ((pnome)::text = 'Alicia')::text	1

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

MySQL

Consultas pela chave primária (empregado a partir do cpf)

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the Navigator with a Schemas view, displaying a tree structure for 'projeto9' and 'projeto9' (expanded) with tables: DEPARTAMENTO, DEPENDENTE, DEPT_LOCALIZACOE, EMPREGADO, PROJETO, and TRABALHA_EM. The main editor shows a query: `EXPLAIN ANALYZE SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'`. The bottom panel shows the 'Form Editor' with the output: `-> Rows fetched before execution (cost=0..0 rows=1) (actual time=188e-6..292e-6 rows=1 loops=1)`. The status bar at the bottom indicates 'Query Completed'.

SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'

Consulta por valor (empregado a partir do primeiro nome)

The screenshot shows the MySQL Workbench interface. On the left, the 'Navigator' pane displays the 'projetos' schema with a tree view of tables including DEPARTAMENTO, DEPENDENTE, DEPT_LOCALIZACOES, EMPREGADO, PROJETO, and TRABALHA_EM. The 'Query Editor' pane contains the following SQL query:

```
1 EXPLAIN ANALYZE SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

The 'Form Editor' pane shows the EXPLAIN output for the query:

```
EXPLAIN:
-> Filter: (E.PNOME = 'Alicia') (cost=3.05 rows=2.8) (actual time=0.111..0.114 rows=1 loops=1)
-> Table scan on E (cost=3.05 rows=28) (actual time=0.0432..0.103 rows=28 loops=1)
```

The 'Information' pane at the bottom left shows 'Schema: projetos'. The 'Result Grid' pane at the bottom right is currently empty, and the 'Form Editor' pane is active.

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

Sub-consulta na projeção

(lista de departamentos com seus respectivos gerentes)

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the Schemas tree with 'grupo9' and 'projetos' (containing tables like DEPARTAMENTO, EMPREGADO, etc.). The main editor displays a query:

```
1 • EXPLAIN ANALYZE SELECT D.*,
2     (SELECT PNOOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOOME
3 FROM DEPARTAMENTO D
4
5
6
7
```

The bottom panel shows the EXPLAIN output:

```
EXPLAIN:
-> Table scan on D (cost=0.55 rows=3) (actual time=0.0233..0.0286 rows=3 loops=1)
-> Select #2 (subquery in projection; dependent)
   -> Single-row index lookup on E using PRIMARY (CPF=D.GERCPF) (cost=0.35 rows=1) (actual
       time=0.00927..0.00937 rows=1 loops=3)
```

The interface also shows 'No object selected' in the Information panel and 'Query Completed' at the bottom.

```
SELECT D.*,
 (SELECT PNOOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOOME
FROM DEPARTAMENTO D
```

Consulta com agregação (quantidade de empregados que trabalham em mais de um projeto)

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'projeto9' expanded to show tables: DEPARTAMENTO, DEPENDENTE, DEPT_LOCALIZACOE, EMPREGADO, PROJETO, and TRABALHA_EM. The main editor shows a query in the 'Form Editor' tab:

```
1 • EXPLAIN ANALYZE SELECT E.PNOME, COUNT(*) AS QTD
2 FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
3 GROUP BY E.PNOME HAVING COUNT(*) > 1
4
5
6
7
8
```

The 'EXPLAIN' output is displayed in the 'Result Grid' tab:

```
EXPLAIN:
-> Filter: (`count(0)` > 1) (actual time=0.178..0.182 rows=6 loops=1)
-> Table scan on <temporary> (actual time=0.175..0.178 rows=8 loops=1)
-> Aggregate using temporary table (actual time=0.172..0.172 rows=8 loops=1)
-> Nested loop inner join (cost=7.45 rows=16) (actual time=0.0474..0.116 rows=16 loops=1)
```

The status bar at the bottom indicates 'Query Completed'.

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Junção com busca por valor (último nome e horas trabalhadas por projeto pelo primeiro nome)

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'grupo9' and 'projetos' (containing tables like EMPREGADO, PROJETO, and TRABALHA_EM). The main editor shows a query:

```
1 • EXPLAIN ANALYZE SELECT E.UNOME, TE.HORAS
2 FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
3 WHERE E.PNOME = 'Alicia'
4
5
```

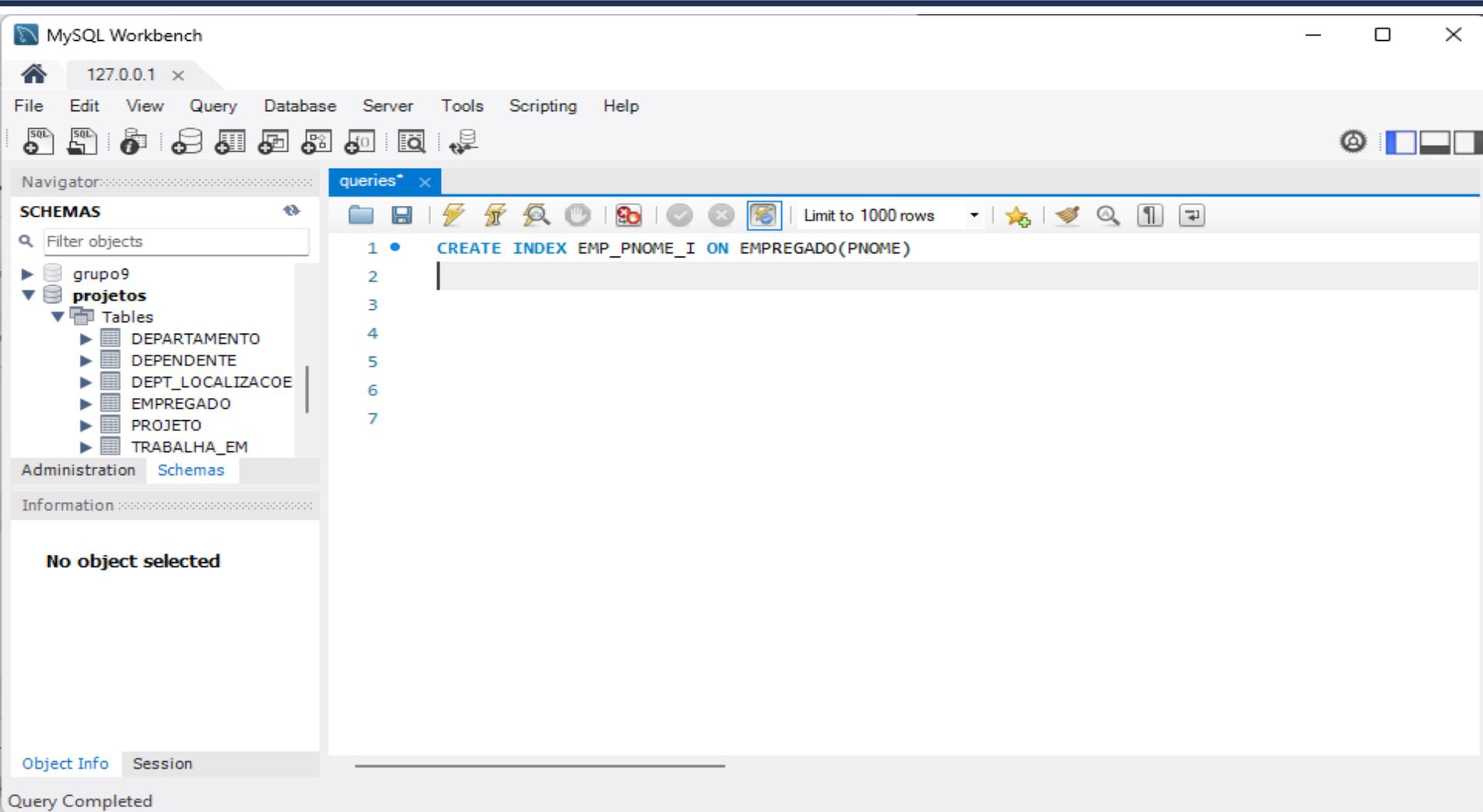
The 'Form Editor' below the query displays the EXPLAIN output:

```
EXPLAIN:
-> Nested loop inner join (cost=5.01 rows=5.6) (actual time=0.0983..0.104 rows=2 loops=1)
  -> Filter: (E.PNOME = 'Alicia') (cost=3.05 rows=2.8) (actual time=0.0726..0.0743 rows=1 loops=1)
    -> Table scan on E (cost=3.05 rows=28) (actual time=0.042..0.0653 rows=28 loops=1)
      -> Index lookup on TE using PRIMARY (ECPF=E.CPF) (cost=0.571 rows=2) (actual time=0.0229..0.026 rows=2)
```

The bottom status bar indicates 'Query Completed'.

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

Criação de índice



CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)

Uso tradicional do índice (busca pelo empregado a partir do primeiro nome)

The screenshot shows the MySQL Workbench interface. The main window displays a query in the SQL editor:

```
1 • EXPLAIN ANALYZE SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

The query has been executed, and the output is shown in the 'Form Editor' tab. The output is an 'EXPLAIN' statement:

```
EXPLAIN: -> Index lookup on E using EMP_PNOME_I (PNOME='Alicia') (cost=0.35 rows=1) (actual time=0.0443..0.0511 rows=1 loops=1)
```

The interface also shows a 'Navigator' pane on the left with a tree view of schemas. The 'projetos' schema is expanded, showing tables: DEPARTAMENTO, DEPENDENTE, DEPT_LOCALIZACOE, EMPREGADO, PROJETO, and TRABALHA_EM. The 'Administration' and 'Schemas' tabs are visible at the bottom of the Navigator pane.

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

Junção tradicional

(índice do nome do empregado foi usado)

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'projeto9' expanded to show tables: DEPARTAMENTO, DEPENDENTE, DEPT_LOCALIZACOE, EMPREGADO, PROJETO, and TRABALHA_EM. The main editor shows a query in the 'Form Editor' tab:

```
1 • EXPLAIN ANALYZE SELECT E.UNOME, TE.HORAS
2 FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
3 WHERE E.PNOME = 'Alicia'
```

The 'EXPLAIN' output is displayed in the 'Result Grid' tab:

```
EXPLAIN:
-> Nested loop inner join (cost=1.05 rows=2) (actual time=0.0634..0.0712 rows=2 loops=1)
-> Index lookup on E using EMP_PNOME_I (PNOME='Alicia') (cost=0.35 rows=1) (actual time=0.0432..0.0456 rows=1 loops=1)
-> Index lookup on TE using PRIMARY (ECPF=E.CPF) (cost=0.7 rows=2) (actual time=0.0179..0.0225 rows=2)
```

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

SQL Server

Consulta por valor (empregado a partir do primeiro nome)

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query editor with the following SQL statement:

```
SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

Below the query editor, the execution plan is visible, showing a **Clustered Index Scan (Clustered)** operation on the **[EMPREGADO].[PK_EMPREGAD_C1F89730...]** index. The cost is indicated as **Cost: 100 %**.

The **Properties** window on the right provides detailed information about the query execution:

SELECT	
Misc	
Cached plan size	24 KB
CardinalityEstimationMod	160
CompileCPU	0
CompileMemory	136
CompileTime	0
Estimated Number of Row	0
Estimated Number of Row	3,5
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0033128
MemoryGrantInfo	
Optimization Level	TRIVIAL
OptimizerHardwareDepen	
OptimizerStatsUsage	

The status bar at the bottom indicates: **Query executed successfully.** | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

Sub-consulta na projeção (lista de departamentos com seus respectivos gerentes)

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query editor with the following SQL code:

```
SELECT D.*,  
    (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME  
FROM DEPARTAMENTO D
```

Below the query editor, the Execution plan tab is active, showing a query plan for "Query 1". The plan consists of the following operations:

- SELECT** (Cost: 0 %)
- Compute Scalar** (Cost: 0 %)
- Nested Loops (Left Outer Join)** (Cost: 0 %)
- Clustered Index Scan (Clustered) [DEPARTAMENTO].[PK_DEPARTAM_FAE85...]** (Cost: 48 %)
- Clustered Index Seek (Clustered) [EMPREGADO].[PK_EMPREGAD_C1F89730...]** (Cost: 52 %)

The Properties window on the right shows the following details for the selected operation:

Properties	
SELECT	
Misc	
Cached plan size	24 KB
CardinalityEstimationMod	160
CompileCPU	3
CompileMemory	232
CompileTime	3
Estimated Number of Row	0
Estimated Number of Row	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0068974
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	

At the bottom of the Properties window, the text reads: "Estimated Subtree Cost Estimated cumulative cost of this operation and all child operations."

The status bar at the bottom of the window indicates: "Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows"

```
SELECT D.*,  
    (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME  
FROM DEPARTAMENTO D
```

Consulta equivalente em junção (lista de departamentos com seus respectivos gerentes)

The screenshot displays the Microsoft SQL Server Management Studio interface. The top window shows the query text:

```
SELECT D.*, E.PNOME  
FROM DEPARTAMENTO D, EMPREGADO E  
WHERE E.CPF = D.GERCPF
```

Below the query, the execution plan is visible, showing a Nested Loops (Inner Join) operator with a cost of 0%. It is connected to two scan/seek operators: a Clustered Index Scan (Clustering) on the DEPARTAMENTO table with a cost of 48%, and a Clustered Index Seek (Clustering) on the EMPREGADO table with a cost of 52%.

On the right side, the Properties window is open, showing the following details for the SELECT operation:

Property	Value
Cached plan size	24 KB
CardinalityEstimationMod	160
CompileCPU	2
CompileMemory	200
CompileTime	2
Estimated Number of Row	0
Estimated Number of Row	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0068971
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	

At the bottom of the Properties window, a summary states: **Estimated Subtree Cost** Estimated cumulative cost of this operation and all child operations.

The status bar at the bottom indicates: Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows

```
SELECT D.*, E.PNOME  
FROM DEPARTAMENTO D, EMPREGADO E  
WHERE E.CPF = D.GERCPF
```

Subconsulta não correlacionada com in (empregados com dependentes)

SQLQuery1.sql - 127.0.0.1.projetos (sa (70))* - Microsoft SQL Server Management Studio

```
SELECT *  
FROM EMPREGADO E  
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Query 1: Query cost (relative to the batch): 100%

```
SELECT * FROM EMPREGADO E WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Execution Plan:

- SELECT (Cost: 0 %)
- Nested Loops (Left Semi Join) (Cost: 1 %)
 - Clustered Index Scan (Clustered) [EMPREGADO].[PK_EMPREGAD_C1F89730...] (Cost: 30 %)
 - Clustered Index Seek (Clustered) [DEPENDENTE].[PK_DEPENDEN_2BB4A8D...] (Cost: 69 %)

Properties:

Property	Value
Cached plan size	24 KB
CardinalityEstimationMod	160
CompileCPU	6
CompileMemory	288
CompileTime	6
Estimated Number of Row	0
Estimated Number of Row	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0109816
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	

Estimated Subtree Cost
Estimated cumulative cost of this operation and all child operations.

Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows

```
SELECT *  
FROM EMPREGADO E  
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Subconsulta correlacionada com exists (empregados com dependentes do mesmo sexo)

SQLQuery1.sql - 127.0.0.1.projetos (sa (70))* - Microsoft SQL Server Management Studio

```
SELECT *  
FROM EMPREGADO E  
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
```

Query 1: Query cost (relative to the batch): 100%

SELECT * FROM EMPREGADO E WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF...

Execution plan details:

- SELECT (Cost: 0 %)
- Nested Loops (Left Semi Join) (Cost: 1 %)
- Clustered Index Scan (Clustered) [EMPREGADO].[PK_EMPREGAD_C1F89730...] (Cost: 30 %)
- Clustered Index Seek (Clustered) [DEPENDENTE].[PK_DEPENDEN_2BB4A8D...] (Cost: 69 %)

Properties window:

Misc	
Cached plan size	32 KB
CardinalityEstimationMod	160
CompileCPU	7
CompileMemory	336
CompileTime	7
Estimated Number of Row	0
Estimated Number of Row	28
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0109951
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	

Estimated Subtree Cost
Estimated cumulative cost of this operation and all child operations.

Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows

```
SELECT *  
FROM EMPREGADO E  
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
```

Consulta com agregação (quantidade de empregados que trabalham em mais de um projeto)

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query window with the following SQL code:

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Below the query window, the Execution plan is visible. The plan consists of the following operators from right to left:

- Clustered Index Scan (Clustered) [TRABALHA_EM].[PK_TRABALHA_CB6423... Cost: 16 %
- Clustered Index Seek (Clustered) [EMPREGADO].[PK_EMPREGAD_C1F89730... Cost: 26 %
- Nested Loops (Inner Join) Cost: 0 %
- Sort Cost: 56 %
- Stream Aggregate (Aggregate) Cost: 0 %
- Compute Scalar Cost: 0 %
- Filter Cost: 0 %
- SELECT Cost: 0 %

The Properties window on the right shows the following details for the SELECT operator:

Properties	
SELECT	
Misc	
Cached plan size	32 KB
CardinalityEstimationMod	160
CompileCPU	4
CompileMemory	296
CompileTime	5
Estimated Number of Row	0
Estimated Number of Row	2,28486
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,020494
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	
Estimated Subtree Cost	
Estimated cumulative cost of this operation and all child operations.	

The status bar at the bottom indicates: Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Junção com busca por valor (último nome e horas trabalhadas por projeto pelo primeiro nome)

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query in the editor:

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

Below the query editor, the Execution plan is visible. The plan shows a 'SELECT' operator (Cost: 0%) connected to a 'Nested Loops (Inner Join)' operator (Cost: 1%). The 'Nested Loops' operator is connected to two 'Clustered Index Scan (Clustered)' operators. The top scan is for '[EMPREGADO].[PK_EMPREGAD_C1F89730...]' (Cost: 47%) and the bottom scan is for '[TRABALHA_EM].[PK_TRABALHA_CB6423...]' (Cost: 52%).

The Properties window on the right shows the following details for the selected 'SELECT' operator:

Misc	
Cached plan size	24 KB
CardinalityEstimationMod	160
CompileCPU	3
CompileMemory	240
CompileTime	3
Estimated Number of Row	0
Estimated Number of Row	7
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0070377
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDepen	
OptimizerStatsUsage	

At the bottom of the Properties window, the text reads: "Estimated Subtree Cost Estimated cumulative cost of this operation and all child operations."

The status bar at the bottom of the window indicates: "Query executed successfully. | 127.0.0.1 (16.0 RTM) | sa (70) | projetos | 00:00:00 | 0 rows"

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

Índice de cobertura de consulta

- O índice de cobertura de consulta possibilita processar a consulta sem selecionar a tabela
- Todos os atributos usados na consulta para uma Tabela T estão presentes no índice
- A ordem de criação dos campos no índice é relevante
 - Primeiro os atributos de seleção (mais restritivos primeiro)
 - Segundo os atributos usados na junção
 - Terceiro os atributos usados na projeção

Índice de cobertura de consulta

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query editor with the following SQL statement:

```
CREATE INDEX EMP_COVER ON EMPREGADO(PNOME) INCLUDE (CPF, UNOME)
```

The query has been executed successfully, as indicated by the status bar at the bottom: "Query executed successfully." The completion time is 2024-11-21T08:09:50.8609699-03:00. The status bar also shows the server name "127.0.0.1 (16.0 RTM)", the user "sa (70)", the database "projetos", and the execution time "00:00:00" with "0 rows" returned.

The Properties window on the right shows the current connection parameters for the connection "127.0.0.1 (sa)".

Aggregate Status	
Connection failures	
Elapsed time	00:00:00.3570063
Finish time	21/11/2024 08:09:50
Name	127.0.0.1
Rows returned	0
Start time	21/11/2024 08:09:50
State	Open

Connection	
Connection name	127.0.0.1 (sa)

Connection Details	
Connection elapsed time	00:00:00.3570063
Connection encryption	Not encrypted
Connection finish time	21/11/2024 08:09:50

Name
The name of the connection.

CREATE INDEX EMP_COVER ON EMPREGADO(PNOME) INCLUDE (CPF, UNOME)

Uso do índice de cobertura de consulta (Consulta de alocação por data de nascimento com índice)

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query in the SQL editor:

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

Below the query, the Execution plan is visible. The plan shows a Nested Loops (Inner Join) operation with a cost of 0. This operation is connected to two index seek operations, each with a cost of 50:

- Index Seek (NonClustered) on [EMPREGADO].[EMP_COVER] [E]
- Clustered Index Seek (Cluste...) on [TRABALHA_EM].[PK_TRABALHA_...]

The Properties window on the right shows details for the SELECT operation:

SELECT	
Misc	
Cached plan size	24 KB
CardinalityEstimate	160
CompileCPU	3
CompileMemory	248
CompileTime	3
Estimated Number	0
Estimated Number	2
Estimated Operato	0 (0%)
Estimated Subtree	0,0065757
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardware	
OptimizerStatsUsa	
QueryHash	0x86586179216741C
Estimated Subtree Cost	
Estimated cumulative cost of this operatio...	

The status bar at the bottom indicates: Query executed successfully. | 127.0.0.1,1433 (16.0 RTM) | sa (58) | projetos | 00:00:00 | 0 rows

```
SELECT E.UNOME, TE.HORAS
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
WHERE E.PNOME = 'Alicia'
```

Oracle

Consultas pela chave primária (empregado a partir do cpf)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Query Builder:

```
SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'
```

Below the query, the execution plan is shown in a table format:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
INDEX	SYS_C004766	UNIQUE SCAN	1	1

The execution plan also includes a tree view showing the query structure and execution details:

- SELECT STATEMENT
 - TABLE ACCESS
 - INDEX
 - Access Predicates
 - E.CPF='999887777'
 - Other XML
 - {info}
 - info type="db_version"
 - 10.2.0.1
 - info type="parse_schema"

SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'

Consulta por valor (empregado a partir do primeiro nome)

The screenshot shows the Oracle SQL Developer interface. The main window displays the following SQL query:

```
SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
```

Below the query editor, the 'Query Result' tab is active, showing the execution plan for the query. The execution plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
TABLE ACCESS	EMPREGADO	FULL	1	3

The execution plan also includes a tree view of the query structure:

- SELECT STATEMENT
 - TABLE ACCESS (EMPREGADO)
 - Filter Predicates
 - E.PNOME='Alicia'
 - Other XML
 - {info}
 - info type="db_version"
 - 10.2.0.1
 - info type="parse_schema"
 - "PROJETOS"

SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'

Sub-consulta na projeção

(lista de departamentos com seus respectivos gerentes)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Query Builder:

```
SELECT D.*,  
  (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME  
FROM DEPARTAMENTO D
```

Below the query, the Execution Plan is shown in a table format:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	3
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
INDEX	SYS_C004766	UNIQUE SCAN	1	1
Access Predicates				
E.CPF=:B1				
TABLE ACCESS	DEPARTAMENTO	FULL	3	3
Other XML				
{info}				
info type="db_version"				
10.2.0.1				

```
SELECT D.*,  
  (SELECT PNOME FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNOME  
FROM DEPARTAMENTO D
```

Consulta equivalente em junção (lista de departamentos com seus respectivos gerentes)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Worksheet:

```
SELECT D.*, E.PNOME  
FROM DEPARTAMENTO D, EMPREGADO E  
WHERE E.CPF = D.GERCPF
```

Below the query, the Execution Plan is shown. The plan consists of a SELECT STATEMENT, a NESTED LOOPS join, and two TABLE ACCESS operations. The second TABLE ACCESS operation uses an INDEX (SYS_C004766) with an Access Predicate (E.CPF=D.GERCPF).

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	6
NESTED LOOPS			3	6
TABLE ACCESS	DEPARTAMENTO	FULL	3	3
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
INDEX	SYS_C004766	UNIQUE SCAN	1	0

```
SELECT D.*, E.PNOME  
FROM DEPARTAMENTO D, EMPREGADO E  
WHERE E.CPF = D.GERCPF
```

Subconsulta não correlacionada com in (empregados com dependentes)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Worksheet:

```
FROM EMPREGADO E  
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Below the query, the Execution Plan is visible. The plan shows a HASH JOIN operation with a RIGHT SEMI join type. The cardinality is 23 and the cost is 7. The plan also shows table access for DEPENDENTE (407 rows, cost 3) and EMPREGADO (28 rows, cost 3).

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			23	7
HASH JOIN		RIGHT SEMI	23	7
Access Predicates E.CPF=D.ECPF				
TABLE ACCESS	DEPENDENTE	FULL	407	3
TABLE ACCESS	EMPREGADO	FULL	28	3

```
SELECT *  
FROM EMPREGADO E  
WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D)
```

Subconsulta correlacionada com exists (empregados com dependentes do mesmo sexo)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Worksheet:

```
SELECT *  
FROM EMPREGADO E  
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
```

Below the query, the 'Query Result' tab is active, showing the execution plan. The plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
HASH JOIN		RIGHT SEMI	1	7
Access Predicates				
AND				
D.ECPF=E.CPF				
D.SEXO=E.SEXO				
TABLE ACCESS	DEPENDENTE	FULL	407	3
TABLE ACCESS	EMPREGADO	FULL	28	3
Other XML				
{info}				

```
SELECT *  
FROM EMPREGADO E  
WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
```

Consulta com agregação (quantidade de empregados que trabalham em mais de um projeto)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Query Builder:

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Below the query, the Execution Plan is shown. The plan consists of several operations:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			16	6
FILTER				
Filter Predicates				
COUNT(*)>1				
HASH		GROUP BY	16	6
NESTED LOOPS			16	5
TABLE ACCESS	EMPREGADO	FULL	28	3
INDEX	SYS_C004778	RANGE SCAN	1	1
Access Predicates				
TE.ECPF=E.CPF				

```
SELECT E.PNOME, COUNT(*) AS QTD
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF
GROUP BY E.PNOME HAVING COUNT(*) > 1
```

Junção com busca por valor

(último nome e horas trabalhadas por projeto pelo primeiro nome)

The screenshot shows the Oracle SQL Developer interface. The main window displays a query in the Worksheet:

```
SELECT E.UNOME, TE.HORAS  
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF  
WHERE E.PNOME = 'Alicia'
```

Below the query, the Execution Plan is shown. The plan consists of the following operations:

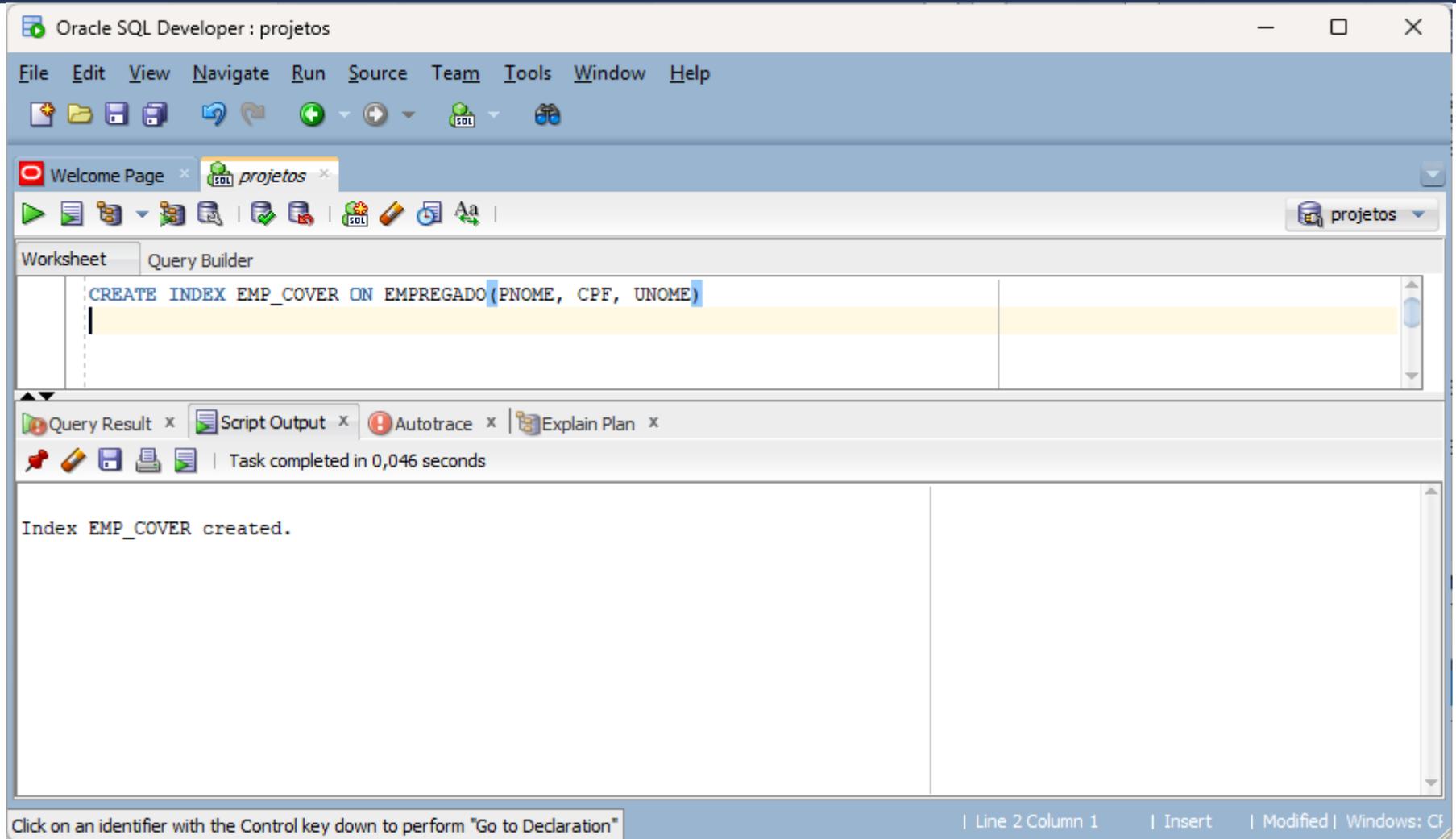
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	7
HASH JOIN			2	7
Access Predicates TE.ECPF=E.CPF				
TABLE ACCESS	EMPREGADO	FULL	1	3
Filter Predicates E.PNOME='Alicia'				
TABLE ACCESS	TRABALHA_EM	FULL	16	3
Other XML				
{info}				

```
SELECT E.UNOME, TE.HORAS  
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF  
WHERE E.PNOME = 'Alicia'
```

Índice de cobertura de consulta

- O índice de cobertura de consulta possibilita processar a consulta sem selecionar a tabela
- Todos os atributos usados na consulta para uma Tabela T estão presentes no índice
- A ordem de criação dos campos no índice é relevante
 - Primeiro os atributos de seleção (mais restritivos primeiro)
 - Segundo os atributos usados na junção
 - Terceiro os atributos usados na projeção

Índice de cobertura de consulta



CREATE INDEX EMP_COVER ON EMPREGADO(PNOME, CPF, UNOME)

Uso do índice de cobertura de consulta (Consulta de alocação por data de nascimento com índice)

The screenshot displays the Oracle SQL Developer interface. The main window shows a query being executed in the 'Query Builder' tab. The query is:

```
SELECT E.UNOME, TE.HORAS  
FROM EMPREGADO E JOIN TRABALHA_EM TE ON E.CPF = TE.ECPF  
WHERE E.PNOME = 'Alicia'
```

The execution time is 0,055 seconds. Below the query, the 'Query Result' tab is active, showing the execution plan (EXPLAIN PLAN) for the query. The plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	4
NESTED LOOPS			2	4
TABLE ACCESS	TRABALHA_EM	FULL	16	3
INDEX	EMP_COVER	RANGE SCAN	1	1

The execution plan diagram shows the following structure:

- SELECT STATEMENT
 - NESTED LOOPS
 - TABLE ACCESS (TRABALHA_EM)
 - INDEX (EMP_COVER) with Access Predicates:
 - AND
 - E.PNOME='Alicia'
 - TE.ECPF=E.CPF

The status bar at the bottom indicates the current cursor position: Line 4 Column 1. A tooltip at the bottom left reads: "Click on an identifier with the Control key down to perform 'Go to Declaration'".

Referências

