



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. The left sidebar displays a tree view of servers and databases. The main window is titled 'bd/postgres@regulus.eic.cefet-rj.br *' and contains a 'Query Editor' with the following SQL query:

```
1 select * from empregado e where e.cpf = '999887777'
```

Below the query editor, the 'Explain' tab is active, showing the execution plan:

#	Node	Rows
1.	→ Index Scan using empregado_pkey on empregado as e (cost=0.14..8.16 rows=1 width=288) Index Cond: ((cpf)::bpchar = '999887777'::bpchar)	1

A green notification box at the bottom right states: 'Successfully run. Total query runtime: 64 msec. 1 rows affected.'

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 servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. The main window shows the 'Query Editor' with the following SQL query:

```
1 select * from empregado e where e.pnome = 'Alicia'
```

Below the query editor, the 'Explain' tab is active, showing the execution plan:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..13.12 rows=1 width=288) 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. The left sidebar displays a tree view of servers and databases. The main window is titled 'bd/postgres@regulus.eic.cefet-rj.br *' and contains a 'Query Editor' with the following SQL query:

```
1 select d.*,
2     (select pnome from empregado e where e.cpf = d.gercpf) as pnome
3 from departamento d
4
5
```

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

#	Node	Rows
1.	→ Seq Scan on departamento as d (cost=0..4423.15 rows=540 width=200)	540
2.	→ Index Scan using empregado_pkey on empregado as e (cost=0.14..8.16 rows=1 width=78) Index Cond: ((cpf)::bpchar = (d.gercpf)::bpchar)	1

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

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. The main window shows the 'Query Editor' with the following SQL query:

```
1 select d.*, e.pnome
2 from departamento d, empregado e
3 where e.cpf = d.gercpf
4
```

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

#	Node	Rows
1.	→ Hash Inner Join (cost=15.62..32.46 rows=540 width=200) Hash Cond: ((d.gercpf)::bpchar = (e.cpf)::bpchar)	540
2.	→ Seq Scan on departamento as d (cost=0..15.4 rows=540 width=122)	540
3.	→ Hash (cost=12.5..12.5 rows=250 width=110)	250
4.	→ Seq Scan on empregado as e (cost=0..12.5 rows=250 width=110)	250

```
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. The Query Editor contains the following SQL query:

```
1 select d.*, e.pnome
2 from departamento d, empregado e
3 where e.cpf = d.gercpf
4
```

The Execution Plan shows a Hash Inner Join. A detailed window for the Hash Inner Join node is open on the right, showing the following properties:

Property	Value
Node Type	Hash Join
Parallel Aware	false
Join Type	Inner
Startup Cost	15.62
Total Cost	32.46
Plan Rows	540
Plan Width	200
Inner Unique	true
Hash Cond	((d.gercpf)::bpchar = (e.cpf)::bpchar)
serial	1

```
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 servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. 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
```

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

#	Node	Rows
1.	→ Hash Inner Join (cost=18.88..33.42 rows=125 width=288) Hash Cond: ((e.cpf)::bpchar = (d.ecpf)::bpchar)	125
2.	→ Seq Scan on empregado as e (cost=0..12.5 rows=250 width=288)	250
3.	→ Hash (cost=16.38..16.38 rows=200 width=32)	200
4.	→ Aggregate (cost=14.38..16.38 rows=200 width=32)	200
5.	→ Seq Scan on dependente as d (cost=0..13.5 rows=350 width=32)	350

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

Subconsulta não correlacionada com exists (empregados com dependentes)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure for 'regulus.eic.cefet-rj.br', with the 'bd' database selected. The main window shows the 'Query Editor' with the following SQL query:

```
1 select *
2 from empregado e
3 where exists (
4     select * from dependente d where e.cpf = d.ecpf
5 )
6
7
```

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

#	Node	Rows
1.	→ Hash Inner Join (cost=18.88..33.42 rows=125 width=288) Hash Cond: ((e.cpf)::bpchar = (d.ecpf)::bpchar)	125
2.	→ Seq Scan on empregado as e (cost=0..12.5 rows=250 width=288)	250
3.	→ Hash (cost=16.38..16.38 rows=200 width=32)	200
4.	→ Aggregate (cost=14.38..16.38 rows=200 width=32)	200
5.	→ Seq Scan on dependente as d (cost=0..13.5 rows=350 width=32)	350

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

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 servers and databases, with the 'bd' database selected. The main window shows the 'Query Editor' with the following SQL query:

```
1 select *
2 from empregado e
3 where e.cpf in (
4     select d.ecpf from dependente d where d.sexo = e.sexo
5 )
6
7
```

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

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..1810.63 rows=125 width=288) Filter: (SubPlan 1)	125
2.	→ Seq Scan on dependente as d (cost=0..14.38 rows=2 width=32) Filter: ((sexo)::text = (e.sexo)::text)	2

```
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. On the left, the server tree is expanded to show the 'bd' database. The main window displays a SQL query in the Query Editor:

```
1 select *
2 from empregado e
3 where exists (
4     select * from dependente d where d.ecpf = e.cpf and d.sexo = e.sexo
5 )
6
7
```

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=20.25..34.82 rows=62 width=288) Hash Cond: (((e.cpf)::bpchar = (d.ecpf)::bpchar) AND ((e.sexo)::text = (d.sexo)::text))	62
2.	→ Seq Scan on empregado as e (cost=0..12.5 rows=250 width=288)	250
3.	→ Hash (cost=17.25..17.25 rows=200 width=40)	200
4.	→ Aggregate (cost=15.25..17.25 rows=200 width=40)	200
5.	→ Seq Scan on dependente as d (cost=0..13.5 rows=350 width=40)	350

```
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. The left sidebar displays a tree view of servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. The main window shows the 'Query Editor' with the following SQL query:

```
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
4 having count(*) > 1
5
6
```

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

#	Node	Rows
1.	→ Aggregate (cost=48.42..50.92 rows=67 width=86) Filter: (count(*) > 1)	67
2.	→ Hash Inner Join (cost=15.62..39.95 rows=1130 width=78) Hash Cond: ((te.ecpf)::bpchar = (e.cpf)::bpchar)	1130
3.	→ Seq Scan on trabalha_em as te (cost=0..21.3 rows=1130 width=32)	1130
4.	→ Hash (cost=12.5..12.5 rows=250 width=110)	250
5.	→ Seq Scan on empregado as e (cost=0..12.5 rows=250 width=110)	250

```
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 pgAdmin 4 interface. The left sidebar displays a tree view of servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. The main window shows the 'Query Editor' with the following SQL query:

```
1 select e.unome, te.horas
2 from empregado e join trabalha_em te on e.cpf = te.ecpf
3 where e.pnome = 'Alicia'
4
```

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

#	Node	Rows
1.	→ Nested Loop Inner Join (cost=4.2..26.85 rows=5 width=86)	5
2.	→ Seq Scan on empregado as e (cost=0..13.12 rows=1 width=110) Filter: ((pnome)::text = 'Alicia'::text)	1
3.	→ Bitmap Heap Scan on trabalha_em as te (cost=4.2..13.67 rows=6 width=40) Recheck Cond: ((ecpf)::bpchar = (e.cpf)::bpchar)	6
4.	→ Bitmap Index Scan using trabalha_em_pkey (cost=0..4.2 rows=6 width=0) Index Cond: ((ecpf)::bpchar = (e.cpf)::bpchar)	6

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

Criação de índice

The screenshot displays the pgAdmin 4 web interface. On the left, a tree view shows the server hierarchy: Servers (3) > albali.eic.cefet-rj.br, localhost, and regulus.eic.cefet-rj.br > Databases (2) > bd (selected). The main area is titled 'bd/postgres@regulus.eic.cefet-rj.br *'. The 'Query Editor' tab is active, showing the SQL command: `1 CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)`. Below the editor, the 'Messages' tab is selected, displaying the output: `CREATE INDEX` and `Query returned successfully in 284 msec.`

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. The left sidebar displays a tree view of servers and databases, with 'bd' selected under 'regulus.eic.cefet-rj.br'. The main window shows the 'Query Editor' with the following SQL query:

```
1 select * from empregado e where e.pnome = 'Alicia'
```

Below the query editor, the 'Explain' tab is active, showing the execution plan:

#	Node	Rows
1.	→ Seq Scan on empregado as e (cost=0..1.1 rows=1 width=288) Filter: ((pnome)::text = 'Alicia'::text)	1

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

Junção tradicional (sem uso do índice criado) (último nome e horas trabalhadas por projeto pelo primeiro nome)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the server tree with 'regulus.eic.cefet-rj.br' selected, and the 'bd' database expanded. The main window shows the 'Query Editor' with the following SQL query:

```
1 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
```

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

#	Node	Rows
1.	→ Hash Inner Join (cost=1.11..2.34 rows=2 width=14) Hash Cond: ((te.ecpf)::bpchar = (e.cpf)::bpchar)	2
2.	→ Seq Scan on trabalha_em as te (cost=0..1.16 rows=16 width=20)	16
3.	→ Hash (cost=1.1..1.1 rows=1 width=18)	1
4.	→ Seq Scan on empregado as e (cost=0..1.1 rows=1 width=18) Filter: ((pnome)::text = 'Alicia'::text)	1

```
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 shows the pgAdmin 4 web interface. The left sidebar displays a tree view of servers and databases. The 'regulus.eic.cefet-rj.br' server is expanded, showing two databases: 'bd' and 'postgres'. The 'bd' database is selected. The main area is the 'Query Editor' for the 'bd/postgres@regulus.eic.cefet-rj.br' connection. The query editor contains the following SQL statement:

```
1 CREATE INDEX EMP_COVER ON EMPREGADO(pnome) include (cpf, unome)
2
3
4
5
```

Below the query editor, the 'Messages' tab is active, displaying the following output:

```
CREATE INDEX
Query returned successfully in 50 msec.
```

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

Junção tradicional (sem uso dos índices criados) (último nome e horas trabalhadas por projeto pelo primeiro nome)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the server tree with 'regulus.eic.cefet-rj.br' selected, and the 'bd' database expanded. The main window shows the 'Query Editor' with the following SQL query:

```
1 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
```

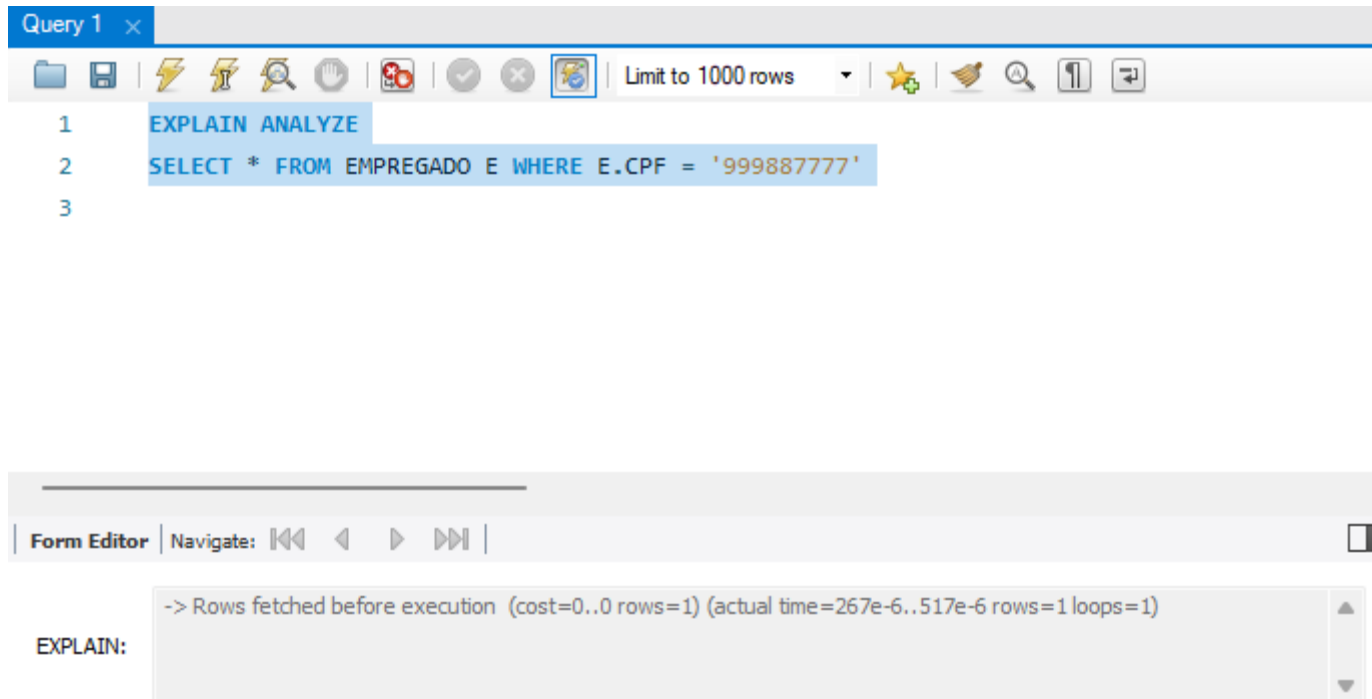
Below the query editor, the 'Data Output' tab is active, showing the 'Explain' view of the query plan. The plan consists of four nodes:

#	Node	Rows
1.	→ Hash Inner Join (cost=1.11..2.34 rows=2 width=14) Hash Cond: ((te.ecpf)::bpchar = (e.cpf)::bpchar)	2
2.	→ Seq Scan on trabalha_em as te (cost=0..1.16 rows=16 width=20)	16
3.	→ Hash (cost=1.1..1.1 rows=1 width=18)	1
4.	→ Seq Scan on empregado as e (cost=0..1.1 rows=1 width=18) Filter: ((pnome)::text = 'Alicia'::text)	1

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

MySQL

Consultas pela chave primária (empregado a partir do cpf)



The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons for file operations, execution, and settings. The query text is as follows:

```
1 EXPLAIN ANALYZE  
2 SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'  
3
```

Below the query editor, there is a section labeled "Form Editor" with navigation controls. The execution plan output is displayed in a scrollable area:

```
-> Rows fetched before execution (cost=0..0 rows=1) (actual time=267e-6..517e-6 rows=1 loops=1)
```

The word "EXPLAIN:" is visible to the left of the execution plan output.

Consulta por valor (empregado a partir do primeiro nome)

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 tree view of Schemas, including a database (BD) with tables like DEPARTAMENTO, EMPREGADO, and TRABALHA_EM. The main window displays a query editor with the following SQL query:

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

The query results are shown in a table with the following columns: id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, and Extra. The results are as follows:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	E	NULL	ALL	NULL	NULL	NULL	NULL	8	12.50	Using where

Below the query editor, the Action Output section shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	13:47:26	EXPLAIN SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'	1 row(s) returned	0.015 sec / 0.000 sec

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 left sidebar displays the 'SCHEMAS' tree with 'BD' expanded to show tables like 'DEPARTAMENTO', 'DEPENDENTE', etc. The main window shows 'Query 1' with the following SQL code:

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

The 'Result Grid' shows the execution plan for the query:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	D	NULL	ALL	NULL	NULL	NULL	NULL	3	100.00	NULL
	2	DEPENDENT SUBQUERY	E	NULL	eq_ref	PRIMARY	PRIMARY	44	BD.D.GERCPF	1	100.00	NULL

The 'Output' pane shows the execution message:

#	Time	Action	Message	Duration / Fetch
✓ 1	13:51:35	EXPLAIN SELECT D.*, (SELECT PNOME FROM EMPREGADO E WHE...	2 row(s) returned	0.031 sec / 0.000 sec

```
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 a database query editor window titled "Query 1". The query text is as follows:

```
1  EXPLAIN ANALYZE
2  SELECT D.*, E.PNOME
3  FROM DEPARTAMENTO D, EMPREGADO E
4  WHERE E.CPF = D.GERCPF
5
6
```

Below the query editor, there is a "Form Editor" section with navigation controls. The execution plan (EXPLAIN) is displayed in a scrollable area:

```
EXPLAIN:
-> Nested loop inner join (cost=1.6 rows=3) (actual time=0.0776..0.135 rows=3 loops=1)
  -> Filter: (D.GERCPF is not null) (cost=0.55 rows=3) (actual time=0.0467..0.0864 rows=3 loops=1)
    -> Table scan on D (cost=0.55 rows=3) (actual time=0.0446..0.0833 rows=3 loops=1)
      -> Single-row index lookup on E using PRIMARY (CPF=D.GERCPF) (cost=0.283 rows=1) (actual
```

```
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 MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'BD' expanded to show tables like 'DEPARTAMENTO', 'DEPENDENTE', 'EMPREGADO', etc. The main window shows 'Query 1' with the following SQL:

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

The 'Result Grid' below the query shows the execution plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	D	NULL	index	PRIMARY	PRIMARY	166	NULL	7	100.00	Using index; LooseScan
	1	SIMPLE	E	NULL	eq_ref	PRIMARY	PRIMARY	44	BD.D.ECPF	1	100.00	NULL

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
✓ 1	13:53:49	EXPLAIN SELECT * FROM EMPREGADO E WHERE E.CPF IN (SELECT ...	2 row(s) returned	0.031 sec / 0.000 sec

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

Subconsulta não correlacionada com exists (empregados com dependentes)

The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

```
1 EXPLAIN ANALYZE
2 SELECT *
3 FROM EMPREGADO E
4 WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE E.CPF = D.ECPF)
5
6
```

Below the query editor, the execution plan is displayed under the heading "EXPLAIN:". The plan details are:

- > Nested loop inner join (cost=3.06 rows=7) (actual time=0.0916..0.13 rows=3 loops=1)
- > Remove duplicates from input sorted on PRIMARY (cost=0.61 rows=7) (actual time=0.051..0.068 rows=3 loops=1)
- > Covering index scan on D using PRIMARY (cost=0.61 rows=7) (actual time=0.0474..0.0578 rows=7 loops=1)

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

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

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 pane with a tree view of the database structure, including tables like DEPARTAMENTO, DEPENDENTE, and EMPREGADO. The main query editor displays the following SQL query:

```
1 EXPLAIN
2 SELECT *
3 FROM EMPREGADO E
4 WHERE E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D WHERE D.SEXO = E.SEXO)
5
6
7
```

Below the query editor, the Result Grid shows the execution plan for the query. The grid has columns for id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, and Extra. The results are as follows:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	E	NULL	ALL	PRIMARY	NULL	NULL	NULL	8	100.00	NULL
1	SIMPLE	D	NULL	ref	PRIMARY	PRIMARY	44	BD.E.CPF	1	14.29	Using where; FirstMatch(E)

The bottom pane shows the Output window with the following message:

#	Time	Action	Message	Duration / Fetch
1	13:55:44	EXPLAIN SELECT * FROM EMPREGADO E WHERE E.CPF IN (SELECT ...	2 row(s) returned	0.000 sec / 0.000 sec

```
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 MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'BD' expanded to show tables like 'DEPENDENTE' and 'EMPREGADO'. The main editor shows 'Query 1' with the following SQL:

```
1 EXPLAIN
2 SELECT *
3 FROM EMPREGADO E
4 WHERE EXISTS (SELECT * FROM DEPENDENTE D WHERE D.ECPF = E.CPF AND D.SEXO = E.SEXO)
5
6
7
```

The 'Result Grid' below the query shows the execution plan:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	E	NULL	ALL	PRIMARY	NULL	NULL	NULL	8	100.00	NULL
	1	SIMPLE	D	NULL	ref	PRIMARY	PRIMARY	44	BD.E.CPF	1	14.29	Using where; FirstMatch(E)

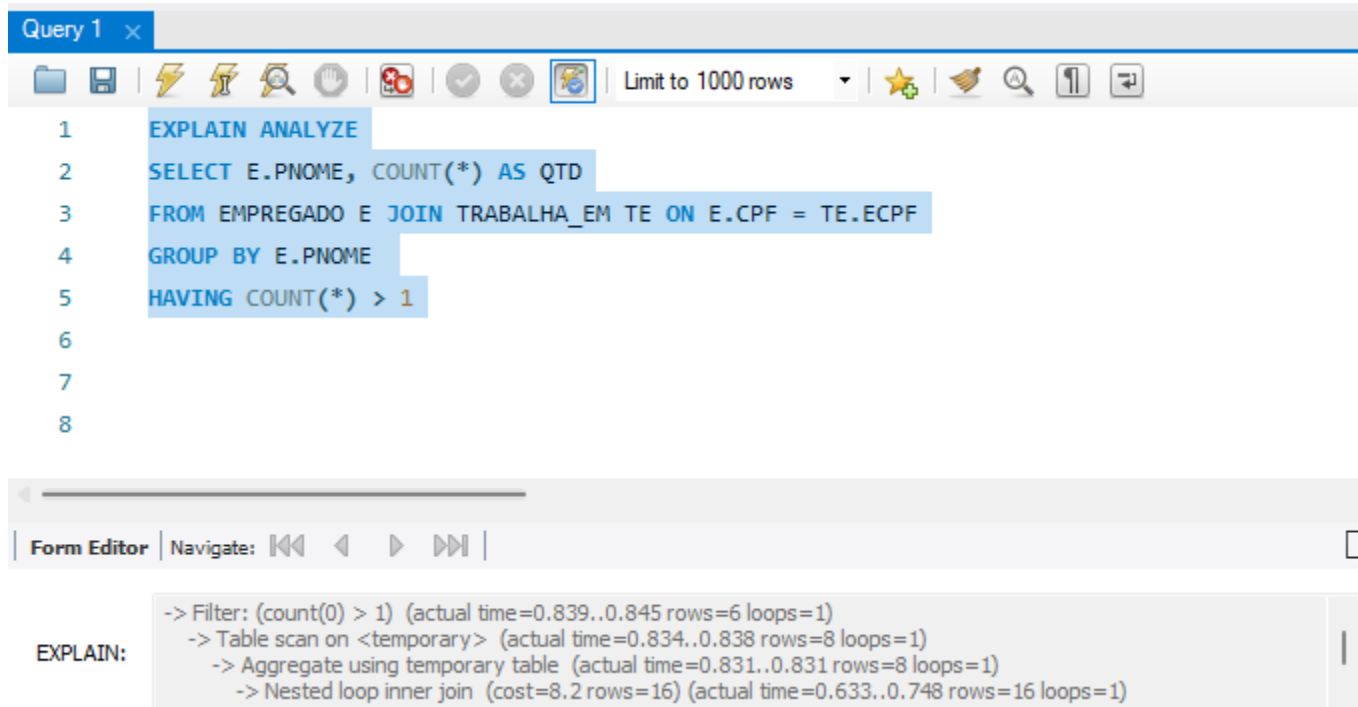
The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
✓ 1	13:57:04	EXPLAIN SELECT * FROM EMPREGADO E WHERE EXISTS (SELECT * ...	2 row(s) returned	0.031 sec / 0.000 sec

```
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 a SQL IDE window titled "Query 1". The query text is as follows:

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

Below the query editor, there is a "Form Editor" section with navigation controls. The "EXPLAIN:" output is displayed in a scrollable area:

```
-> Filter: (count(0) > 1) (actual time=0.839..0.845 rows=6 loops=1)
-> Table scan on <temporary> (actual time=0.834..0.838 rows=8 loops=1)
-> Aggregate using temporary table (actual time=0.831..0.831 rows=8 loops=1)
-> Nested loop inner join (cost=8.2 rows=16) (actual time=0.633..0.748 rows=16 loops=1)
```

```
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 pgAdmin 4 interface. The left sidebar displays the server hierarchy: Servers (3) including alballi.eic.cefet-rj.br, localhost, and regulus.eic.cefet-rj.br. Under regulus.eic.cefet-rj.br, Databases (2) are listed, with 'bd' selected. The main window shows the Query Editor with the following SQL query:

```
1 select e.unome, te.horas
2 from empregado e join trabalha_em te on e.cpf = te.ecpf
3 where e.pnome = 'Alicia'
4
```

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

#	Node	Rows
1.	→ Nested Loop Inner Join (cost=4.2..26.85 rows=5 width=86)	5
2.	→ Seq Scan on empregado as e (cost=0..13.12 rows=1 width=110) Filter: ((pnome)::text = 'Alicia'::text)	1
3.	→ Bitmap Heap Scan on trabalha_em as te (cost=4.2..13.67 rows=6 width=40) Recheck Cond: ((ecpf)::bpchar = (e.cpf)::bpchar)	6
4.	→ Bitmap Index Scan using trabalha_em_pkey (cost=0..4.2 rows=6 width=0) Index Cond: ((ecpf)::bpchar = (e.cpf)::bpchar)	6

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

Criação de índice

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 tree view of Schemas, including a database named 'BD' with tables like DEPARTAMENTO, EMPREGADO, and PROJETO. The main editor window displays a SQL query: `CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)`. The bottom panel shows the Output window with a table of execution results.

#	Time	Action	Message	Duration / Fetch
✓ 1	13:58:52	CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.094 sec

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 left sidebar displays the 'SCHEMAS' tree with 'BD' expanded to show tables like 'DEPARTAMENTO', 'DEPENDENTE', 'DEPT_LOCALIZACOES', 'EMPREGADO', 'PROJETO', and 'TRABALHA_EM'. The main editor shows a query window with the following SQL code:

```
1 • EXPLAIN
2 SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'
3
4
5
6
7
```

Below the query editor is the 'Result Grid' showing the execution plan for the query:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	E	NULL	ref	EMP_PNOME_I	EMP_PNOME_I	123	const	1	100.00	NULL

The bottom panel shows the 'Output' window with the following message:

#	Time	Action	Message	Duration / Fetch
✓ 1	13:59:31	EXPLAIN SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'	1 row(s) returned	0.015 sec / 0.000 sec

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

Junção tradicional

(índice do nome do empregado foi usado)

The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

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

Below the query, the execution plan is displayed under the heading "EXPLAIN:". The plan details the execution strategy and performance metrics:

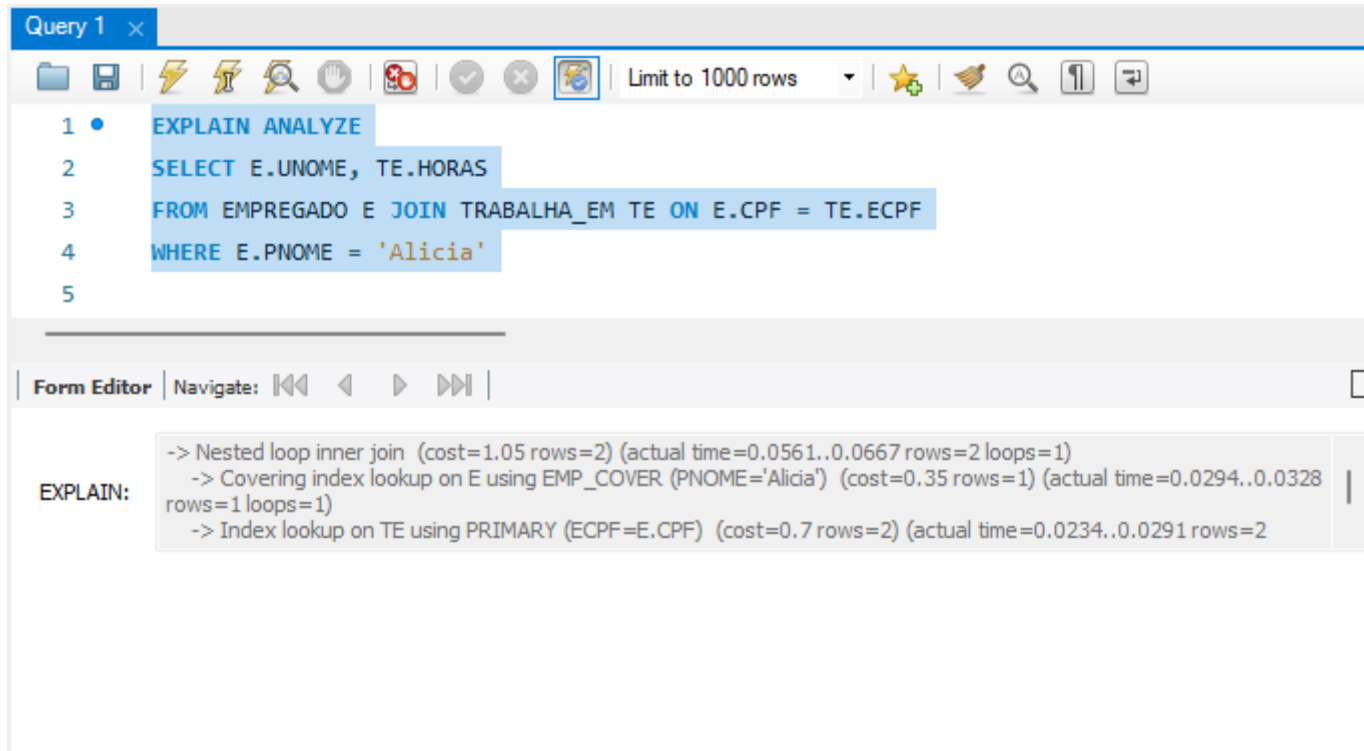
- > Nested loop inner join (cost=3.25 rows=2) (actual time=0.55..0.564 rows=2 loops=1)
- > Filter: (E.PNOME = 'Alicia') (cost=1.05 rows=1) (actual time=0.0693..0.0733 rows=1 loops=1)
- > Table scan on E (cost=1.05 rows=8) (actual time=0.0556..0.0651 rows=8 loops=1)
- > Index lookup on TE using PRIMARY (ECPF=E.CPF) (cost=2.2 rows=2) (actual time=0.477..0.485 rows=2)

```
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 shows a database query editor window titled "Query 1". The query text is as follows:

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

Below the query editor, the "Form Editor" section displays the execution plan for the query:

```
EXPLAIN:
-> Nested loop inner join (cost=1.05 rows=2) (actual time=0.0561..0.0667 rows=2 loops=1)
  -> Covering index lookup on E using EMP_COVER (PNOME='Alicia') (cost=0.35 rows=1) (actual time=0.0294..0.0328
rows=1 loops=1)
  -> Index lookup on TE using PRIMARY (ECPF=E.CPF) (cost=0.7 rows=2) (actual time=0.0234..0.0291 rows=2)
```

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



The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

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

The execution plan (EXPLAIN) is displayed below the query:

```
EXPLAIN:
-> Nested loop inner join (cost=3.25 rows=2) (actual time=0.55..0.564 rows=2 loops=1)
  -> Filter: (E.PNOME = 'Alicia') (cost=1.05 rows=1) (actual time=0.0693..0.0733 rows=1 loops=1)
    -> Table scan on E (cost=1.05 rows=8) (actual time=0.0556..0.0651 rows=8 loops=1)
      -> Index lookup on TE using PRIMARY (ECPF=E.CPF) (cost=2.2 rows=2) (actual time=0.477..0.485 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

Consultas pela chave primária (empregado a partir do cpf)

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the execution plan for the query: `SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'`. The plan consists of a single operator: **Clustered Index Seek (Clustered)** on the `[EMPREGADO].[PK_EMPREGAD_C1F89730...]` index. A tooltip for this operator shows: `1 of 1 (100%)`. The **Properties** pane on the right shows the **Actual Execution Mode** as **Actual Execution Mode**.

Object Explorer: regulus.eic.cefet-rj.br,8088 (SQL Ser)

- Databases
 - System Databases
 - Database Snapshots
 - BD
 - Database Diagrams
 - Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs dis)
 - XFvent Profiler

SQLQuery1.sql - reg...br,8088.BD (sa (56))*

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

Estimated query progress:100% Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [EMPREGADO] [E] WHERE [E].[CPF]=@1

Clustered Index Seek (Clustered)
[EMPREGADO].[PK_EMPREGAD_C1F89730...]
1 of 1 (100%)

Actual Execution Mode
Actual Execution Mode

Query executed successfully. regulus.eic.cefet-rj.br,808... sa (56) BD 00:00:00 3 rows

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

Consulta por valor (empregado a partir do primeiro nome)

The screenshot displays the Microsoft SQL Server Management Studio interface. The central query editor contains the following SQL query:

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

Below the query editor, the execution plan is visible. A callout box highlights the operation:

```
Clustered Index Scan (Clustered)  
[EMPREGADO].[PK_EMPREGAD_C1F89730...]  
Cost: 100 %  
0.000s
```

The Properties window on the right shows the following statistics for the operation:

Misc	
Actual Execution Mode	Row
Actual I/O Statistics	
Actual Number of I/Os	0
Actual Number of Logical Reads	1
Actual Rebinds	0
Actual Rewinds	0
Actual Time Statistics	
CloseTime	0
CompletionEstimate	1
Defined Values	[BD],[dbo],[EMPREG
Description	Scanning a clustered
ElapsedTime	0
Estimated CPU Cost	0,0001658
Estimated Execution Mode	Row
Estimated I/O Cost	0,003125
Estimated Number of Rows	1
Estimated Number of Rows	1

The status bar at the bottom indicates: Query executed successfully. | regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:00 | 3 rows

Sub-consulta na projeção

(lista de departamentos com seus respectivos gerentes)

The screenshot displays the Microsoft SQL Server Management Studio interface. The central pane shows the execution plan for a query. The query text is: `SELECT D.*, (SELECT PNome FROM EMPREGADO E WHERE E.CPF = D.GERCPF) AS PNome FROM DEPARTAMENTO D`. The execution plan consists of a **SELECT** operator, a **Compute Scalar** operator (3 of 3, 100%), a **Nested Loops (Left Outer Join)** operator (3 of 3, 100%), and two **Clustered Index Scan (Clustered)** operators. One scan is for `[DEPARTAMENTO].[PK_DEPARTAM_FAE85...]` (3 of 3, 100%) and the other is for `[EMPREGADO].[PK_EMPREGAD_C1F89730...]` (24 of 24, 100%).

The **Properties** pane on the right shows details for the **Compute Scalar** operator:

Property	Value
Defined Values	[Expr1003] = Scalar (...)
Description	Compute new value
ElapsedTime	0
Estimated CPU Cost	0,0000003
Estimated Execution Row	Row
Estimated I/O Cost	0
Estimated Number	1
Estimated Number	3
Estimated Number	3
Estimated Operator	0,0000003 (0%)
Estimated Rebinds	0
Estimated Rewinds	0
Estimated Row Size	61 B
Estimated Subtree	0,0068585
Logical Operation	Compute Scalar
Node ID	0
Number of Execution	1
OpenTime	0

The **Actual Number of Rows for All Executions** section indicates: **Actual number of rows for All Executions output by this operator. For rows of type P...**

The status bar at the bottom shows: **Query executed successfully.** | regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:00 | 8 rows

Ready

```
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 central pane shows the following SQL query:

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

Below the query, the execution plan is visualized. It consists of three main components:

- SELECT**: A leaf operator with a cost of 0% and 0.000s.
- Nested Loops (Inner Join)**: The root operator with a cost of 0% and 0.000s, which connects to the other two operators.
- Clustered Index Scan (Clustered)**: An operator for the DEPARTAMENTO table with a cost of 48% and 0.000s.
- Clustered Index Seek (Clustered)**: An operator for the EMPREGADO table with a cost of 52% and 0.000s.

The status bar at the bottom indicates: "Query executed successfully. regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:01 | 7 rows".

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

Property	Value
Actual Number of Rows	3
Cached plan size	24 KB
CardinalityEstimate	150
CompileCPU	2
CompileMemory	200
CompileTime	2
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number of Rows	0
Estimated Number of Rows for All Executions	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0068971
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareAcceleration	
OptimizerStatsUsage	

Subconsulta não correlacionada com in (empregados com dependentes)

SQLQuery1.sql - regulus.eic.cefet-rj.br,8088.BD (sa (56))* - 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:

- Clustered Index Scan (Clustered) [DEPENDENTE].[PK_DEPENDENTE_2B84A8D] Cost: 48% 0.000s
- Stream Aggregate (Aggregate) Cost: 0% 0.000s
- Clustered Index Seek (Clustered) [EMPREGADO].[PK_EMPREGADO_C1F89730...] Cost: 52% 0.000s
- Nested Loops (Inner Join) Cost: 0% 0.000s
- SELECT Cost: 0%

Properties

SELECT

Cached plan size	32 KB
CardinalityEstimate	150
CompileCPU	9
CompileMemory	272
CompileTime	10
Degree of Parallelism	1
Estimated Number of Rows	0
Estimated Number of Operators	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0069065

Query executed successfully. regulus.eic.cefet-rj.br,8088... sa (56) BD 00:00:01 8 rows

Ready

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

Subconsulta não correlacionada com exists (empregados com dependentes)

SQLQuery1.sql - regulus.eic.cefet-rj.br,8088.BD (sa (56))* - Microsoft SQL Server Management Studio

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

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

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

Execution Plan:

- Clustered Index Seek (Clustered) [EMPREGADO]. [PK_EMPREGAD_C1F89730...]
Cost: 52 %
- Stream Aggregate (Aggregate)
Cost: 0 %
0.000s
- Nested Loops (Inner Join)
Cost: 0 %
0.000s
- SELECT
Cost: 0 %

Properties:

SELECT	
Misc	
Cached plan size	32 KB
CardinalityEstimate	150
CompileCPU	5
CompileMemory	280
CompileTime	5
Degree of Parallelism	1
Estimated Number of Rows	0
Estimated Number of Subtrees	3
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0069065
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareAssess	
OptimizerStatsUsage	
QueryHash	0x4B5B7ACCAD9479
QueryPlanHash	0xD9B8C812E470569
QueryTimeStats	
Cached plan size	
Cached plan size.	

Query executed successfully. regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:00 | 8 rows

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

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

SQLQuery1.sql - regulus.eic.cefet-rj.br,8088.BD (sa (56))* - Microsoft SQL Server Management Studio

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

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

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

Execution Plan:

- Clustered Index Scan (Clustered) [EMPREGADO].[PK_EMPREGAD_C1F89730...] Cost: 45 % 0.000s
- Clustered Index Scan (Clustered) [DEPENDENTE].[PK_DEPENDEN_2BB4A8D...] Cost: 53 % 0.000s
- Nested Loops (Left Semi Join) Cost: 3 % 0.000s
- SELECT Cost: 0 %

Properties:

SELECT	
Actual Number of	2
Cached plan size	32 KB
CardinalityEstimati	150
CompileCPU	7
CompileMemory	320
CompileTime	7
CompletionEstima	1
Degree of Parallelis	1
ElapsedTime	0
Estimated Number	0
Estimated Number	5,08558
Estimated Operato	0 (0%)
Estimated Subtree	0,0073703
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwar	
OptimizerStatsUsag	

Actual Number of Rows for All Executions
Actual number of rows for All Executions output by this operator. For rows of type P...

Query executed successfully. regulus.eic.cefet-rj.br,808... sa (56) BD 00:00:00 6 rows

```
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)

SQLQuery1.sql - regulus.eic.cefet-rj.br,8088.BD (sa (56))* - 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...

Execution Plan:

- SELECT (Cost: 0 %)
- Nested Loops (Left Semi Join) (Cost: 3 %)
- Clustered Index Scan (Clustered) [EMPREGADO].[PK__EMPREGAD__C1F89730... (Cost: 45 %)
- Clustered Index Scan (Clustered) [DEPENDENTE].[PK__DEPENDEN__2BB4A8D... (Cost: 53 %)

Properties

SELECT	
Actual Number of	2
Cached plan size	32 KB
CardinalityEstimate	150
CompileCPU	7
CompileMemory	320
CompileTime	7
CompletionEstimate	1
Degree of Parallelis	1
ElapsedTime	0
Estimated Number	0
Estimated Number	5,08558
Estimated Operato	0 (0%)
Estimated Subtree	0,0073703
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwar	
OptimizerStatsUsa	

Actual Number of Rows for All Executions

Actual number of rows for All Executions output by this operator. For rows of type P...

Ready

```
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 Enterprise Manager interface. The central pane shows the following SQL query:

```
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
```

The execution plan below the query shows the following operators and their costs:

- compute Scalar** (Cost: 0 %)
- Stream Aggregate (Aggregate)** (Cost: 0 %)
- Nested Loops (Inner Join)** (Cost: 0 %)
- Sort** (Cost: 59 %)
- Clustered [EMPREGADO]** (Cost: 0.000s)
- Clustered Index Seek (Clustered)** ([TRABALHA_EM].[FK_TRABALHA_CB6423...]) (Cost: 23 %)

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

Property	Value
Actual Number of	8
CloseTime	0
CompletionEstima	1
Defined Values	[Expr1002] = Scalar C
Description	Compute new value
ElapsedTime	0
Estimated CPU Cos	0
Estimated Executio	Row
Estimated I/O Cost	0
Estimated Number	1
Estimated Number	8
Estimated Number	8
Estimated Number	8
Estimated Operato	0 (0%)
Estimated Rebinds	0
Estimated Rewinds	0
Estimated Row Size	20 B
Estimated Subtree	0,0191686

The status bar at the bottom indicates: Query executed successfully. | regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:00 | 14 rows

Ready

```
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 central pane shows the following SQL query:

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

The execution plan below the query shows a **Nested Loops (Inner Join)** operator with a cost of 0% and 0.000s. It is connected to two **Clustered Index Scan (Clustered)** operators. The first scan is on the **EMPREGADO** table with a cost of 50% and 0.000s. The second scan is on the **TRABALHA_EM** table with a cost of 50% and 0.000s. The status bar at the bottom indicates the query was executed successfully, returning 6 rows.

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

Property	Value
Actual Number of	2
Cached plan size	24 KB
CardinalityEstimate	150
CompileCPU	3
CompileMemory	232
CompileTime	3
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number	0
Estimated Number	2
Estimated Operator	0 (0%)
Estimated Subtree	0,0065872
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardware	
OptimizerStatsUsage	

Below the table, it states: **Actual Number of Rows for All Executions** and **Actual number of rows for All Executions output by this operator. For rows of type P...**

Criação de índice

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query: `CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)`. The query has been executed successfully, as indicated by the status bar and the Messages pane. The Messages pane shows: "Commands completed successfully." and "Completion time: 2022-04-25T15:11:53.5300775-03:00". The Properties pane on the right shows connection details for the current connection, including the connection name, elapsed time, and state.

Object Explorer: regulus.eic.cefet-rj.br,8088 (SQL Ser)

- Databases
 - System Databases
 - Database Snapshots
 - BD
 - Database Diagrams
 - Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs dis)
 - XFvent Profiler

SQLQuery1.sql - reg...br,8088.BD (sa (56))*

```
CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)
```

100 %

Results Messages Live Query Statistics Execution plan

Commands completed successfully.

Completion time: 2022-04-25T15:11:53.5300775-03:00

100 %

Query executed successfully. regulus.eic.cefet-rj.br,808... sa (56) BD 00:00:00 4 rows

Properties

Current connection parameters

Aggregate Status

Connection failure	
Elapsed time	00:00:00.8367147
Finish time	25/04/2022 15:11:53
Name	regulus.eic.cefet-rj.b
Rows returned	4
Start time	25/04/2022 15:11:52
State	Open

Connection

Connection name	regulus.eic.cefet-rj.b
-----------------	------------------------

Connection Details

Connection elapse	00:00:00.8367147
Connection encryp	Not encrypted
Connection finish t	25/04/2022 15:11:53
Connection rows r	4
Connection start ti	25/04/2022 15:11:52
Connection state	Open
Display name	regulus.eic.cefet-rj.b

Name

The name of the connection.

Ready Ln 4 Col 1 Ch 1 INS

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 Microsoft SQL Server Management Studio interface. The query editor contains the following SQL statement:

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

The execution plan shows a **Clustered Index Scan (Clustered)** on the **[EMPREGADO].[PK_EMPREGAD_C1F89730...]** index. The cost is 100% and the execution time is 0.000s. The status bar indicates the query executed successfully, returning 3 rows.

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

Property	Value
Actual Number of Rows	1
Cached plan size	24 KB
CardinalityEstimate	150
CompileCPU	2
CompileMemory	184
CompileTime	2
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number of Rows	0
Estimated Number of Rows	1
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0,0032908
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareAcceleration	
OptimizerStatsUsage	

Actual Number of Rows for All Executions
Actual number of rows for All Executions output by this operator. For rows of type P...

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

Junção tradicional (sem uso do índice criado) (Consulta de alocação por data de nascimento com índice)

SQLQuery1.sql - regulus.eic.cefet-rj.br,8088.BD (sa (56))* - Microsoft SQL Server Management Studio

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

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

SELECT E.UNOME, TE.HORAS FROM EMPREGADO E JOIN TRABALHA_EM TE ON E...

Execution Plan:

- SELECT (Cost: 0%)
- Nested Loops (Inner Join) (Cost: 0%)
- Clustered Index Scan (Clustered) [EMPREGADO].[PK__EMPREGAD__C1F89730...] (Cost: 50%)
- Clustered Index Seek (Clustered) [TRABALHA_EM].[PK__TRABALHA__CB6423...] (Cost: 50%)

Properties

SELECT	
Actual Number of	2
Cached plan size	24 KB
CardinalityEstimate	150
CompileCPU	3
CompileMemory	272
CompileTime	3
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number	0
Estimated Number	2
Estimated Operator	0 (0%)
Estimated Subtree	0,0065872
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardware	
OptimizerStatsUsage	

Actual Number of Rows for All Executions
Actual number of rows for All Executions output by this operator. For rows of type P...

Query executed successfully. regulus.eic.cefet-rj.br,8088... sa (56) BD 00:00:00 6 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 (SSMS) interface. The main window shows a query editor with the following SQL code:

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

The Results pane below the query editor shows the following output:

```
Commands completed successfully.  
  
Completion time: 2022-04-25T15:14:50.2137275-03:00
```

The Properties pane on the right side of the window displays the following connection details:

Current connection parameters	
Aggregate Status	
Connection failure	
Elapsed time	00:00:00.8356172
Finish time	25/04/2022 15:14:50
Name	regulus.eic.cefet-rj.b
Rows returned	5
Start time	25/04/2022 15:14:49
State	Open
Connection	
Connection name	regulus.eic.cefet-rj.b
Connection Details	
Connection elapse	00:00:00.8356172
Connection encrypt	Not encrypted
Connection finish t	25/04/2022 15:14:50
Connection rows r	5
Connection start ti	25/04/2022 15:14:49
Connection state	Open
Display name	regulus.eic.cefet-rj.b
Name	
The name of the connection.	

The status bar at the bottom of the window indicates: Ready | Ln 4 | Col 1 | Ch 1 | INS | Query executed successfully. | regulus.eic.cefet-rj.br,808... | sa (56) | BD | 00:00:00 | 5 rows

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 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'
```

The execution plan below the query shows the following steps:

- SELECT**: Cost: 0 %, 0.000s
- Nested Loops (Inner Join)**: Cost: 0 %, 0.000s
- Index Seek (NonClustered) [EMPREGADO].[EMP_COVER] [E]**: Cost: 50 %, 0.000s
- Clustered Index Seek (Clustered) [TRABALHA_EM].[PK_TRABALHA_CB6423...]**: Cost: 50 %, 0.000s

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

Property	Value
Actual Number of	2
Cached plan size	24 KB
CardinalityEstimate	150
CompileCPU	3
CompileMemory	248
CompileTime	3
CompletionEstimate	1
Degree of Parallelism	1
ElapsedTime	0
Estimated Number	0
Estimated Number	2
Estimated Operator	0 (0%)
Estimated Subtree	0,0065757
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardware	
OptimizerStatsUsage	

The status bar at the bottom indicates: Query executed successfully. regulus.eic.cefet-rj.br,808... sa (56) BD 00:00:00 6 rows

Ready

```
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 displays the Oracle SQL Developer interface. The main window shows a query in the Query Builder: `SELECT * FROM EMPREGADO E WHERE E.CPF = '999887777'`. Below the query, the Explain Plan is visible, showing the execution path: SELECT STATEMENT, TABLE ACCESS (EMPREGADO), and INDEX (SYS_C004112) with a UNIQUE SCAN. The execution time for the query is 0,056 seconds.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
INDEX	SYS_C004112	UNIQUE SCAN	1	1

The interface also shows a tree view of database objects on the left, including tables like DEPARTAMENTO, EMPREGADO, and TRABALHA_EM. The bottom status bar indicates the current cursor position: | Line 2 Column 1 | Insert | Modified | Windows: Cf

`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 components:

- Connections:** A tree view on the left showing the database structure for 'BD@localhost', including tables like EMPREGADO, DEPARTAMENTO, and others.
- Worksheet:** The top right pane contains the SQL query: `SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'`.
- Query Result:** The bottom right pane shows the execution plan (EXPLAIN PLAN) for the query. It details the operations performed by the database engine.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
TABLE ACCESS	EMPREGADO	FULL	1	3
Filter Predicates				
E.PNOME='Alicia'				
Other XML				

`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:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	3
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
INDEX	SYS_C004112	UNIQUE SCAN	1	1
Access Predicates		E.CPF=:B1		
TABLE ACCESS	DEPARTAMENTO	FULL	3	3
Other XML				

The interface also shows the Connections pane on the left and the Reports pane at the bottom left. The status bar at the bottom indicates the current cursor position and available actions.

```
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 Oracle SQL Developer interface. The main window shows 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 visible, showing the following operations:

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_C004112	UNIQUE SCAN	1	0

The execution plan also includes an Access Predicate: `E.CPF=D.GERCPF`.

The bottom status bar indicates: | Line 4 Column 1 | Insert | Modified | Windows: Cf

```
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 Query Builder:

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

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

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	5
NESTED LOOPS		SEMI	3	5
TABLE ACCESS	EMPREGADO	FULL	8	3
INDEX	SYS_C004129	RANGE SCAN	3	1

The execution plan also shows an Access Predicate: `E.CPF=D.ECPF`.

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

Subconsulta não correlacionada com exists (empregados com dependentes)

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 E.CPF = D.ECPF)
```

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

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	5
NESTED LOOPS		SEMI	3	5
TABLE ACCESS	EMPREGADO	FULL	8	3
INDEX	SYS_C004129	RANGE SCAN	3	1

The execution plan also includes a diagram showing the relationship between the operations: a NESTED LOOPS join between the TABLE ACCESS (EMPREGADO) and the INDEX (SYS_C004129). The index is used for an Access Predicate: E.CPF=D.ECPF.

At the bottom of the window, a status bar indicates: | Line 4 Column 1 | Insert | Modified | Windows: CF

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

Subconsulta correlacionada com in (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 E.CPF IN (SELECT D.ECPF FROM DEPENDENTE D WHERE D.SEXO = E.SEXO)
```

Below the query, the Explain Plan is visible, showing the execution strategy:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	7
HASH JOIN		RIGHT SEMI	1	7
Access Predicates				
AND				
E.CPF=D.ECPF				
D.SEXO=E.SEXO				
TABLE ACCESS	DEPENDENTE	FULL	7	3
TABLE ACCESS	EMPREGADO	FULL	8	3

The interface also shows the Connections pane on the left and the Reports pane at the bottom left. The status bar at the bottom indicates the current cursor position: | Line 4 Column 1 | Insert | Modified | Windows: C

```
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 displays the Oracle SQL Developer interface. The main window shows 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 Explain Plan is visible, showing the execution strategy:

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	7	3
TABLE ACCESS	EMPREGADO	FULL	8	3
Other XML				

The interface also shows the Connections pane on the left and the Reports pane at the bottom left. The status bar at the bottom indicates the current cursor position: | Line 4 Column 1 | Insert | Modified | Windows: C

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

```
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 visible, showing the following 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	8	3
INDEX	SYS_C004124	RANGE SCAN	2	1
Access Predicates		TE.ECPF=E.CPF		

At the bottom of the window, a status bar indicates: | Line 4 Column 1 | Insert | Modified | Windows: Cf

```
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 SQL 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 in a table format:

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				

The interface also shows the Connections pane on the left and the Reports pane at the bottom left. The status bar at the bottom indicates the current cursor position: Line 4 Column 1.

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

Criação de índice

The screenshot displays the Oracle SQL Developer interface. The main window is titled "Oracle SQL Developer : BD@localhost". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations and database actions. On the left, the "Connections" pane shows a tree view of database connections, including "Orade Connections", "BD@localhost", "localhost", and "sys@localhost". The "sys@localhost" connection is expanded to show a list of database objects: Tables (Filtered), Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, and Materialized Views. Below this is the "Reports" pane, which lists "All Reports", "Analytic View Reports", "Data Dictionary Reports", "Data Modeler Reports", and "OLAP Reports". The main workspace is divided into two panes: "Worksheet" and "Query Builder". The "Worksheet" pane contains the SQL statement: `CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)`. The "Query Builder" pane is currently empty. Below the workspace, there are tabs for "Autotrace", "Query Result", "Explain Plan", and "Script Output". The "Query Result" tab is active, displaying the message: "Index EMP_PNOME_I created." and "Task completed in 0,041 seconds". The status bar at the bottom indicates "Line 2 Column 1 | Insert | Modified | Windows: CF".

CREATE INDEX EMP_PNOME_I ON EMPREGADO(PNOME)

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

The screenshot displays the Oracle SQL Developer interface. The main window shows a query in the Query Builder: `SELECT * FROM EMPREGADO E WHERE E.PNOME = 'Alicia'`. Below the query, the Explain Plan is visible, showing the execution path:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	2
INDEX	EMP_PNOME_I	RANGE SCAN	1	1

The diagram below the table shows the flow from the SELECT STATEMENT to the TABLE ACCESS (EMPREGADO), which then uses the INDEX (EMP_PNOME_I) with the predicate E.PNOME='Alicia'.

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

Junção tradicional (sem uso do índice criado) (Consulta de alocação por data de nascimento com índice)

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 Explain Plan is shown, detailing the execution strategy:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			2	4
TABLE ACCESS	EMPREGADO	BY INDEX ROWID	1	1
Filter Predicates				
TE.ECPF=E.CPF				
NESTED LOOPS			2	4
TABLE ACCESS	TRABALHA_EM	FULL	16	3
INDEX	EMP_PNOME_I	RANGE SCAN	1	0
Access Predicates				
E.PNOME='Alicia'				
Other XML				

The interface also shows the Connections pane on the left and the Reports pane at the bottom left. The status bar at the bottom indicates the current cursor position: "Line 4 Column 1 | Insert | Modified | Windows: Cf".

```
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 Oracle SQL Developer interface. The main window shows a SQL script in the Query Builder:

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

The script has been executed, and the output pane shows the following messages:

```
Index EMP_PNOME_I created.  
  
Index EMP_COVER created.
```

The status bar at the bottom indicates "Task completed in 0,042 seconds".

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 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 Explain Plan is visible, showing the execution strategy:

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 diagram below the table illustrates the execution flow: SELECT STATEMENT -> NESTED LOOPS -> TABLE ACCESS (EMP_COVER) -> INDEX (EMP_COVER). The index is used for the join condition E.CPF = TE.ECPF and the filter E.PNOME = 'Alicia'.

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

Referências

