Relational Algebra

- Relational algebra is a **procedural** language … operations that can be implemented by a database
- SQL is **non-procedural**
- We will consider some operators and the mapping of SQL Select to RA statements and to an RA tree.

Relational Algebra

- Operators:
  - SELECT
  - PROJECT
  - JOIN

There are more …. ACS-4902

We need to consider a **SQL Select** in terms of **RA operators** select, project, and join

SELECT

- The SELECT Operation
  - Yields a relation, a subset of the tuples from the operand that satisfies a selection condition:

$$\sigma_{<\text{selection condition}>}(R)$$

R is a relation or a variable representing a relation

- Boolean expression contains clauses of the form <attribute name> <comparison op> <constant value>

  or

- <attribute name> <comparison op> <attribute name>

AND, **OR**, and **NOT**

SELECT

- Example:

$$\sigma_{(\text{Dno}=4 \textbf{ AND } \text{Salary}>25000) \textbf{ OR } (\text{Dno}=5 \textbf{ AND } \text{Salary}>30000)}(\text{EMPLOYEE})$$

- <selection condition> applied independently to each individual tuple *t* in *R*
  - If condition evaluates to TRUE, tuple selected
- **unary**

PROJECT

- Yields a relation with specific columns (or arithmetic expressions involving columns) from the operand

  -discards the other columns:

$$\pi_{<\text{attribute list}>}(R)$$

R is a relation or a variable representing a relation

- **unary**
- **degree**
  - Number of attributes in <attribute list>
- **duplicate elimination**
  - Result of PROJECT operation is a set of distinct tuples

Sequences of Operations

- **In-line** expression:

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

- Sequence of operations:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{DEP5\_EMPS})$$

Same result

The CARTESIAN PRODUCT

- **CARTESIAN PRODUCT**
  - **CROSS PRODUCT** or **CROSS JOIN**
  - Denoted by $\times$
  - Binary set operation
  - Useful when followed by a selection that matches values of attributes

A Product can yield attributes of the same name … the dot notation to disambiguate

JOIN

- The **JOIN** Operation
  - Denoted by ⋈
  - Combine related tuples from two relations
  - Inner join
  - binary

Join condition

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$
$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

Theta JOIN

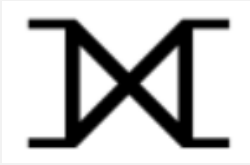
- **Natural join** *
- **THETA** JOIN
  - Each <condition> of the form A_i θ B_j
  - A_i is an attribute of R
  - B_j is an attribute of S
  - A_i and B_j have the same domain
  - θ (theta) is one of the comparison operators:
    - {=, <, ≤, >, ≥, ≠}

OUTER JOIN SYMBOLS

LEFT

RIGHT

FULL



Notation for Query Trees

- **Query tree for RA operations**
  - Represents the input relations of a query as leaf nodes of the tree
  - Represents the relational algebra operations as internal nodes
  - Execution is bottom-up; results move up the tree as an input/operand

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Q2:

```
SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber
AND Mgr_ssn=Ssn
AND Plocation = 'Stafford';
```

Data Output | Explain | Messages | History

pnumber integer	dnum integer	lname character varying	address character varying	bdate date
30	4	Wallace	291 Berry, Bellaire...	1941-06-20
10	4	Wallace	291 Berry, Bellaire...	1941-06-20

Q2: SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber
AND Mgr_ssn=Ssn
AND
Plocation='Stafford';

Given the Selects statement, what is the RA query tree? Given the RA query tree, what is the Select statement?

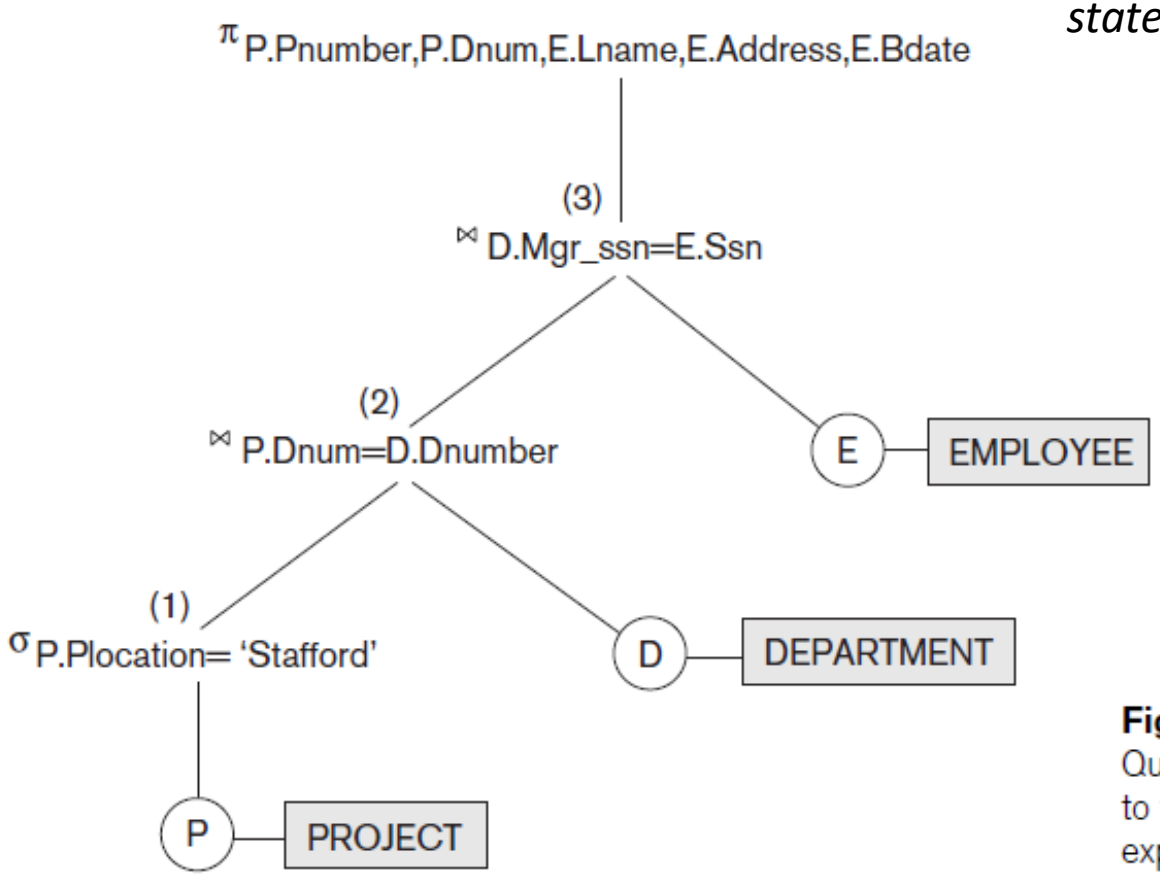


Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

From **December 2015 exam**:

Consider the XXX database.
Consider the SQL statement:

```
select city
from Offices
where state = "CA";
```

Translate the SQL statement into a sequence of relational algebra **statements**.

From **December 2015 exam**:

Consider the SQL statement:

```
select city, amount
from customers
inner join orders
on (customers.customerNumber = orders.customerNumber)
inner join payments
on (customers.customerNumber = payments.customerNumber)
where city = "NYC"
and status = "shipped"
;
```

Illustrate a relational algebra **query tree** for the execution of the statement.

Example
…. and next 3 slides

Example : Consider this query that retrieves from the text's Company database the SSN and last name of employees who earn more than 10000 and have a daughter and a son

```
SELECT ssn, lname

FROM dependent x
INNER JOIN employee e
        ON (x.essn = e.ssn)
INNER JOIN dependent y
        ON (y.essn = e.ssn)

where x.relationship = 'Son'
and y.relationship = 'Daughter'
and e.salary > 10000
```

Create a query tree for the above

Example

First: how do we apply our study of relational algebra to this?

```
SELECT ssn, lname
```

a projection

```
FROM dependent x
INNER JOIN employee e
        ON (x.essn = e.ssn)
INNER JOIN dependent y
        ON (y.essn = e.ssn)
```

inner join between x and e

inner join between y and e

```
where x.relationship = 'Son'
and y.relationship = 'Daughter'
and e.salary > 10000
```

a selection on x

a selection on y

a selection on e

Example

First: how do we apply our study of relational algebra to this?

```
SELECT ssn, lname
FROM dependent x
INNER JOIN employee e
        ON (x.essn = e.ssn)
INNER JOIN dependent y
        ON (y.essn = e.ssn)

where x.relationship = 'Son'
and y.relationship = 'Daughter'
and e.salary > 10000
```

a projection

inner join between x and e

inner join between y and e

a selection on x

a selection on y

a selection on e

Our approach:

We need to create *the query tree with the appropriate joins, selections, and projections, and then push operations downwards as far as possible*

Example

a projection

inner join between x and e

inner join between y and e

a selection on x

a selection on y

a selection on e

Query trees

In class:
Create the diagram, then check that operations are as low as possible in tree.

Remaining slides are for interest only

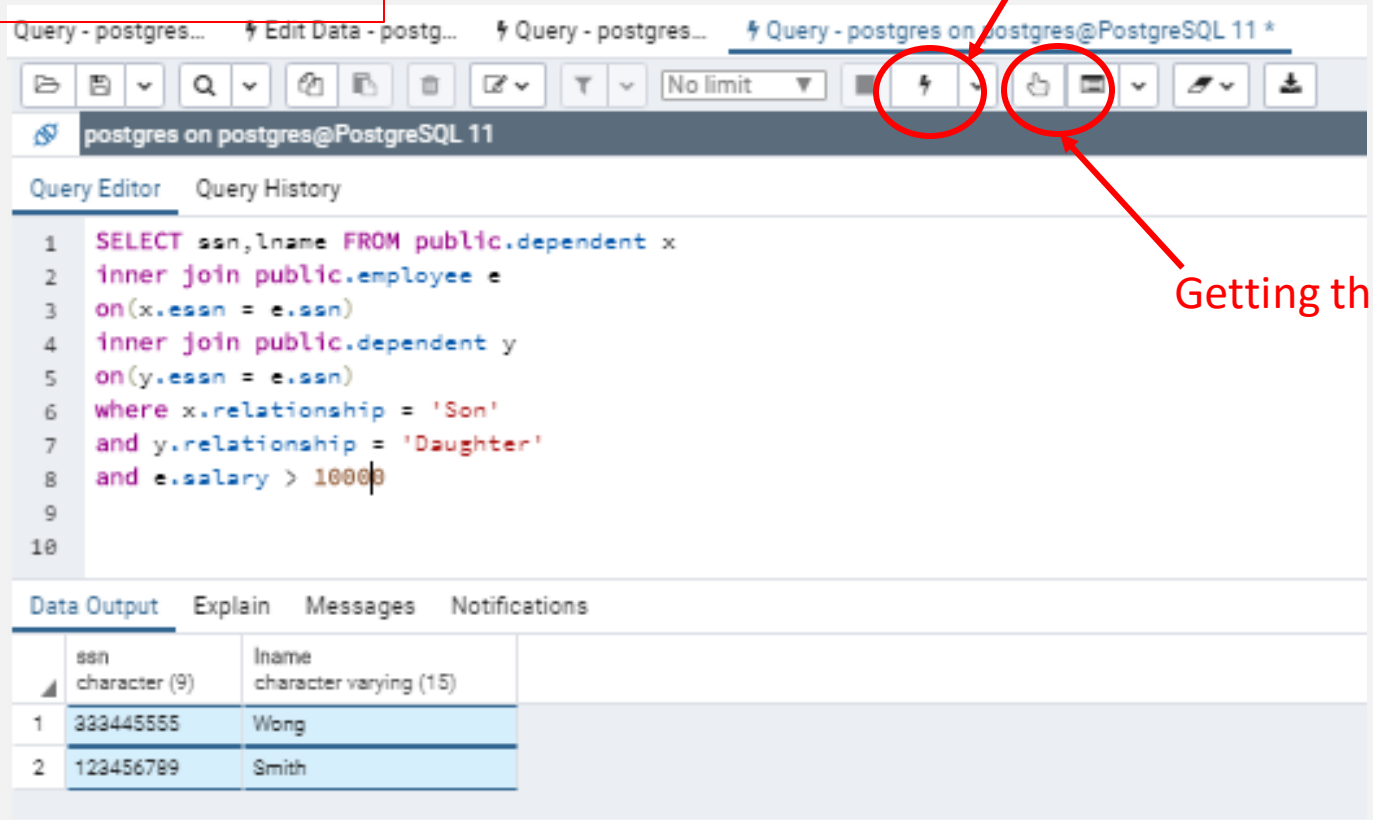
For interest only

Aside: Seeing the query plan in PostgreSQL

Two employees have both a daughter and son and earn more than 10000

Running the query

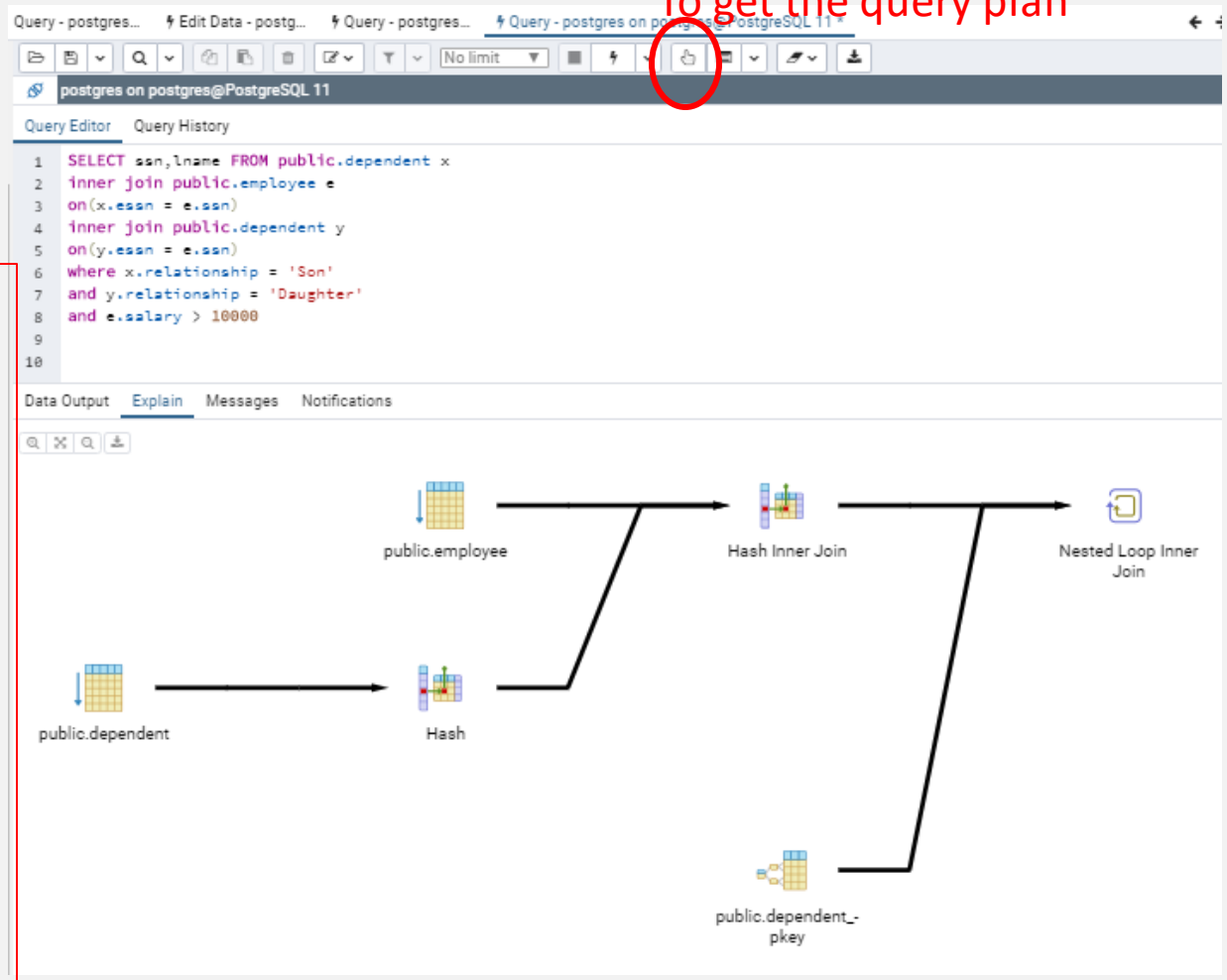
Getting the plan



For interest only

ASIDE: query plans in PostgreSQL

To get the query plan

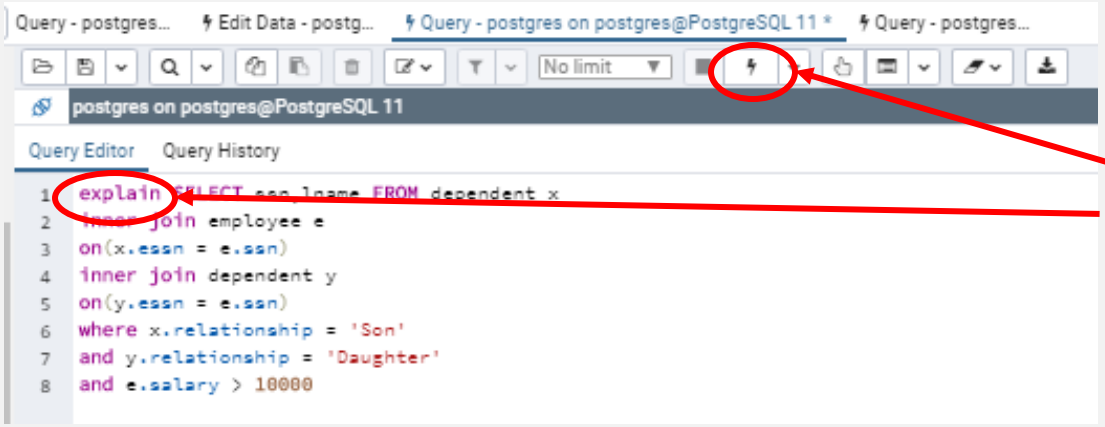


On the right, if one lets the mouse hover over a node you get more information… for example, the output from each step, algorithm, filtering

… to get the query plan in text → next slide

For interest only

ASIDE: query plans in PostgreSQL



Execute the query but prefix the Select with "explain"

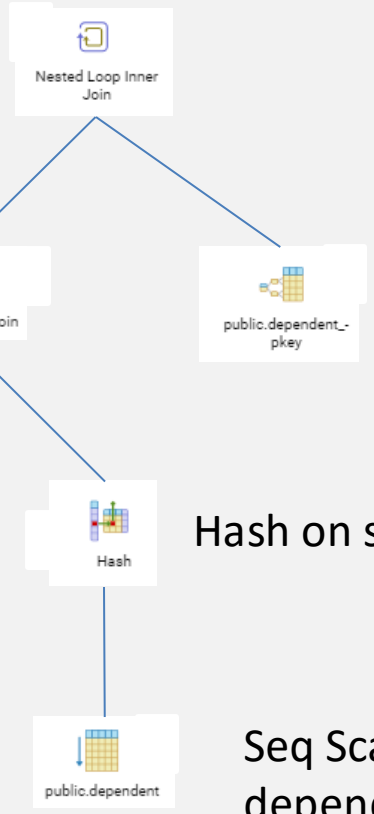
Query plan in **text** form – note that it shows join algorithms and filters

```
Nested Loop  (cost=16.42..30.50 rows=1 width=88)
  ->  Hash Join  (cost=16.27..29.72 rows=1 width=128)
        Hash Cond: (e.ssn = y.essn)
        ->  Seq Scan on employee e  (cost=0.00..13.13 rows=83 width=88)
              Filter: (salary > '10000'::numeric)
        ->  Hash  (cost=16.25..16.25 rows=2 width=40)
              ->  Seq Scan on dependent y  (cost=0.00..16.25 rows=2 width=40)
                    Filter: ((relationship)::text = 'Daughter'::text)
  ->  Index Scan using dependent_pkey on dependent x  (cost=0.15..0.77 rows=1
width=40)
        Index Cond: (essn = e.ssn)
        Filter: ((relationship)::text = 'Son'::text)
```

For interest only

ASIDE: query plans in PostgreSQL → similar to a query tree

Redrawing the query plan as a tree:



Joining using a hash inner join algorithm

Index scan & applies a filter: Relationship='Son' & access rows pertinent to "the nested inner loop join"

Hash on ssn

Seq Scan on employee e & applies a filter: (salary > '10000')

Seq Scan on dependent y & Applies a filter Relationship='Daughter'

For interest only

Aside: Query plans in real database systems are trees where nodes are operations the database system Implements

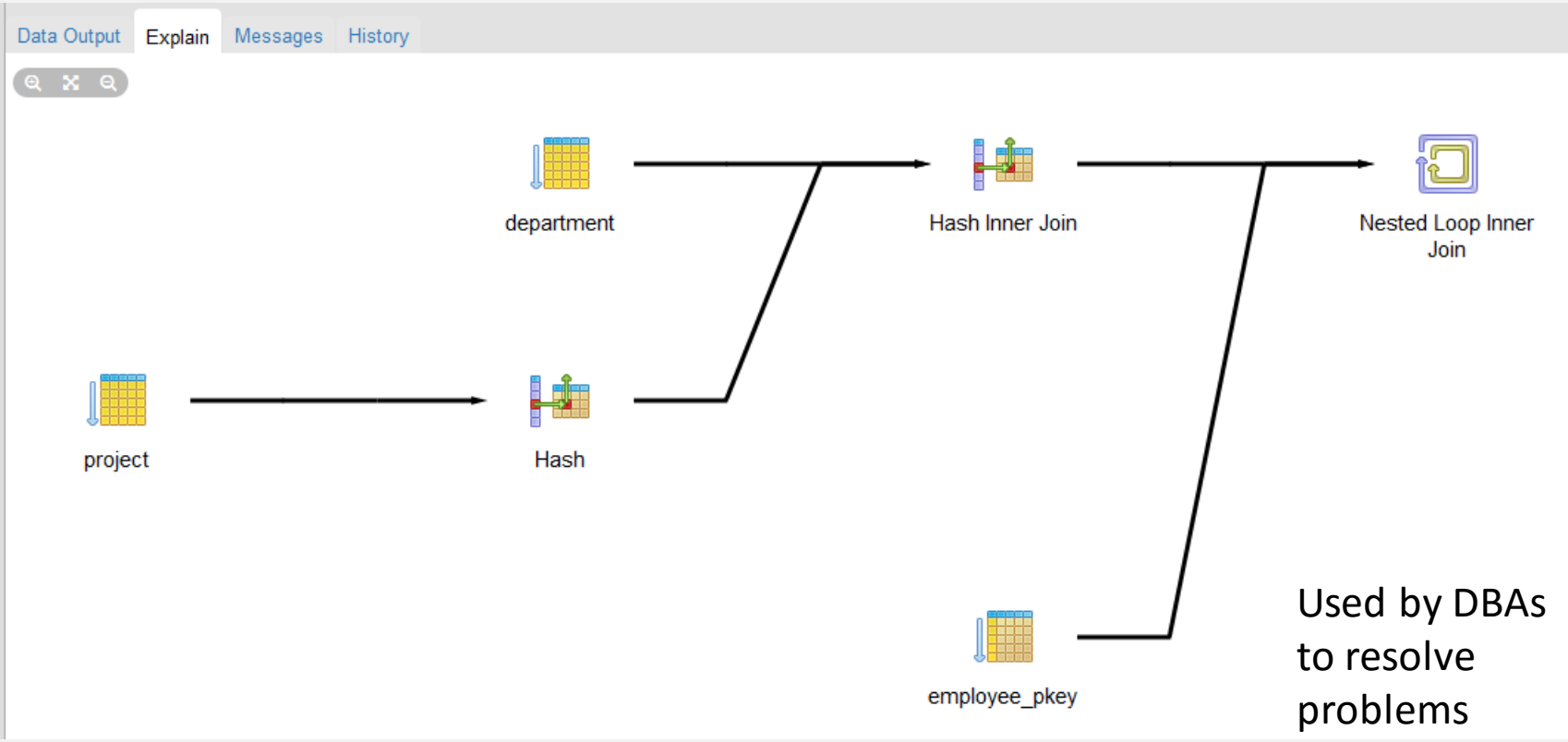
Some examples→

https://www.postgresql.org/docs/current/static/using-explain.html

For interest only

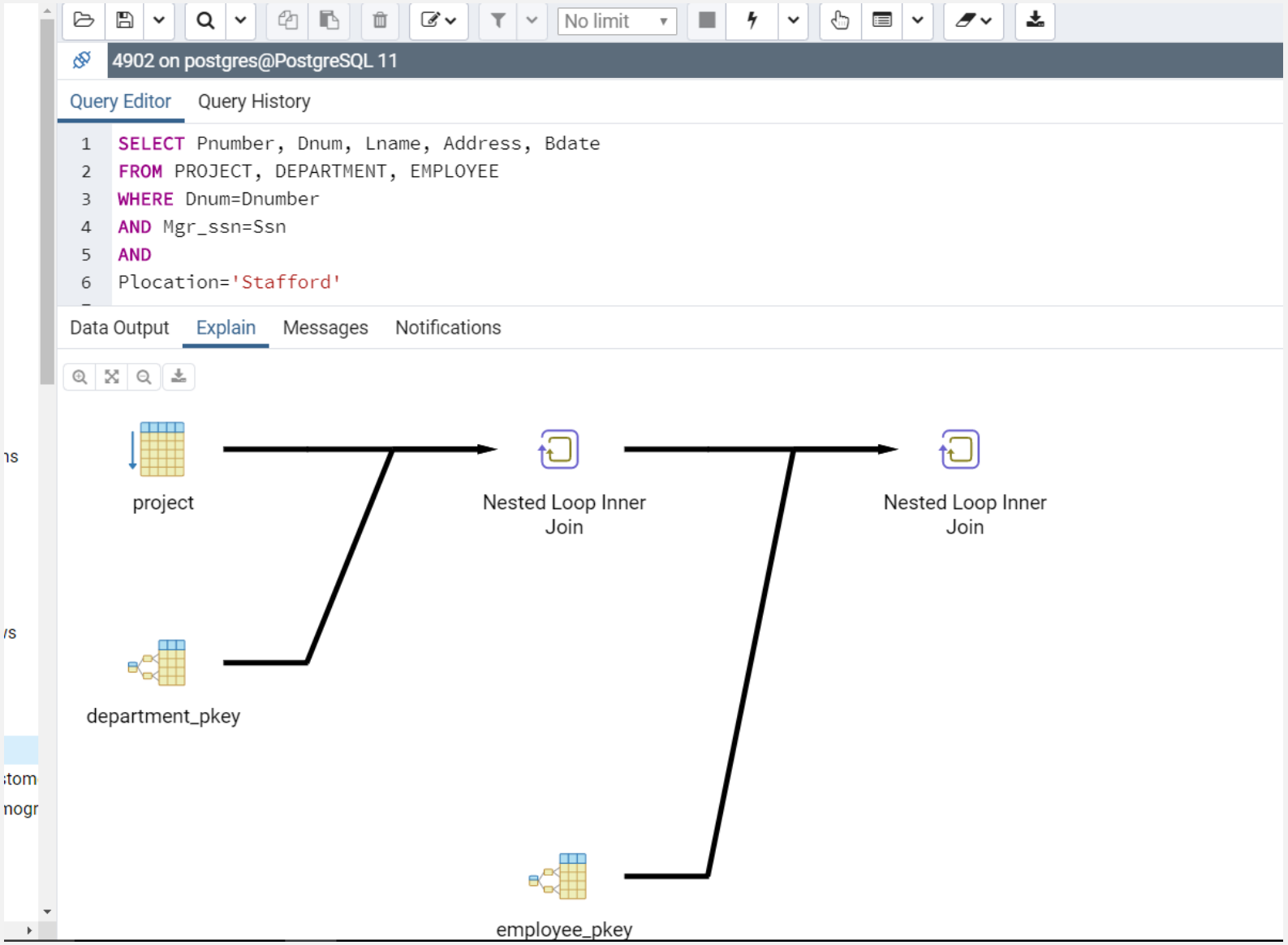
Query 2. For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

An actual explain plan from PostgreSQL



For interest only

Query 2. (again…rerun today)

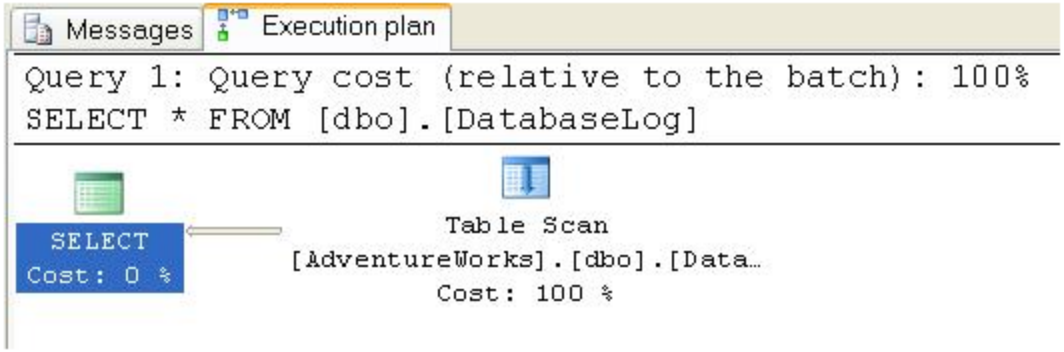


From https://www.simple-talk.com/sql/performance/execution-plan-basics/

For interest only

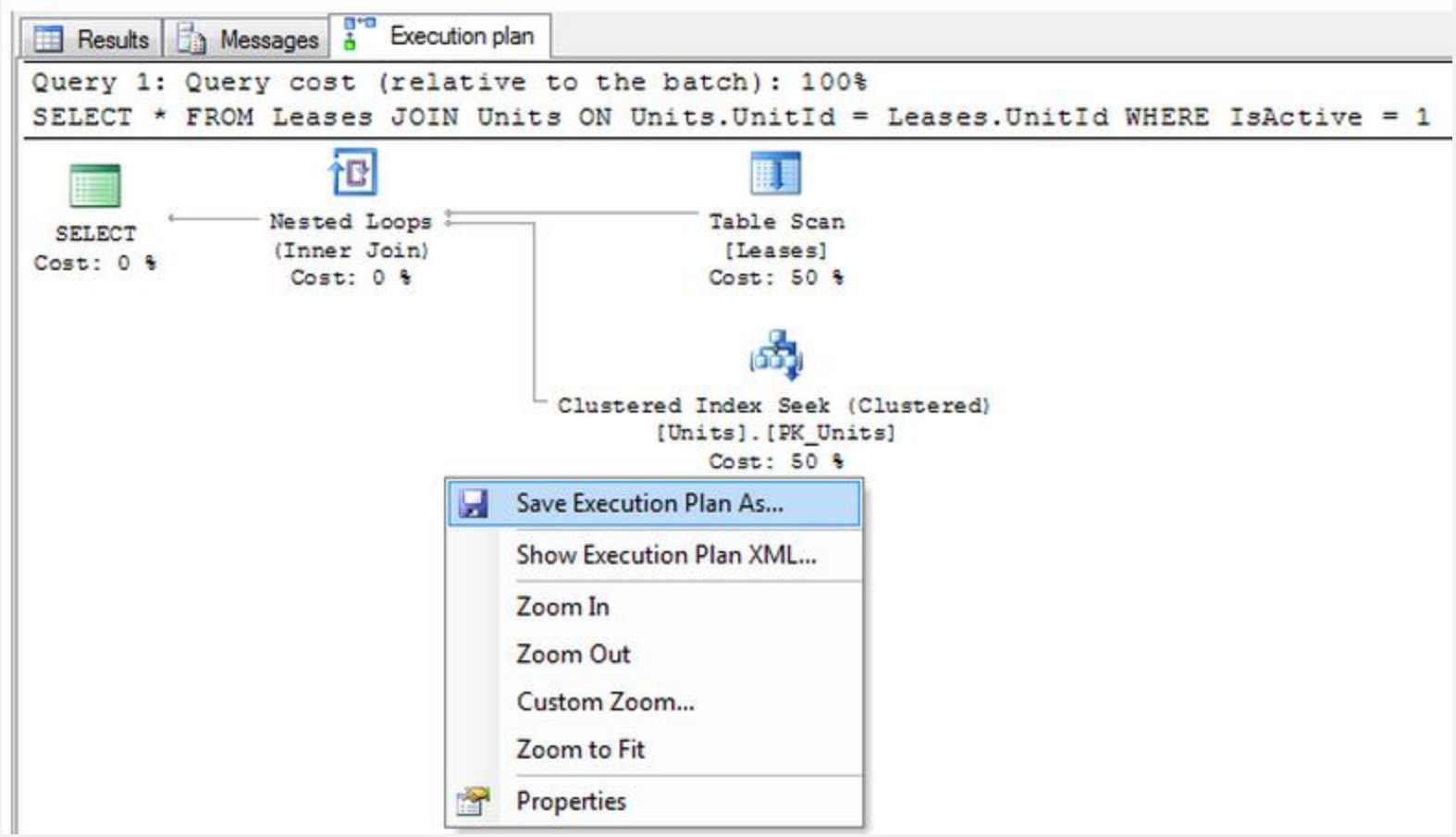
```
SELECT *
FROM [dbo].[DatabaseLog];
```

I tend to click the icon more often than not but, either way, we see our very first **Estimated execution plan**, as in Figure 1.



From http://stackoverflow.com/questions/7359702/how-do-i-obtain-a-query-execution-plan

For interest only



From https://dev.mysql.com/doc/workbench/en/wb-performance-explain.html

For interest only

```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer, address.phone, film.title
FROM rental
INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
LIMIT 5;
```