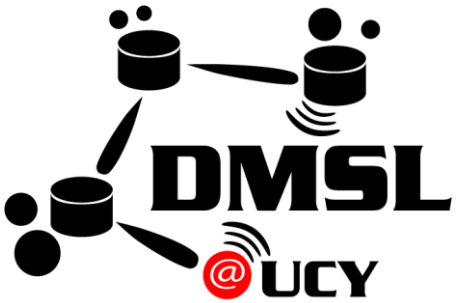
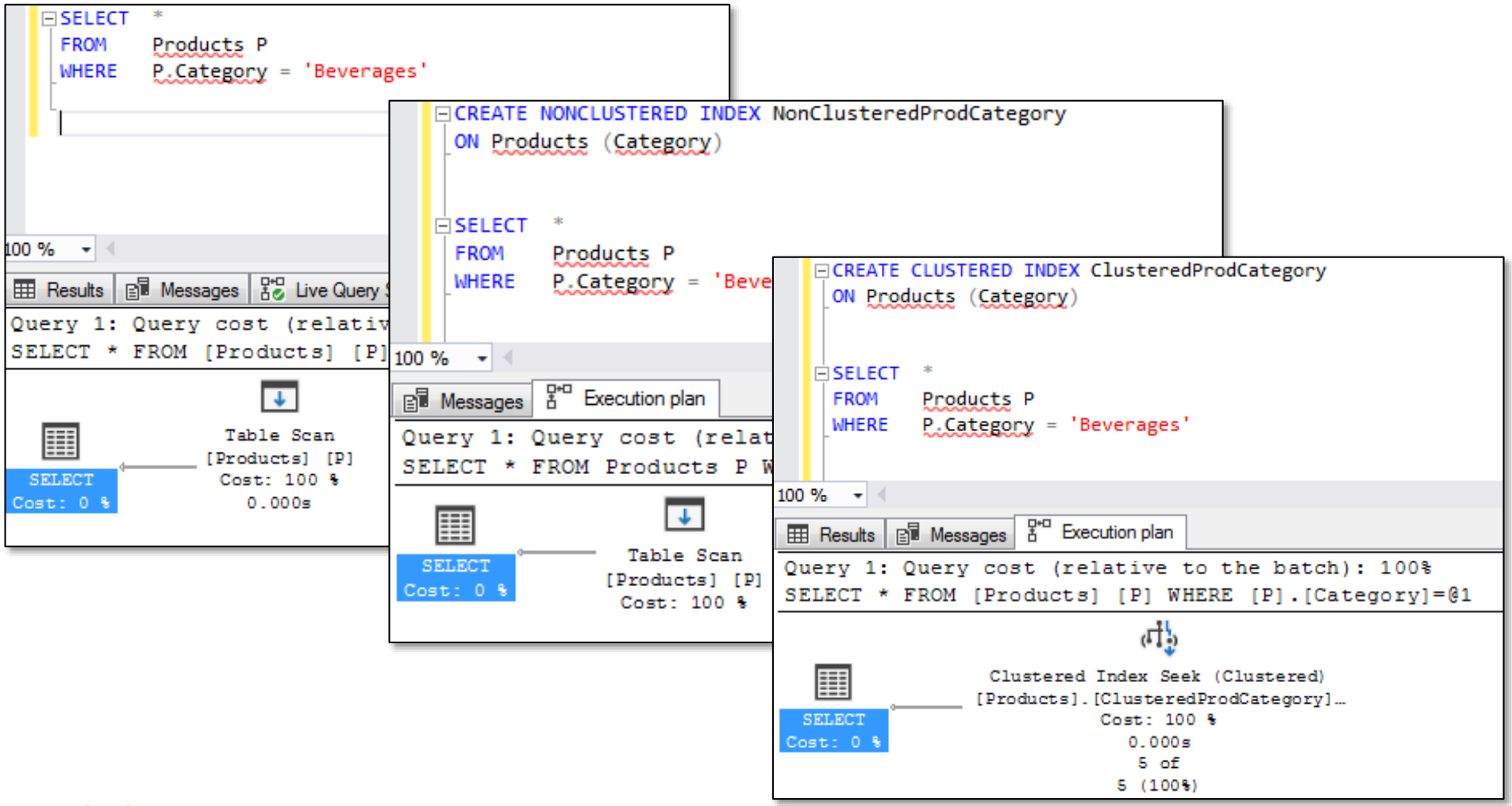
EPL646 – Advanced Topics in Databases

Query Optimizer

http://www.cs.ucy.ac.cy/~dzeina/courses/epl646/labs/lab.html



SQL server actual query plan: Simple SELECT



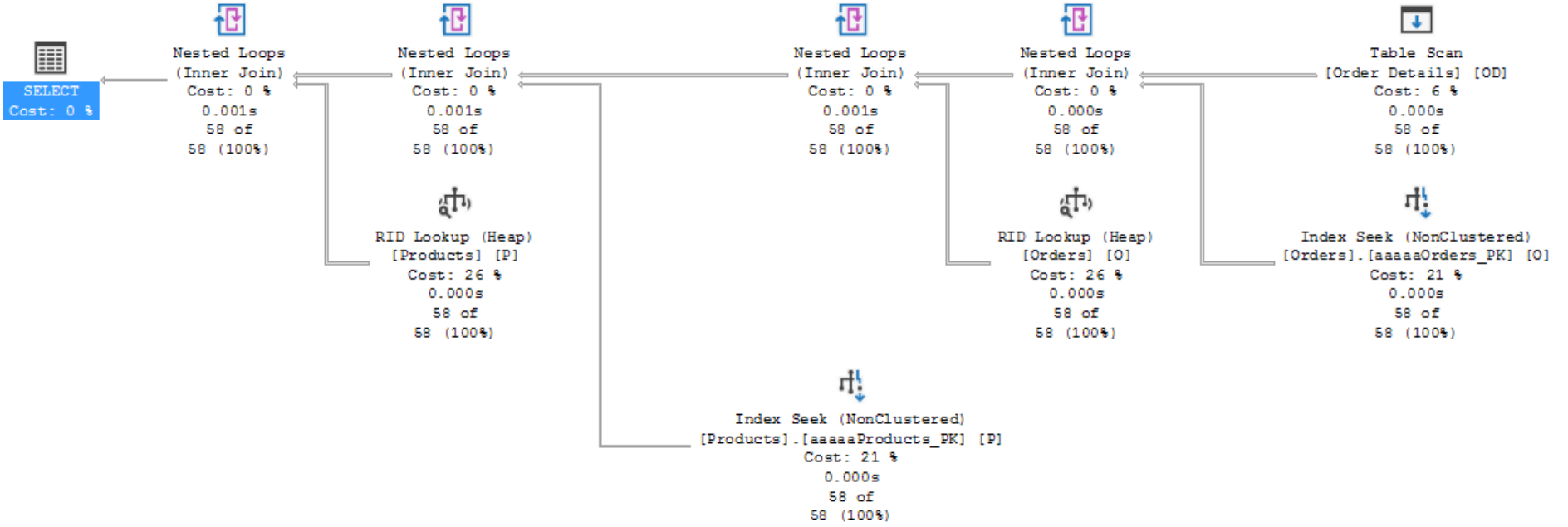
SQL server actual query plan: Simple JOIN – No index

```
SELECT  O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name]
FROM    Orders O INNER JOIN [Order Details] OD on (O.[Order ID] = OD.[Order ID])
        INNER JOIN Products P on (OD.[Product ID] = P.[ID])
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name] FROM Orders O INNER JOIN [Ord



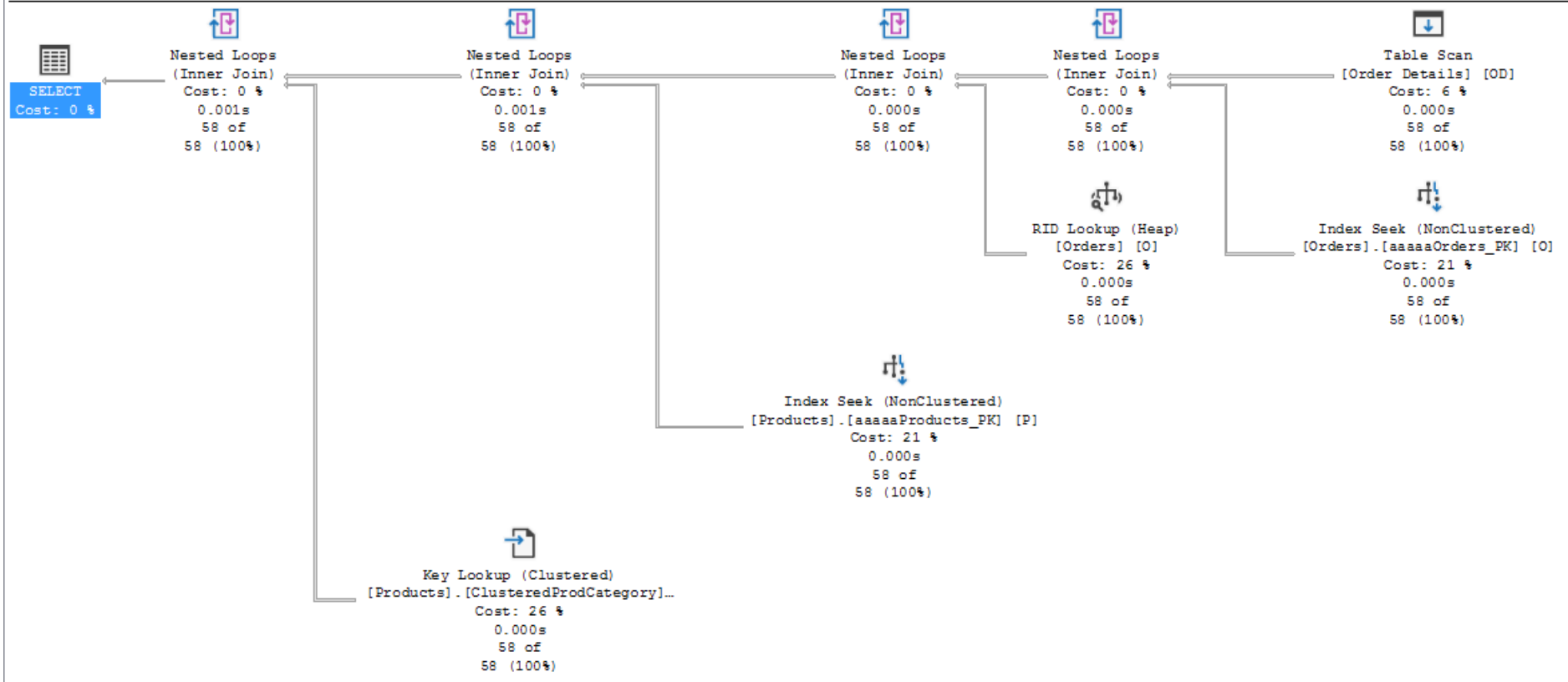
SQL server actual query plan: Simple JOIN – Clustered index

```
SELECT  O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name]
FROM    Orders O INNER JOIN [Order Details] OD on (O.[Order ID] = OD.[Order ID])
        INNER JOIN Products P on (OD.[Product ID] = P.[ID])
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name] FROM Orders O INNER JOIN [Order De…



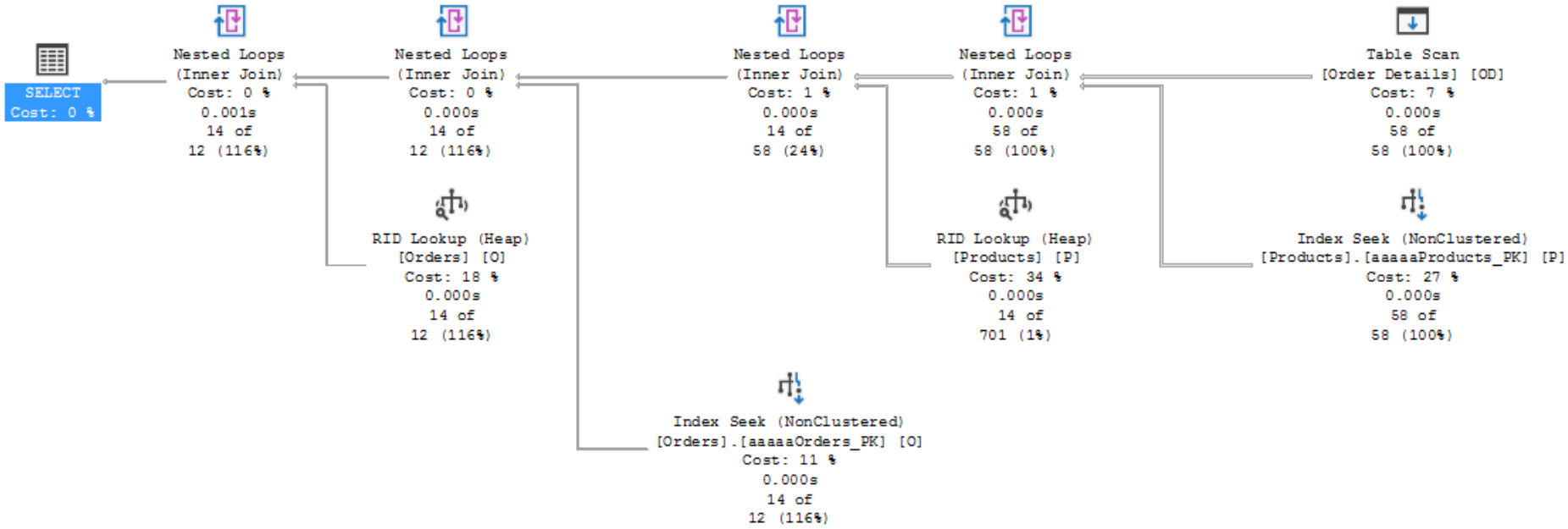
SQL server actual query plan: Simple JOIN with WHERE – No index

```
SELECT  O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name]
FROM    Orders O INNER JOIN [Order Details] OD on (O.[Order ID] = OD.[Order ID])
        INNER JOIN Products P on (OD.[Product ID] = P.[ID])
WHERE   P.Category = 'Beverages'
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name] FROM Orders O INNER JOIN [Order



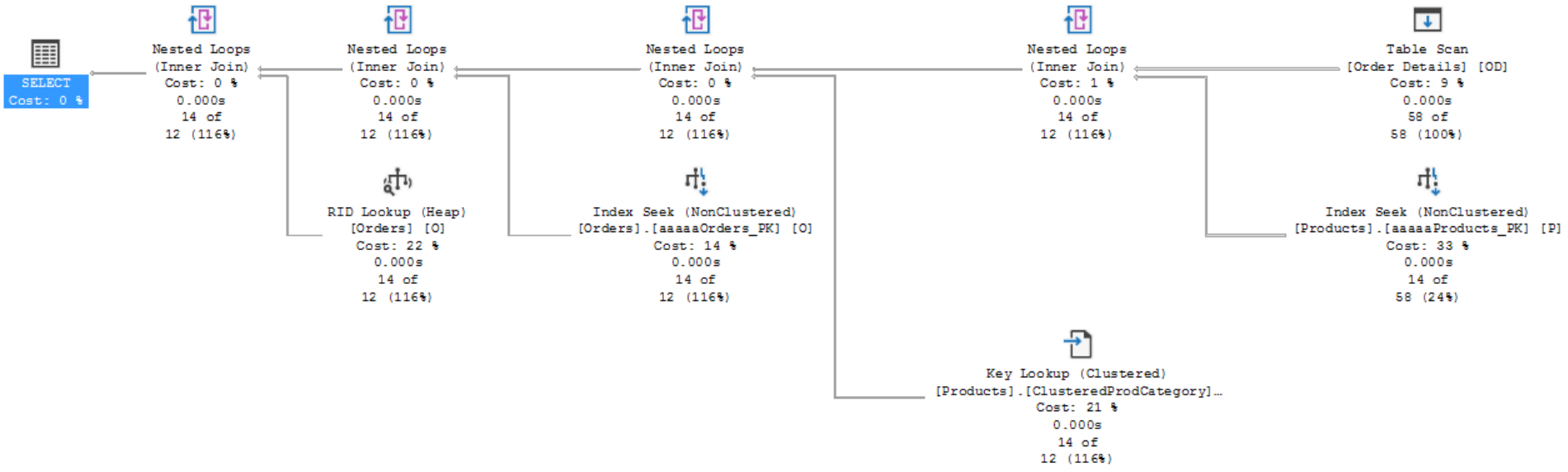
SQL server actual query plan:
Simple JOIN with WHERE – Clustered index

```
SELECT  O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name]
FROM    Orders O INNER JOIN [Order Details] OD on (O.[Order ID] = OD.[Order ID])
        INNER JOIN Products P on (OD.[Product ID] = P.[ID])
WHERE   P.Category = 'Beverages'
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT O.[Order ID], O.[Employee ID], O.[Customer ID], O.[Order Date], OD.[Product ID], P.[Product Name] FROM Orders O INNER JOIN [Order Details] O…



Exercise 15.3

(Exercise 15.3) For each of the following SQL queries, for each relation involved, list the attributes that must be examined to compute the answer. All queries refer to the following relations:

*Emp(**eid**: integer, **did**: integer, **sal**: integer, **hobby**: char(20))*
*Dept(**did**: integer, **dname**: char(20), **floor**: integer, **budget**: real)*

a. SELECT COUNT(*) FROM Emp E, Dept D WHERE E.did = D.did
b. SELECT MAX(E.sal) FROM Emp E, Dept D WHERE E.did = D.did
c. SELECT MAX(E.sal) FROM Emp E, Dept D WHERE E.did = D.did AND D.floor = 5
d. SELECT E.did, COUNT(*) FROM Emp E, Dept D WHERE E.did = D.did GROUP BY D.did
e. SELECT D.floor, AVG(D.budget) FROM Dept D GROUP BY D.floor HAVING COUNT(*) > 2
f. SELECT D.floor, AVG(D.budget) FROM Dept D GROUP BY D.floor ORDER BY D.floor

Exercise 15.7

(Exercise 15.7) Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances (organized on a per department basis).

*Emp(**eid**: integer, **did**: integer, **sal**: integer, **hobby**: char(20))*
*Dept(**did**: integer, **dname**: char(20), **floor**: integer, **phone**: char(10))*
*Finance(**did**: integer, **budget**: real, **sales**: real, expenses: real)*

Consider the following query:

```
SELECT D.dname, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.did=D.did AND D.did=F.did AND D.floor=1
AND E.sal ≥ 59000 AND E.hobby = 'yodeling'
```

Exercise 15.7

(Exercise 15.7)

a. Identify a relational algebra tree (or a relational algebra expression if you prefer) that reflects the order of operations a decent query optimizer would choose.

b. List the join orders (i.e., orders in which pairs of relations can be joined to compute the query result) that a relational query optimizer will consider. (Assume that the optimizer follows the heuristic of never considering plans that require the computation of cross-products.) Briefly explain how you arrived at your list.

Exercise 15.7

(Exercise 15.7)

c. Suppose that the following additional information is available: Unclustered B+ tree indexes exist on Emp.did, Emp.sal, Dept.floor, Dept.did, and Finance.did.

The system's statistics indicate that employee salaries range from 10,000 to 60,000, employees enjoy 200 different hobbies, and the company owns two floors in the building. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the data-base. The DBMS used by the company has just one join method available, index nested loops.

I. For each of the query's base relations (Emp, Dept, and Finance) estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.

II. Given your answer to the preceding question, which of the join orders considered by the optimizer has the lowest estimated cost?

Questions?

http://www.cs.ucy.ac.cy/~dzeina/courses/epl646/labs/lab.html