Theorem: If you take into account all the different algebraic formulations, join orders and algorithms for row retrieval and joins then there all together MANY ways to execute any query. Proof: some other time.

The heuristic: Make things as small as you can as rapidly as you can.

**The algorithm for optimizing query execution.**

Input: a query tree representing a query expressed either in SQL or relational algebra
Output: a query tree in which the heuristic rule has been applied.

1. Translate the SQL statement into a relational algebra statement.

2. Use rules 1, 2, 4, 6, 7 & 10 to move selections and projections as far to the bottom of the query tree as possible.

3. If possible perform selections and projections at the same time.

4. If possible combine select and products into joins.

5. In the case of multiple joins, move the most selective to the left.

6. Look for common subexpressions and consider materializing them as temporary tables

Example:

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE SALARY > (SELECT MAX (SALARY) FROM EMPLOYEE WHERE DNO = 5)
```

1. In this case, the query looks nested, but that are independent queries. First we have to work with:

```
SELECT MAX(SALARY)
FROM EMPLOYEE
WHERE DNO = 5
```

It will become: \( \gamma(\pi_{\text{SALARY}}(\sigma_{\text{DNO} = 5}(\text{EMPLOYEE}))) \), where \( \gamma \) represents the maximum of the salaries obtained. Assume that value is \( c \).

Then the outer block

```
SELECT LNAME, FNAME
FROM EMPLOYEE
```
WHERE SALARY > c

will be transformed into $\pi_{LNAME, FNAME} (\sigma_{SALARY > c} (EMPLOYEE))$

We don’t have much choice here. 
So the parsing tree will look like:

```
\pi_{LNAME, FNAME}
\mid
\sigma_{SALARY > \gamma}
\mid
\pi_{SALARY}
\mid
\sigma_{DNO = 5}
\mid
EMPLOYEE
```

Example of how a Query optimization is done in Ingres.

A query execution plan for

```
SELECT DISTINCT t.conf, s.season
FROM teams t, seasons s
WHERE t.conf = s.conf AND t.conf != "
AND t.type='MB' AND s.season < '1980';
```

According to our approach, we could write

```
\pi_{t.conf, s.season}(\sigma_{\text{conf} \neq " AND \text{type} = 'MB'(t)} \triangleright \sigma_{\text{conf} = \text{season} < '1980'(s)})
```

and then we have to discard all the nondistinct tuples.

We could modify using rule number 7.

```
\pi_{t.conf}(\sigma_{\text{conf} \neq " AND \text{type} = 'MB'(t)}) \triangleright \sigma_{\text{conf} = \text{season} < '1980'(s)}
\pi_{s.conf, s.season}(\sigma_{\text{season} < '1980'(s)})
```

Then because table s is bigger, (see values below), we can switch the order of the join.

```
\pi_{s.conf, s.season}(\sigma_{\text{season} < '1980'(s)}) \triangleright \sigma_{\text{conf} = \text{conf} \neq " AND \text{type} = 'MB'(t)}
```

This would be our approach.
The parsing tree corresponding to this query could be represented as:

```
  Unique
    ▷◁ s.conf = t.conf
      ▷◁ s.conf, s.season
          ▷◁ t.conf
            ▷◁ season < '1980'
                ▷◁ season = 'MB'
                    s
                    t
```

On the other hand, if we look at the same query, performed in Ingress, the optimizer changes the same query into the following relational statement

\[
(\pi_{\text{conf}}(\sigma_{\text{season} < '1980'}(s)) \bowtie s.\text{conf} = t.\text{conf}) \bowtie t.\text{conf} \left(\sigma_{\text{conf} \neq 'M'} \text{ AND } \text{type} = 'MB'(t))
\]

Then, of course, you have to discard all the nondistinct tuples.

Check the tree corresponding to this statement, with all the extra information that the DBMS supplies to us.

```
Sort Unique
Pages 1 Tups 81
D271 C720

T Join(tidp)
Heap
Pages 1 Tups 81
D271 C713

PSM Join(CO)(conf)
Heap
Pages 1 Tups 81
D190 C711

Proj-rest
Heap
Pages111 Tups 18432
D190 C369

confidx
(s)
B-Tree(NU)
Pages 1762 Tups 36864

Proj-rest
SortU on(conf)
Pages 3 Tups 650
D121 C54

fullteams
(t)
isam(conf)
Pages 241 Tups 5372
```
Checking Optimization in Oracle

a. Include the line

```
ALTER session set sql_trace=true;
```

as the first line of your SQL file, named here tracetest.sql

b. On the second line enter a "!" without quotes, when this statement executes, it will put you back to the OS.

c. Finish typing your sql query.

d. Enter the Sql buffer by typing

```
% sqlplus username/userid @tracetest
```

e. You will get a message saying "statement altered", then when the second line is executed, you will thrown back into the operating system where you must enter the command

```
% ps -fu <yourusername>
```

to produce a list of oracle processes. Note the pid (process id ) for the sqlplus, process, it will be a 5 digit integer like 25607 . Type

```
%exit
```
to exit from the cshell and go back to your sql session. Your query will execute.

f. Quit Sqlplus.

g. At OS prompt issue the command:

```
% tkprof /db/app/oracle/admin/OR73/udump/or73_ora_nnnnn.trc output= <filename>
explain=<userid/password>
```

where nnnnn is the process id obtained earlier.

h. Examine the output file.