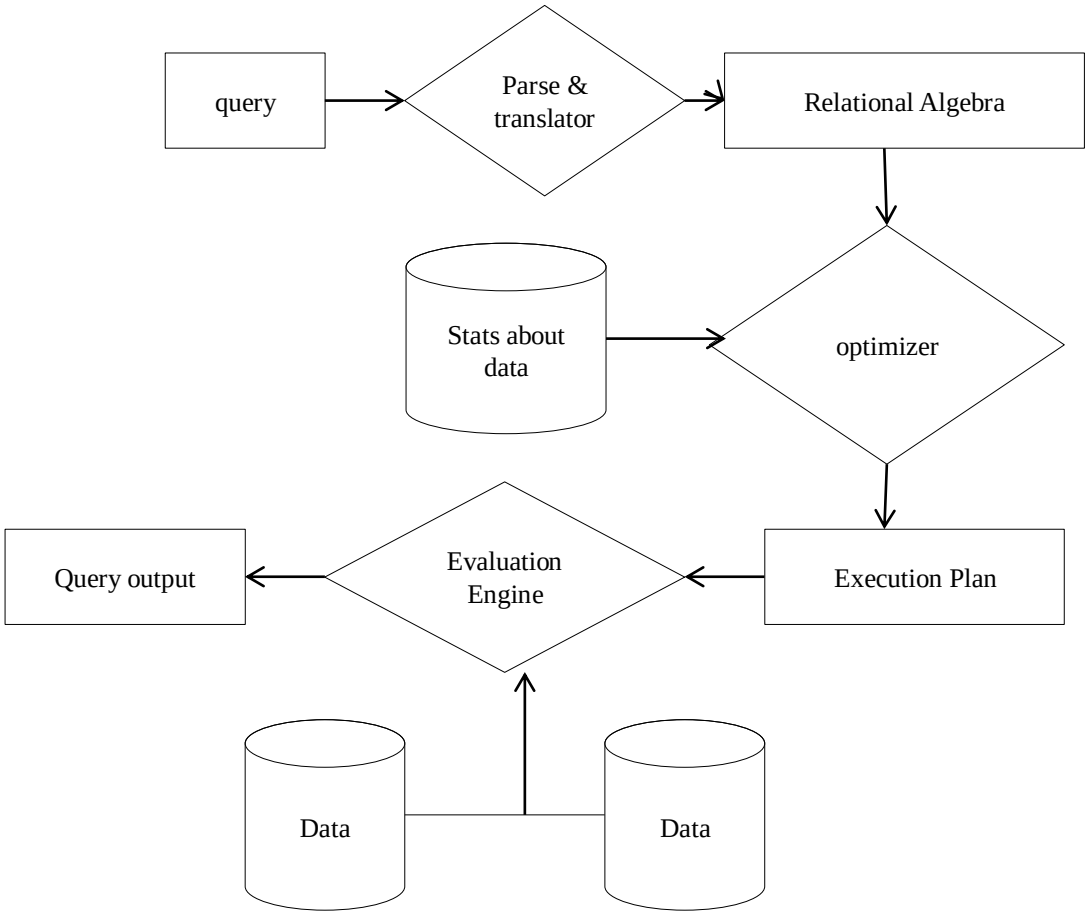
Lecture Notes
Topic: *Query Optimization*

Date: 18 Oct 2011 Made By: Naresh Mehra
Shyam Sunder Singh

Query Processing:

Query processing refers to activities including translation of high level language(HLL) queries into operations at physical file level, query optimization transformations, and actual evaluation of queries.

Steps in query processing :



Function of query parser is parsing and translation of HLL query into its immediate form relational algebraic expression.

A parse-tree of the query is constructed and then translated in to relational algebra expression.

example: consider the following SQL query:

```
SELECT S.sname
FROM Reserves R,Sailors S
WHERE R.sid = S.sid
    AND R.bid = 100 AND S.rating > 5
```

this query can be expressed in relational algebra as follows:

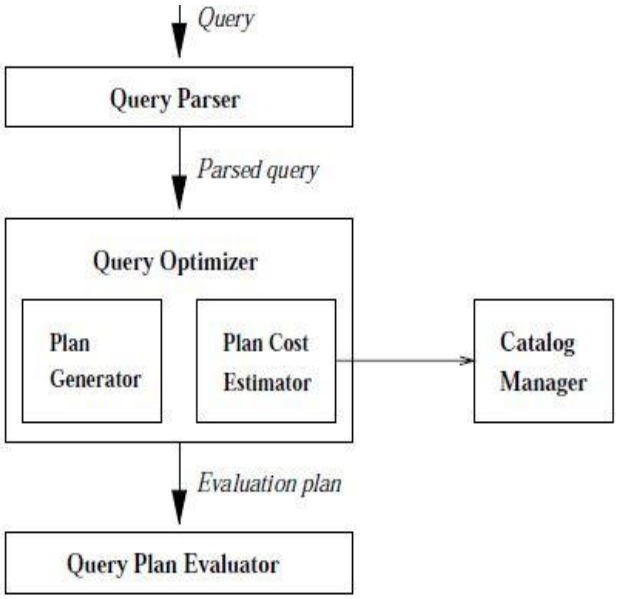
$$\pi_{sname}(\sigma_{bid=100 \wedge rating>5}(Reserves \bowtie_{sid=sid} Sailors))$$

Query Optimization:

Query optimization is the process of selecting an efficient execution plan for evaluating the query.

After parsing of query, parsed query is passed to query optimizer, which generates different execution plans to evaluate parsed query and select the plan with least estimated cost.

Catalog manager helps optimizer to choose best plan to execute query by generating cost of each plan.



Evaluation plans for different operators involved in parsed query(relation algebric expression):
Let we have F1 with m pages of records and F2 having n pages:Let we have a cross product of two data files:
F1 X F2

Cross product(X):

We can have different scenario and based on those scenario we will choose different execution plan for query.

Scenerio1:- when both data files are small.

Load both F1 and F2 in main memory and then do cross product.

Pick up one record at a time from F1 and do cross product with every element of F2 and repeat this process for every element of F1. As both data files are small so no require of lot memory space.

Scenerio2: when F1 is very large and F2 is small.

In this case we cannot load F1 in main memory as it will requires large space and it is not an efficient plan.

Solution:- In this case load only F2 in main memory and read records from F1 direct from secondary memory and do cross product since we don't require to write records into F1 hence we have least amount of I/O in this process and also no large space overhead.

Amount of I/O = size of F1(no. of pages in F1) + size of F2(no. of pages in F2) = m+n

Above one is the best execution plan for given scenario as in this we have to read at least once from both the files.

Scenerio3: When we have a limited constant space available of main memory.

case1:- Let we have two pages p1 and p2 main memory available to perform cross product of F1 and F2.

Then we read a page from F1 and load it into p1 and a page from F2 into p2 and then we don't change p1 and change contents of p2 by

loading elements of F2 till n and perform cross product every time. After this change p1 and thus repeat this process till m times.
Amount of I/O = m x n

Case2: find the best way to perform cross product when pages of main memory available equals to 4(p1, p2, p3, p4).
We have two solution for this:
Solution1:- read 2 pages from F1 at a time and also 2 pages from F2 then perform cross product between them and then remove last two pages of F2 and read two new pages and repeat this process till n. thus we repeat same process n/2 times then we read 2 next pages from F1 and repeat process m/2 times.
Hence Amount of I/O = (m/2)*(n/2)

Solution2:- read 3 pages from F1 at a time and also 1 page from F2 then perform cross product between them and then remove last pages of F2 and read new page and repeat this process till n. thus we repeat same process n times then we read 3 next pages from F1 and repeat process m/3 times.
Hence Amount of I/O = (m/3)*n

Scenerio4: When we have amount of memory not constant i.e. some fraction of data files first half pages (p1) from F1 and read half pages (p2) from F2 and do cross product then read another half part of F2 (p3) and do cross product with same part of F1(p1) and then read another half part of F1 (p4) and do cross product with latest read part of F2 (p3) then do cross product with first read part of F2 (p2).
Amount of I/O = (m/2)+(n/2)+(n/2)+(m/2)+(n/2)

so we conclude that if we have size of main memory available is constant then amount of total I/O will be multiplication and if is fraction part of data then linear.

On the basis of following two factors we select algorithms for query execution:

1. Size of dbms.
2. Amount of main memory available.

Natural join (|X|):

Scenerio1: if we sort the file in common attribute
First take record from F1 then from F2 if matches then store in temporary file and repeat this process for all records of F1.
Amount of I/O = size of F1 + size of F2

Like cross product we also require different sorting method on the basis of different scenario present like size of data files. Best sorting method does not require main memory.

Scenario_X: if we have both F1 and F2 very-2 large we should use merge sort for this case.

Scenario_Y: if we have very-2 large data files and some memory in fraction of data files:
Let we have 5% pages main memory available of data file having 100 pages then we read 5 pages from F1 and sort them and assign one page and again next 5 pages and sort them and assign one page thus we have different sorted group assigned into one page. Then keep on merging between these all sorted groups and get a final sorted file.
Amount of I/O = whole scan of file
= to create 20 sorted file + cost of merging

Comparison between above types of memory available (constant and fraction of data file):

1. In first one we have sufficient memory available but in second one we have some fraction of data file
2. in scenario_X we sort records in decreasing order i.e. first we have more sorted no. of files those reduces to

one file. but in scenario_Y in starting we have fix no. of files those after sorting reduces to one file.

Selection operation(σ):

If given data file is large then instead of load it into main memory we take a temporary page in main
memory and pick records from data files(secondary memory)one by one, if any particular record qualify required
condition then store it.

Projection(Π):

In given data files we search table by table as above and if we found required attribute in particular table then store that attribute and values.