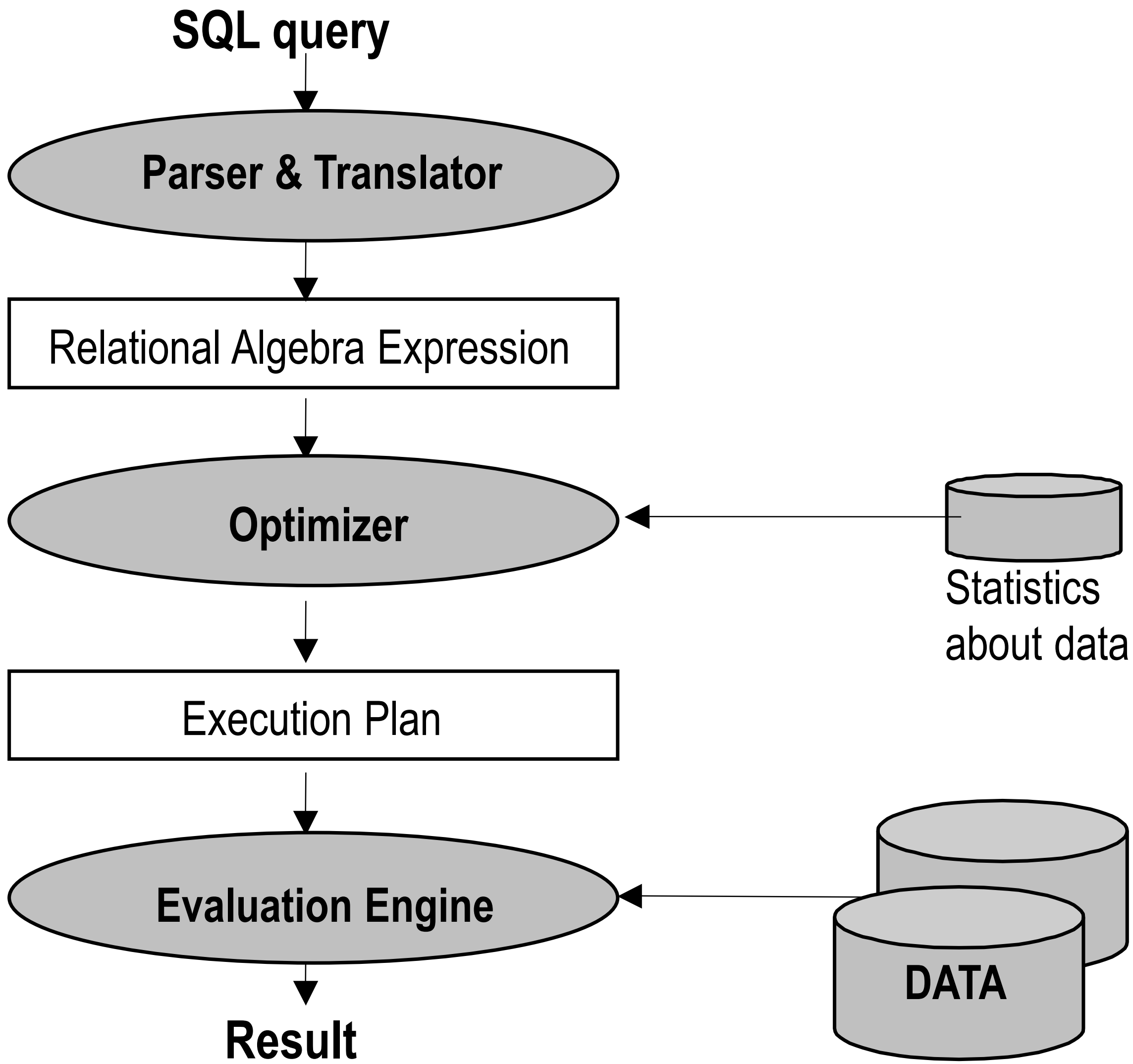
Query Processing and Optimization

Introduction

- Users are expected to write "efficient" queries, but they don't always do that :
  - Users typically don't have enough information about the database to write efficient queries.
  - E.g.: no information on table size

- DBMS's job is to optimize the user's query by:
  - Converting the query to an internal representation (tree or graph)
  - Evaluate the costs of several possible ways of executing the query and find the best one.

Steps in Query Processing



Select Operation

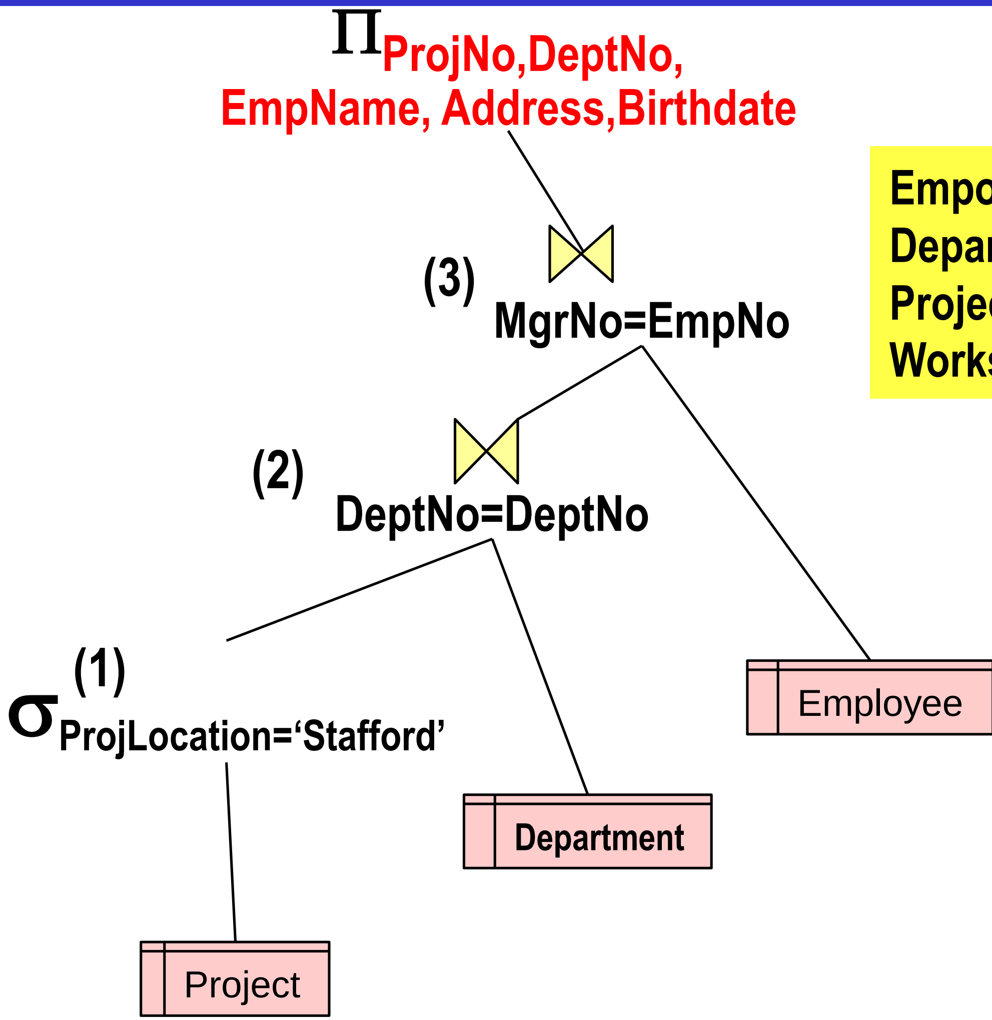
- File scan : **scan all records of the file to find records that satisfy selection condition**
- Binary search : **when the file is sorted on attributes specified in the selection condition**
- Index scan : **using index to locate the qualified records**
  - Primary index,
    - single record retrieval : **equality comparison on a primary key attribute with a primary index**
    - multiple records retrieval : **comparison condition <, >, etc. on a key field with primary index**
  - **Clustering index to retrieve multiple records**
  - **Secondary index to retrieve single or multiple records**

Query Optimization

- **Give a relational algebra expression, how do we transform it to a more efficient one?**

Use the query tree **as a tool to rearrange the operations of the relational algebra expression**

A Query Tree



Empolyee (EmpNo, EmpName, Address, Birthdate, DeptNo)
Department (DeptNo, DeptName, MgrNo)
Project (ProjNo, ProjName, ProjLocation)
WorksOn (EmpNo, ProjNo, Hours)

Structure and Execution of a Query Tree

- A query tree is a tree structure that corresponds to a relational algebra expression by representing:
  - the input relations as leaf nodes
  - and the relational algebra operations as internal nodes of the tree

- An execution of the query tree consists of:
  - executing an internal node operation whenever its operands are available
  - and then replacing that internal node by the relation that results from executing the operation

Heuristics for Optimizing a Query

A query may have several equivalent query trees

A query parser generates a standard canonical query tree from a SQL query tree

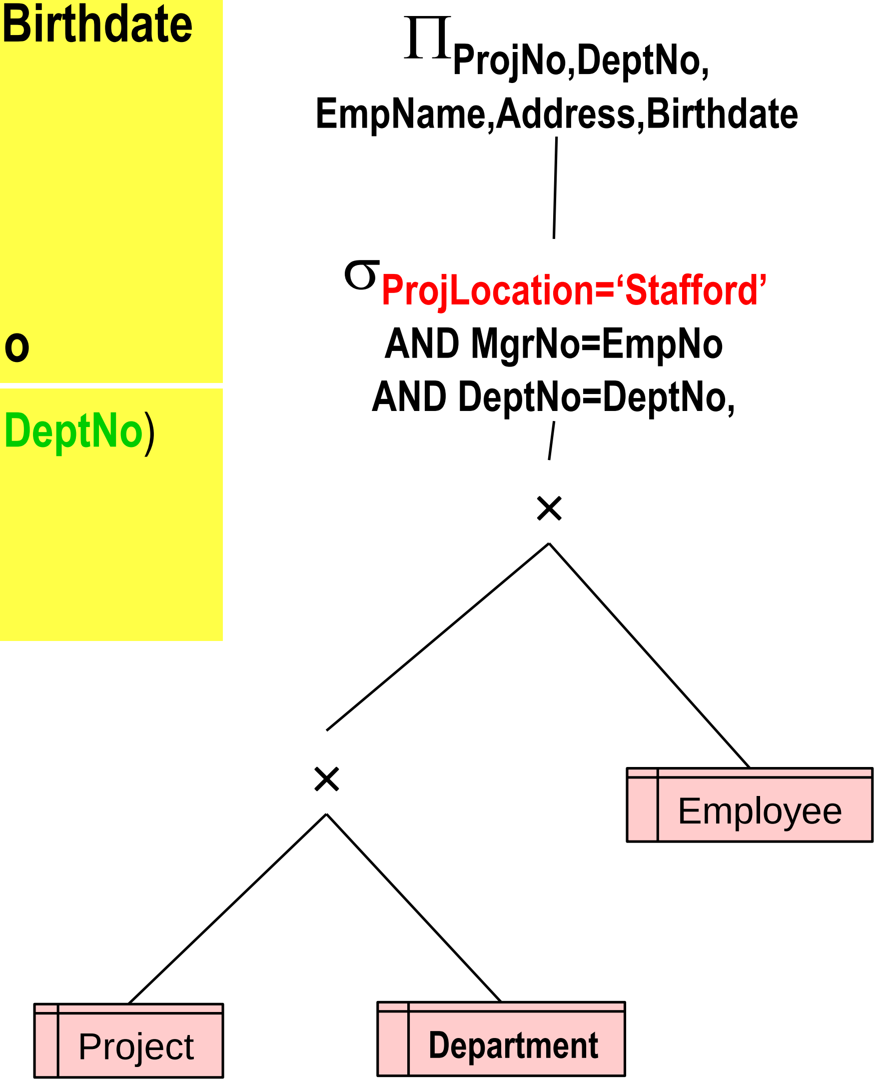
1. Cartesian products are first applied (FROM)
2. then the conditions (WHERE)
3. and finally projection (SELECT)

Heuristics for Optimizing a Query

select ProjNo, DeptNo, EmpName, Address, Birthdate
from Project, Department, Employee
where ProjLocation='Stafford' and
MrgNo=EmpNo and
Department.DeptNo=Employee.DeptNo

Empolyee (EmpNo, EmpName, Address, Birthdate, DeptNo)
Department (DeptNo, DeptName, MgrNo)
Project (ProjNo, ProjName, ProjLocation)
WorksOn (EmpNo, ProjNo, Hours)

The **query optimizer**
transforms this **canonical** query
into an efficient final query

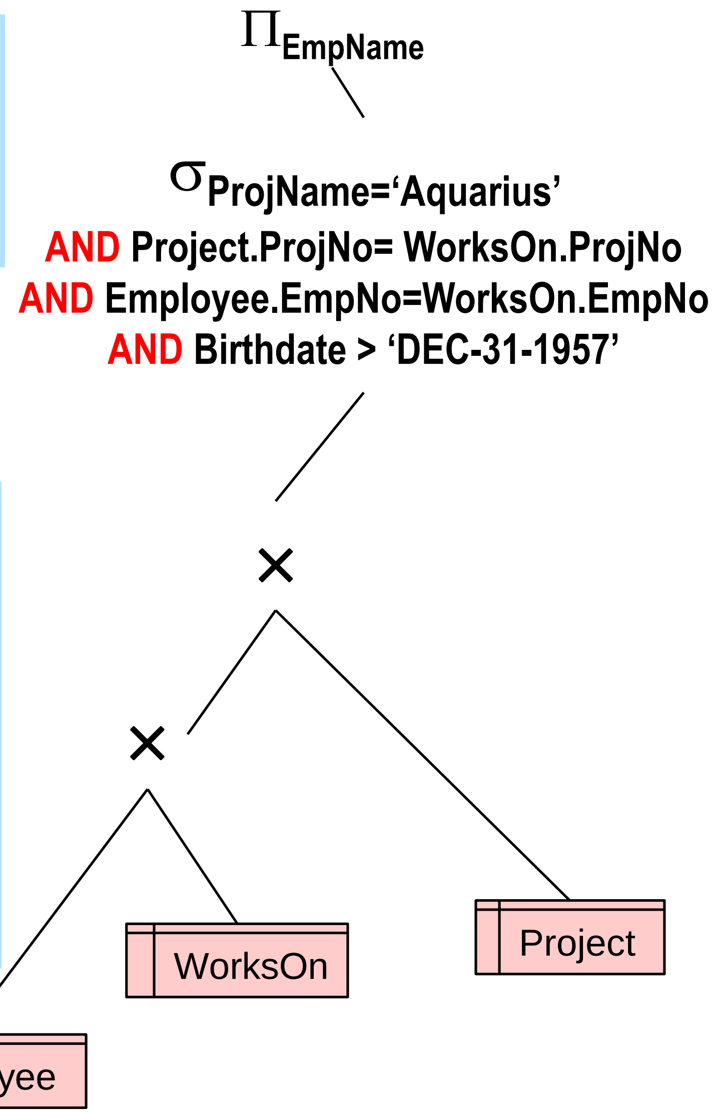


Example

Empolyee (EmpNo, EmpName, Address, Birthdate, DeptNo)
Department (DeptNo, DeptName, MgrNo)
Project (ProjNo, ProjName, ProjLocation)
WorksOn (EmpNo, ProjNo, Hours)

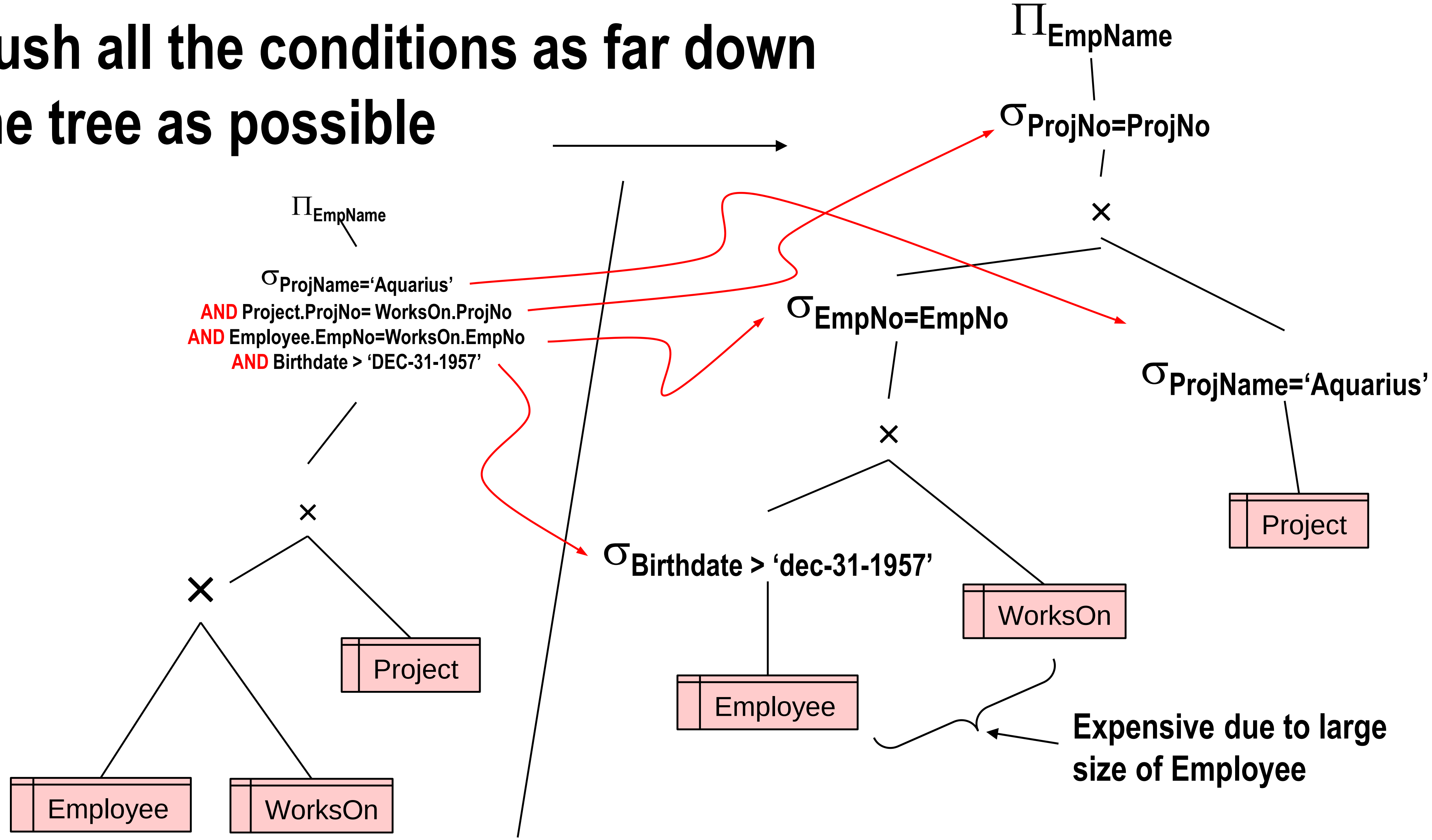
Find the names of employees born after 1957 who work on a project named 'Aquarius'

```
select  EmpName
from    Employee, WorksOn, Project
where   ProjName='Aquarius' AND
        Project.ProjNo=WorksOn.ProjNo AND
        Employee.EmpNo = WorksOn.EmpNo AND
        Birthdate >'DEC-31-1957'
```



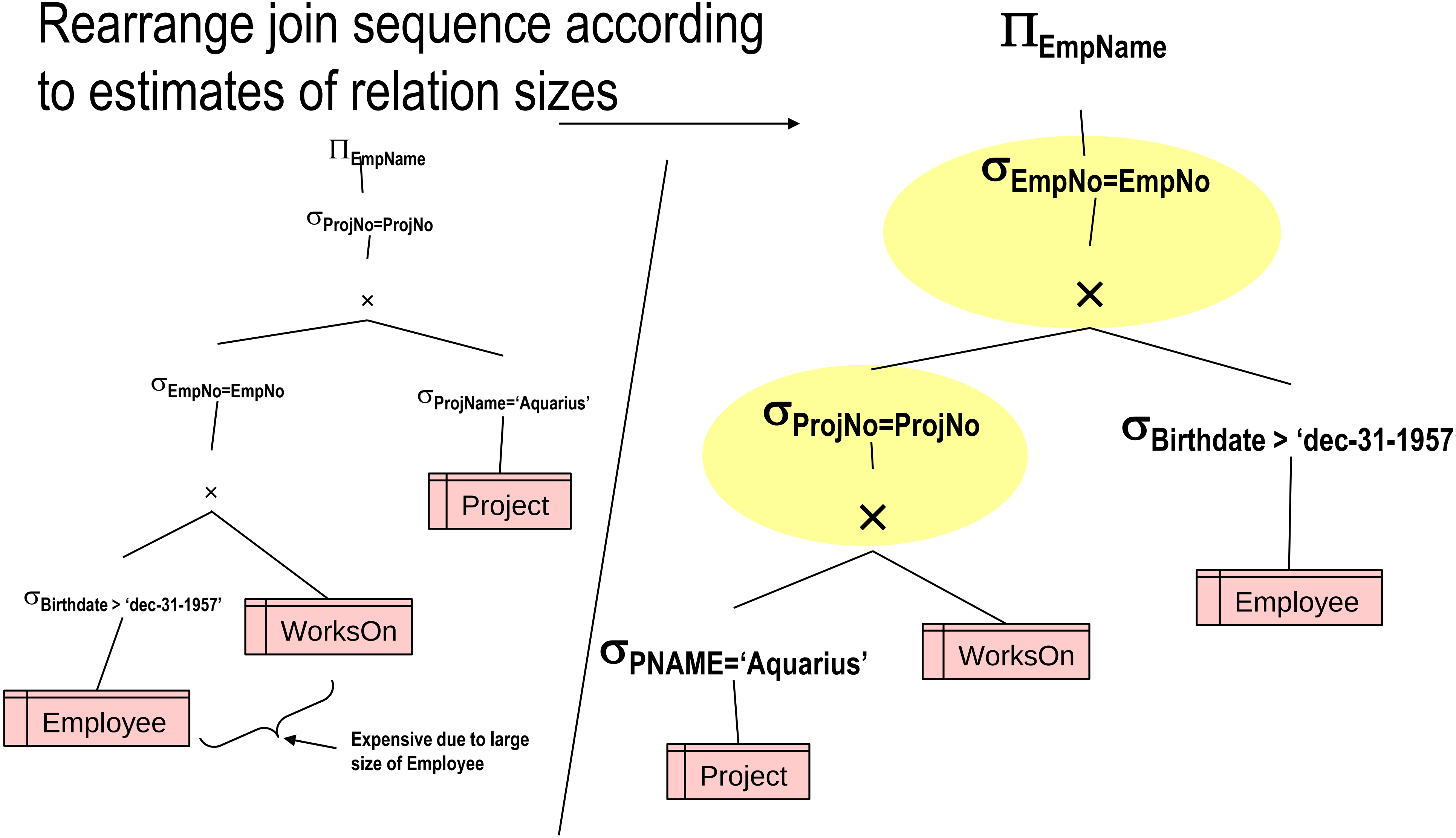
Example

Push all the conditions as far down the tree as possible



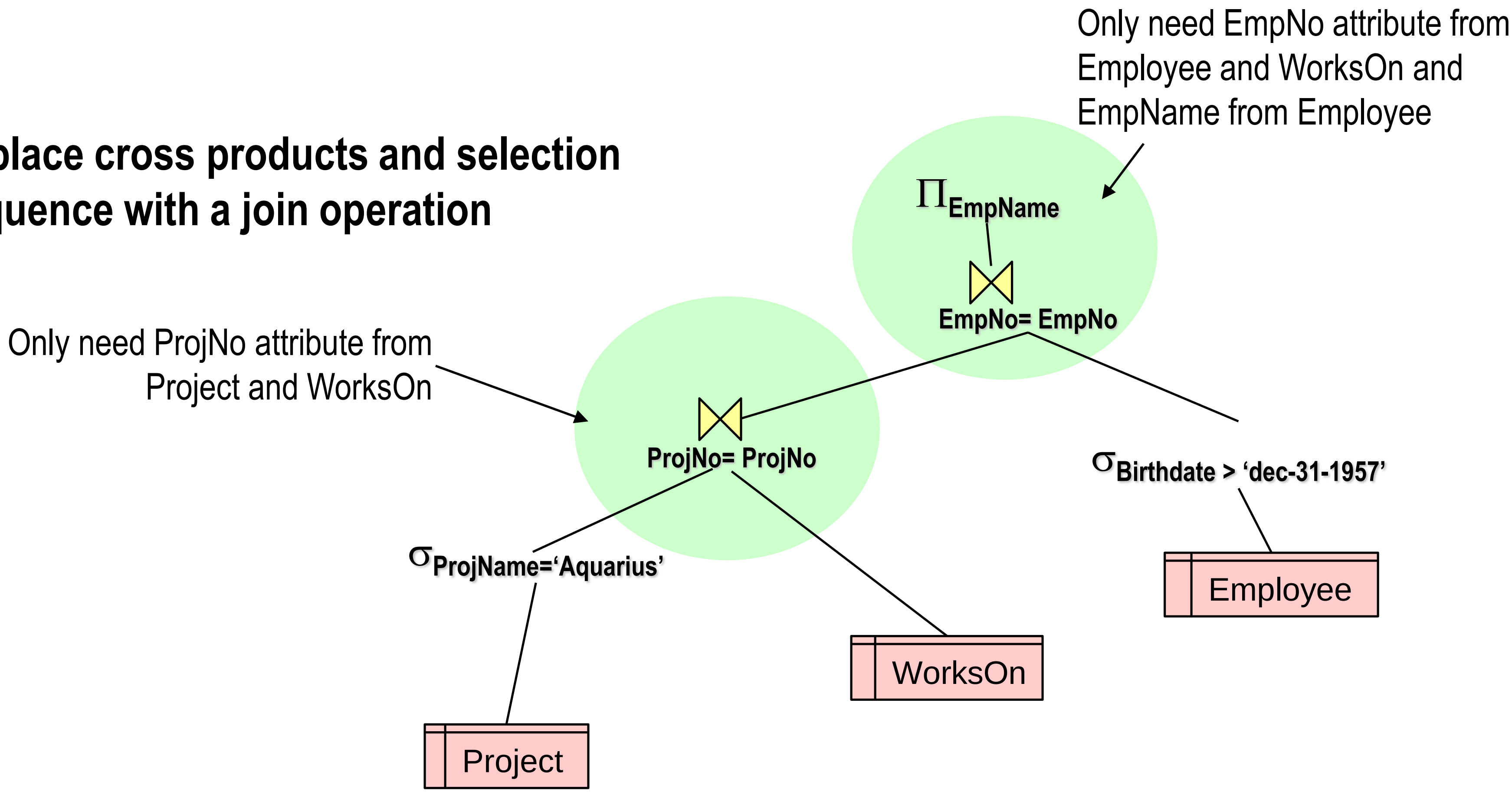
Example

Rearrange join sequence according to estimates of relation sizes



Example

Replace cross products and selection sequence with a join operation



Example

Push projection as far down the query tree as possible