Exercise

Let's consider the following relational schema for a company, with branches in different cities:

EMPLOYEE(ENumber, Name, Age, Salary, BranchCode)
PROJECT(ProjCode, Description, Budget)
BRANCH(BranchCode, BranchName, BranchCity)
WORK(ENumber, ProjCode, PercentageOfTime)

Primary keys are underlined in the relations. Moreover, ENumber in WORK is foreign key of EMPLOYEE; ProjCode in WORK is foreign key of PROJECT and BranchCode in EMPLOYEE is foreign key of BRANCH.
An employee works at a branch. Tasks in different projects can be assigned to an employee. PercentageOfTime is the percentage of time that an employee works in a project.

Assume that:

$n_{EMPLOYEE}$ 1000
$n_{PROJECT}$ = 100
n_{BRANCH} = 20
n_{WORK} = 5000

V(BranchCity, BRANCH) = 10
V(BranchCode, EMPLOYEE) = 20
V(ENumber, WORK) = 1000
V(ProjCode, WORK) = 100

Given the query:

Codes of projects with workers in Pisa branches

1) express the query as a relational-algebra expression;
2) show the basic steps of the query optimization process in terms of relational-algebra expression transformations
3) give an efficient strategy for computing the query.

Point 1

$\Pi_{WORK.ProjCode}\ (\sigma_{BRANCH.BrachCity=Pisa}(\ (EMPLOYEE\ \ |X|_{EMPLOYEE.ENumber=WORK.ENumber}\ WORK)\ |X|_{EMPLOYEE.BranchCode=BRANCH.BranchCode}\ BRANCH))$

Let E, W and P denote EMPLOYEE, WORK and BRANCH, respectively.

$\Pi_{W.ProjCode}\ (\sigma_{B.BranchCity=Pisa}(\ (E\ \ |X|_{E.ENumber=W.ENumber}\ W)\ |X|_{E.BranchCode=B.BranchCode}\ B))$

Point 2

Push selection down

$\Pi_{W.ProjCode}\ (\ (E\ \ |X|_{E.ENumber=W.ENumber}\ W)\ \ |X|_{E.BranchCode=B.BranchCode}\ \ (\sigma_{BranchCity=Pisa}(\ B)))$

Push projection down

$\Pi_{W.ProjCode}\ (\ (\ (\Pi_{ENumber,BranchCode}\ \ E)\ \ |X|_{E.ENumber=W.ENumber}\ (\Pi_{Enumber,ProjCode}\ W))\ |X|_{E.BranchCode=B.BranchCode}\ (\ \Pi_{BranchCode}\ (\sigma_{BranchCity=Pisa}(\ B)))$

Estimate of size and different values for the new relations.

Let $B' = \sigma_{BranchCity=Pisa}(B)$
$n_{B'} = n_{BRANCH} / V(BranchCity, BRANCH) = (20/10) = 2$

Let $B'' = \Pi_{BranchCode}(B')$
$n_{B''} = n_{B'} = 2$ BranchCode is a key
$V(BranchCode, B'') = 2$

Let $E' = \Pi_{ENumber,BranchCode}\ E$
$n_{E'} = n_{EMPLOYEE} = 1000$ ENumber is a key
$V(BranchCode, E') = V(BranchCode, E) = 20$

Let $W' = \Pi_{Enumber,ProjCode}\ W$
$n_{W'} = n_{WORK} = 5000$ ENumber,ProjCode is a key

Point 3

The query expression can be rewritten using natural join operator. Natural join is commutative.
$\Pi_{W.ProjCode}(E' |X| W' |X| B'')$

We estimate the size of different combinations of join.

$T1 = (E' |X|_{E'.ENumber=W'.ENumber} W')$
Number of records in the result:
ENumber in W' is foreign key of E'
$n_{T1} = n_{W'} = 5000$

$T2 = (E' |X|_{E'.BranchCode=B''.BranchCode} B'')$
BranchCode in E' is not foreign key of B" (B" has less values of BranchCode than B)

BranchCode in E' is a key of B"
$n_{T2} < n_{E'} < 1000$
More precisely :
n_{T2} = number of employees for each branch * number of branches
-employees for each branch: $n_{EMPLOYEE} / V(BranchCode, EMPLOYEE) = 1.000 / 20 = 50$
-number of branches: $n_{B''} = 2$
$n_{T2} = 50 * 2 = 100$
Rule applied by the optimizer:
$\min(n_{E'} * (n_{B''} / V(BranchCode, B''), n_{B''} * (n_{E'} / V(BranchCode, E')) =$
$\min(1.000 * (2/2), 2 * (1000/20)) = 100$

T3 = (W' |X| B") cartesian product
$N_{T3} = 5.000 * 2 = 10.000$

The best ordering of join is: $((E' |X|_{E'.BranchCode=B''.BranchCode} B'') |X|_{E'.ENumber=W'.ENumber} W')$

An efficient strategy for solving the query is:
$\Pi_{W.ProjCode}((E' |X|_{E'.BranchCode=B''.BranchCode} B'') |X|_{E'.ENumber=W'.ENumber} W')$