Exercise

Let's consider the following relational schema for a group of insurance companies located in different cities:

CUSTOMER(Id_cust, Name, Age, City_cust)
INSURANCE_COMPANY(Id_company, Id_Director, nEmployee, City)
POLICY(Id_policy, Id_cust, Id_company, expiry_date)

Primary keys are underlined in the relations. Moreover, Id_cust in POLICY is foreign key of CUSTOMER; Id_company in POLICY is foreign key of INSURANCE_COMPANY and Id_Director in INSURANCE_COMPANY foreign key of CUSTOMER.
A customer can have more than one policy in the same company or in different companies.
Expiry_date in POLICY is a year.

Assume that:

$n_{CUSTOMER}$ =2000	V(Id_cust, POLICY) = 2000
$n_{INSURANCE\_COMPANY}$ = 20	V(Id_company, POLICY) = 20
n_{POLICY} = 100.000	V(expiry_date, POLICY) = 20
	V(City, INSURANCE_COMPANY) = 5

Given the query:
"Name of customers holding policies with companies located in Pisa and with expiry date 2010"

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.

Let C, IC and P denote CUSTOMER, INSURANCE_COMPANY and POLICY, respectively.
Let |X| be the natural join

Point 1

$\Pi_{C.Name} (\sigma_{IC.City=Pisa \text{ and } P.expiry\_date=2010} ((C\ |X|\ P)\ |X|\ IC))$

Point 2

$\sigma_{IC.City=Pisa \text{ and } P.expiry\_date=2010} (\ldots..)$ can be rewritten as: $\sigma_{IC.City=Pisa} (\sigma_{P.expiry\_date=2010} (\ldots..)\ )$

$\Pi_{C.Name} (\sigma_{IC.City=Pisa} (\sigma_{P.expiry\_date=2010} ((C\ |X|\ P)\ |X|\ IC))\ )$

Push selection down

$\Pi_{C.Name} ((C\ |X|\ (\sigma_{P.expiry\_date=2010} (P)))\ |X|\ (\sigma_{IC.City=Pisa} (IC))\ )$

Push projection down

$\Pi_{C.Name} ((\Pi_{C.Name, C.Id\_cust} C)\ |X|\ (\Pi_{P.Id\_cust, P.Id\_company} (\sigma_{P.expiry\_date=2010} P))$
$|X|\ (\Pi_{IC.Id\_company} (\sigma_{IC.City=Pisa} (IC))\ )$

We evaluate the size and the number of different values for the new relations.

Let C' = $\Pi_{C.Name, C.Id\_cust} (C)$

$n_{C'} = n_{CUSTOMER}$ = 2000 Id_cust is a key

Let P' = $\sigma_{P.expiry\_date=2010} (P)$

$n_{P'} = n_{POLICY}$ / V(expiry_date, POLICY) = (100.000/20) = 5.000
V(Id_cust, P') = min($n_{P'}$, V(Id_cust, P)) = min(5.000, 2.000) = 2.000
V(Id_company, P') = min($n_{P'}$, V(Id_company, P)) = min(5.000, 20) = 20

Let P" = $\Pi_{P.Id\_cust,\ P.Id\_company}$ (P')
$n_{P''}$ = min($n_{P'}$, V(Id_cust, P') * V(Id_company, P')) = min (5.000, 2.000 * 20) = 5.000
V(Id_cust, P") = 2.000
V(Id_company, P") = 20

Let IC' = $\sigma_{IC.\ City=Pisa}$ (IC)
$n_{IC'}$ = ($n_{INSURANCE\_COMPANY}$ / V(City, INSURANCE_COMPANY) = (20/5) = 4
V(Id_company, IC') = $n_{IC'}$ = 4

Let IC" = $\Pi_{IC.Id\_company}$ (IC')
$n_{IC''} = n_{IC'}$ = 4 (Id_company is a key)

Point 3
Natural join is commutative.
$\Pi_{C.Name}$ (C' |*X*| P" |*X*| IC")

We estimate the size of different combinations of join.

Let T1 = (C' |X| P") Attribute of the join: Id_cust
Number of records in the result:
Id_cust in P" is foreign key of C' (note that C' and C have the same values of Id_cust)
$n_{T1} = n_{P''}$ = 5000

Let T2 = (C' |X| IC") Cartesian product
Number of records in the result:
n_{T2} = ($n_{C'}$ * $n_{IC''}$) = 2000 * 4 = 8000

Let T3 = (P" |X| IC") Attribute of the join: Id_company
Number of records in the result:
Id_company in P" is not foreign key of IC"
Id_company in P" is a key of IC"
$n_{T3} < n_{P''} < 5.000$

More precisely (rule applied by the optimizer):
min($n_{P''}$ * ($n_{IC''}$ / V(Id_company, IC"), $n_{IC''}$ * ($n_{P''}$ / V(Id_company, P")) =
min(5000 * (4/4), 4*(5.000/ 20)) = min(5.000, 1.000) = 1.000

The best ordering of join is : (C' |*X*| (P" |*X*| IC"))

An efficient strategy for solving the query is:
$\Pi_{C.Name}$ (C' |*X*| (P" |*X*| IC"))