**Qu. 1**.

Consider the attribute set R = ABCDEGH and the FD set F = {AB → C, AC → B, AD → E, B → D, BC →A, E → G}.

a.) For attribute set ABCEG, do the following:

(i) Compute the set of dependencies that hold over the set and write down a minimal cover.

{AB->C, AC->B, BC->A, E->G} holds.

It is already a minimal cover.

(ii) Name the strongest normal form that is not violated by the relation containing these attributes.

Key: ABE, ACE, BCE

AB->C is violated if considering it as BCNF.

E->G is violated since G is not prime if considering it as 3NF.

It is a 1NF.

(iii) Decompose it into a collection of BCNF relations if it is not in BCNF.

{ABE, ABC, EG}

b.) Which of the following decompositions of R = ABCDEG, with the same set of dependencies F, is (a) dependency-preserving? (b) lossless-join? Explain your answer. (you may use examples to explain)

(i) {AB, BC, ABDE, EG }

After decomposition:

{AB} : no FD

{BC}: no FD

{ABDE}: B->D, AD->E

{EG}: E->G

It is not dependency-preserving. AB-> C can not be recovered

It is not lossless. {ABDE} ∩ {BC} = {B} and B->BC does not hold.

(ii) {ABC, ACDE, ADG }

After decomposition:

{ABC} : AC->B, AB->C, BC->A

{ACDE}: AD->E, AC->D

{ADG}: AD->G

It is not dependency-preserving. E->G can not be recovered

It is lossless. {ACDE} ∩ {ADG} = {AD} and AD->ADG hold since AD->G holds. {ACDE} ∩ {ABC} = {AC} and AC ->ACB holds since AC->B holds.

**Qu. 2**.

Suppose you are given a relation R(A,B,C,D). For each of the following sets of FDs, assuming they are the only dependencies that hold for R, do the following:

(a) Identify the candidate key(s) for R.

(b) State whether or not the proposed decomposition of R into smaller relations is a good decomposition and briefly explain why or why not.

i.) B → C, D → A; decompose into BC and AD.

Key: BD.

FDs after decomposition:

BC: B->C

AD: D->A

The decomposition is not lossless: {BC} ∩ {AD} = NULL

ii.) AB → C, C → A, C → D; decompose into ACD and BC.

Key: AB, BC,

FDs after decomposition:

ACD: C->A,C->D

BC: no FD

The decomposition is lossless:

{ACD} ∩ {BC} = {C} and C -> ACD holds.

It is also not dependence preserved: AB->C can not be recovered.

iii.) A → BC, C → AD; decompose into ABC and AD.

Key: A, C

FDs after decomposition:

ABC: A->BC, C->ABC

AD: A->D

The decomposition is lossless:

{ABC} ∩ {AD} = {A} while A -> AD holds.

It is dependence preserved.

iv.) A → B, B → C, C → D; decompose into AB and ACD.

Key: A,

FDs after decomposition:

AB: A->B

ACD: C->D, A->CD

The decomposition is lossless:

{AB} ∩ {ACD} = {A} while A -> AB holds.

It is not dependence preserved: B->C can not be recovered.

v.) A → B, B → C, C → D; decompose into AB, AD and CD.

Key: A,

FDs after decomposition:

AB: A->B

AD: A->D

CD: C->D

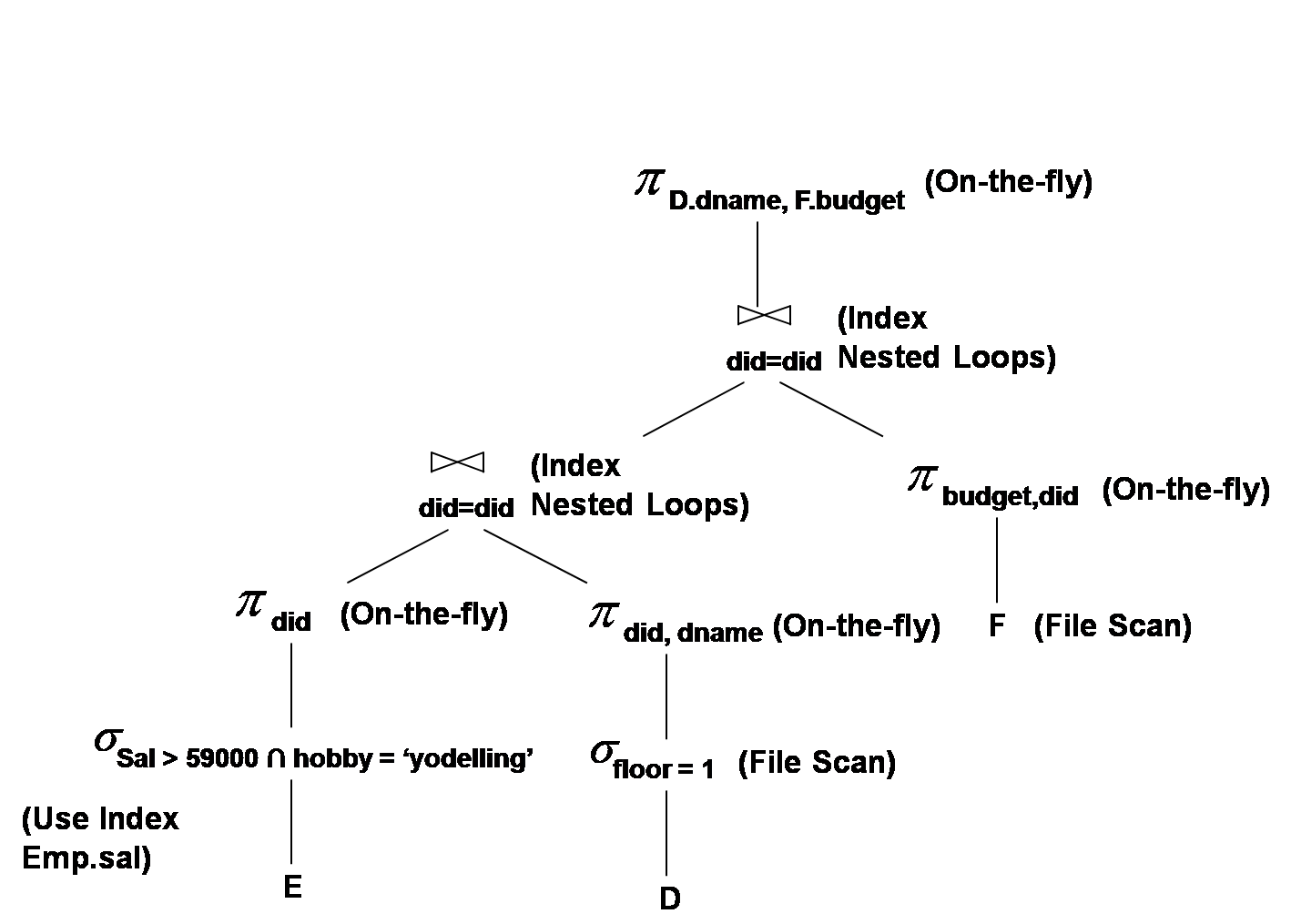
The decomposition is not lossless {AB} ∩ {AD} = {A}, {AD} ∩ {CD} = {D}, while D->CD and D->AD do not hold.

It is not dependence preserved: B->C can not be recovered.

**Qu. 3.**

Suppose that the following additional information is available: Unclustered B+ tree indexes exist on Emp.did, Emp.sal, Dept.floor, Dept.did, and Finance.did. The system’s statistics indicate that employee salaries range from 10,000 to 60,000, employees enjoy 200 different hobbies, and the company owns two floors in the building. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the database. The DBMS used by the company has just one join method available, index nested loops.

(a) Draw a relational expression plan tree (nodes are annotated with physical operators) that is equivalent to the expression.



(b) For each of the query’s base relations (Emp, Dept, and Finance) estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.

E: 50000 \* ((60000 - 59000) / (60000-10000)) = 1000 (Salary > 59000)

1000 \* (1 / 200) = 5 (hobby = ‘yodelling’)

D: 5000 \* (1 / 2) = 2500 (floor = 1)

F: 5000 (5000 departments)

(c) Given your answer to the preceding question, estimated cost of evaluating the plan if the order of order of join processing is as given i.e. ((E join D) join F)

Assume that it requires (3 + 1) = 4 cost to access a tuple using unclustered B+ tree index

sal>59000 and hobby = ‘yodelling’: Locate the first sal > 59000: 4

Access the remain: 4 + (1+1) \* 999 = 2002

E join D : 5 + 5 \* 4 = 25

E join D produce 5 \* (1/2) = 3 tuples

(E join D) join F: 3 + 3 \* 4 = 15

Total: 2002 + 25 + 15 = 2042