ÔN TẬP GIỮA KỲ

Phần 1: Thiết kế (5đ)

1. Phân mảnh ngang
2. Phân mảnh dọc:

Câu 1:

Employee(Name, Area, Skill, Salary)

Department(DeptId, Name, Area)

There are two areas (A1,A2) and three application centers (C1,C2,C3).

The applications requiring access to the tables:

Application 1: it will access the database . The likelihood of accesses:

- from C1 : A1:80%, A2:20%;
- from C2 : A1:50%, A2:50%;
- from C3 : A1:20%, A2: 80%

Application 2: it will access departments with the following probabilities:

- from C1: A1:100%, from C3:A2:100%

Application 3: it will access departments with the following probabilities:

- from C1: DeptId<100:100%,
- from C2: Deptid between 100 and 200: 100%,
- from C3: DeptId>200:100%

Derive the primary horizontal fragmentation of Department using the foregoing information.

Câu 2: Cho lược đồ CSDL gồm 2 quan hệ như sau:

HOIVIEN (MaHV, TenHV, Ngaysinh, Diachi)

THAMGIA (MaHV, UYBAN, THỜI GIAN, LOAI_HV)

Ghi chú: Có 4 loại hội viên: “thường”, “bạc”, “vàng”, “kim cương”. Thời gian là thời gian mà hội viên tham gia vào ủy ban, được tính bằng tháng. CSDL được phân bổ tại 5 vị trí. Mỗi vị trí sẽ quản lý thông tin của từng loại hội viên.

Có 3 ứng dụng quan trọng:

q1: với một mã hội viên, tìm thời gian tham gia vào các ủy ban của hội viên đó

q2: danh sách các ủy ban mà hội viên loại "vàng" có thời gian tham gia nhỏ hơn 24 tháng

q3: danh sách các ủy ban mà hội viên "vàng", có thời gian tham gia từ 24 tháng trở lên

q1 có thể xuất phát từ mọi vị trí, q2 luôn xuất phát từ vị trí của hội viên "vàng" , q3 xuất phát từ một vị trí khác của hội viên vàng

Nên phân mảnh ngang quan hệ Tham gia như thế nào để thực hiện tốt các câu truy vấn? Với mỗi phân mảnh hãy cung cấp các vị từ đơn giản ban đầu, cách xây dựng và loại bỏ các vị từ hội sơ cấp. Cho biết các vị từ hội sơ cấp của phân mảnh ngang cuối cùng của quan hệ tham gia

Phần 2: Truy vấn trên CSDLPT (5điểm)

Câu 3: Cho quan hệ *EMP (EmpID, Name, LOC, SAL, DoB, Dept);*

EMP được phân mảnh dựa trên các vị từ sau:

p1: LOC = 'Minneapolis';

p2: LOC = 'LA';

p3: LOC = 'NY';

Cho câu truy vấn sau:

Select Name

From EMP

Where LOC = 'LA' and Sal > 30,000.

a) Vẽ cây toán tử của câu truy vấn
b) Biến đổi câu truy vấn trên thành câu truy vấn trên các mảnh

Câu 4: Cho các CSDL gồm các quan hệ sau:

SV(Masv, Hoten, Tuoi, Gioitinh, Malop)

Lop(Malop, Tenlop, Maltr, TenKhoa)

Monhoc(Mamh, Tenmh)

Hoc (Masv, mamh, Diem)

Giả sử chúng ta có hai khoa tên là ‘CNTT’ và ‘DIEN’. Quan hệ lop được phân mảnh ngang dựa vào tenkhoa thành hai mảnh *lop1* và *lop2*
Lop1 = σtenkhoa=’CNTT’(lop)
Lop2 = σtenkhoa=’DIEN’(lop)

Quan hệ **Sinh viên** được phân mảnh ngang dẫn xuất theo lớp

SV1 = SV ⋉ Lop1

SV2 = SV ⋉ Lop2

Quan hệ **Học** được phân mảnh ngang dẫn xuất theo Sinh viên

Hoc1 = Hoc ⋉ SV1

Hoc2 = Hoc ⋉ SV2

Quan hệ **môn học** được giữ nguyên không có phân mảnh

Viết các câu truy vấn sau đây, vẽ cây toán tử và tối ưu cây toán tử trên các mảnh

1. Liệt kê họ tên sinh viên và điểm của môn học “Cơ sở dữ liệu” của lớp có mã “IT” với điều kiện đạt điểm lớn hơn 5
2. Cho biết họ tên các sinh viên không phải tên “Nam” học môn Cơ sở dữ liệu đạt điểm 10.
3. Cho biết họ tên sinh viên có mã lớp trưởng là “123” và các sinh viên này có tuổi không lớn hơn 20.
4. Cho biết kết quả học tập môn “Hệ cơ sở dữ liệu” của sinh viên “Nguyễn Văn An”
5. Liệt kê tên các môn học được sinh viên khoa “Công nghệ thông tin” đang học.

REVIEW

EMP (EmpID, Name, LOC, SAL, DoB, Dept);

Suppose application “AP1” queries the table “EMP” , looking for those employees who work in Los Angeles (LA). The set “Pr = {p1: Loc = “LA”}” shows all the required simple predicates used by AP1. Therefore, the set “M = {m1: Loc = “LA”, m2: Loc<>“LA”}” is a minimal and complete set of minterm

predicates for AP1. M fragments EMP into the following two fragments:

Fragment F1:

Create table LA_EMPS as

Select * from EMP

Where Loc = "LA";

Fragment F2:

Create table NON_LA_EMPS as

Select * from EMP

Where Loc <> "LA";

If AP1 were to also (in addition to checking the value of Loc) exclude any employee whose salary was less than or equal to 30000, then the set of predicates would no longer be minimal or complete. This additional check would mean that the rows in F1 would be accessed by AP1 differently depending on the salary of each employee in F1.

Applying the minimality rule mentioned above, we would need to further fragment the EMP table, if this were the case. The new simple predicates for AP1 would require changing Pr and M to the following:

Pr = {p1: Loc = "LA",

p2: salary > 30000}

M = {m1: Loc = "LA" Sal > 30000,

m2: Loc = "LA" Sal <= 30000,

m3: Loc <>"LA" Sal > 30000,

m4: Loc <>"LA" Sal <= 30000}

Observation 1: Since there are two simple predicates in Pr, and since for each predicate we have to consider the predicate and its negative, M will now have four minterm predicates in it. In general, if there are N simple predicates in Pr, M will have 2N minterm predicates. As explained later in this section, not all minterm predicates are relevant—the irrelevant predicates must be removed from the set. Therefore, the actual number of horizontal fragments is usually fewer than 2N .

Observation 2: In forming minterm predicates, we only use conjunctive normal form (“^”) and do not use disjunctive normal form (“OR”). That is because disjunctive normal forms create coarser fragments than fragments generated by conjunctive normal form and hence are not needed.

Example 2.8 As another example, consider table “PROJ (PNO, Pname, Funds, Dno,Loc),” where the PNO column is the primary key as shown in Figure 2.28.

Also assume two applications (“AP1” and “AP2”) query the PROJ table based on the following set of simple predicates:

The AP1’s simple predicates:

p1: Loc = "MPLS"

p2: Loc = "NY"

p3: Loc = "LA"

The AP2’s simple predicates:

p4: Funds <= 300000

The two applications have a combined set of four simple predicates in Pr, defined as:

Pr = {Loc = "MPLS",

Loc ="NY",

Loc ="LA",

Funds <= 300000}

Given the four simple predicates in Pr, M will have “24 = 16” different minterm predicates. The following depicts all these predicates.
m1 = {Loc = "MPLS" ^ Loc = "NY" ^ Loc = "LA" ^ Funds <= 300000}

m2 = {Loc = "MPLS" ^ Loc = "NY" ^ Loc = "LA" ^ Funds > 300000}

m3 = {Loc = "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m4 = {Loc = "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds > 300000}

m5 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds <= 300000}

m6 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds > 300000}

m7 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m8 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds > 300000}

m9 = {Loc <> "MPLS" ^ Loc = "NY" ^ Loc = "LA" ^ Funds <= 300000}

m10= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc = "LA" ^ Funds > 300000}

m11= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m12= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds > 300000}

m13= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds <= 300000}

m14= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds > 300000}

m15= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m16= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds > 300000}

Observation 3: Simple predicates p1, p2, and p3 are mutually exclusive. This means that only p1 or p2 or p3 can be true and not any combination of them. For example, if Loc is set to “MPLS,” then it cannot be equal to “NY” or “LA.” As a result, m1, m2, m3, m4, m5, m6, m9, and m10 are invalid—we will remove them. This will leave the following eight minterm candidates:

m7 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m8 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds > 300000}

m11= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m12= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds > 300000}

m13= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds <= 300000}

m14= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds > 300000}

m15= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m16= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds > 300000}

Observation 4: m15 and m16 are invalid predicates, because Loc must have one of the three values mentioned (no blank values, no null values, and no other location values are allowed). After removing m15 and m16, we will have the following six minterm predicates:

m7 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m8 = {Loc = "MPLS" ^ Loc <> "NY" ^ Loc <> "LA" ^ Funds > 300000}

m11= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds <= 300000}

m12= {Loc <> "MPLS" ^ Loc = "NY" ^ Loc <> "LA" ^ Funds > 300000}

m13= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds <= 300000}

m14= {Loc <> "MPLS" ^ Loc <> "NY" ^ Loc = "LA" ^ Funds > 300000}

Observation 5: Since Loc can only have one value at a time, we can simplify the predicates to the following:

m7 = {Loc = "MPLS" ^ Funds <= 300000}

m8 = {Loc = "MPLS" ^ Funds > 300000}

m11= {Loc = "NY" ^ Funds <= 300000}

m12= {Loc = "NY" ^ Funds > 300000}

m13= {Loc = "LA" ^ Funds <= 300000}

m14= {Loc = "LA" ^ Funds > 300000}

Applying these minterm predicates to PROJ, we can generate fragments Although there are six minterm predicates, only five of them produce useful results from PROJ. For the current state of the table, only the five fragments shown actually contain rows, the sixth fragment does not. This fragmentation is minimal since rows within each fragment are accessed the same by both applications. This fragmentation is also complete since any two rows within each fragment have the same access probability for AP1 and AP2