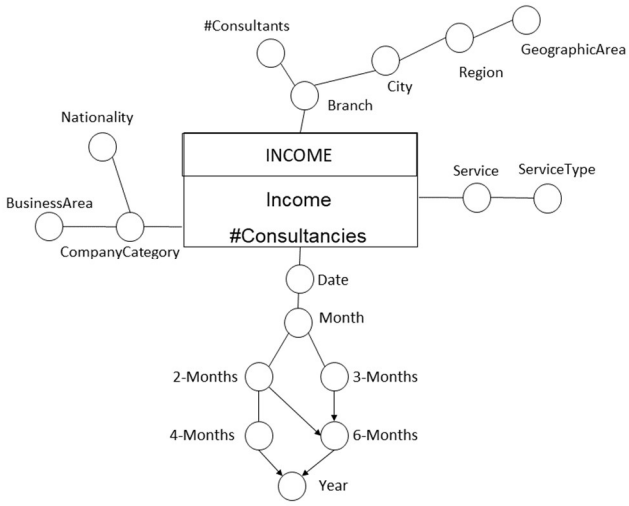
Exercise on materialized views

A data mart is characterized by the following conceptual schema and corresponding logical schema



INCOME (BranchID, ServiceID, CompanyCategoryID, TimeID, #Consultancies, Income)
SERVICE (ServiceID, Service, ServiceType)
TIME (TimeID, Date, Month, 2M, 3M, 4M, Semester, Year)
CONSULTANTS_BRANCH (BranchID, Branch, City, Region, GeographicArea, #Consultants)
COMPANY (CompanyCategoryID, CompanyCategory, BusinessArea, Nationality)

Point 1

Given the above logic schema, consider the following queries of interest:

a) For each pair (type of services, semester), show the total income and the total number of consultancies carried out by the consultants of the branches located in the Italian region Lombardia.
b) Considering only Italian and German companies, show for each pair (Region of consultants' branch, Service), the total income and number of consultancies, separately for each year.
c) Considering only the income of years 2017, 2018, and 2019, for each pair (Type of services, Nationality of the company), show the six-months income and the average six-months income per consulting.

Considering the above information, define a materialized view with CREATE MATERIALIZED VIEW, to reduce the response time of queries (a) to (c) above. In particular, specify the SQL query associated with Block A in the following statement:

CREATE MATERIALIZED VIEW ViewIncome

BUILD IMMEDIATE

REFRESH FAST ON COMMIT

AS

Block A

Point 2

Specify the minimal combination of attributes that constitutes an identifier for the materialized ViewIncome view.

Point 3

Assume that the ViewIncome materialized view is managed **without using** the CREATE MATERIALIZED VIEW statement. The derived table that implements the materialized view is defined by the following SQL statement

```
CREATE TABLE ViewIncome
    (
    Service             VARCHAR(20),
    ServiceType         VARCHAR(20),
    Nationality         VARCHAR(20),
    Semester            VARCHAR(20),
    Year                VARCHAR(20),
    Region              VARCHAR(20),
    TotIncome           INTEGER
                CHECK (TotIncome IS NOT NULL and TotIncome >0),
    TotNumConsulting    INTEGER
                CHECK (TotNumConsulting IS NOT NULL and TotNumConsulting >0),
    PRIMARY KEY (Nationality, Semester, Region, Service)
    )
```

Write the INSERT statement for the first data loading into the ViewIncome materialized view, using the INSERT (SELECT ...) statement.

Point 4

Assume that the updating of the materialized view is done using triggers. Write the trigger to propagate the changes to the ViewIncome materialized view when a new record is inserted in the INCOME fact table.

Point 5

Assume that the updating of the materialized view is done using triggers. Write the trigger to propagate the update of the ServiceType attribute in the SERVICE table to the ViewIncome materialized view. Note: Consider the Type 1 mode for time management.

Point 6

The VM1 materialized view is built with the following SQL statement

```
CREATE MATERIALIZED VIEW
BUILD IMMEDIATE
REFRESH FAST ON DEMAND
ENABLE QUERY REWRITE
AS      <Block A>
```

For the definition of Block A see Point 1.

Write the instructions that define the MATERIALIZED VIEW LOG in Oracle needed for the automatic FAST update of the ViewIncome materialized view. Indicate ***all and only*** the necessary logs and within each log definition indicate ***all and only*** the necessary attributes.

DRAFT SOLUTION

Point 1

A: GROUP BY: ServiceType, Semester
WHERE Region = ‘Lombardia’
Aggregates: SUM(Income), SUM(#Consultancies)
FROM: INCOME, SERVICE, TIME, CONSULTANTS_BRANCH

Query A
SELECT ServiceType, Semester, SUM(Income), SUM(#Consultancies)
FROM INCOME, SERVICE, TIME, CONSULTANTS_BRANCH
WHERE <join>
AND Region = ‘Lombardia’
GROUP BY ServiceType, Semester

B: GROUP BY: Region, Service, Year
Aggregates: SUM(Income), SUM(#Consultancies)
WHERE Nationality= ‘Italian’ OR Nationality=’German’
FROM: INCOME, SERVICE, TIME, COMPANY, CONSULTANTS_BRANCH

Query B
SELECT Region, Service, Year, SUM(Income), SUM(#Consultancies)
FROM INCOME, SERVICE, TIME, COMPANY, CONSULTANTS_BRANCH
WHERE <join>
AND (Nationality= ‘Italian’ OR Nationality=’German’)
GROUP BY Region, Service, Year

C: GROUP BY ServiceType, Nationality, Semester
Aggregates: SUM(Income), SUM(#Consultancies)
WHERE Year >= 2017 AND Year <= 2019
FROM: INCOME, SERVICE, TIME, COMPANY

Query C
SELECT ServiceType, Nationality, Semester, SUM(Income), SUM(Income)/ SUM(#Consultancies)
FROM INCOME, SERVICE, TIME, COMPANY
WHERE <join>
AND Year >= 2017 AND Year <= 2019
GROUP BY ServiceType, Nationality, Semester

Blocco A

```
SELECT Service, ServiceType, Nationality, Semester, Year, Region,
        SUM(Income) AS TotIncome,
        SUM(#Consultancies) AS TotNumConsultancies
FROM INCOME I, SERVICE S, TIME T, COMPANY C, CONSULTANTS_BRANCH CB
WHERE I.ServiceID=S. ServiceID AND I.TimeID=T. TimeID AND
          C.CompanyCategoryID=I.CompanyCategoryID AND CB.BranchID=I. BranchID
GROUP BY Service, ServiceType, Nationality, Semester, Year, Region
```

Punto 2

Minimal identifier: Service, Nationality, Semester, Region

Note: Semester → Year

Service → ServiceType

Punto 3

```
INSERT INTO ViewIncome (Service, ServiceType, Nationality, Semester, Year, Region,
                        TotIncome, TotNumConsultancies)
        (SELECT Service, ServiceType, Nationality, Semester, Year, Region,
                        SUM(Income), SUM(#Consultancies)
        FROM INCOME I, SERVICE S, TIME T, COMPANY C, CONSULTANTS_BRANCH CB
        WHERE I.ServiceID=S. ServiceID AND I.TimeID=T. TimeID AND
                    C.CompanyCategoryID=I.CompanyCategoryID AND CB.BranchID=I. BranchID
        GROUP BY Service, ServiceType, Nationality, Semester, Year, Region);
```

Punto 4

```
CREATE TRIGGER RefreshViewIncome
AFTER INSERT ON INCOME
FOR EACH ROW
DECLARE
N number;
VarService, VarSType VARCHAR(20); VarNationality VARCHAR(20);
VarSemester, VarYear VARCHAR(20); VarRegion VARCHAR(20);

BEGIN

--- Read from the dimension tables values of attributes Service, Nationality, Semester, Region corresponding
to the new inserted values of :NEW.ServiceID, :NEW.CompanyCategoryID, :NEW.TimeID, :NEW.BranchID

SELECT Service, ServiceType INTO VarService, VarSType
FROM SERVICE
WHERE ServiceID = :NEW.ServiceID;

SELECT Region INTO VarRegion
FROM CONSULTANTS_BRANCH
WHERE BranchID = :NEW.BranchID;

SELECT Semestre, Anno INTO VarSemestre, VarAnno
FROM TEMPO
WHERE TimeID = :NEW.TimeID;

SELECT Nationality INTO VarNationality
FROM COMPANY
WHERE CompanyCategoryID = :NEW.CompanyCategoryID;

--- Check if there is a record in ViewIncome associated with the values Service, Nationality, Semester, Region
just read (minimal record identifier in ViewIncome).

SELECT Count(*) INTO N
FROM ViewIncome
WHERE Service = VarService AND Nationality = VarNationality AND Semestre= VarSemestre AND Region =
VarRegion;

IF (N > 0) THEN
--- If the record exists the value of the aggregates must be updated
        UPDATE ViewIncome
        SET TotIncome = TotIncome + :NEW.Income
              TotNumConsultancies = TotNumConsultancies + :NEW.#Consultancies
        WHERE Service = VarService AND Nationality = VarNationality
        AND Semester = VarSemester AND Region = VarRegion;
ELSE
--- If the record does not exist, a new record must be inserted
        INSERT INTO ViewIncome (Service, ServiceType, Nationality, Semester, Year,
                Region, TotIncome, TotNumConsultancies)
        VALUES (VarService, VarSType, VarNationality, VarSemester, VarYear, VarRegion,
                :NEW.Income, :NEW.#Consultancies);
END IF;
END;
```

Punto 5

```
CREATE TRIGGER UpdateServiceType
AFTER UPDATE OF ServiceType ON SERVICE
FOR EACH ROW
DECLARE
N number;
BEGIN

---Check if the ServiceType value is present in the materialized view
SELECT Count(*) INTO N
FROM ViewIncome
WHERE ServiceType = :OLD.ServiceType;

IF (N > 0) THEN
--- The TypeService value is present.
--- All records associated with the previous value of ServiceType must be updated
        UPDATE ViewIncome
        SET ServiceType = :NEW.ServiceType
        WHERE ServiceType = :OLD.ServiceType;

END IF;
END;
```

Punto 6

```
CREATE MATERIALIZED VIEW LOG ON INCOME
WITH SEQUENCE, ROWID
(BranchID, ServiceID, TimeID, CompanyCategoryID, #Consultancies, Income)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON SERVICE
WITH SEQUENCE, ROWID
(ServiceID, Service, ServiceType)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON TIME
WITH SEQUENCE, ROWID
(TimeID, Semester, Year)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON CONSULTANTS_BRANCH
WITH SEQUENCE, ROWID
(BranchID, Region)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON COMPANY
WITH SEQUENCE, ROWID
(CompanyCategoryID, Nationality)
INCLUDING NEW VALUES;
```