Σχεδίαση Βάσεων Δεδομένων και Κατανεμημένες ΒΔ

Ερώτημα 1 - Δημιουργία και γέμισμα σχήματος

Κώδικας:

```
-- FUNCTIONS --

-- Map age to CUSTOMERS_AGE_CHK --
CREATE OR REPLACE FUNCTION get_age_group(birth_date DATE)
  RETURN VARCHAR2 IS
  age CONSTANT NUMBER(3) := FLOOR((CURRENT_DATE - birth_date) / 365);
  BEGIN
    IF age <= 30 THEN RETURN 'under 30';
    ELSIF age <= 40 THEN RETURN '30-40';
    ELSIF age <= 50 THEN RETURN '40-50';
    ELSIF age <= 60 THEN RETURN '50-60';
    ELSIF age <= 70 THEN RETURN '60-70';
    ELSE RETURN 'above 70';
    END IF;
  END;
/

-- Map income level to CUSTOMERS_INCOME_CHK --
CREATE OR REPLACE FUNCTION get_income_level(income CHAR)
  RETURN VARCHAR2 IS
  identifier CONSTANT CHAR(1) := SUBSTR(income, 1, 1);
  BEGIN
    IF identifier IN ('A', 'B', 'C', 'D', 'E') THEN RETURN 'low';
    ELSIF identifier IN ('F', 'G', 'H', 'I') THEN RETURN 'medium';
    ELSE RETURN 'high';
    END IF;
  END;
/

-- Map marital status to CUSTOMERS_MARRIAGE_CHK --
CREATE OR REPLACE FUNCTION fix_status(status CHAR)
  RETURN VARCHAR2 IS
  married CONSTANT CHAR(27) := '([Mm]arried|Mabsent|Mar-AF)';
  BEGIN
    IF REGEXP_LIKE(status, married) THEN RETURN 'married';
    ELSIF status IS NOT NULL THEN RETURN 'single';
    ELSE RETURN 'unknown';
    END IF;
  END;
/

-- TABLES --

-- Customers --
CREATE TABLE CUSTOMERS AS
  SELECT ID AS CUSTOMER_ID, CAST(GENDER AS VARCHAR2(6)) AS GENDER,
    CAST(get_age_group(BIRTH_DATE) AS VARCHAR2(8)) AS AGEGROUP,
    CAST(fix_status(MARITAL_STATUS) AS VARCHAR2(7)) AS MARITAL_STATUS,
    CAST(get_income_level(INCOME_LEVEL) AS VARCHAR2(6)) AS INCOME_LEVEL
  FROM XSALES.CUSTOMERS;

ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_PK
PRIMARY KEY (CUSTOMER_ID);

ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_GENDER_CHK
CHECK (GENDER IN ('Male', 'Female'));
```

```
57
58  ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_AGE_CHK
59  CHECK (AGEGROUP IN ('under 30', '30-40', '40-50',
60                      '50-60', '60-70', 'above 70'));
61
62  ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_MARRIAGE_CHK
63  CHECK (MARITAL_STATUS IN ('married', 'single', 'unknown'));
64
65  ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_INCOME_CHK
66  CHECK (INCOME_LEVEL IN ('low', 'medium', 'high'));
67
68  -- Products --
69  CREATE TABLE PRODUCTS AS
70    SELECT P.IDENTIFIER AS PRODUCT_ID,
71      CAST(P.NAME AS VARCHAR2(50)) AS PRODUCTNAME,
72      CAST(C.NAME AS VARCHAR2(50)) AS CATEGORYNAME,
73      CAST(TO_NUMBER(P.LIST_PRICE) AS NUMBER(7, 2)) AS LIST_PRICE
74    FROM XSALES.PRODUCTS P JOIN XSALES.CATEGORIES C
75        ON P.SUBCATEGORY_REFERENCE = C.ID;
76
77  ALTER TABLE PRODUCTS
78    ADD CONSTRAINT PRODUCTS_PK
79  PRIMARY KEY (PRODUCT_ID);
80
81  -- Orders --
82  CREATE TABLE ORDERS AS
83    SELECT O.ID AS ORDER_ID, I.PRODUCT_ID AS PRODUCT_ID,
84      O.CUSTOMER_ID AS CUSTOMER_ID,
85      CAST(FLOOR(AVG((O.ORDER_FINISHED - I.ORDER_DATE)))
86           AS NUMBER(4)) AS DAYS_TO_PROCESS,
87      CAST(TRUNC(AVG(I.AMOUNT), 2) AS NUMBER(7, 2)) AS PRICE,
88      CAST(TRUNC(AVG(I.COST), 2) AS NUMBER(7, 2)) AS COST,
89      CAST(O.CHANNEL AS VARCHAR2(20)) AS CHANNEL
90    FROM XSALES.ORDERS O JOIN XSALES.ORDER_ITEMS I ON O.ID = I.ORDER_ID
91    GROUP BY (O.ID, I.PRODUCT_ID, O.CUSTOMER_ID, O.CHANNEL);
92
93  ALTER TABLE ORDERS ADD CONSTRAINT ORDERS_PK
94  PRIMARY KEY (ORDER_ID, PRODUCT_ID);
95
96  ALTER TABLE ORDERS ADD CONSTRAINT ORDERS_PRODUCT_FK
97  FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCTS (PRODUCT_ID);
98
99  ALTER TABLE ORDERS ADD CONSTRAINT ORDERS_CUSTOMER_FK
100 FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMERS (CUSTOMER_ID);
```

Παραδοχές:

Στον πίνακα `ORDERS` πήραμε `AVG` τιμές ώστε να υπάρχει μοναδικός συνδυασμός `ORDER_ID, PRODUCT_ID`.

Ερώτημα 2 – Εντοπισμός ζημιογόνων παραγγελιών

i) Για κάθε παραγγελία βρείτε τη μέγιστη καθυστέρηση εκτέλεσης σε ημέρες λαμβάνοντας υπόψη ότι η καθυστέρηση μετρά μετά τις 10 πρώτες ημέρες (days_to_process>10 για το σύνολο των προϊόντων της παραγγελίας).

Κώδικας:

```
SELECT ORDER_ID, MAX(DAYS_TO_PROCESS) AS BIGGEST_DELAY
FROM ORDERS WHERE DAYS_TO_PROCESS > 10 GROUP BY ORDER_ID;
```

ii) Υπολογίστε το τελικό κέρδος της αφού πρώτα αφαιρέσετε (ανά προϊόν) από την τιμή πώλησης (price) την τιμή κόστους (cost) και για κάθε ημέρα καθυστέρησης της εκτέλεσης επιπλέον το 0.0001 της τιμής καταλόγου (list_price).

Κώδικας:

```
SELECT O.ORDER_ID, P.PRODUCT_ID,
  TRUNC(O.PRICE - O.COST - 0.0001 * P.LIST_PRICE, 2) AS PROFIT
FROM PRODUCTS P JOIN ORDERS O ON P.PRODUCT_ID = O.PRODUCT_ID;
```

iii) Δημιουργήστε έναν cursor που να διατρέχει τις παραγγελίες και να υπολογίζει ανά παραγγελία το τελικό κέρδος (για όλα τα προϊόντα της), στη συνέχεια να ελέγχει αν το κέρδος είναι αρνητικό ή θετικό. Στην περίπτωση ι) που είναι αρνητικό να καταχωρει σε ένα πίνακα deficit τα (orderid,customerid,channel,amount) όπου η amount θα έχει τη ζημιά με θετικό πρόσημο, ιι) που είναι θετικό να καταχωρεί σε ένα πίνακα profit τα αντίστοιχα orderid,customerid,channel,amount.

Κώδικας:

```
CREATE TABLE PROFIT (
  ORDER_ID NUMBER(*),
  CUSTOMER_ID NUMBER(*),
  CHANNEL VARCHAR2(20),
  AMOUNT NUMBER(*)
);
CREATE TABLE DEFICIT AS SELECT * FROM PROFIT;

DECLARE
  profit_tuple PROFIT%ROWTYPE;
  CURSOR profit_cursor IS
    SELECT O.ORDER_ID, O.CUSTOMER_ID, O.CHANNEL,
      TRUNC(SUM(O.PRICE - O.COST - 0.0001 * P.LIST_PRICE), 2) AS AMOUNT
    FROM PRODUCTS P JOIN ORDERS O ON P.PRODUCT_ID = O.PRODUCT_ID
    GROUP BY P.PRODUCT_ID, O.ORDER_ID, O.CUSTOMER_ID, O.CHANNEL;
BEGIN
  FOR profit_tuple IN profit_cursor LOOP
    IF profit_tuple.AMOUNT > 0 THEN
      INSERT INTO PROFIT VALUES (
        profit_tuple.ORDER_ID,
        profit_tuple.CUSTOMER_ID,
        profit_tuple.CHANNEL,
        profit_tuple.AMOUNT
      );
    ELSE
      INSERT INTO DEFICIT VALUES (
        profit_tuple.ORDER_ID,
        profit_tuple.CUSTOMER_ID,
        profit_tuple.CHANNEL,
        ABS(profit_tuple.AMOUNT)
      );
    END IF;
  END LOOP;
  COMMIT;
END;
/
```

iv) Ποια τα συνολικά έσοδα και ζημιές σε παραγγελίες που έγιναν από άνδρες και γυναίκες αντίστοιχα;

Κώδικας:

```
SELECT
  C.GENDER AS CUSTOMER_GENDER,
  SUM(P.AMOUNT) AS TOTAL_PROFIT,
  SUM(D.AMOUNT) AS TOTAL_DEFICIT
FROM PROFIT P
  JOIN DEFICIT D ON P.CUSTOMER_ID = D.CUSTOMER_ID
  JOIN CUSTOMERS C ON P.CUSTOMER_ID = C.CUSTOMER_ID
GROUP BY C.GENDER;
```

Αποτέλεσμα:

CUSTOMER_GENDER	TOTAL_PROFIT	TOTAL_DEFICIT
Male	1288541.18	58107.68
Female	856590.6	36187.7

v) Ποια τα συνολικά έσοδα και ζημιές ανά κανάλι παραγγελιών;

Κώδικας:

```
SELECT
  P.CHANNEL AS ORDER_CHANNEL,
  SUM(P.AMOUNT) AS TOTAL_PROFIT,
  SUM(D.AMOUNT) AS TOTAL_DEFICIT
FROM PROFIT P
  JOIN DEFICIT D ON P.CUSTOMER_ID = D.CUSTOMER_ID
GROUP BY P.CHANNEL;
```

Αποτέλεσμα:

ORDER_CHANNEL	TOTAL_PROFIT	TOTAL_DEFICIT
Direct Sales	1026668.39	41957.73
Partners	612302.71	29623.31
Internet	506160.68	22714.34

Ερώτημα 3 - Βελτιστοποίηση ερωτήματος ισότητας

Χρησιμοποιώντας την εντολή EXPLAIN ελέγξτε πώς λειτουργεί ο optimizer για το ακόλουθο ερώτημα:

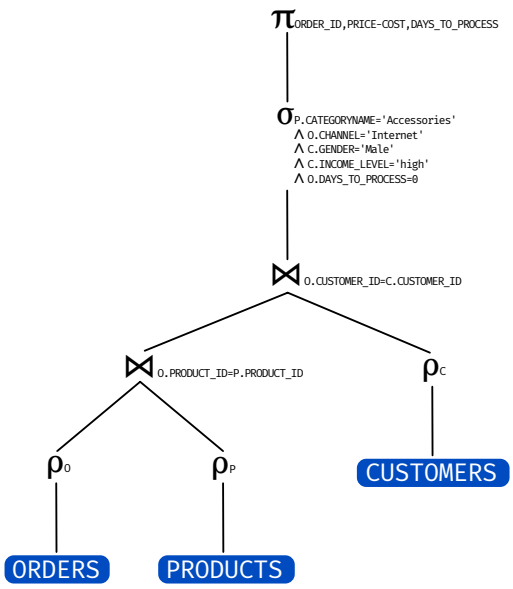
```
SELECT ORDER_ID, PRICE - COST, DAYS_TO_PROCESS
FROM PRODUCTS P
  JOIN ORDERS O ON O.PRODUCT_ID = P.PRODUCT_ID
  JOIN CUSTOMERS C ON O.CUSTOMER_ID = C.CUSTOMER_ID
WHERE P.CATEGORYNAME = 'Accessories'
    AND O.CHANNEL = 'Internet'
    AND C.GENDER = 'Male'
    AND C.INCOME_LEVEL = 'high'
    AND O.DAYS_TO_PROCESS = 0;
```

1) Σύμφωνα με την EXPLAIN ποιο είναι το εκτιμώμενο συνολικό κόστος για την εκτέλεση του καλύτερου πλάνου για το ερώτημα αυτό; Ποια τα CPU_COST και IO_COST; Ποια είναι η πιο χρονοβόρα ενέργεια;

Το εκτιμώμενο συνολικό κόστος για το ερώτημα είναι **868**.
Τα **CPU_COST** και **IO_COST** είναι **205864078** και **863** αντίστοιχα.
Η ποιο χρονοβόρα ενέργεια είναι η εκτέλεση **TABLE ACCESS FULL** στον πίνακα **ORDERS** με κόστος **786**.

2) Ποιο είναι το εκτιμώμενο πλήθος αποτελεσμάτων για το ερώτημα; Πόσες πλειάδες επιστρέφει πραγματικά το ερώτημα; Χρησιμοποιώντας συμβολισμούς σχεσιακής άλγεβρας, σχεδιάστε το πλάνο εκτέλεσης που επέλεξε ο optimizer.

Το εκτιμώμενο πλήθος αποτελεσμάτων για το ερώτημα είναι **883**, ενώ τελικά επιστρέφει **105** πλειάδες.



Πλάνο εκτέλεσης

3) Ποιο το τελικό κόστος μετά τη βελτιστοποίηση της σχεδίασης;

```
CREATE INDEX PRODUCTS_CATEGORYNAME_IDX
  ON PRODUCTS (CATEGORYNAME);
CREATE UNIQUE INDEX CUSTOMERS_IDX
  ON CUSTOMERS (CUSTOMER_ID, GENDER, INCOME_LEVEL);
```

Αφού υλοποιήσαμε τα παραπάνω ευρετήρια, το τελικό κόστος μειώθηκε σε **786**.
Επίσης, τα **CPU_COST** και **IO_COST** μειώθηκαν σε **188418202** και **783** αντίστοιχα.

Ερώτημα 4 - Βελτιστοποίηση ερωτήματος ανισότητας

Τι θα αλλάξει αν κάνετε το ακόλουθο ερώτημα;

```
SELECT
  /*+
  NO_INDEX(C CUSTOMERS_IDX)
  NO_INDEX(P PRODUCTS_CATEGORYNAME_IDX)
  */ ORDER_ID, PRICE - COST, DAYS_TO_PROCESS
FROM PRODUCTS P
  JOIN ORDERS O ON O.PRODUCT_ID = P.PRODUCT_ID
  JOIN CUSTOMERS C ON O.CUSTOMER_ID = C.CUSTOMER_ID
WHERE P.CATEGORYNAME = 'Accessories'
    AND O.CHANNEL = 'Internet'
    AND C.GENDER = 'Male'
    AND C.INCOME_LEVEL = 'high'
    AND O.DAYS_TO_PROCESS > 100;
```

1) Σύμφωνα με την EXPLAIN ποιο είναι το εκτιμώμενο συνολικό κόστος για την εκτέλεση του καλύτερου πλάνου χωρίς ευρετήρια για το ερώτημα αυτό; Ποια τα CPU_COST και IO_COST; Ποια είναι η πιο χρονοβόρα ενέργεια;

Το εκτιμώμενο συνολικό κόστος για το ερώτημα είναι **868**.
Τα **CPU_COST** και **IO_COST** είναι **211452900** και **863** αντίστοιχα.
Η ποιο χρονοβόρα ενέργεια είναι η εκτέλεση **TABLE ACCESS FULL** στον πίνακα **ORDERS** με κόστος **786**.

2) Επιχειρήστε να βελτιστοποιήσετε το ερώτημα δημιουργώντας τα κατάλληλα ευρετήρια. Ποιο το τελικό κόστος μετά τη βελτιστοποίηση της σχεδίασης;

Χρησιμοποιώντας τα ίδια ευρετήρια με το προηγούμενο ερώτημα, το τελικό κόστος μειώθηκε σε **788**, ενώ τα **CPU_COST** και **IO_COST** μειώθηκαν σε **20093774** και **783** αντίστοιχα.
Δεν παρατηρήσαμε κάποια περαιτέρω βελτίωση μετά την δημιουργία επιπλέον ευρετηρίων.