درس طرحی پایگاه داده

تمرین تئوری سری سوم **۱۴۰۴/۰۲/۲۲**

متین باقری **۴۰۲۱۰۵۷۲۷**

---

۱) آ)

```
CREATE VIEW EmployeeDepartment AS
SELECT
    e.name AS EmployeeName,
    d.name AS DepartmentName,
    e.salary AS EmployeeSalary
FROM
    Employees e
JOIN
    Departments d ON e.department_id = d.department_id;
```

این view در چه سناریو هایی میتواند کاربردی و مفید باشد؟

- مشاهده ساده ۳ داده خواسته شده توسط کاربر بدون نیاز به انجام دستی join

- محدود کردن دسترسی افراد به ۳ attribute گفته شده و افزایش امنیت دیتابیس

ب) اگر از inner join استفاده شده باشد (مانند این view)، کارمندان آن دپارتمان دیگر در view نمایش داده نمی‌شوند.

اگر از left join استفاده می‌کردیم، کارمندان آن دپارتمان نمایش داده می‌شدند ولی ستون نام دپارتمنت آن ها مقدار null می‌گرفت.

ج)

```
CREATE VIEW Managers AS
SELECT
    e.name AS EmployeeName,
    d.name AS DepartmentName,
    e.salary AS EmployeeSalary
FROM
    Employees E
JOIN
    Departments D ON E.department_id = D.department_id
WHERE
    E.salary > 50000000;
```

د) الزامی نیست، ولی می‌تواند باعث کنترل بیشتر بر روی کاربر شده و اجازه اضافه کردن یا ویرایش tuple هایی که شروط view را نقض می‌کنند نمی‌دهد. با این کار از صحت داده ها مطمئن می‌شود.

برای مثال در view بالا، اگر کاربری بخواهد کارمندی را با حقوق کمتر از ۵۰ میلیون اضافه یا ویرایش کند، عملیات رد می‌شود.

۲) آ)

ممکن است تعداد صندلی های موجود یک درس در ترم های مختلف متفاوت باشد. و یا در جدول enrollment یک دانشجو یک درس را بیفتد و بخواهد دوباره آن درس را بردارد، در این صورت primary key یکتا نخواهد بود.

تکمیل پیش‌نیاز قبل از ثبت‌نام: یعنی اگر یک دانشجو می‌خواهد در درسی ثبت‌نام کند که پیش‌نیاز دارد، ابتدا باید آن پیش‌نیاز را گذرانده باشد. در صورت نقض این شرط، دانشجو بدون داشتن دانش پایه لازم وارد درس شده و فهم کافی از مطالب درس نخواهد داشت.

محدودیت تعداد ثبت‌نام: یعنی تعداد ثبت‌نامی های هر درس نباید بیشتر از ظرفیت آن باشد. در صورت عدم رعایت آن دانشجو و استاد از کمبود منابع و تجهیزات ناراضی خواهند بود.

ب)

```
CREATE ASSERTION MaxStudents
CHECK (
    NOT EXISTS (
        SELECT course_id
        FROM Course c
        WHERE (
            SELECT COUNT(*)
            FROM Enrollment e
            WHERE e.course_id = c.course_id
        ) > c.max_seats
    )
);
```

این assertion بررسی می‌کند که تعداد دانشجویان ثبت‌نامی هیچ درسی بیش از ظرفیت عملی آن نباشد. البته که توانایی جلوگیری از نقض آن را ندارد، چرا که assertion یک مفهوم تئوری است و پیاده سازی عملی ندارد. در تئوری:

اگر هنگام ثبت‌نام یک دانشجو درسی را انتخاب کند که ظرفیتش پر شده، عملیات ثبت‌نام شکست می‌خورد. درواقع قید نقض شده و هنگام اجرای تراکنش (مثلاً INSERT در جدول Enrollment) عملیات COMMIT رد شده و سیستم خطا می‌دهد و اجازه نهایی کردن تراکنش را نمی‌دهد.

ج)

```
CREATE OR REPLACE FUNCTION check_availability()
RETURNS TRIGGER AS $$
DECLARE
    max INT;
    current INT;
BEGIN
    SELECT max_seats INTO max
    FROM Course
    WHERE course_id = NEW.course_id;

    SELECT COUNT(*) INTO current
    FROM Enrollment
    WHERE course_id = NEW.course_id;

    IF current >= max THEN
        RAISE EXCEPTION 'Oops, this course is already full.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_capacity
BEFORE INSERT ON Enrollment
FOR EACH ROW
EXECUTE FUNCTION check_availability();
```

۳) آ)

```
CREATE VIEW order_summary AS
SELECT
    o.order_id,
    o.customer_id,
    o.order_date,
    SUM(oi.quantity * oi.price) AS total_amount
FROM
    orders o
NATURAL JOIN
    order_items oi
GROUP BY
    o.order_id,
    o.customer_id,
    o.order_date;
```

ب)

```
CREATE OR REPLACE FUNCTION update_order_summary()
RETURNS TRIGGER AS $$
DECLARE
    old_total    NUMERIC;
    new_total    NUMERIC;
    diff         NUMERIC;
    bonus_prod   INT     := 777;
    bonus_price  NUMERIC := 0;
    item_record  RECORD;
BEGIN
    SELECT SUM(quantity * price)
      INTO old_total
    FROM order_items
    WHERE order_id = OLD.order_id;

    new_total := NEW.total_amount;
    diff      := new_total - old_total;

    IF diff > 0 THEN
        INSERT INTO order_items(order_id, product_id, quantity, price)
        VALUES (OLD.order_id, bonus_prod, FLOOR(diff), bonus_price);

    ELSIF diff < 0 THEN
        FOR item_record IN
            SELECT *
              FROM order_items
             WHERE order_id = OLD.order_id
             ORDER BY quantity DESC
             LIMIT 2
        LOOP
            IF item_record.price != 0 THEN
                UPDATE order_items
                   SET quantity = FLOOR(0.8 * quantity)
                 WHERE order_id  = item_record.order_id
                   AND product_id = item_record.product_id;
                EXIT;
            END IF;
        END LOOP;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_update_order_summary
INSTEAD OF UPDATE ON order_summary
FOR EACH ROW
WHEN (OLD.total_amount IS DISTINCT FROM NEW.total_amount)
EXECUTE FUNCTION update_order_summary();
```

ج) چالش های مختلفی وجود دارد که برخی از آن‌ها به TCL مرتبط اند:

- همگام‌سازی داده‌ها با جدول‌های وابسته: وقتی داده ای با توجه به مقدار موجود در جدول دیگری محاسبه شود، در صورت تغییر داده آن جدول، باید این تغییر در داده محاسبه شده از آن اعمال شود. برای ایجاد این sync بودن کامل بین view و جداول وابسته، نیاز به استفاده از triggerهای دقیقی داریم.

- Concurrency: اگر چند کاربر به‌صورت هم‌زمان سطری را ویرایش کنند، ممکن است داده‌های view دچار مشکل شود. که البته به طور خودکار از مکانیزم‌هایی مثل transaction و locking استفاده می‌شود.

- محاسباتی که از چندین جدول استفاده می‌کنند و روی داده ها محاسبات طولانی و پیچیده انجام می‌دهند، منابع زیادی مصرف می‌کنند. برای کاهش این محاسبات می‌توان از materialized view استفاده کرد، البته که باید به بررسی آپدیت بودن آن توجه کرد.

- در کل برای مدیریت این اتفاقات نیاز به نوشتن trigger های دقیق و گاها پیچیده داریم که خود طراحی را دشوار می‌کند.

- در میان این پیچیدگی‌ها، باید constraint های مناسب تعریف کرد تا invalid شدن داده های جداول و ویو جلوگیری شود.

۴) آ)

```
INSERT INTO MovieProd
SELECT 'New DB', 1404, pName
FROM Producer WHERE pID = 123;
```

ب)

```
DELETE FROM MovieProd
WHERE title = 'Old DB' AND year = 1403;
```

ج)

```
INSERT INTO MovieProd
SELECT title, year, 'Rox'
FROM Movie WHERE producerID = 456;
```

د)

```
DELETE FROM MovieProd
WHERE pName = 'Bob';
```

ه) نیاز به انجام کاری نیست چون در view تغییری نیاز نیست.