## **Initial Relational Algebra Expression (Before Optimization)**

π\_{S.Name, C.Title, E.Grade}

(σ\_{S.StudentID = E.StudentID ∧ C.CourseID = E.CourseID ∧ S.Major = 'Computer Science' ∧ C.Credits ≥ 3}

(STUDENT × ENROLLMENT × COURSE))

## **Initial Query Tree Representation**

π\_{S.Name, C.Title, E.Grade}

|

σ\_{Conditions}

|

-----------------------------

| | |

STUDENT ENROLLMENT COURSE

‘

## **Optimized Relational Algebra Expression (After Applying Heuristics)**

π\_{S.Name, C.Title, E.Grade}

((σ\_{S.Major = 'Computer Science'}(STUDENT))

⋈\_{S.StudentID = E.StudentID}

(ENROLLMENT

⋈\_{E.CourseID = C.CourseID}

(σ\_{C.Credits ≥ 3}(COURSE))))

## **Optimized Query Tree Representation**

π\_{S.Name, C.Title, E.Grade}

|

⋈\_{S.StudentID = E.StudentID}

/ \

σ\_{S.Major = 'Computer Science'} ⋈\_{E.CourseID = C.CourseID}

| / \

STUDENT ENROLLMENT σ\_{C.Credits ≥ 3}

|

COURSE

## **Explanation of Optimization Steps**

**Step1 (Selection Pushdown):** أول حاجة بنفلتر الـ **STUDENT** على الناس اللي الـ **Major** بتاعهم "Computer Science"، وبنفلتر الـ **COURSE** على الكورسات اللي عدد الـ **Credits** فيها أكبر من أو يساوي ٣.  
 كده احنا قللنا عدد الصفوف اللي هتدخل على الـ join، فالأداء هيبقى أسرع.

**Step2(Join Ordering):** بعد ما فلترنا، بنعمل join الأول بين **ENROLLMENT** والـ **COURSE** اللي اتفلتر، على أساس إن الـ **CourseID** متساوي.  
 وبعد كده بنربط النتيجة دي مع **STUDENT**، بحيث الـ **StudentID** يساوي بعض.  
 الترتيب ده بيقلل حجم البيانات المؤقتة وبيسرع الاستعلام.

**Step3(Projection at the End):** بعد ما نخلص كل الـ joins، بنعمل **projection** عشان نطلع الأعمدة اللي احنا محتاجينها بس: اسم الـ student، عنوان الـ course، والـ grade.  
 وبكده البيانات اللي طالعة بتبقى مظبوطة ومن غير تحميل زياد.