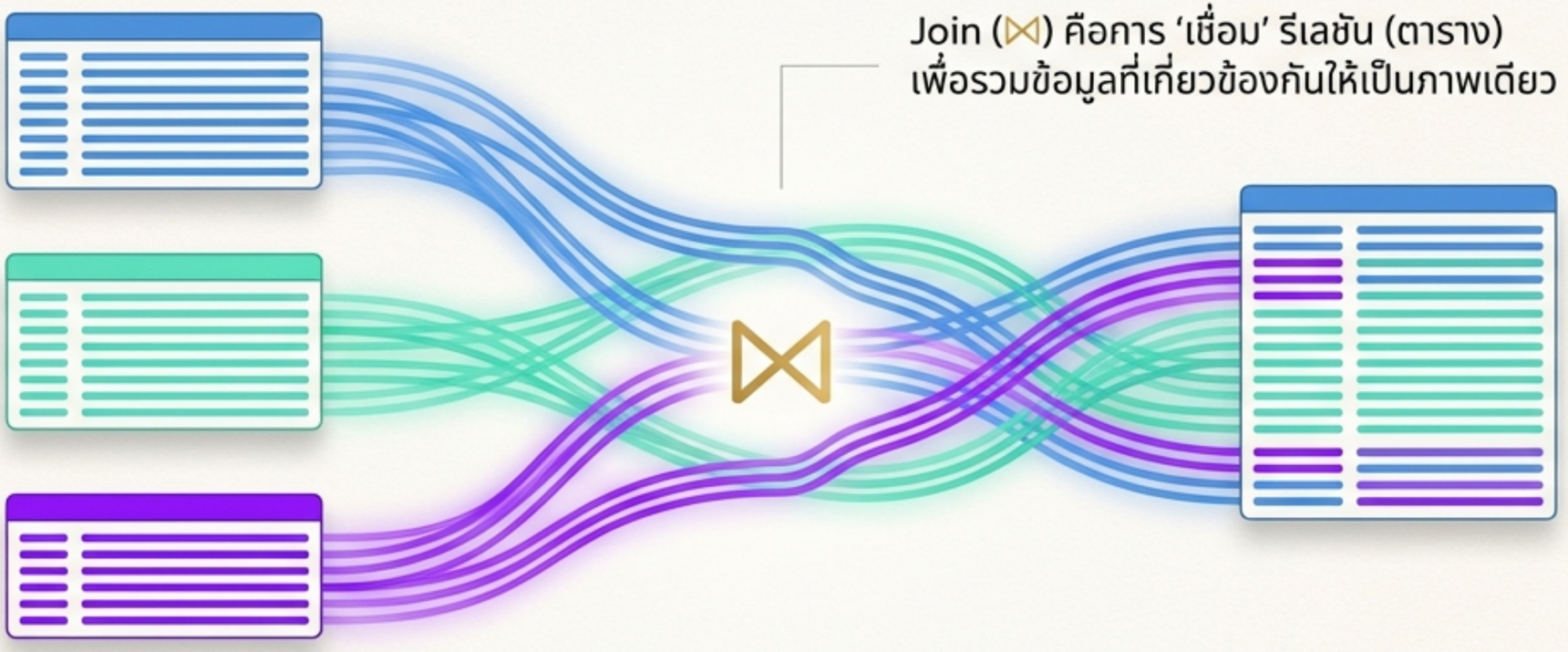
หัวใจของการสืบค้นข้อมูล: ไขความลับของ Relational Algebra Joins

จากแนวคิดพื้นฐานสู่การประยุกต์ใช้ขั้นสูง



Join (⋈) คือการ ‘เชื่อม’ รีเลชัน (ตาราง)
เพื่อรวมข้อมูลที่เกี่ยวข้องกันให้เป็นภาพเดียว

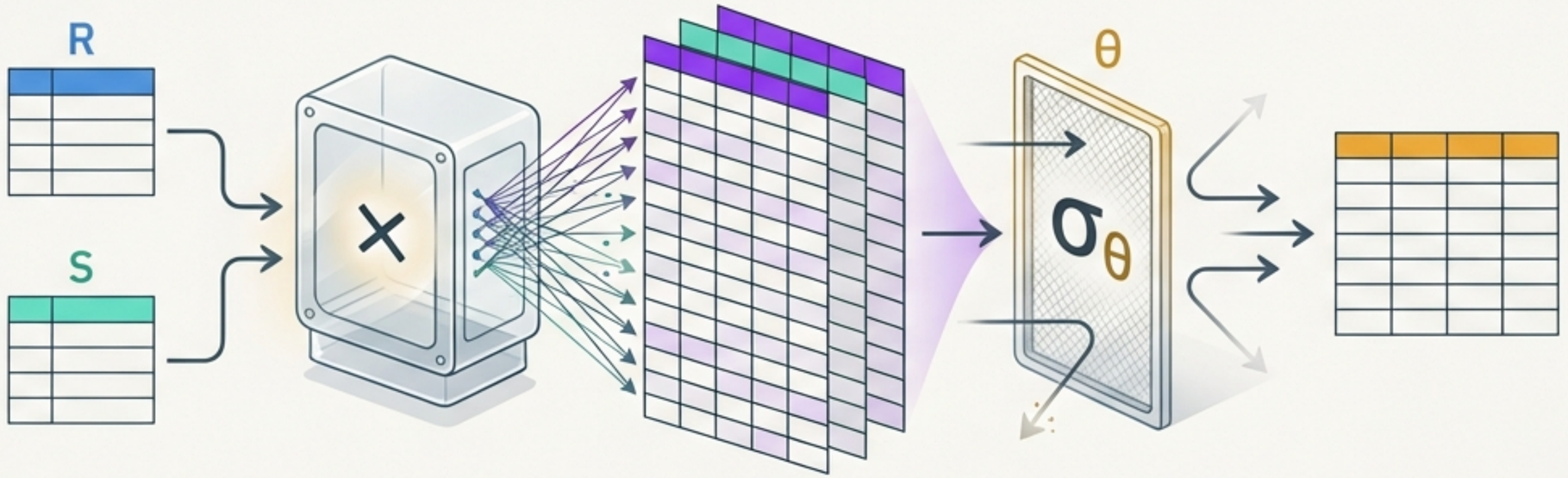
ทำไม Join จึงสำคัญ?

เพราะข้อมูลในโลกความจริงมักถูกจัดเก็บแยกกันเพื่อความเป็นระเบียบ (Normalization)
แต่การจะดึงข้อมูลมาใช้ประโยชน์ ต้องอาศัย Join เพื่อประกอบร่างเรื่องราวทั้งหมดขึ้นมาใหม่

สไลด์ชุดนี้จะพาคุณเดินทางตั้งแต่จุดกำเนิดของ Join ไปจนถึงการเลือกใช้เครื่องมือแต่ละชิ้นอย่างมืออาชีพ

แก่นแท้ของ Join: ไม่ใช่เวทมนตร์ แต่คือกระบวนการสองขั้นตอน

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$



Step 1: Cartesian Product (×) - "สร้างความเป็นไปได้ทั้งหมด**"*:
จับคู่ทุกแถวจากตารางแรกกับทุกแถวจากตารางที่สอง
ผลลัพธ์คือข้อมูลดิบชุดมหาศาล

Step 2: Selection (σ) - "คัดกรองเฉพาะคู่ที่ใช่**"*:
เลือกเก็บเฉพาะคู่แถวที่ตรงตามเงื่อนไข (θ) ที่กำหนด

"Join คือการสร้างทุกความเป็นไปได้ แล้วคัดกรองสิ่งที่ถูกต้องออกมา" นี่คือแนวคิดพื้นฐานที่ใช้สร้าง Join ทุกประเภท

สนามทดลองของเรา: ทำความรู้จักกับ 3 ตารางข้อมูล

Student (SID, SName, Major)

SID	SName	Major
1	Red	IT
2	Blue	DMK
3	Green	IT

Enroll (SID, CID, Grade)

SID	CID	Grade
1	C1	A
1	C2	B
2	C2	A

Course (CID, CName, Credits)

CID	CName	Credits
C1	DB	3
C2	Marketing	3
C3	AI	3

เราจะใช้ข้อมูลชุดนี้เป็นตัวอย่างหลักในการอธิบาย Join แต่ละประเภท โปรดสังเกตสีประจำของแต่ละตาราง

จุดเริ่มต้นแห่งการเชื่อม: Theta Join ($\bowtie_\theta$) – Join ด้วยเงื่อนไขใดก็ได้

- **ลักษณะ**: Join ที่ยืดหยุ่นที่สุด ใช้ตัวเปรียบเทียบทางคณิตศาสตร์ได้ทั้งหมด: $<, >, \leq, \geq, \neq, =$
- **หัวใจ**: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- **การนำไปใช้**: เหมาะกับการจับคู่ข้อมูลที่ไม่จำเป็นต้อง "เท่ากัน" เช่น การหาข้อมูลที่อยู่ในช่วง (range) หรือการจับคู่แบบไม่เท่ากัน (หาพนักงานคนละแผนก)

- **Example**: "จับคู่การลงทะเบียนกับวิชาที่มีหน่วยกิตตั้งแต่ 3 หน่วยกิตขึ้นไป"
- **Relational Algebra Expression**:

Enroll $\bowtie_{(\text{Enroll.CID} = \text{Course.CID} \wedge \text{Course.Credits} \geq 3)}$ Course

Enroll

SID	CID	Grade
1	C1	A
1	C2	B
2	C2	A

$\bowtie_\theta$

Course

CID	CName	Credits
C1	DB	3
C2	Marketing	3
C3	AI	3

SID	CID	Grade	CName	Credits
1	C1	A	DB	3
1	C2	B	Marketing	3
2	C2	A	Marketing	3

เครื่องมือหลักในชีวิตจริง: Equi Join ($\bowtie_{=}$) – Join ด้วยเงื่อนไข "เท่ากัน"

- **ลักษณะ:** เป็น Theta Join ที่จำกัดเงื่อนไขเป็น "เท่ากัน (=)" เท่านั้น
- **การนำไปใช้:** เป็น Join ที่ใช้บ่อยที่สุดในระบบฐานข้อมูลจริง โดยเฉพาะการเชื่อม Primary Key กับ Foreign Key
- **ตัวอย่างโจทย์:** ต้องการดูรายละเอียดของนักศึกษาควบคู่ไปกับข้อมูลการลงทะเบียน
- **Relational Algebra Expression:** Student $\bowtie$(Student.SID = Enroll.SID) Enroll

SID	SName	Major	SID	CID	Grade
1	Red	IT	1	C1	A
1	Red	IT	1	C2	B
2	Blue	DMK	2	C2	A

ข้อสังเกตสำคัญ: Equi Join จะเก็บคอลัมน์ที่ใช้เชื่อม (SID) ไว้ทั้งสองชุด ทำให้เกิดความซ้ำซ้อน

εξυπνότερη ένωση: Natural Join (⋈) – Join อัจฉริยะที่รู้ใจ (แต่ต้องระวัง)

- **ลักษณะ:** Join อัตโนมัติโดยใช้คอลัมน์ที่มี "ชื่อเหมือนกัน" เป็นเงื่อนไข และ **ลบคอลัมน์ที่ซ้ำออกให้เหลือเพียงชุดเดียว**
- **ข้อดี:** เขียนนิพจน์ได้สั้นกระชับ อ่านง่าย
- **ข้อควรระวัง:** หากมีคอลัมน์ชื่อเหมือนกันโดยบังเอิญ อาจทำให้ผลลัพธ์ผิดพลาดได้!

Relational Algebra Expression: `Student ⋈ Enroll`

↖ ระบบจะรู้ทันทีว่าต้อง Join ด้วย `SID` เพราะเป็นคอลัมน์เดียวที่ชื่อเหมือนกัน

SID	SName	Major	SID	CID	Grade
1	Red	IT	1	C1	A
1	Red	IT	1	C2	B
2	Blue	DMK	2	C2	A

SID	SName	Major	CID	Grade
1	Red	IT	C1	A
1	Red	IT	C2	B
2	Blue	DMK	C2	A

แล้วข้อมูลที่ "ไม่มีคู่" หายไปไหน?

Student

SID	SName	Major
1	Red	IT
2	Blue	DMK
3	Green	IT

Enroll

SID	CID	Grade
1	C1	A
1	C2	B
2	C2	A

Student ⋈ Enroll

SID	SName	Major	CID	Grade
1	Red	IT	C1	A
1	Red	IT	C2	B
2	Blue	DMK	C2	A

เราจะทำอย่างไร ถ้าต้องการเห็นรายชื่อนักศึกษา *ทุกคน* ในระบบ พร้อมข้อมูลการลงทะเบียน *ถ้ามี*?

Join ประเภท Theta, Equi, และ Natural (เรียกรวมว่า Inner Joins) จะแสดงผลเฉพาะแถวที่หาคู่เจอเท่านั้น แถวที่ไม่แมตช์จะถูกทิ้งไป... แต่ในหลายกรณี เราต้องการเก็บข้อมูลเหล่านั้นไว้

การเก็บรักษาทุกแถว: Outer Joins – เติม NULL ให้ข้อมูลที่ไม่พบคู่

- **หลักการ**: เก็บทุกแถวของตารางฝั่งซ้ายไว้เสมอ หากหาคู่ในตารางฝั่งขวาไม่เจอ ให้เติมค่า `NULL` ในคอลัมน์ของตารางฝั่งขวา
- **ประโยชน์**: เหมาะสำหรับทำรายงานที่ต้องการเห็นภาพรวมทั้งหมด เช่น "รายชื่อนักศึกษาทุกคนและวิชาที่ลงทะเบียน (ถ้ามี)"
- **Relational Algebra Expression**: Student ⟕ Enroll

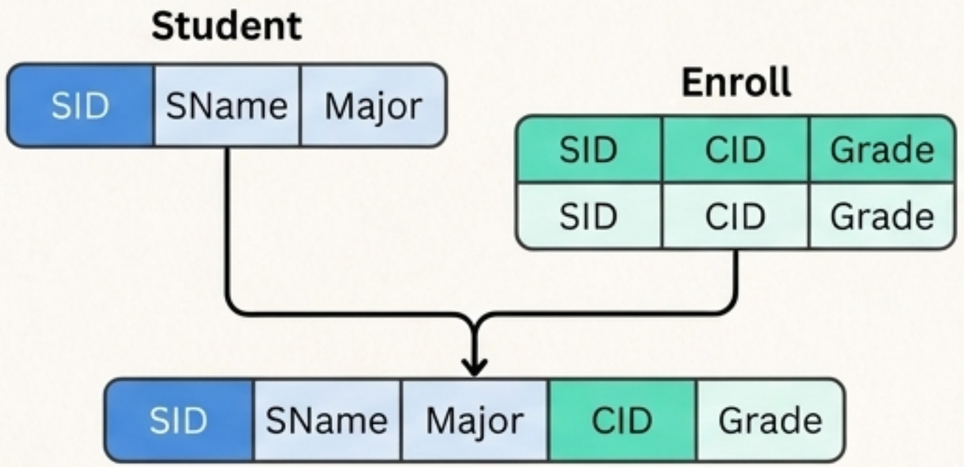
ผลลัพธ์ของ Left Outer Join

SID	SName	Major	CID	Grade
1	Red	IT	C1	A
1	Red	IT	C2	B
2	Blue	DMK	C2	A
3	Green	IT	*NULL*	*NULL*

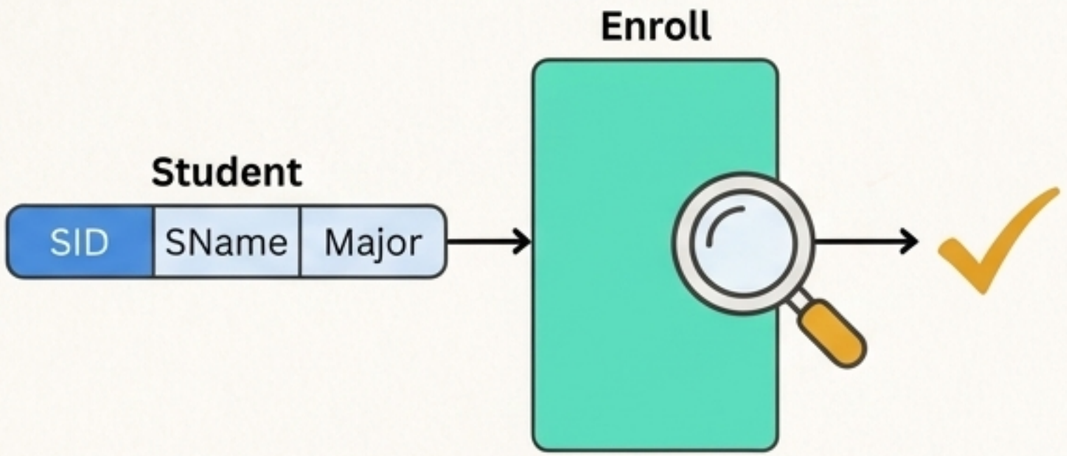
Right Outer Join (⟖): ทำตรงกันข้าม, เก็บทุกแถวฝั่งขวา	**Full Outer Join (⟗):** เก็บทุกแถวจากทั้งสองฝั่ง

เปลี่ยนคำถามใหม่: ไม่ใช่ "ข้อมูลคืออะไร" แต่เป็น "มีข้อมูลหรือไม่"

ต้องการ *ชื่อ* และ *เกรด*



ต้องการแค่รู้ว่า *เคยลงทะเบียนหรือไม่*



- บางครั้งเราไม่ต้องการข้อมูลจากตารางที่สองเลย เราแค่ต้องการใช้มันเป็น 'ตัวกรอง' เพื่อคัดเลือกแถวในตารางแรก
- **คำถามตัวอย่าง:**
  - นักศึกษาคนไหนบ้างที่เคยลงทะเบียนเรียน *อย่างน้อยหนึ่งวิชา*?
  - นักศึกษาคนไหนบ้างที่ *ยังไม่เคยลงทะเบียนเรียนเลย*?
- Inner และ Outer Joins ให้ข้อมูลมากเกินความจำเป็นสำหรับคำถามประเภทนี้

เครื่องมือสำหรับคัดกรอง: Semi Join (⋉) และ Anti Join (▷)

Semi Join (⋉) – "หาคนที่มีคู่"

- **หลักการ:** คืนค่าเฉพาะแถวจากตารางฝั่งซ้ายที่สามารถหาคู่ในตารางฝั่งขวาเจอ
- **เปรียบเสมือน:** `EXISTS` ใน SQL
- **คำถาม:** นักศึกษาที่มีการลงทะเบียนอย่างน้อย 1 รายวิชา
- **Expression:** `Student ⋉ Enroll`

SID	SName	Major
1	Red	IT
2	Blue	DMK

Anti Join (▷) – "หาคนที่ไม่มีคู่"

- **หลักการ:** ตรงข้ามกับ Semi Join, คืนค่าเฉพาะแถวจากตารางฝั่งซ้าย ที่ **ไม่** สามารถหาคู่ในตารางฝั่งขวาเจอ
- **เปรียบเสมือน:** `NOT EXISTS` ใน SQL
- **คำถาม:** นักศึกษาที่ยังไม่เคยลงทะเบียนเรียน
- **Expression:** `Student ▷ Enroll`

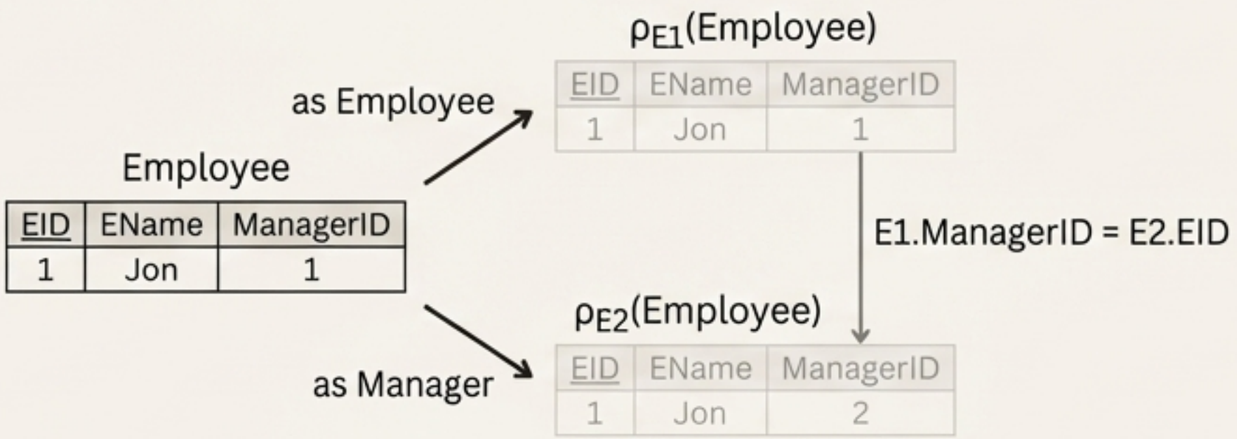
SID	SName	Major
3	Green	IT

เมื่อตารางสัมพันธ์กับตัวเอง: Self Join – การใช้ Rename (ρ) เพื่อสร้างมุมมอง

ลักษณะ: คือการ Join ตารางเดิมเข้ากับตัวเอง ใช้เมื่อข้อมูลมีความสัมพันธ์แบบลำดับชั้น (hierarchical) อยู่ภายในตารางเดียว
ปัญหา: เราจะอ้างอิงถึงคอลัมน์ EID จากฝั่ง "ลูกน้อง" และฝั่ง "หัวหน้า" ได้อย่างไรในเมื่อมันคือตารางเดียวกัน?
ทางออก: ใช้โอเปอเรเตอร์ Rename (ρ) เพื่อสร้าง "ชื่อเล่น" หรือ "alias" ให้กับตาราง ทำให้เรามองเหมือนว่ามีตารางสองตารางที่แยกจากกัน

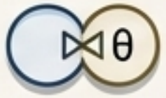
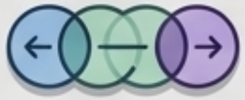
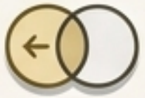
Example Scenario: "ต้องการหารายชื่อพนักงานทุกคนคู่กับชื่อหัวหน้าของพวกเขา"

Employee(EID, EName, ManagerID)



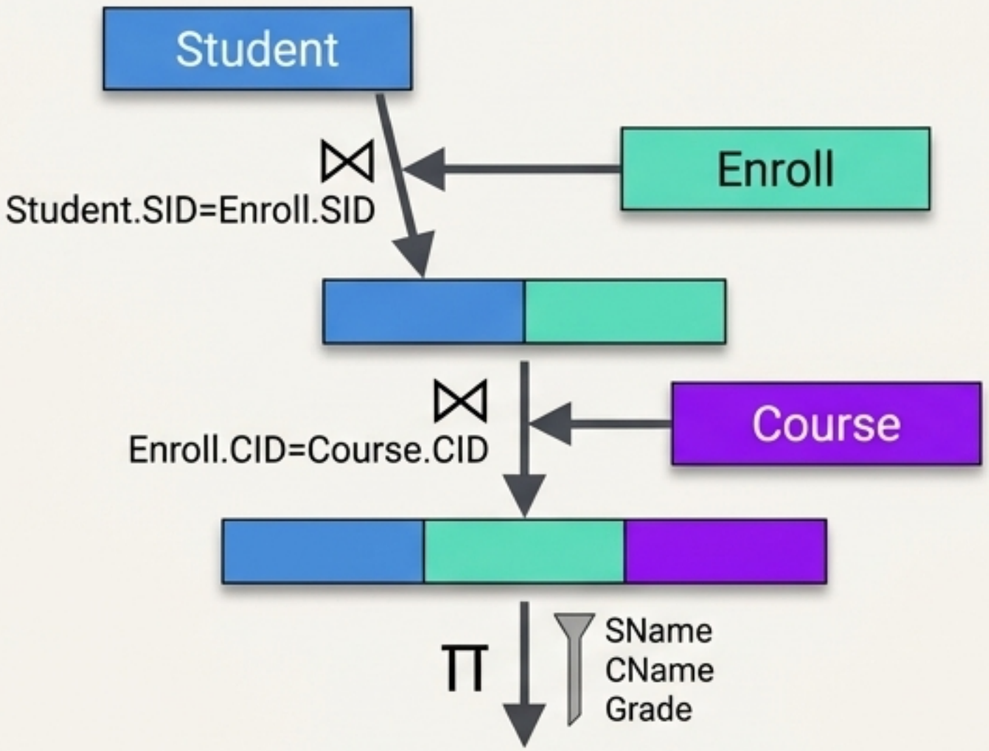
$$\rho_{E1}(\text{Employee}) \bowtie_{(E1.\text{ManagerID} = E2.\text{EID})} \rho_{E2}(\text{Employee})$$

The Join Playbook: เลือกใช้เครื่องมือให้ถูกกับงาน

Join Type (สัญลักษณ์)	เป้าหมายหลัก (Goal)	ผลลัพธ์ที่ได้ (Result)	สถานการณ์ที่ควรใช้ (When to Use)
 Theta Join (⋈θ)	Join ด้วยเงื่อนไขทั่วไป	แถวที่ผ่านเงื่อนไข θ	เปรียบเทียบข้อมูลแบบไม่เท่ากัน, จับคู่ตามช่วง
 Equi Join (⋈=)	Join ด้วยเงื่อนไข "="	แถวที่คีย์ตรงกัน (คอลัมน์คีย์ซ้ำ)	การ Join พื้นฐานที่สุด, เชื่อม PK-FK (ใช้บ่อยสุด)
 Natural Join (⋈)	Join ด้วยคอลัมน์ชื่อเหมือนกัน	แถวที่คีย์ตรงกัน (คอลัมน์คีย์ไม่ซ้ำ)	เมื่อ Schema ออกแบบดี, ต้องการความกระชับ
 Outer Join (⟕, ⟗)	เก็บแถวที่ไม่แมตช์ไว้	แถวที่เมตช์ + แถวที่ไม่แมตช์ (เติม NULL)	ทำรายงานที่ต้องการข้อมูลครบทุกคน/ทุกรายการ
 Semi Join (⋉)	คัดกรองแถวที่ "มี" คู่	เฉพาะแถวจากฝั่งซ้ายที่มีคู่	ตรวจสอบการมีอยู่ (Exists), หาคนที่มี transaction
 Anti Join (▷)	คัดกรองแถวที่ "ไม่มี" คู่	เฉพาะแถวจากฝั่งซ้ายที่ไม่มีคู่	หาข้อมูลกำพร้า, หาลูกค้าที่ไม่เคยซื้อของ
 Self Join (ρ)	เชื่อมข้อมูลภายในตารางเดียว	แถวที่สัมพันธ์กันในตารางเดิม	โครงสร้างลำดับชั้น (หัวหน้า-ลูกน้อง), prerequisite

ประกอบร่างเรื่องราวทั้งหมด: ตัวอย่างการ Join หลายตาราง

****เป้าหมาย**** แสดงรายงานที่มี **ชื่อนักศึกษา**, **ชื่อวิชาที่ลงทะเบียน**, และ **เกรดที่ได้**

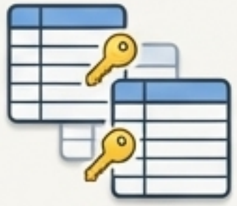


SName	CName	Grade
Red	DB	A
Red	Marketing	B
Blue	Marketing	A

$$\Pi_{SName,\ CName,\ Grade}\ (\ (Student \bowtie_{(Student.SID=Enroll.SID)} Enroll) \bowtie_{(Enroll.CID=Course.CID)} Course\ )$$

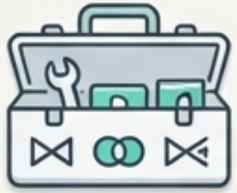
กระบวนการคิดแบบ Master: จากโจทย์สู่คำตอบด้วย Join

1



ข้อมูลที่ต้องการมาจากตารางไหนบ้าง?
และคอลัมน์ใดคือตัวเชื่อมความสัมพันธ์ (keys)?

2



ถามตัวเอง: ต้องการเฉพาะที่แมตซ์ (Inner)?
หรือต้องการเก็บข้อมูลที่ไม่แมตซ์ด้วย (Outer)?
หรือแค่ต้องการตรวจสอบว่า 'มี/ไม่มี' (Semi/Anti)?

3

มีเงื่อนไขเฉพาะเจาะจงที่ต้องกรองข้อมูลออกหรือไม่?
(เช่น Major='IT', Grade='A')
ควรทำ Selection ก่อนเพื่อลดขนาดข้อมูล

4

สุดท้ายแล้ว ผลลัพธ์สุดท้ายต้องการแสดงคอลัมน์ใดบ้าง?
ใช้ Projection เพื่อเลือกเฉพาะสิ่งที่จำเป็น

การเลือก Join ที่ถูกต้อง คือการเลือกเลนส์ที่เหมาะสมในการมองความสัมพันธ์ของข้อมูล