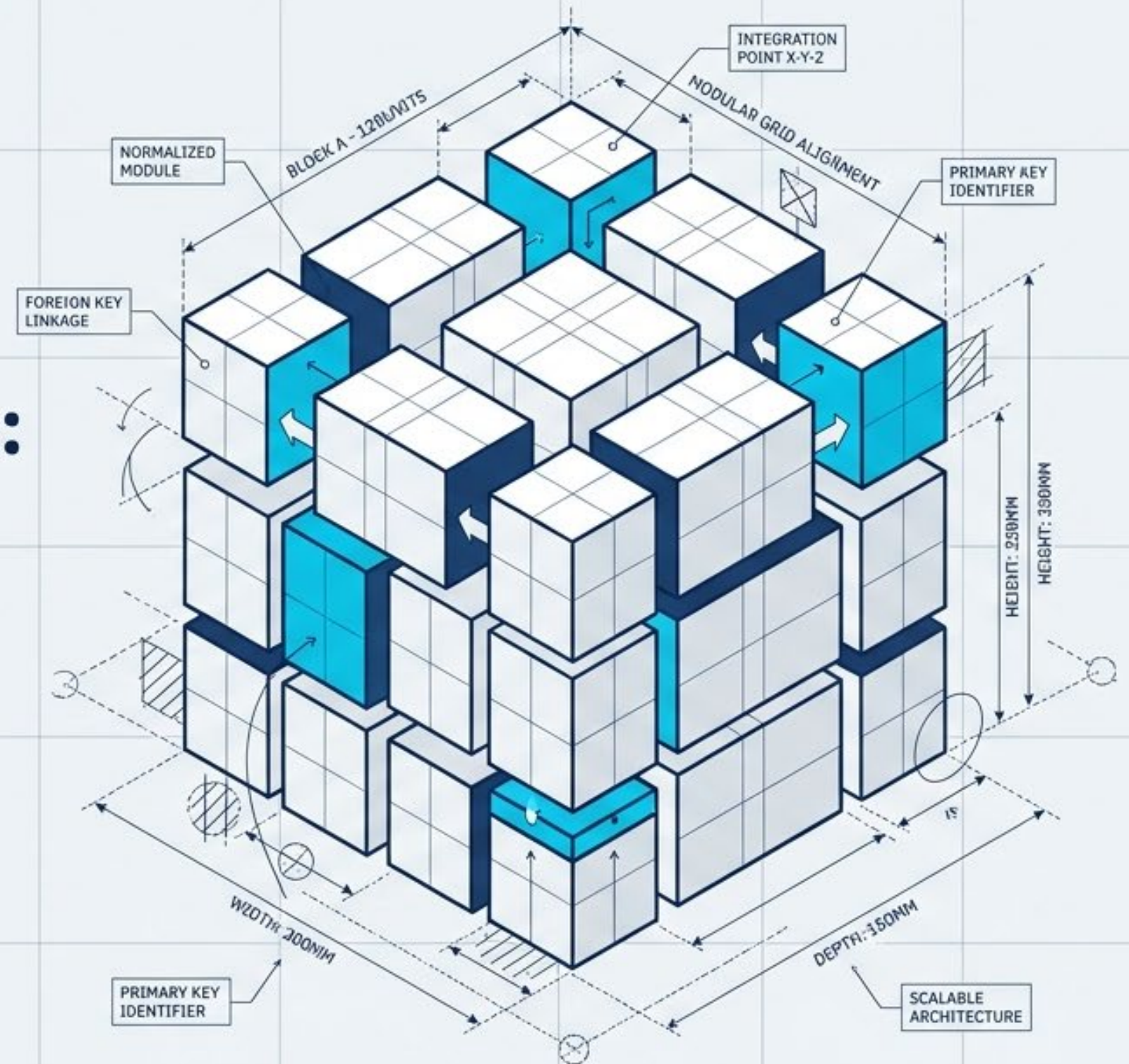


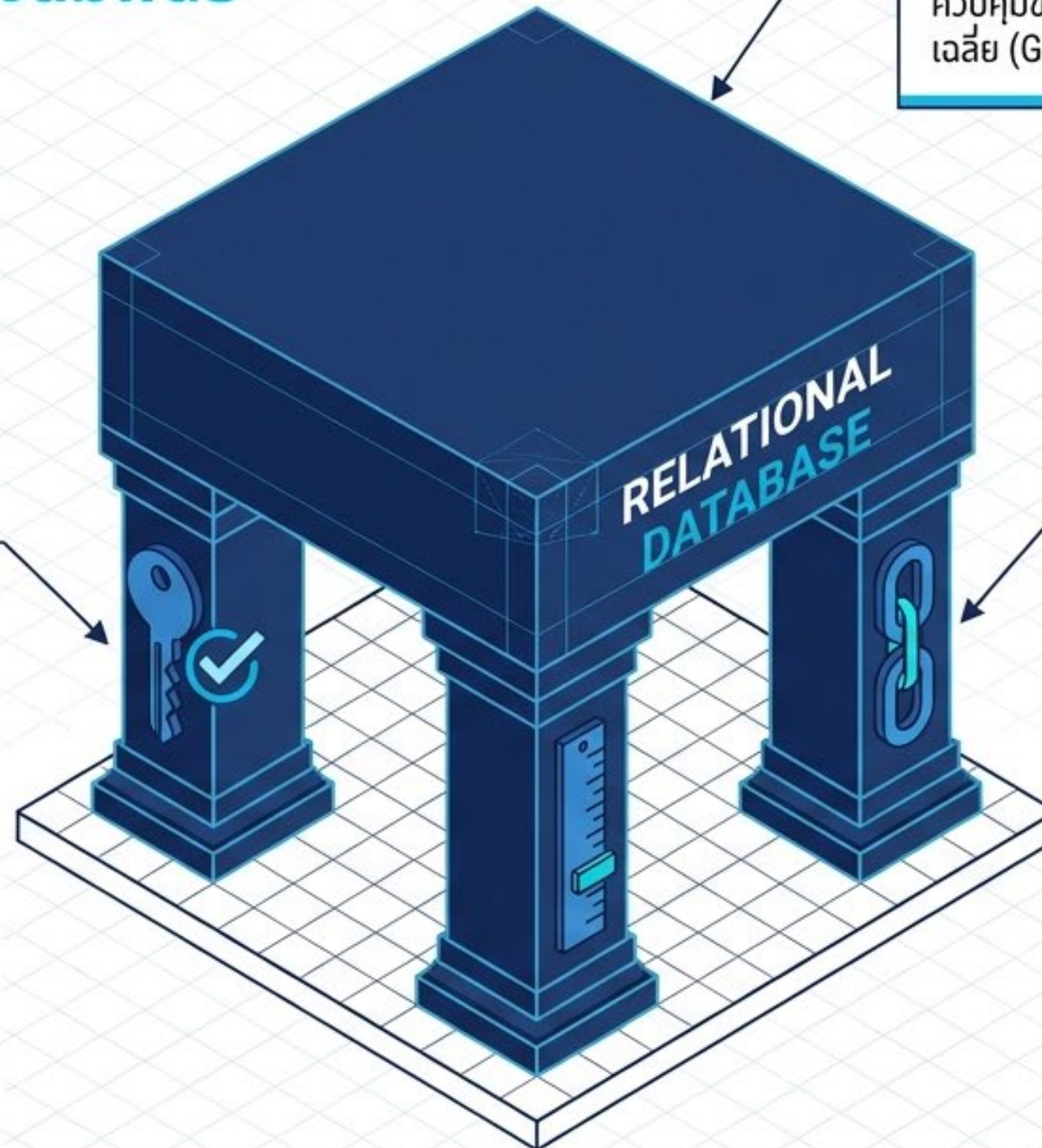
Architecting Relational Data: ศาสตร์แห่งการออกแบบและ การนอร์มัลไลเซชันฐานข้อมูล

A Visual Blueprint for Data Architecture



The 3 Pillars of Data Integrity

รากฐานของฐานข้อมูลเชิงสัมพันธ์



กฎความคงสภาพของเอนทิตี (Entity Integrity)

Primary Key ต้องไม่เป็นค่าว่าง (NULL)
รับประกันเอกลักษณ์ของแต่ละทิวเพิล

กฎความคงสภาพของโดเมน (Domain Integrity)

ควบคุมขอบเขตข้อมูลให้ถูกต้อง เช่น เกรดเฉลี่ย (GPA) ต้องอยู่ระหว่าง 0.00 - 4.00

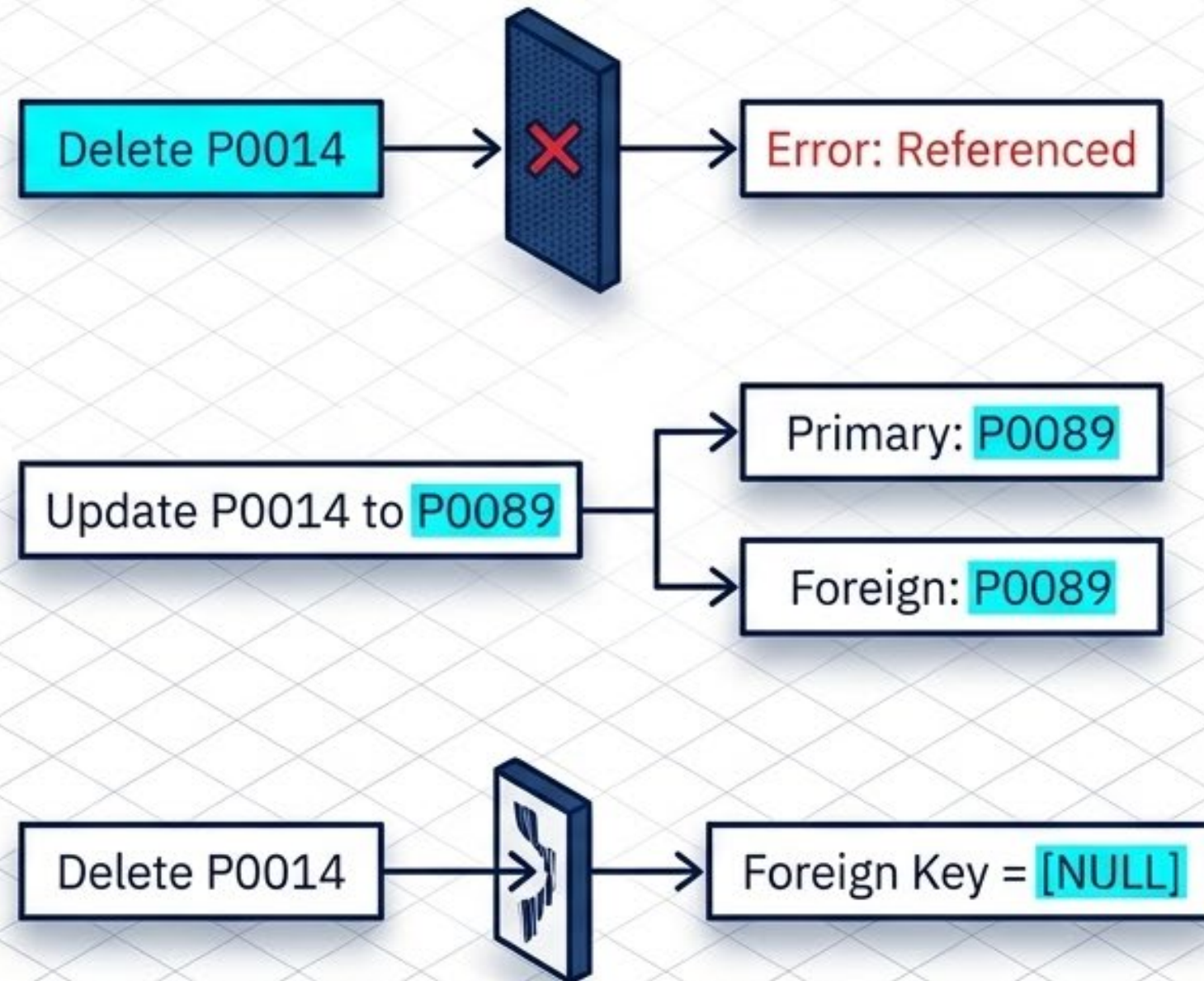
กฎความคงสภาพของการอ้างอิง (Referential Integrity)

Foreign Key ต้องอ้างอิงไปยัง Primary Key ที่มีอยู่จริง
เชื่อมโยงความสัมพันธ์ไม่ให้ขาดตอน

Referential Actions Matrix

กลไกการทำงานเมื่อมีการอัปเดตหรือลบข้อมูลต้นทาง

	Restricted (จำกัด/ห้าม) ไม่อนุญาตให้แก้ไข/ลบ Primary Key หากยังมี Foreign Key อ้างอิงถึงอยู่
	Cascades (อัปเดตต่อเนื่อง) เปลี่ยน Primary Key แล้ว Foreign Key ที่เกี่ยวข้องจะเปลี่ยนตามโดยอัตโนมัติ
	Nullify (กำหนดเป็นค่าว่าง) ยอมให้ลบ Primary Key ต้นทางได้ แต่ปลายทางจะถูกปรับเป็นค่าว่าง (NULL)



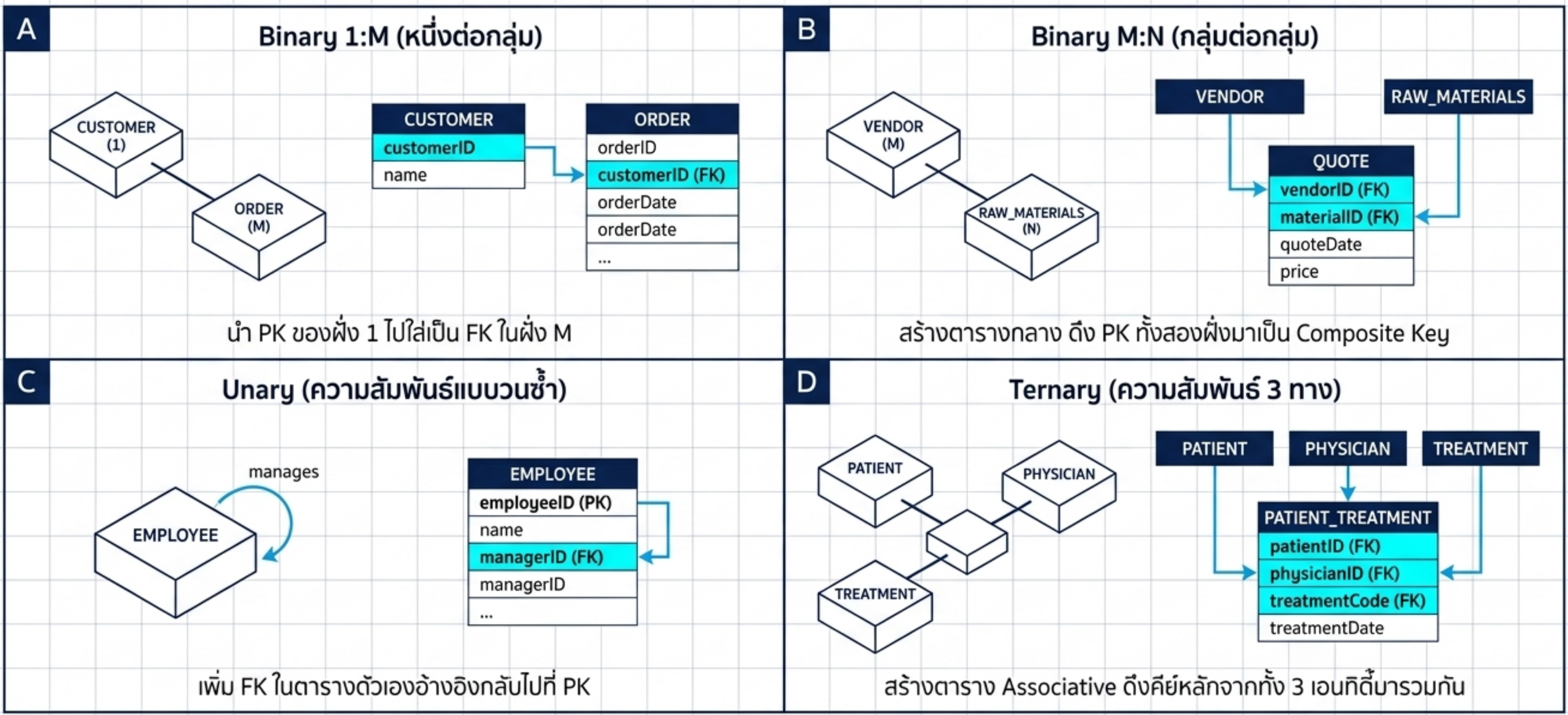
Entity Mapping Cheat Sheet

การแปลง E-R Diagram ขั้นพื้นฐานสู่โครงสร้างตาราง

	E-R Concept (Conceptual)		Relational Schema (Logical)
Strong Entity (เอนทิตีปกติ)	CUSTOMER	→	CUSTOMER CUSTOMER [customerID (PK), name]
Weak Entity (เอนทิตีอ่อนแอ)	DEPENDENT — EMPLOYEE	→	DEPENDENT DEPENDENT [employeeID (PK/FK), dependentName (PK)]
Composite Attributes (แอตทริบิวต์ผสม)	Address street city zipCode	→	 ~~Address~~ [... , street, city, zipCode]
Multivalued Attributes (แอตทริบิวต์หลายค่า)	Skill — EMPLOYEE	→	EMPLOYEE [employeeID] — EMPLOYEE_SKILL [employeeID (FK), skill]

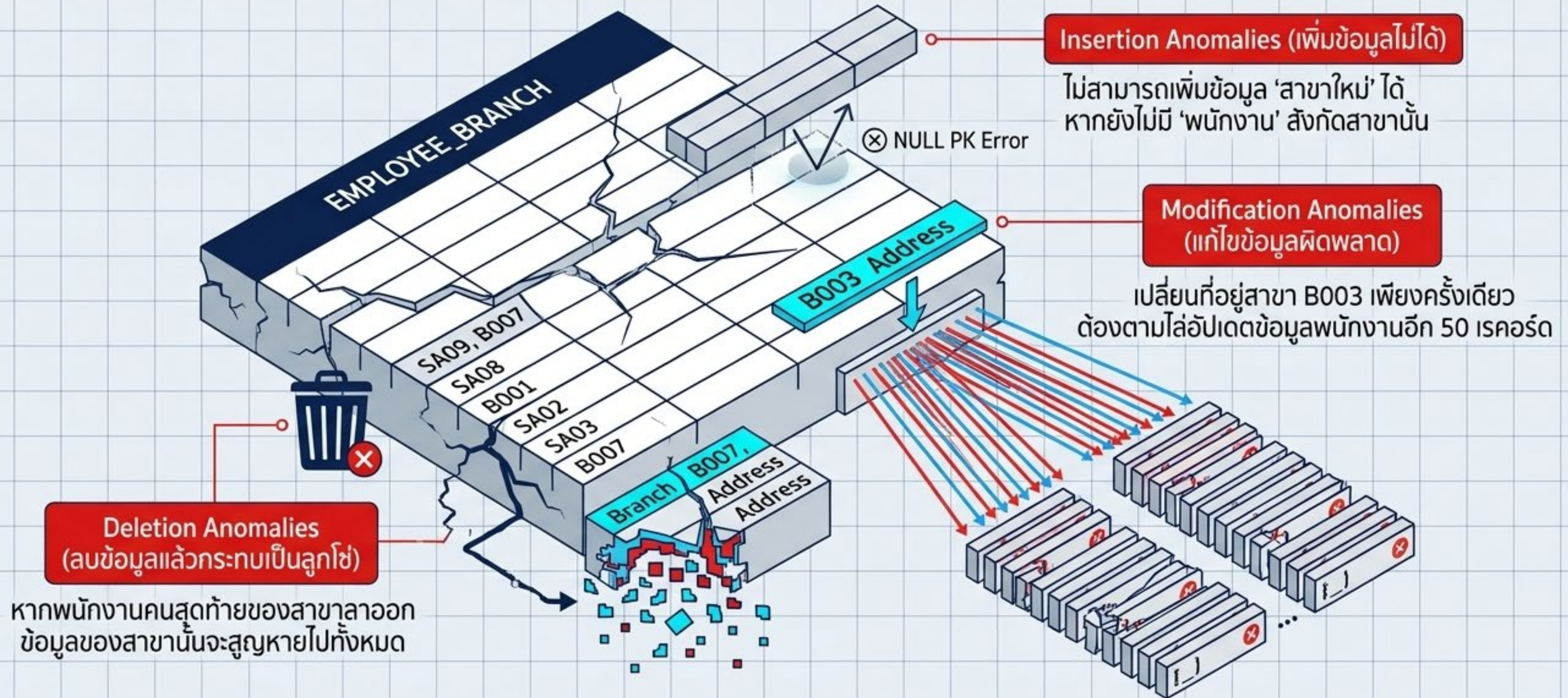
Relationship Mapping Cheat Sheet

ตรรกะการเชื่อมโยงความสัมพันธ์และการคัดลอกคีย์



The Cost of Redundancy: เมื่อโครงสร้างข้อมูลล้มเหลว

ปัญหา (Anomalies) จากการรวมตารางพนักงานและสาขาไว้ด้วยกัน



Anatomy of Functional Dependencies

กลไก 'ตัวกำหนด' และ 'ตัวตาม' ภายใต้อำนาจของ Normalization



ค่าของ A สามารถล็อกสเปกค่าของ B ได้
(เช่น employeeNo -> position)

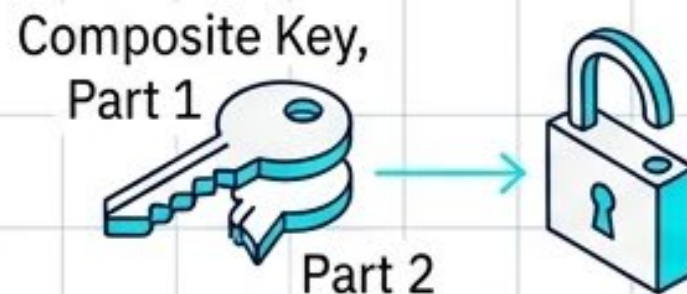
Fully Functional Dependency



ใช้ตัวกำหนดที่เล็กและตรงจุดที่สุด

employeeNo ชี้เป้า branchNo ได้ทันที

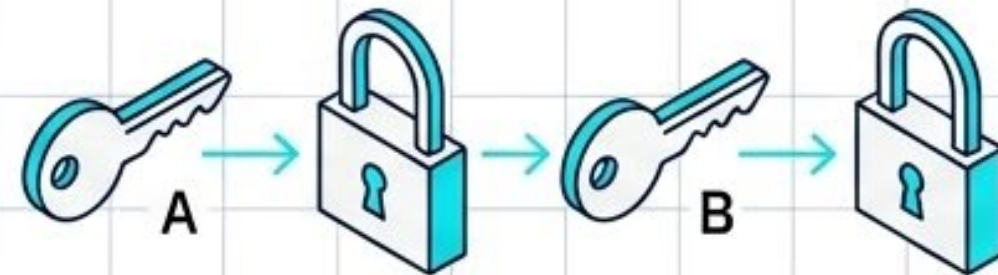
Partial Dependency



ตัวตาม ขึ้นอยู่กับ 'แค่บางส่วน'
ของคีย์หลักแบบกลุ่ม (Composite Key)

propertyNo รู้อยู่แล้วว่า propertyAddr
คืออะไร ไม่ต้องพึ่ง clientNo

Transitive Dependency

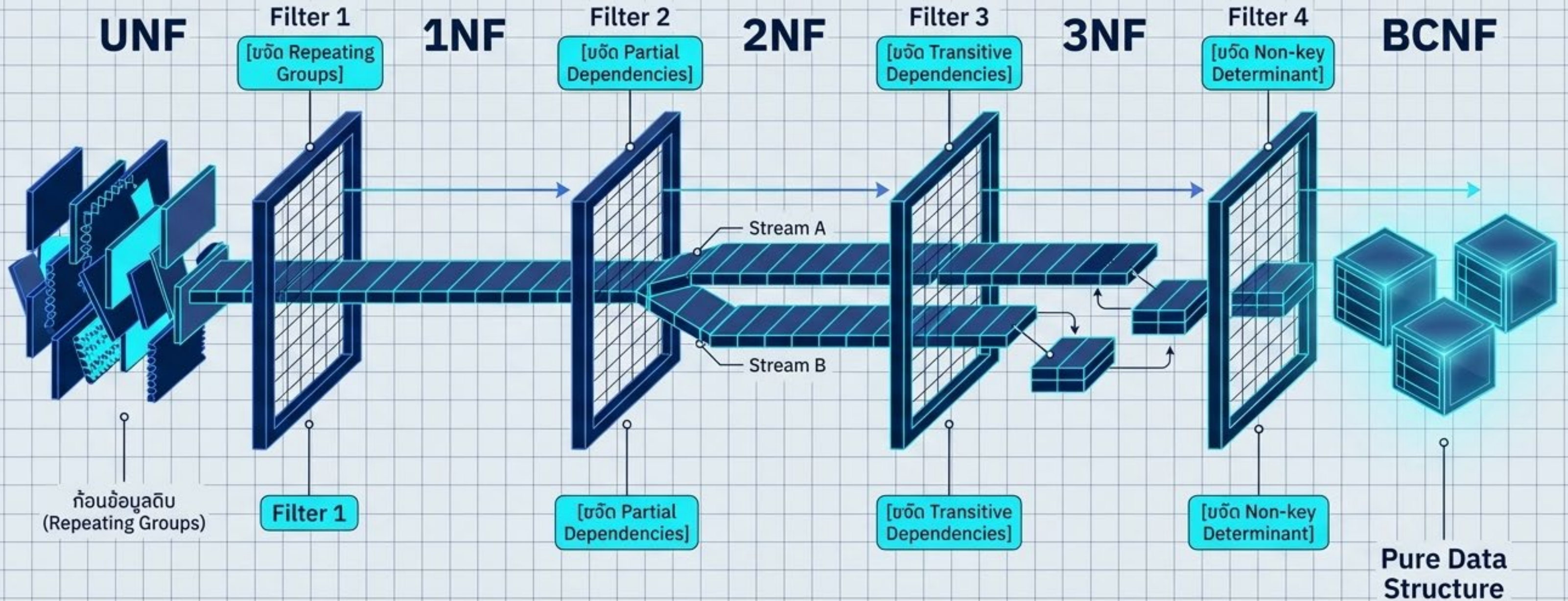


แอตทริบิวต์ที่ไม่ใช่คีย์
ไม่เป็นตัวกำหนดแอตทริบิวต์อื่นต่อเป็นทอดๆ

ownerNo ระบุ ownerName ได้เอง
โดยไม่เกี่ยวกับรหัสบ้านพัก

The Normalization Pipeline

กระบวนการสกัดและกรองข้อมูล สู่โครงสร้างที่สมบูรณ์แบบ



Step 1: UNF to 1NF (The Flattening)

เป้าหมาย: ลบกลุ่มข้อมูลที่ซ้ำซ้อน (Repeating Groups)

BEFORE: UNF
(Unnormalized Form)

CLIENT_RENTAL		(propertyNo, propertyAddr, rentStart)		
clientNo	clientName			
CR56	Aline Stewart	propertyNo: PG04	propertyAddr: 6 Lawrence St, Glasgow	rentStart: 01-Jul-00
		propertyNo: PG36	propertyAddr: 2 Manor Rd, Glasgow	rentStart: 10-Oct-01
		propertyNo: PG16	propertyAddr: 5 Novar Dr, Glasgow	rentStart: 01-Nov-02

ข้อมูลซ้อนทับกันหลายบรรทัดใน 1 เรคอร์ด

THE FLATTENING

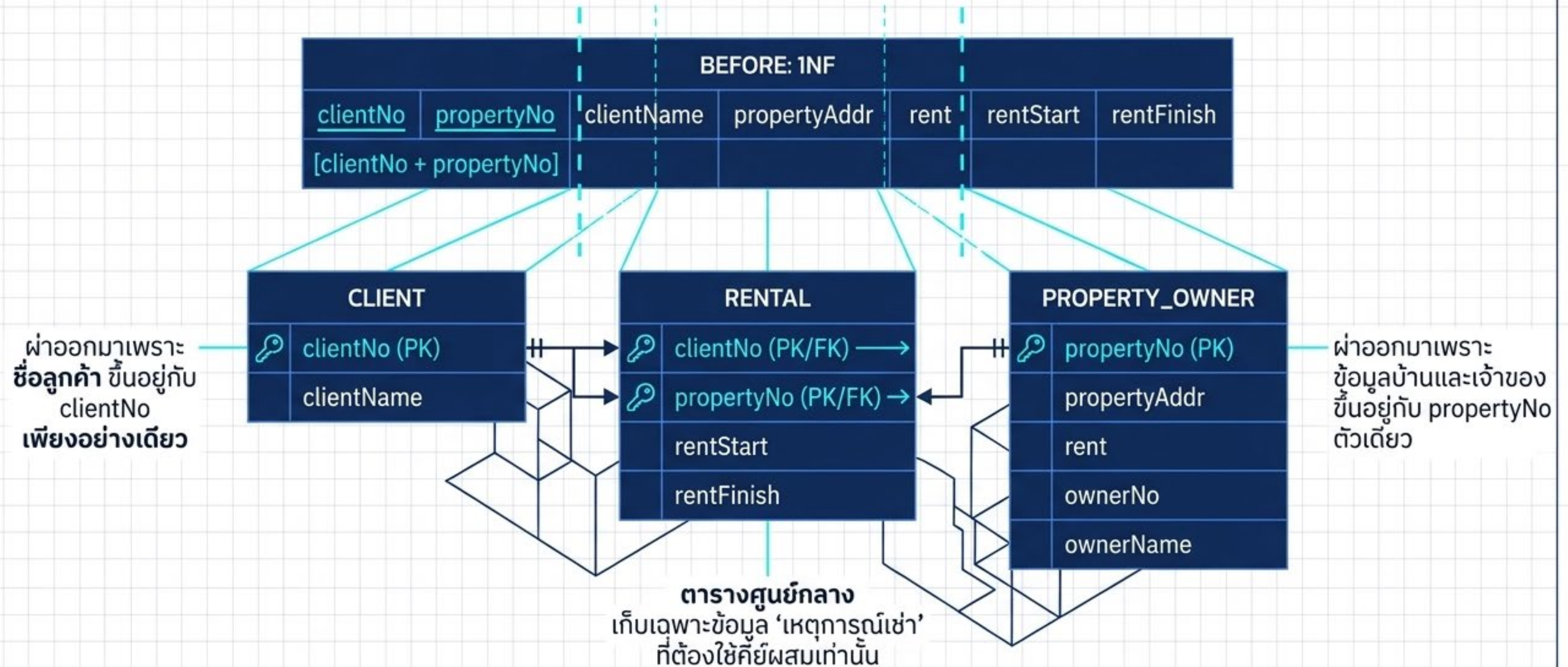
AFTER: 1NF
(First Normal Form)

Composite Primary Key: [clientNo + propertyNo]		clientName	propertyAddr	rentStart
clientNo	propertyNo			
CR56	PG04	Aline Stewart	6 Lawrence St, Glasgow	01-Jul-00
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-01
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	01-Nov-02

ตารางถูกเติมเต็มช่องว่าง (Fill)
1 แถว = 1 เหตุการณ์การเช่า

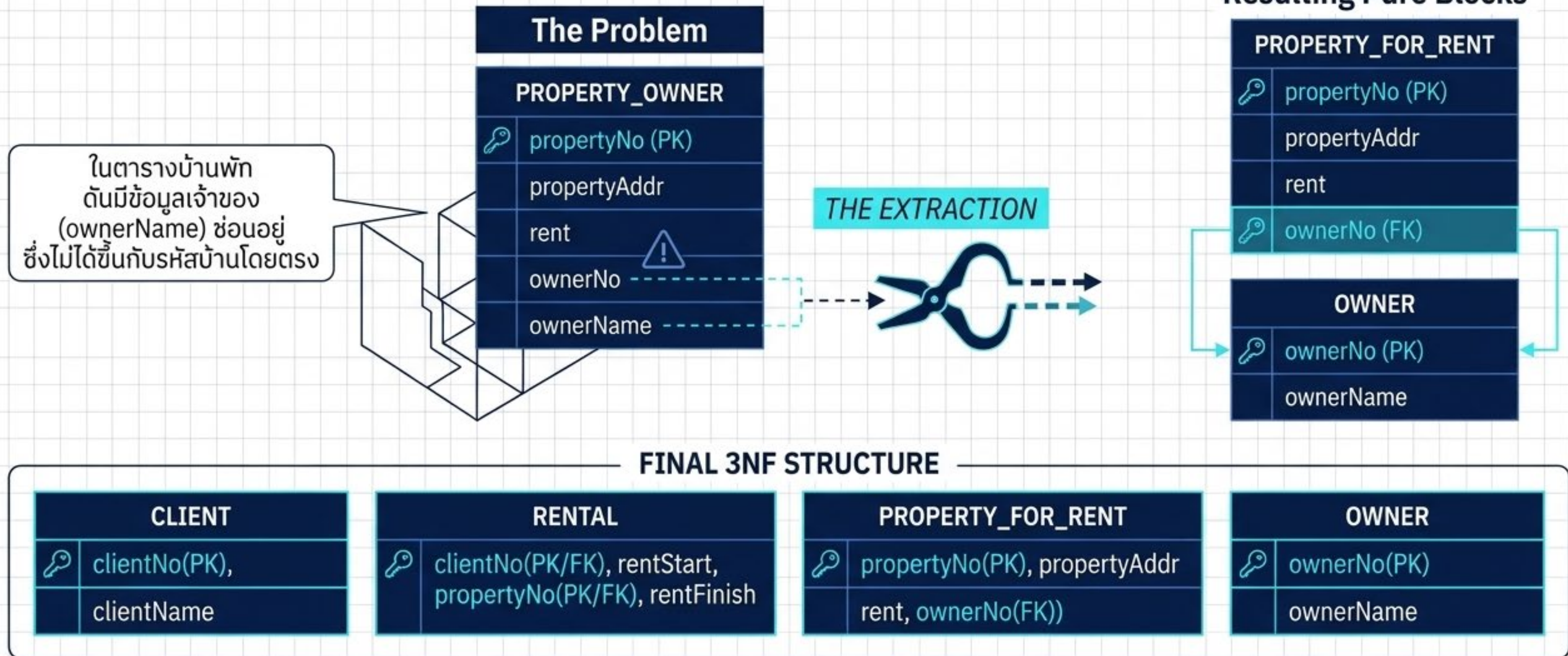
Step 2: 1NF to 2NF (The Splitting)

เป้าหมาย: ขจัด Partial Dependency (แอตทริบิวต์ที่ขึ้นกับคีย์หลักแค่ครึ่งเดียว)



Step 3: 2NF to 3NF (The Refining)

เป้าหมาย: ขจัด Transitive Dependency (ข้อมูลแฝงที่ระบุค่ากันเอง)



โครงสร้าง 3NF ปราศจากความซ้ำซ้อน 100%

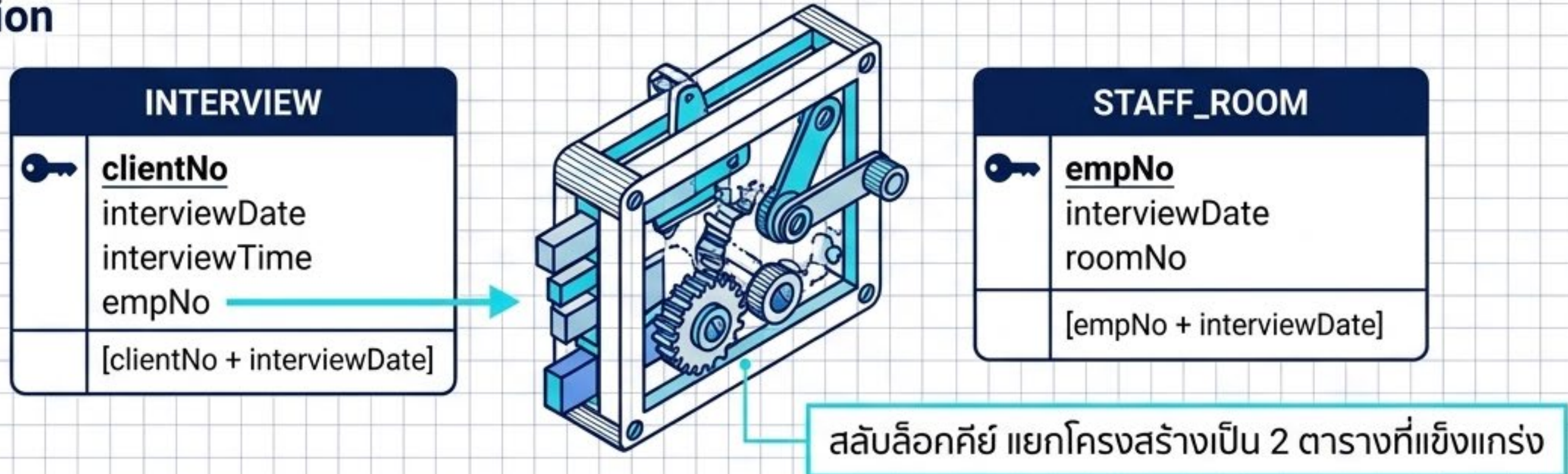
Advanced Refinement: Boyce-Codd Normal Form (BCNF)

ขจัดแอตทริบิวต์ที่ไม่ใช่คีย์ แต่ไประบุค่าส่วนหนึ่งของคีย์หลักได้

The BCNF Anomaly



The Resolution



The Trade-off: Normalization vs. Denormalization

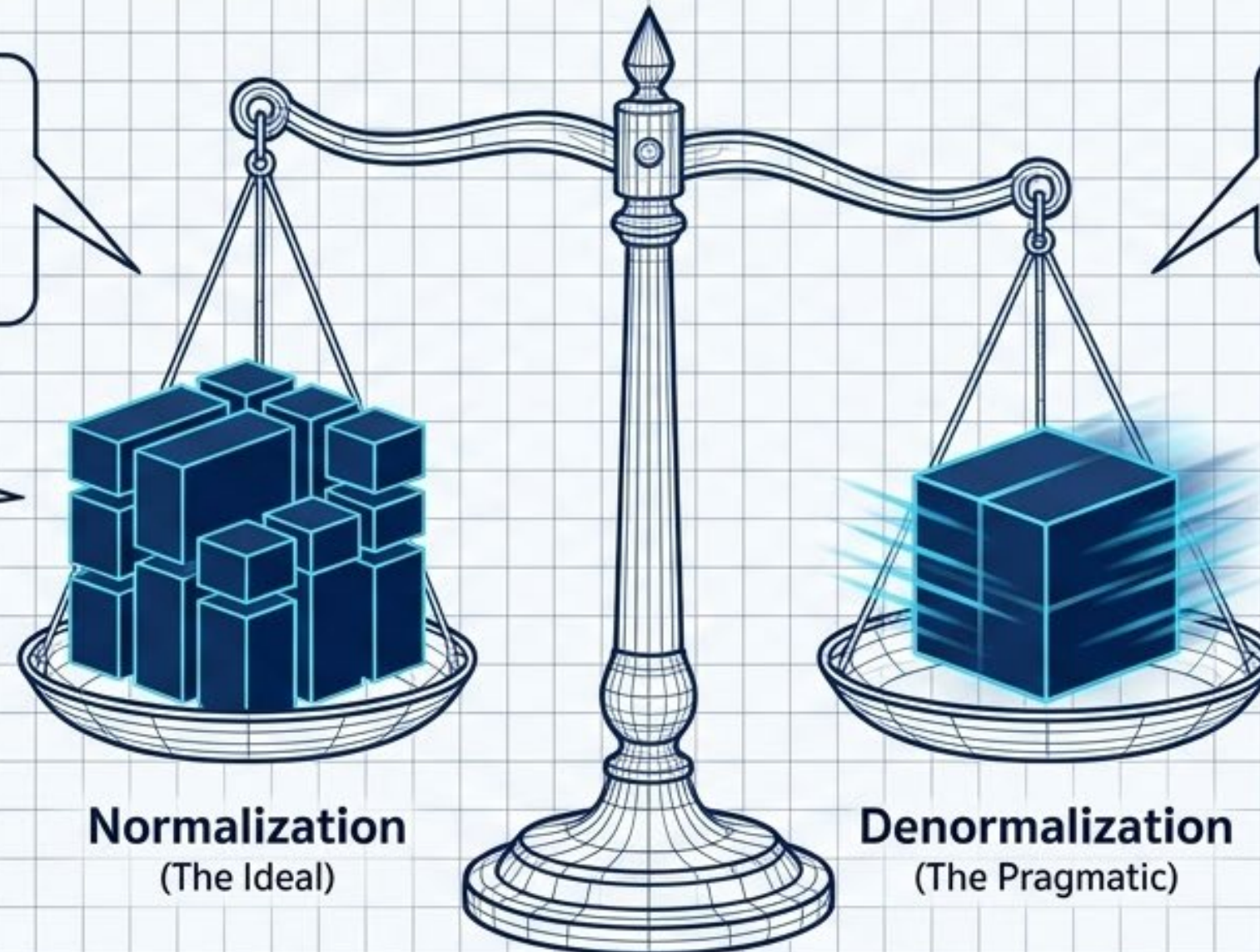
ความสมดุลระหว่าง 'ความถูกต้องของข้อมูล' และ 'ความเร็วในการอ่าน'

Pros:

ลดความซ้ำซ้อน 100%,
ปรับปรุงข้อมูลจุดเดียว
ประหยัดพื้นที่

Cons:

การดึงข้อมูลซับซ้อน
(ต้อง JOIN หลายตาราง)
ประมวลผลช้า



Pros:

ความเร็วสูงสุดในการ Query
ลดความยุ่งยากของโค้ด (No JOIN)

Cons:

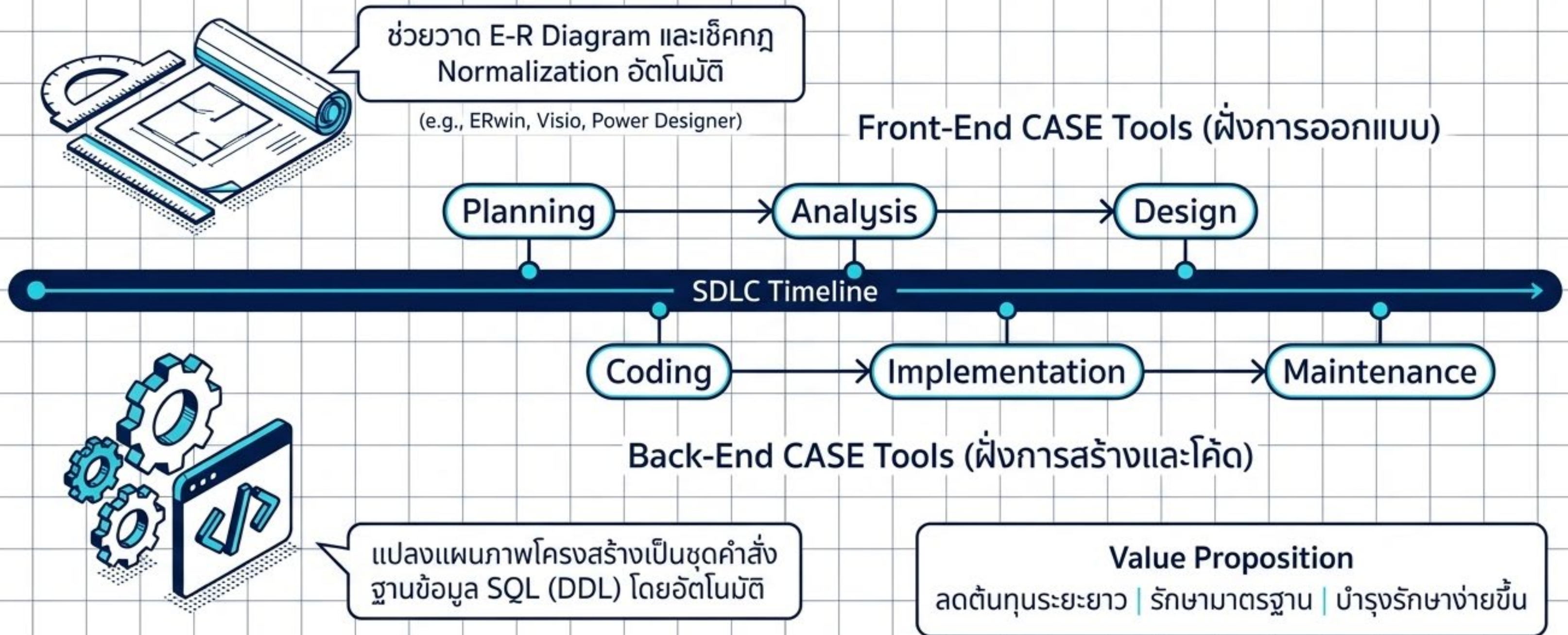
เสี่ยงต่อ Update Anomalies
เปลืองพื้นที่จัดเก็บ

Real-World Example

ยอมฝังข้อมูล PROPERTY_TYPE (เช่น 01=แฟลต, 02=บ้าน) ลงในตารางบ้านพัก เพื่อข้ามการ JOIN ตารางเวลาค้นหา

The Architect's Toolkit: CASE Tools

เครื่องมือระดับมืออาชีพในวงจรการพัฒนาระบบ (SDLC)



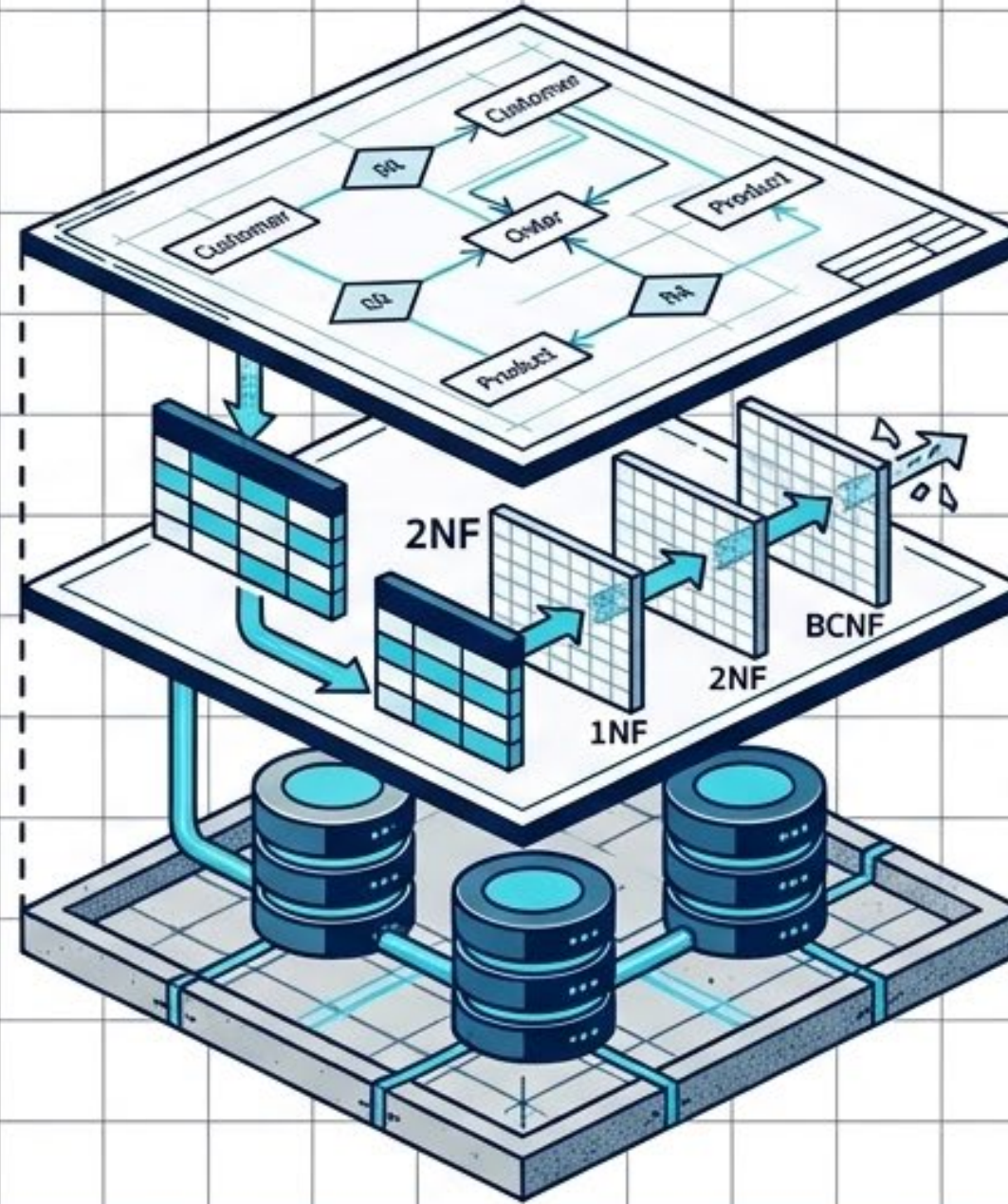
The Holistic Database Architecture

จากแนวคิดสู่ระบบจริง: บทสรุปสถาปัตยกรรมข้อมูล

Conceptual World
(โลกของแนวคิด)

Logical World
(โลกของตรรกะ)

Physical World
(โลกแห่งความจริง)



ความเข้าใจกฎธุรกิจ
และการออกแบบเอนทิตีที่ชัดเจน

ใช้กฎ Functional Dependency
ขจัด Anomalies เพื่อ Data Integrity

ปรับแต่ง Denormalization
เพื่อประสิทธิภาพ สู่ฐานข้อมูลที่ทรงพลัง

Normalization ไม่ใช่แค่การหั่นตาราง แต่มันคือสะพานเชื่อมระหว่างตรรกะของมนุษย์กับประสิทธิภาพของเครื่องจักร