

Full Stack Development Base on Reference Data Model

นำเสนอด้วย นายสิทธิพงศ์ จำรัสฤทธิรงค์
อาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร.วรา วรรवิทย์

เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

01

ศึกษา Full Stack
Cloud Native
(Docker,
Postgres, PHP,
React)

02

เรียนรู้ Data Model
จาก Data Model
Resource Book

03

ออกแบบ Database
ด้วย Visual
Paradigm

04

สร้าง CRUD App
จัดการ Person ด้วย
Database จาก
หนังสือ

เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

05

สร้าง CRUD App จัดการ
Person ด้วย Database
จาก Visual Paradigm

06

สร้าง CRUD App จัดการ
Person ด้วย MVC
(React เป็น View)

07

สร้าง CRUD App
จัดการ Geo Data
(Laravel, Artisan)

08

ใช้ Pandas ETL(extract
transform load) นำ Dataset
Geo จากการปกรองเข้าสู่
app

เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

09

พัฒนา Dynamic
Dropdown
(Country, Province,
District)

10

ปรับปรุง Laravel App
(Response Time: 7s
→ 1s)

11

สร้าง CRUD App
จัดการ Person-
Organization ด้วย
Slim PHP

12

พัฒนา CRUD App
เพิ่ม Role,
Relationship,
Communication
Event

เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

13

สร้าง CRUD App
เดิม เพิ่ม JWT
Authentication
ด้วย FastAPI

14

ปรับ UX/UI ตามคำ
แนะนำจาก
ผู้ช่วยศาสตราจารย์
ภาษิศร์ ณ รังษี
ครุสาขาออกแบบ

15

ปรับปรุง UI ตามข้อเสนอ
แนะจากผู้ใช้ Gen Z

16

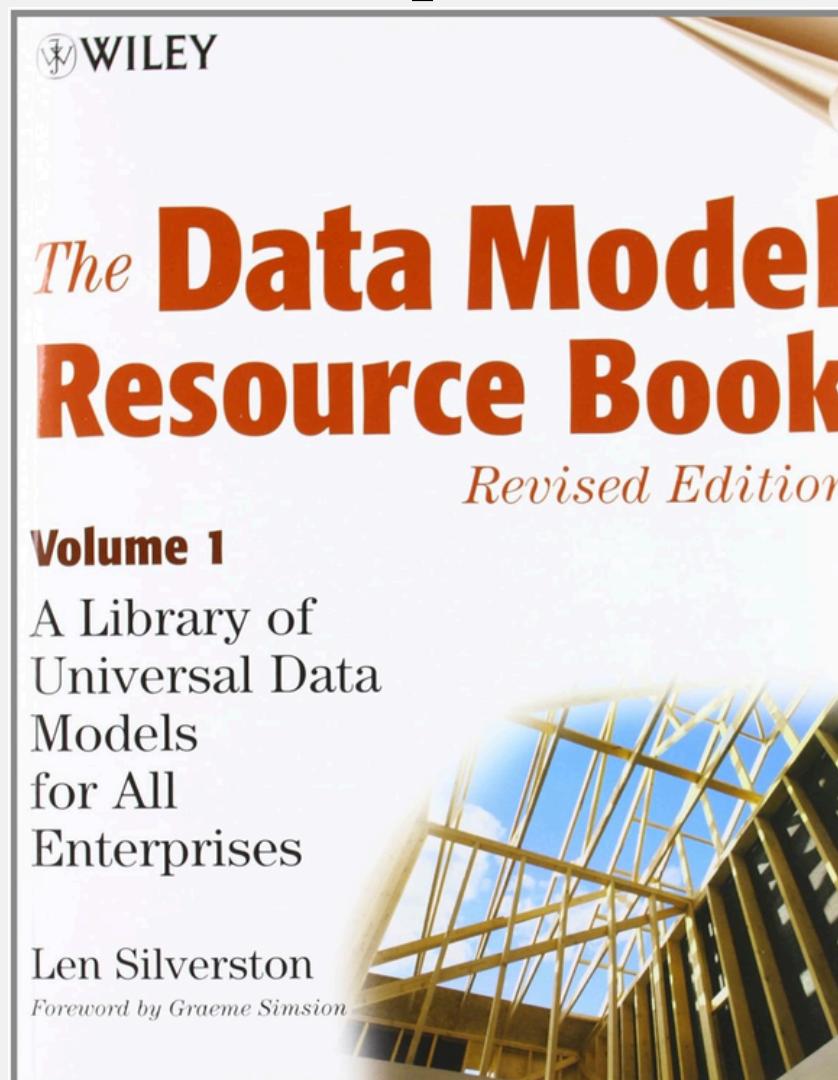
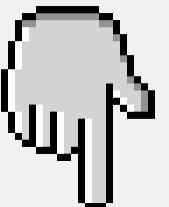
สร้าง Role-Based CRUD
App (6 บทบาท)

เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

17

ออกแบบ Diagram ประกอบซอฟต์แวร์ (Activity, ERD,
Sequence, State, Use Case)

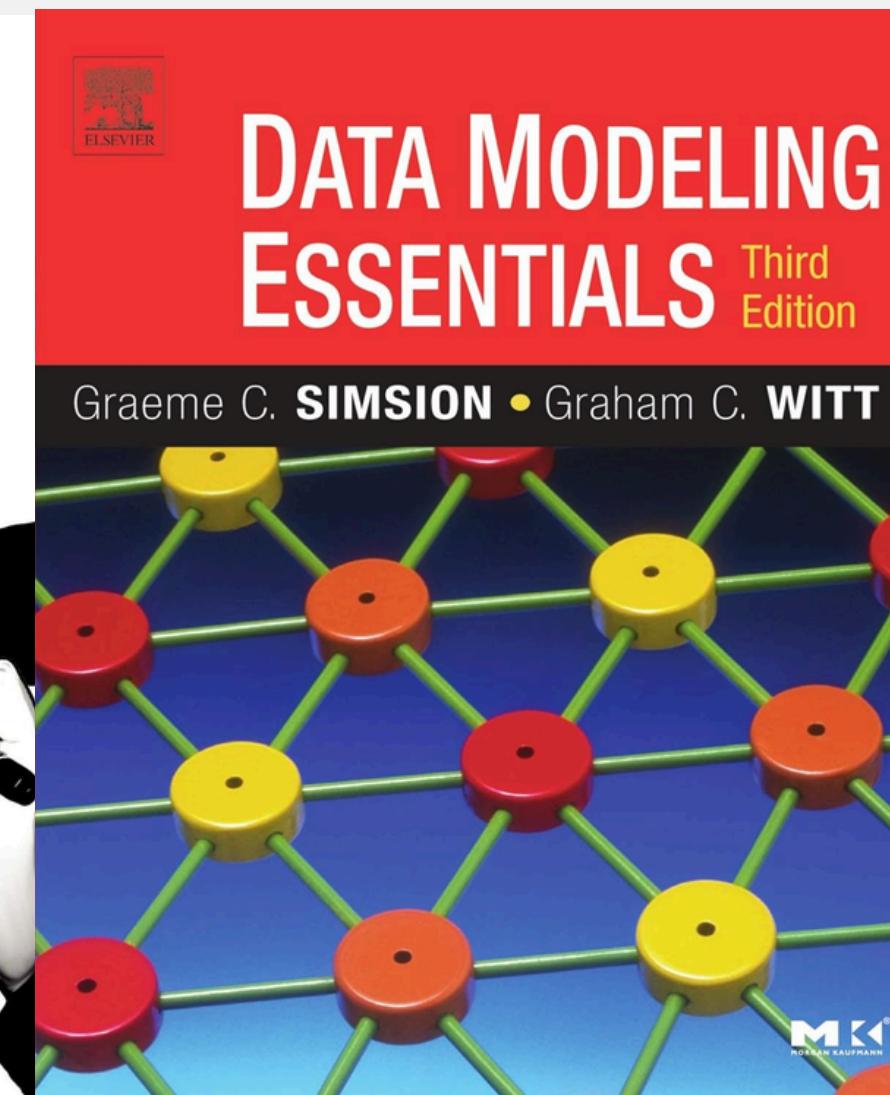
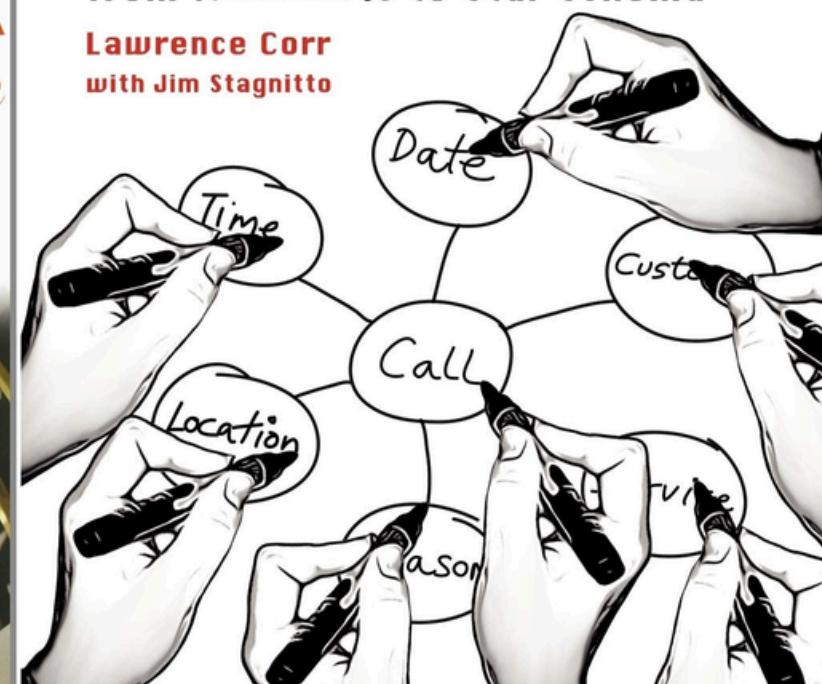
Book About Data Model



Agile Data Warehouse Design

Collaborative Dimensional Modeling,
from Whiteboard to Star Schema

Lawrence Corr
with Jim Stagnitto

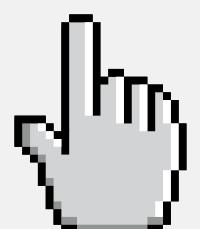


NOSQL AND SQL DATA MODELING

Bringing Together Data, Semantics, and Software



TED HILLS



Data Model

- Data คือข้อมูล
- Model คือรูปปั้น แบบจำลอง
- Data Model คือการสร้างแบบจำลองของข้อมูลในระบบ
- เมื่อ Developer ในทีมเห็นโครงสร้างชัดเจนก็จะเข้าใจข้อมูลในระบบมากขึ้น

Database

- ระบบจัดเก็บข้อมูลที่มีความสัมพันธ์ (Relation)
- ลดความซ้ำซ้อนของข้อมูลในระบบ
- ป้องกันการขัดแย้งของข้อมูล
- ช่วยให้ค้นหาและอัปเดตข้อมูลได้ง่ายและแม่นยำ

Web 1.0

- เว็บยุคแรกแบบ static เน้นอ่านข้อมูลอย่างเดียว
- ผู้ใช้แค่ดู ไม่มีโต้ตอบหรือสร้างเนื้อหา
- หน้าเว็บเรียบง่าย เหมือนหนังสือออนไลน์

Web 2.0

- เว็บแบบ dynamic ให้ผู้ใช้โต้ตอบและสร้างเนื้อหา
- มี social media, blog, แชร์ข้อมูลกันได้
- ใช้ AJAX (Asynchronous JavaScript and XML) ทำหน้าเว็บอัปเดตแบบเรียลไทม์
- ตัวอย่างคือ Facebook, YouTube ผู้ใช้กระทำกับ web ได้

Evolution of Web 2.0

- เริ่มจาก SSR สร้างหน้าเว็บที่เซิร์ฟเวอร์
- เปลี่ยนเป็น CSR ประมวลผลหน้าเว็บผู้ใช้
- CSR ผลักภาระการ render ไปให้เครื่องผู้ใช้
- ส่งเฉพาะ JSON ไม่ต้องส่ง HTML ทั้งหน้า

MVC

- แบ่งงานเป็น Model, View, Controller
- Model จัดการข้อมูล
- View แสดงผลหน้าตาให้ผู้ใช้
- Controller ควบคุมการโต้ตอบ

General Framework

- โครงสร้างพื้นฐานสำหรับพัฒนาแอป
- ให้เครื่องมือและไลบรารีใช้งานง่าย
- เช่น Django, Spring สำหรับงานทั่วไป
- ลดเวลาและความซับซ้อน

Backend

- application ที่ติดต่อกับ Frontend และ Database
- หน้าที่เป็นส่วนเชื่อม Frontend และ Database
- เปรียบเทียบได้กับเด็ก Serv

Frontend

- แอปพลิเคชันหรือเว็บไซต์ที่ผู้ใช้เห็นและโต้ตอบได้โดยตรง
- เช่น การออกแบบหน้าตาเว็บ (UI) การจัดวางข้อมูล ปุ่มกด กล่องข้อความ ตาราง แสดงผล sidebar ข้อความที่แสดงผล ลูกเล่น animation ต่างๆ บนเว็บ
- เปรียบเทียบได้กับ หน้าร้านอาหารที่ลูกค้านั่งกินข้าว

Application-Specific Framework

- เฟรมเวิร์กสำหรับงานเฉพาะ
- เช่น CMS (WordPress) สำหรับเว็บเนื้อหา
- Google Sites สำหรับสร้างเว็บง่ายๆ
- Shopify สำหรับร้านออนไลน์
- ปรับให้เหมาะสมกับเป้าหมายของ

Frontend Framework

- เพرمเวิร์กสำหรับสร้าง UI และการโต้ตอบ
- จัดการหน้าเว็บแบบ dynamic
- เช่น React, Vue.js, Angular
- ทำให้หน้าเว็บตอบสนองและใช้งานง่าย

Backend Framework

- เพرمเวิร์กสำหรับจัดการ logic ฝั่งเซิร์ฟเวอร์
- ช่วยพัฒนาการทำงานของฐานข้อมูลและ API
- เช่น Node.js, Django, Ruby on Rails
- ทำให้ระบบหลังบ้านเร็วและปลอดภัย

Framework Stack (Tech Stack)

- ชุดเทคโนโลยีที่ใช้พัฒนาแอปหรือเว็บ
- รวม frontend, backend และฐานข้อมูล
- เช่น React + Node.js + MongoDB
- เลือก stack ให้เหมาะสมกับเป้าหมายโปรเจกต์

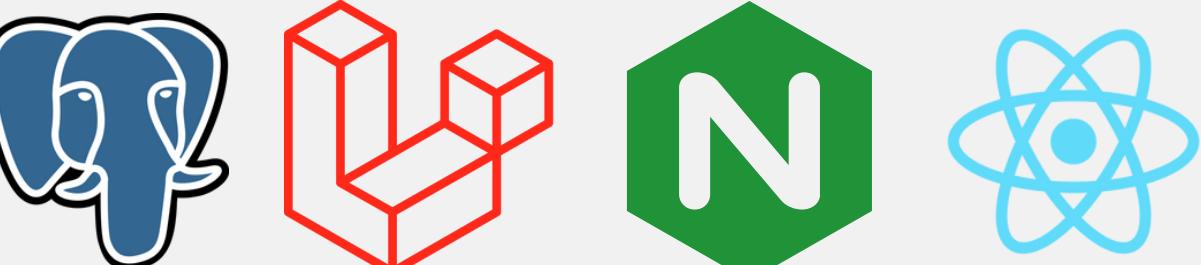
Tech Stack Timeline



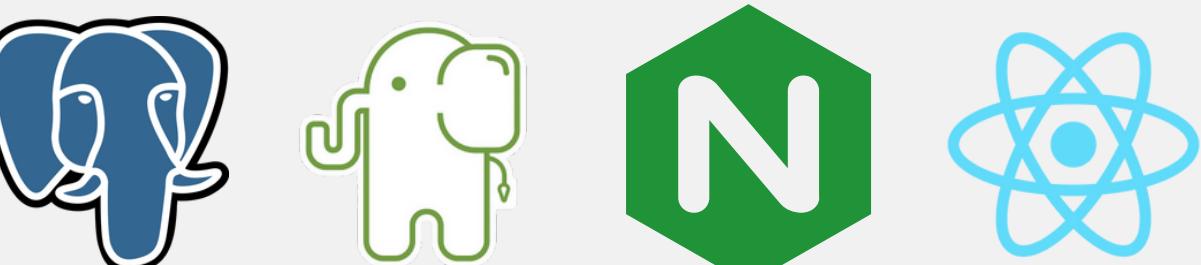
Postgres + Native PHP + React JS + Apache



Postgres + Laravel PHP + React TS + Nginx



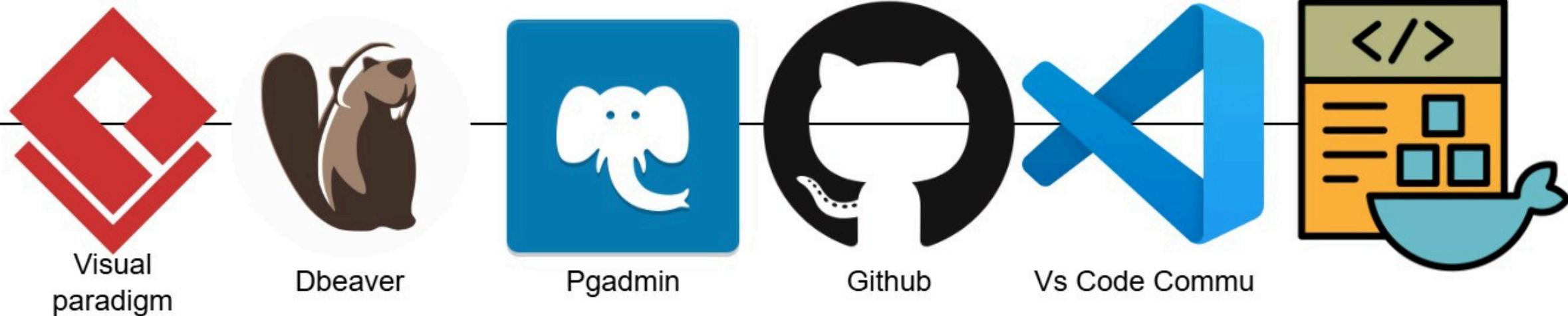
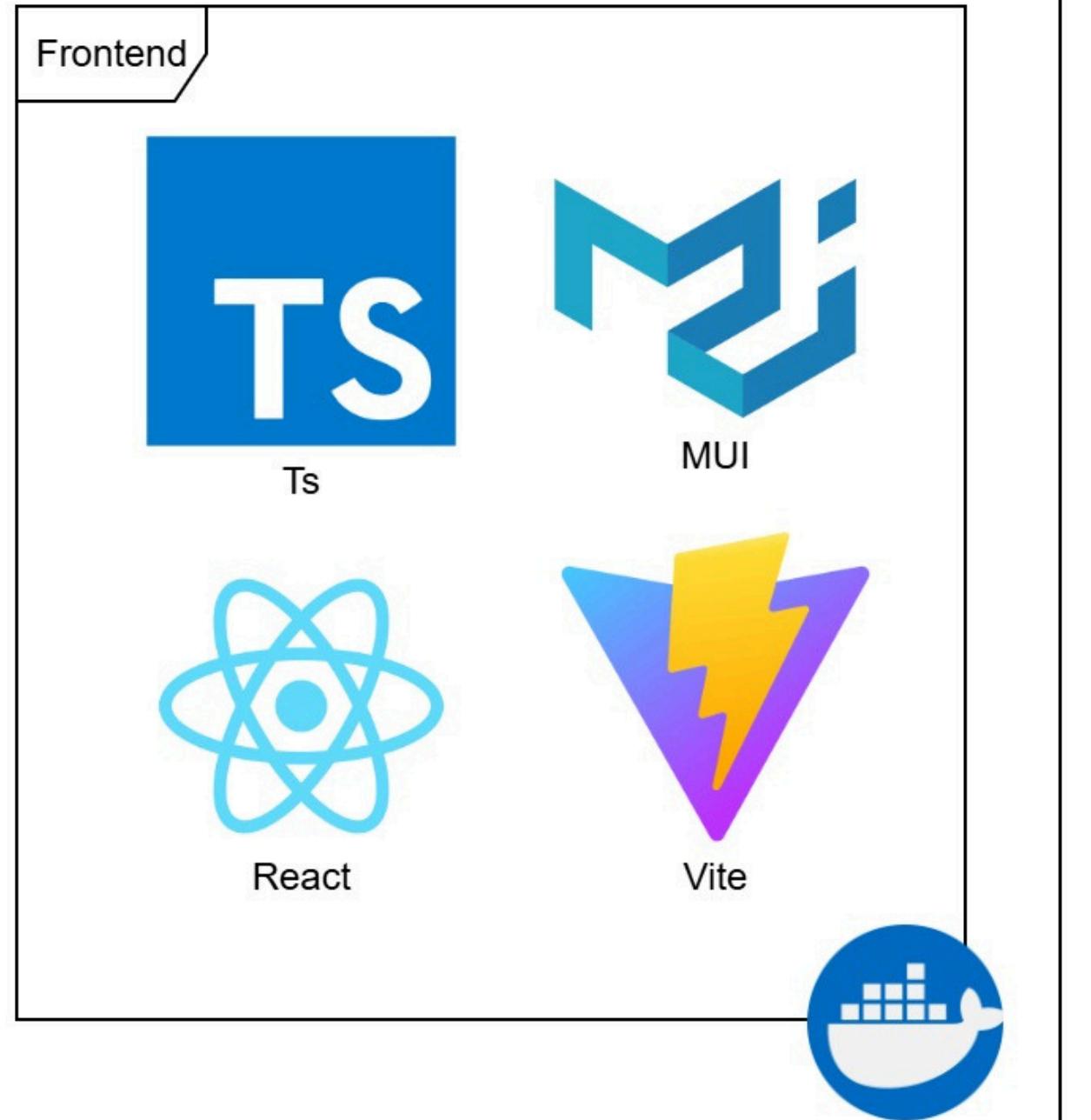
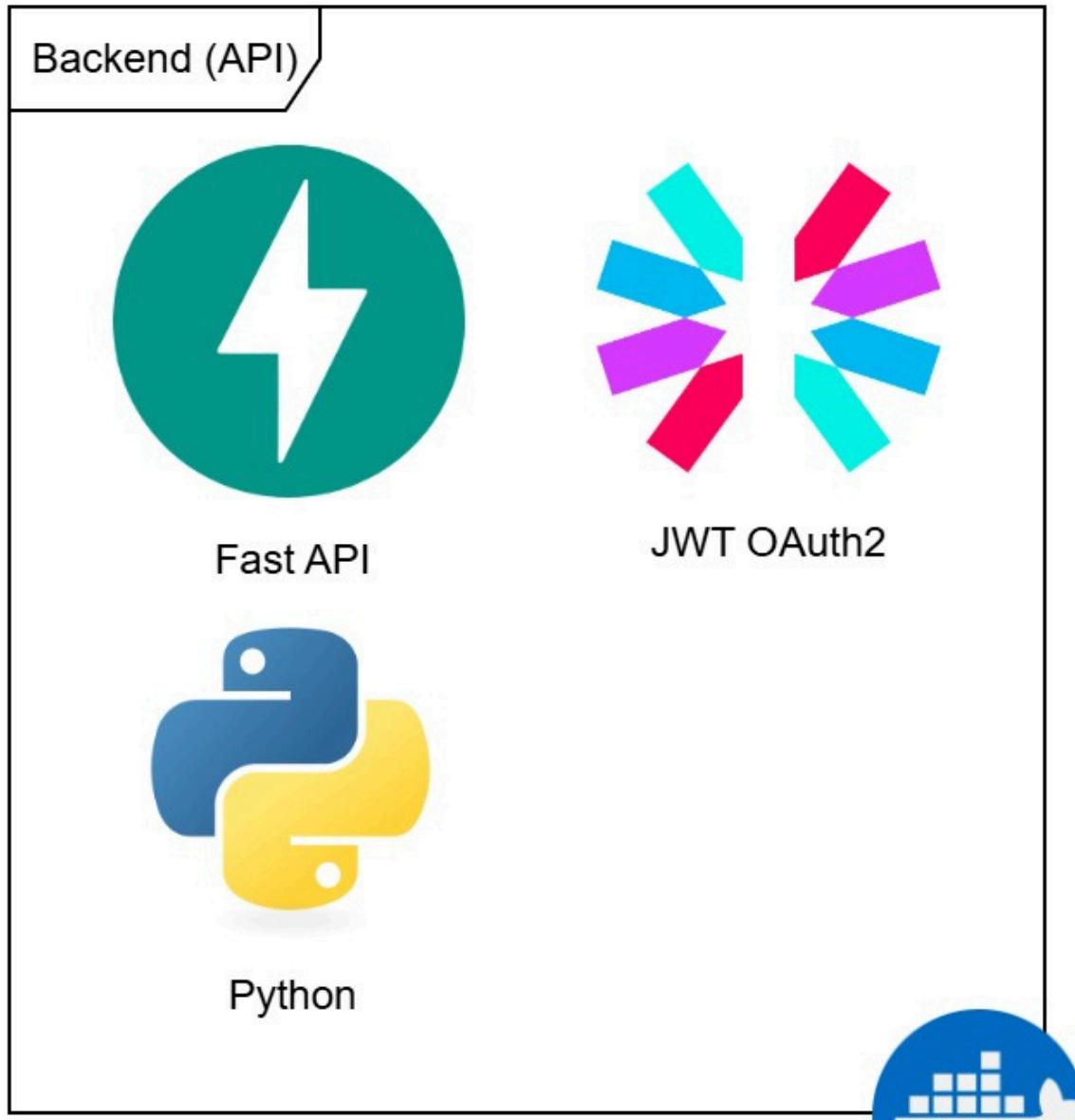
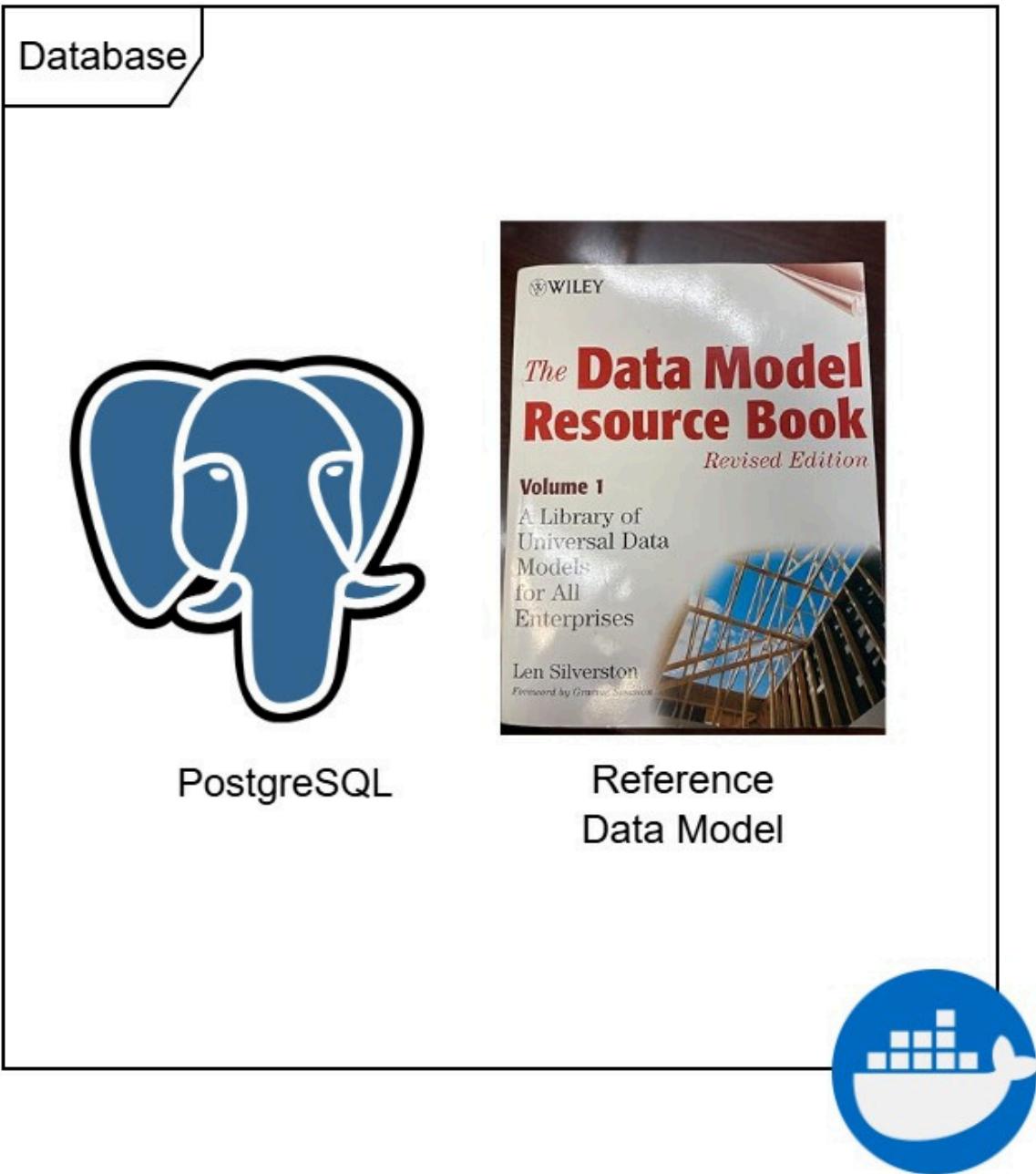
Postgres + Slim PHP + React TS + Nginx



Postgres + FastAPI Python + React TS + Uvicorn



Docker Compose



แนวคิดของโครงงาน

- ใช้ Party Model จากหนังสือสร้างแอป
- CRUD Base Layer เชื่อมบุคคลและองค์กร
- CRUD Activity Layer เชื่อมการสื่อสารระหว่างบุคคล/องค์กร
- แอป Cloud Native พร้อม Authen ด้วย JWT และ OAuth 2.0

เลือก Party Model จากหนังสือมาต่อยอด

60 Chapter 2

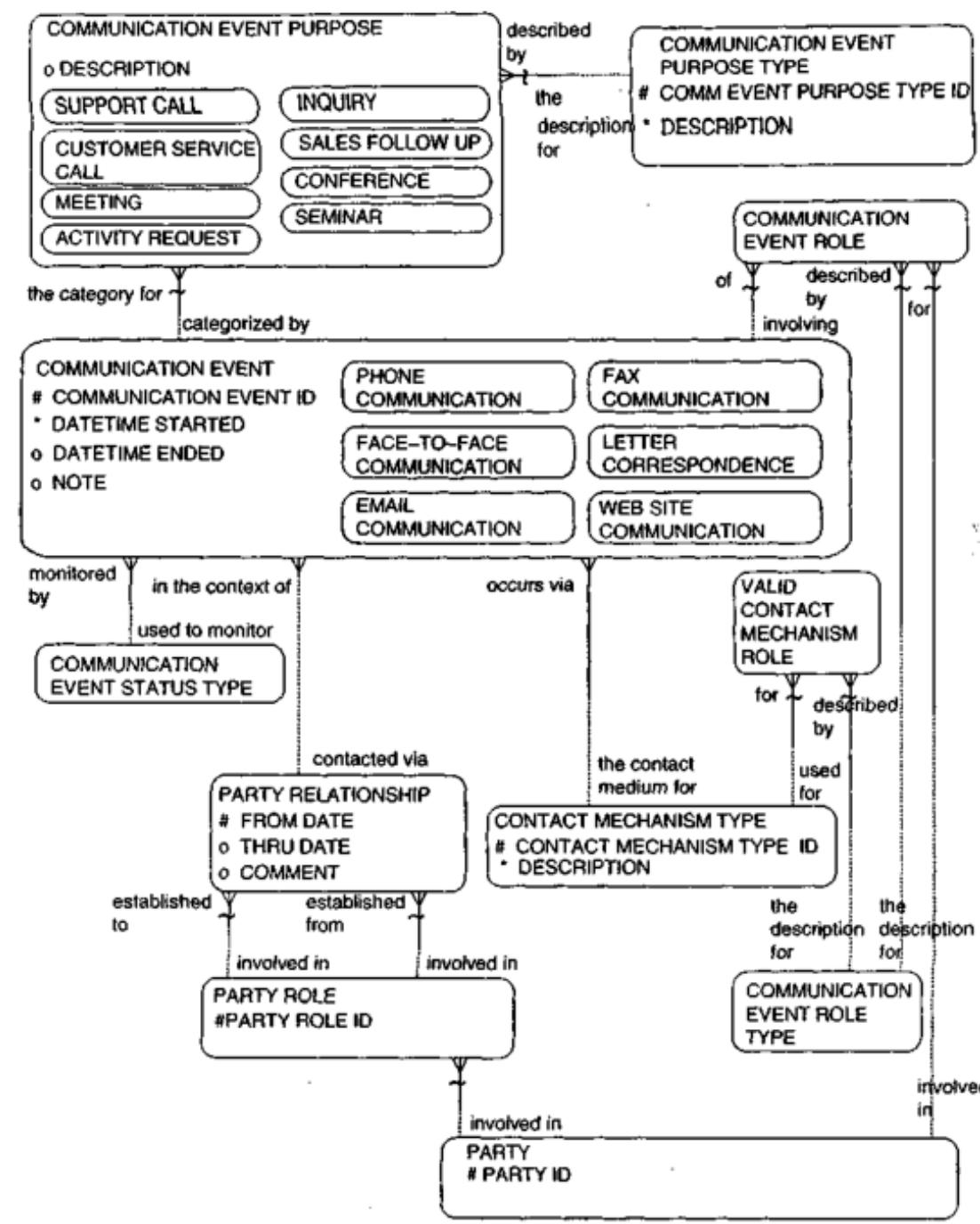


Figure 2.12 Communication event.

30 Chapter 2

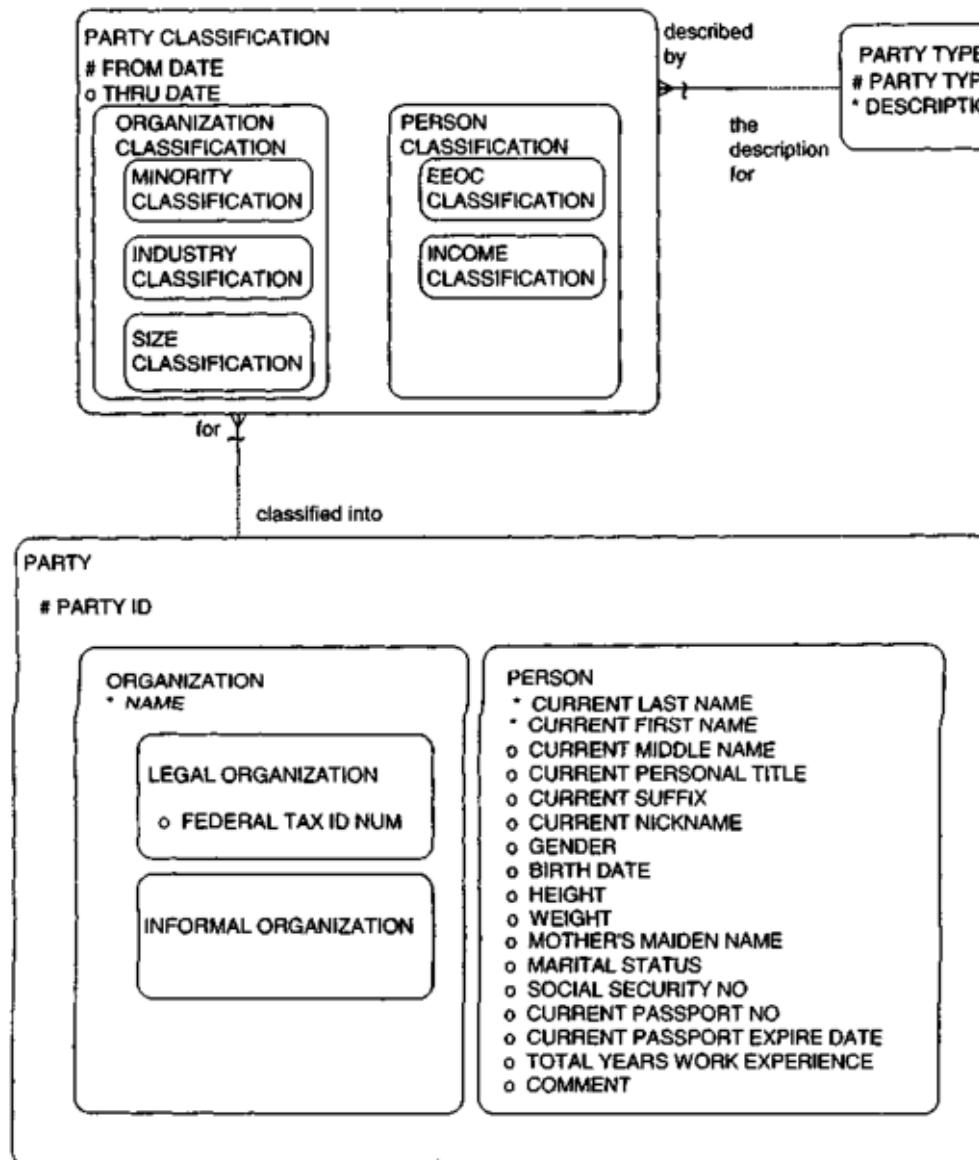


Figure 2.3 Party

Therefore, Figure 2.3 shows a superentity named PARTY that has as its two subtypes PERSON and ORGANIZATION. This PARTY entity will enable storage of some of the common characteristics and relationships that people and organizations share.

People and Organizations 43

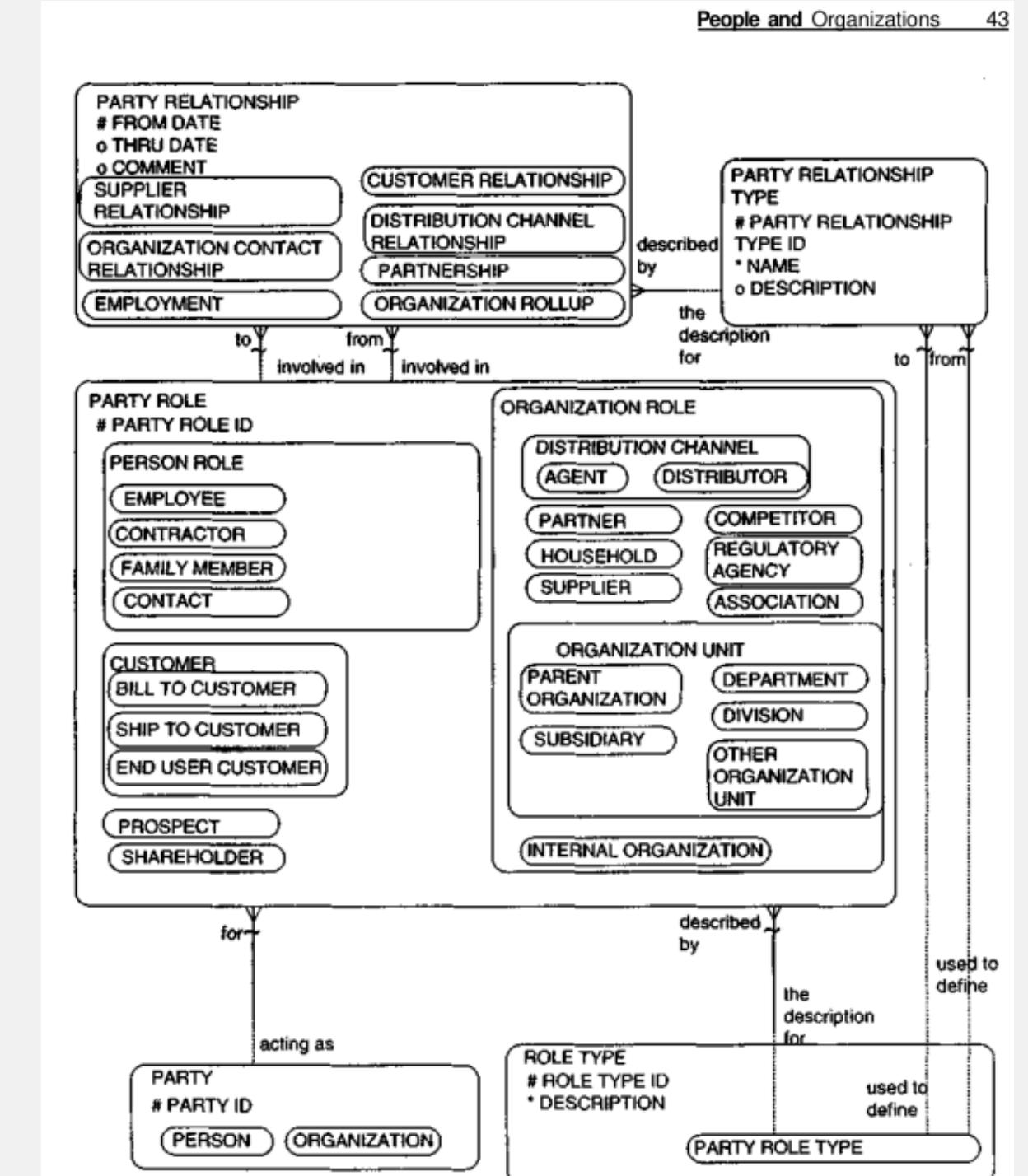
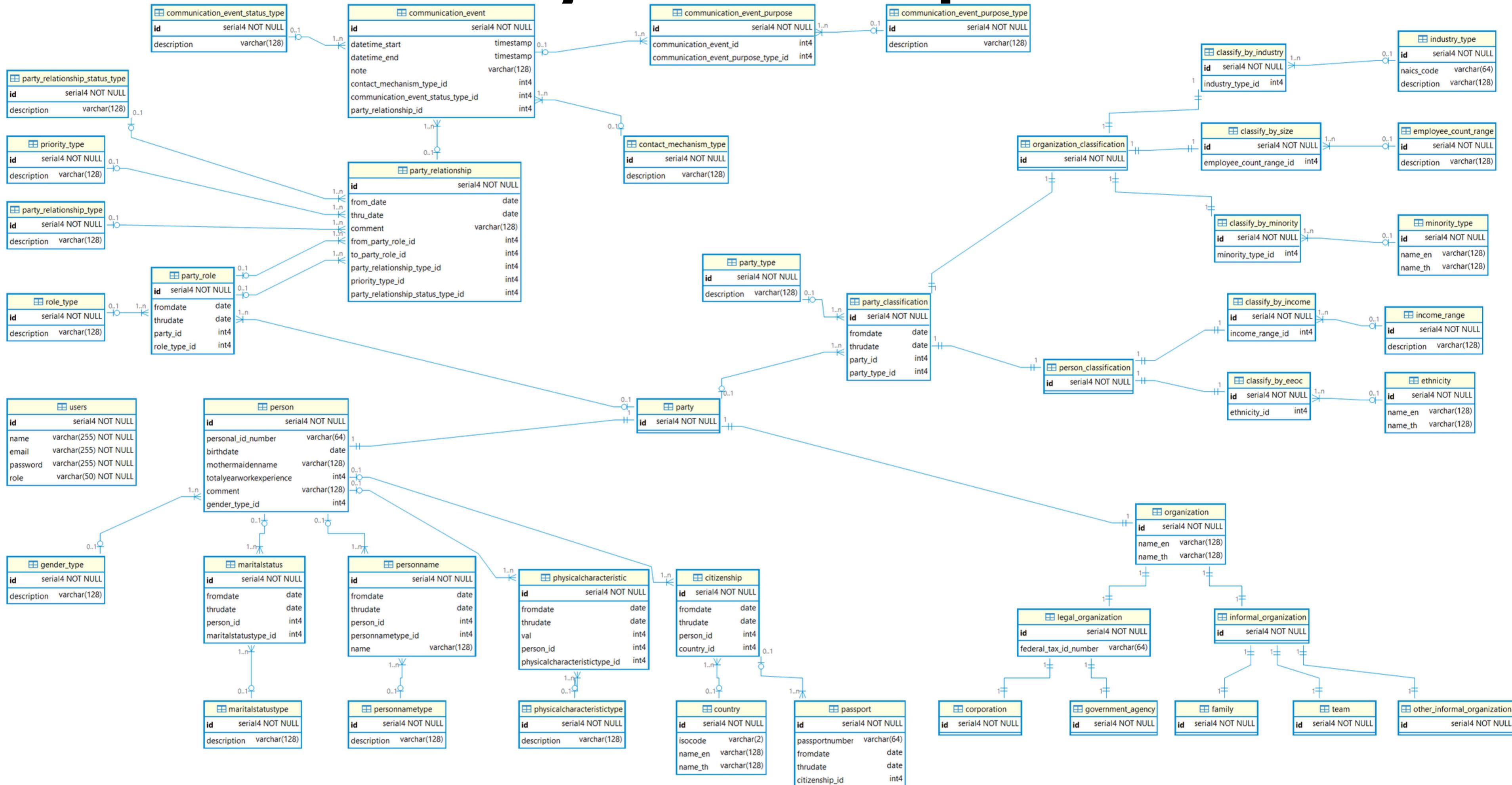
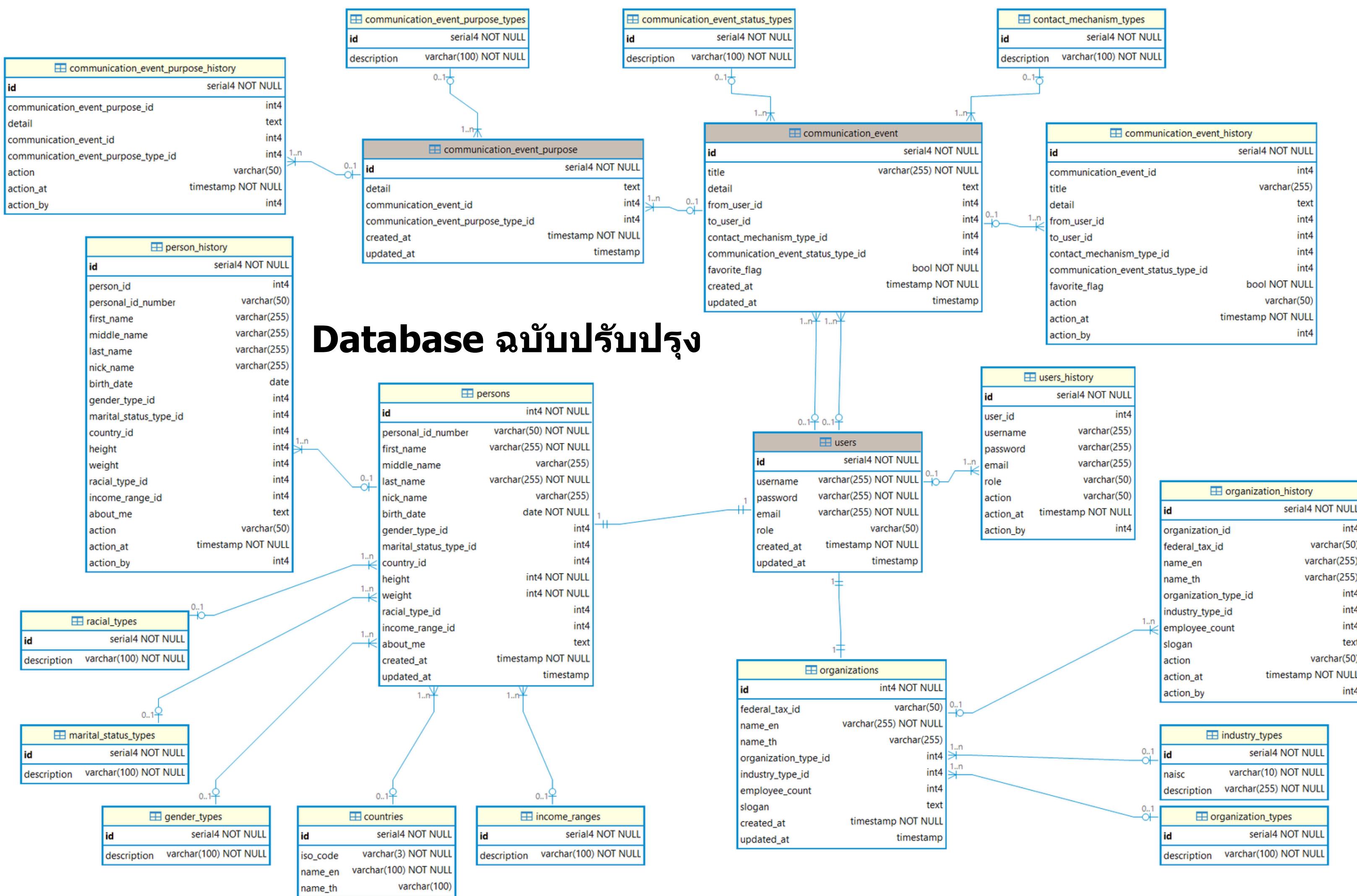


Figure 2.6a Common party relationships

ນຳ Party Model ມາ implement





ปัญหาที่เจอ

- ออกรอบ Database ให้ต่อรองง่ายและมีประสิทธิภาพมาก
- สร้าง Full Stack Cloud Native App ขั้นตอน ต้องเรียนรู้แยกส่วน
- Data Model Resource Book เก่า อ่านยาก เข้าใจลำบาก
- JWT Authentication สำหรับ 6 บทบาทผู้ใช้จัดการยาก
- จัดการ hashed password ในระบบ login ขั้นตอน
- ออกรอบ UX/UI ใช้งานง่ายสำหรับ Gen Z ท้าทาย
- จัดการข้อมูลวันเวลาอยุ่งยาก
- เก็บประวัติข้อมูลว่าคร่าทำอะไร เมื่อไหร เป็นเรื่องขั้นตอน

แนวทางต่อยอด

- พัฒนา Full Stack App ให้รองรับทุกภาษา
- ใช้ Reference Data Model อีนๆ ที่ชัดเจนและมีประสิทธิภาพมากกว่า
- UX/UI ตั้งค่าเปลี่ยน theme เว็บได้ตามต้องการ
- เพิ่ม AI/ML วิเคราะห์ Communication
- ขยายสู่ Mobile App (ใช้ React Native แบบ via bus)
- นำ App ที่ได้ไป Deploy ในระบบ Cloud Computing อีนๆ

Q

&

A