สยามชำนาญกิจ
SIAM CHAMNAN KIT

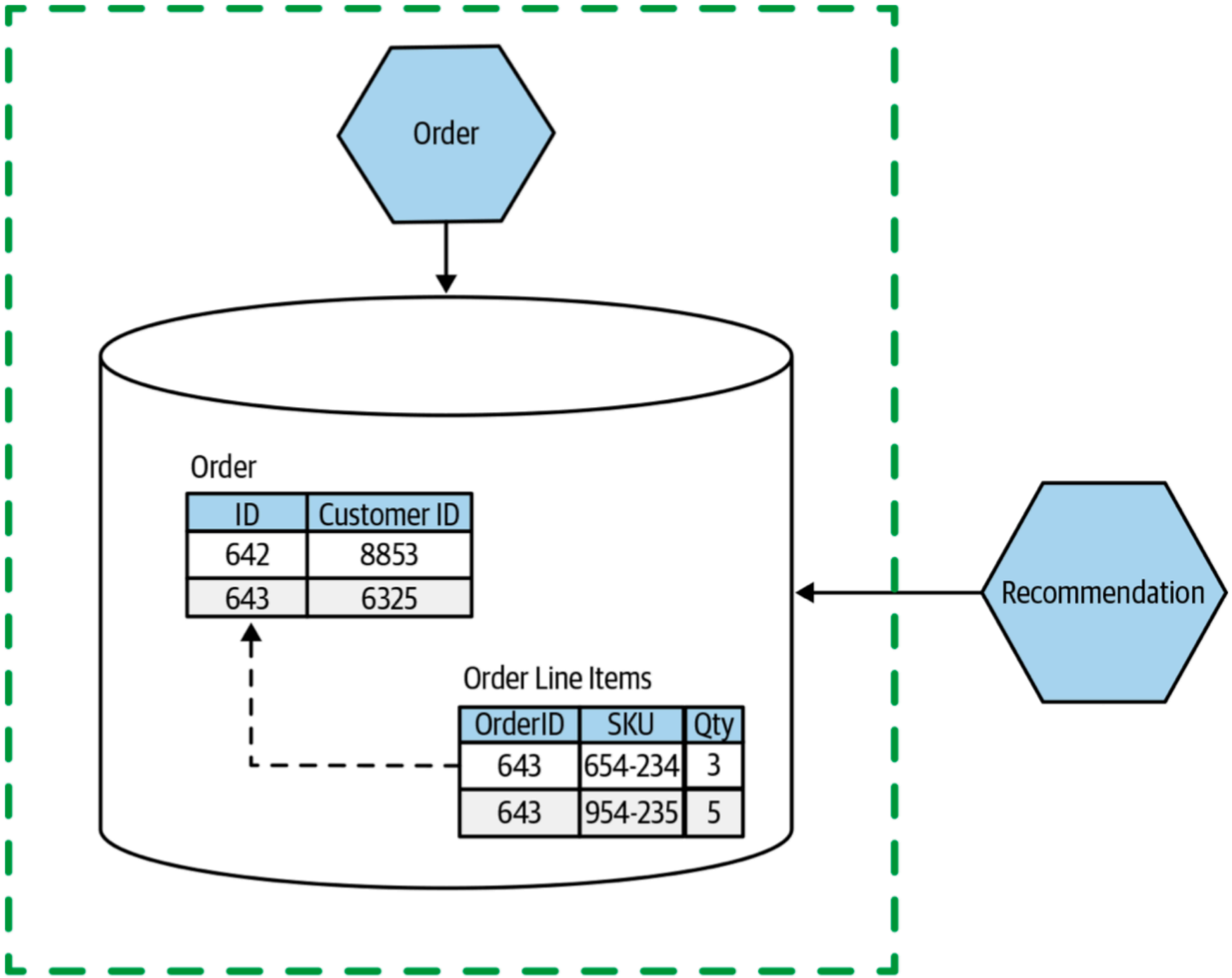
Microservices

Goals

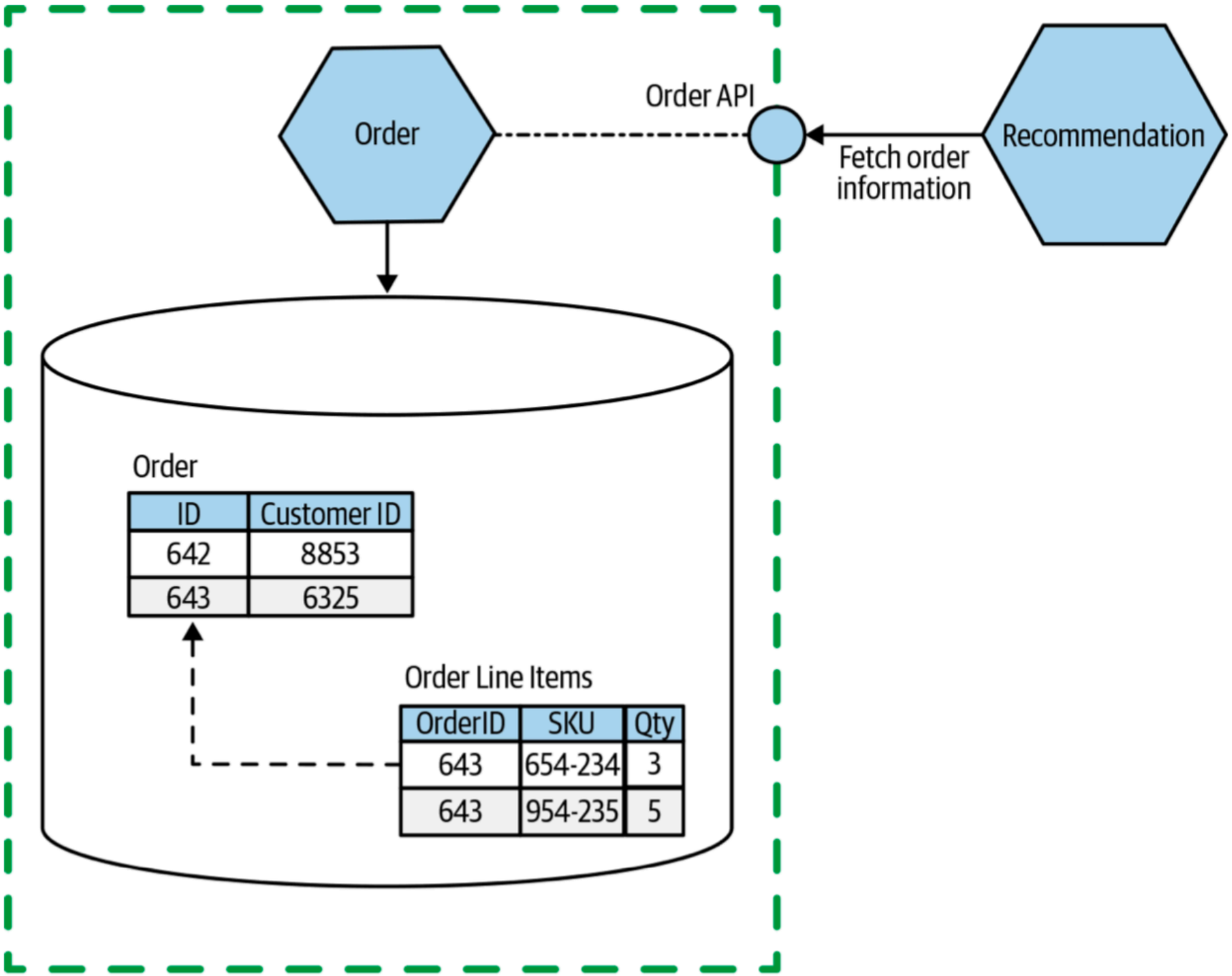
ตำแหน่งของคุณในองค์กร (Job Position)	เป้าหมายในการเข้าเวิร์คชอปในครั้งนี้
Technical Specialist	ประยุกต์ใช้ในงาน
Engineer	เรียนรู้เกี่ยวกับ Microservices
Project Manager	วางแผนการใช้และออกแบบระบบ microservice ของบริษัท
Database Administrator, ดูแล api ทั้งหมดของหน่วยเองและ api ที่จะพัฒนาให้ลูกค้า	นำความรู้ที่ได้มาวางแผนการทำงานของหน่วย
ว่างงาน	หาความรู้เพิ่มเติม
ผู้ช่วยนักวิจัย	นำมาต่อยอดระบบในที่ทำงาน
.NET backend developer	ทำความเข้าใจการออกแบบระบบในรูปแบบ microservices ที่ถูกต้อง เพื่อเสริมความรู้ และประยุกต์ใช้ในการออกแบบระบบสำหรับ product ของบริษัทได้
ผู้ช่วยวิจัย (Developer)	ทราบ micro service design ที่เหมาะสม (Best Practice) เพื่อนำมาวางแผนใช้งาน/ปรับปรุง งานที่ทำอยู่

Coupling

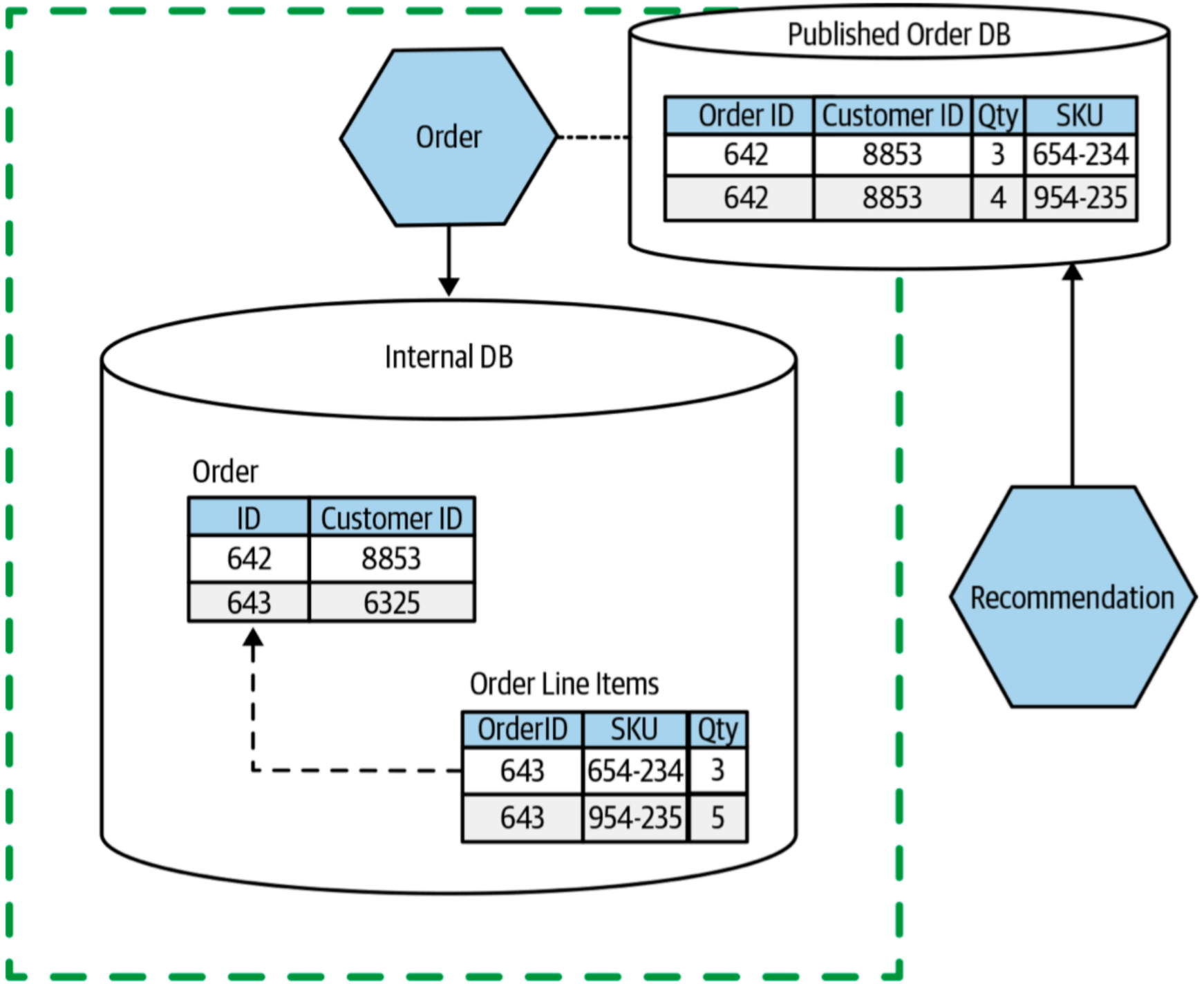
Direct access to database



Access data from API

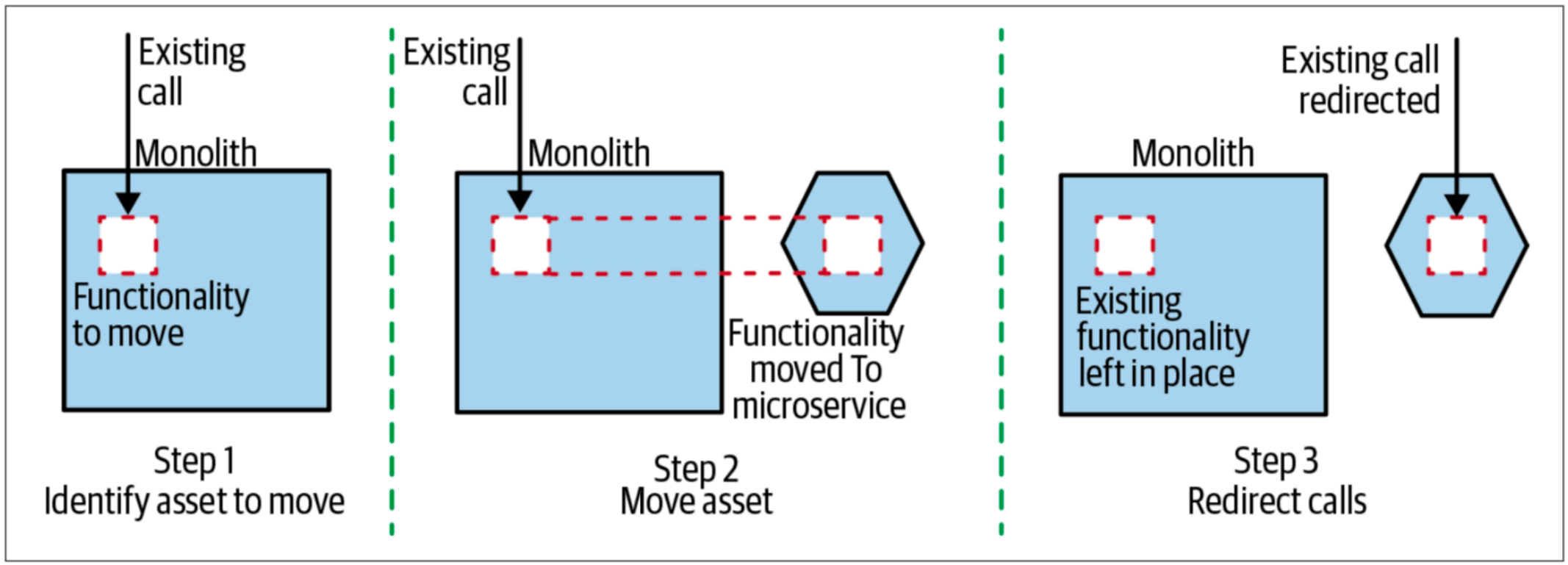


Published database

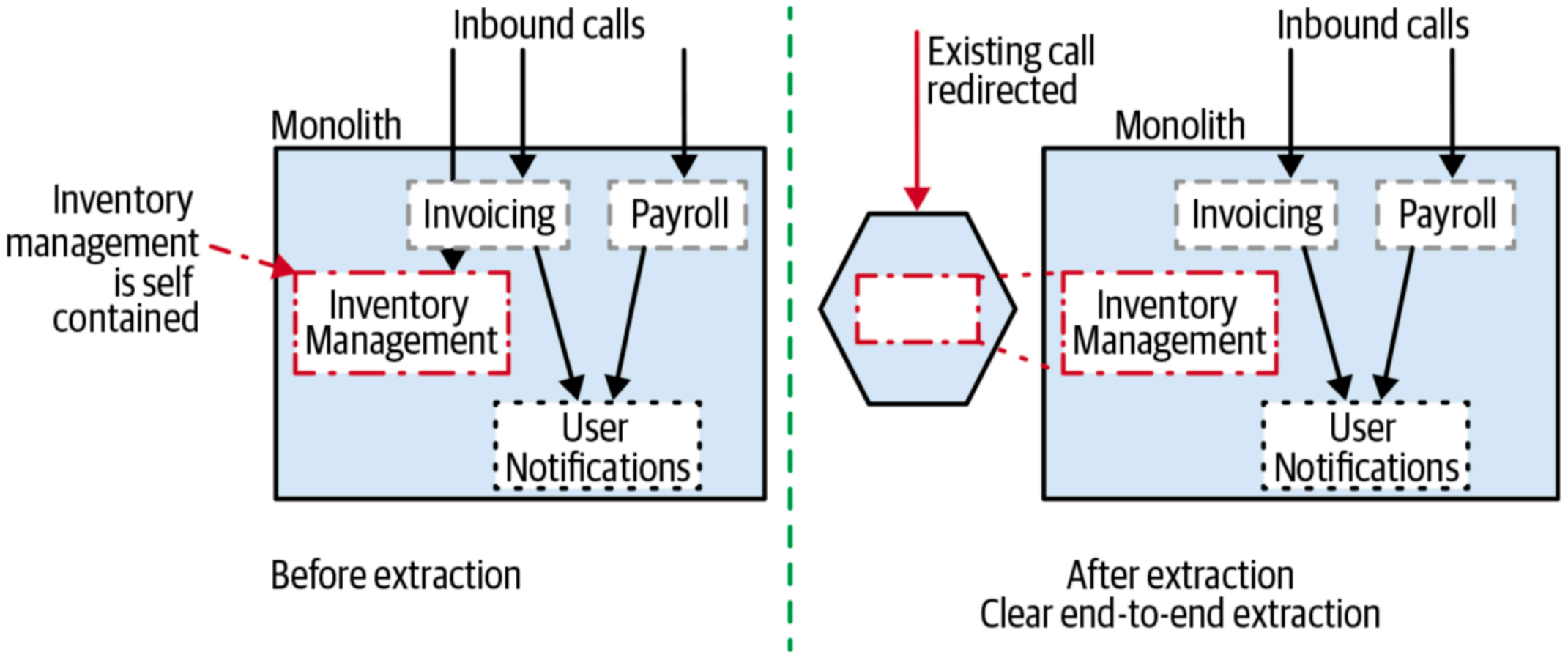


Migration pattern

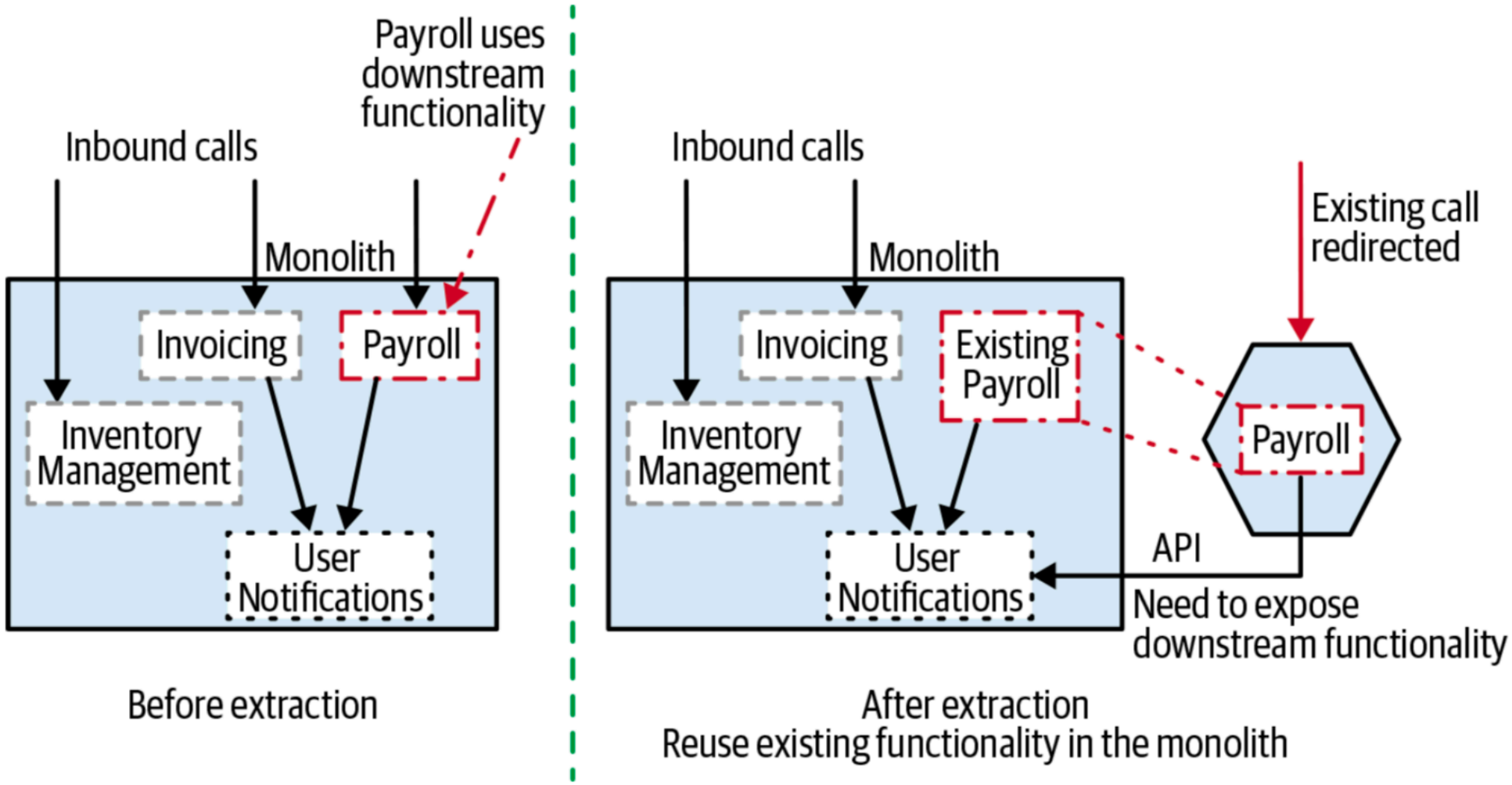
Strangler pattern



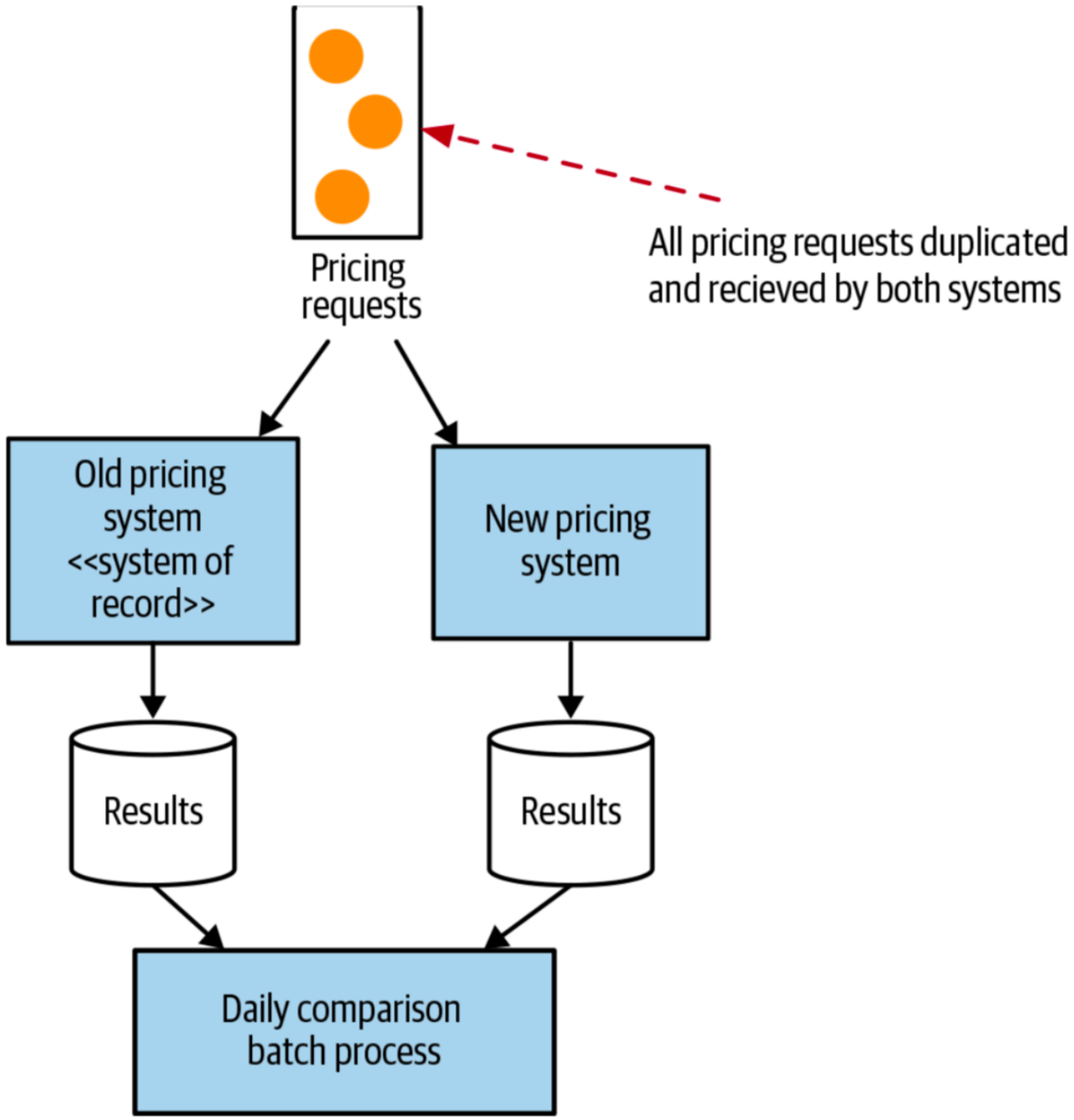
Strangler pattern



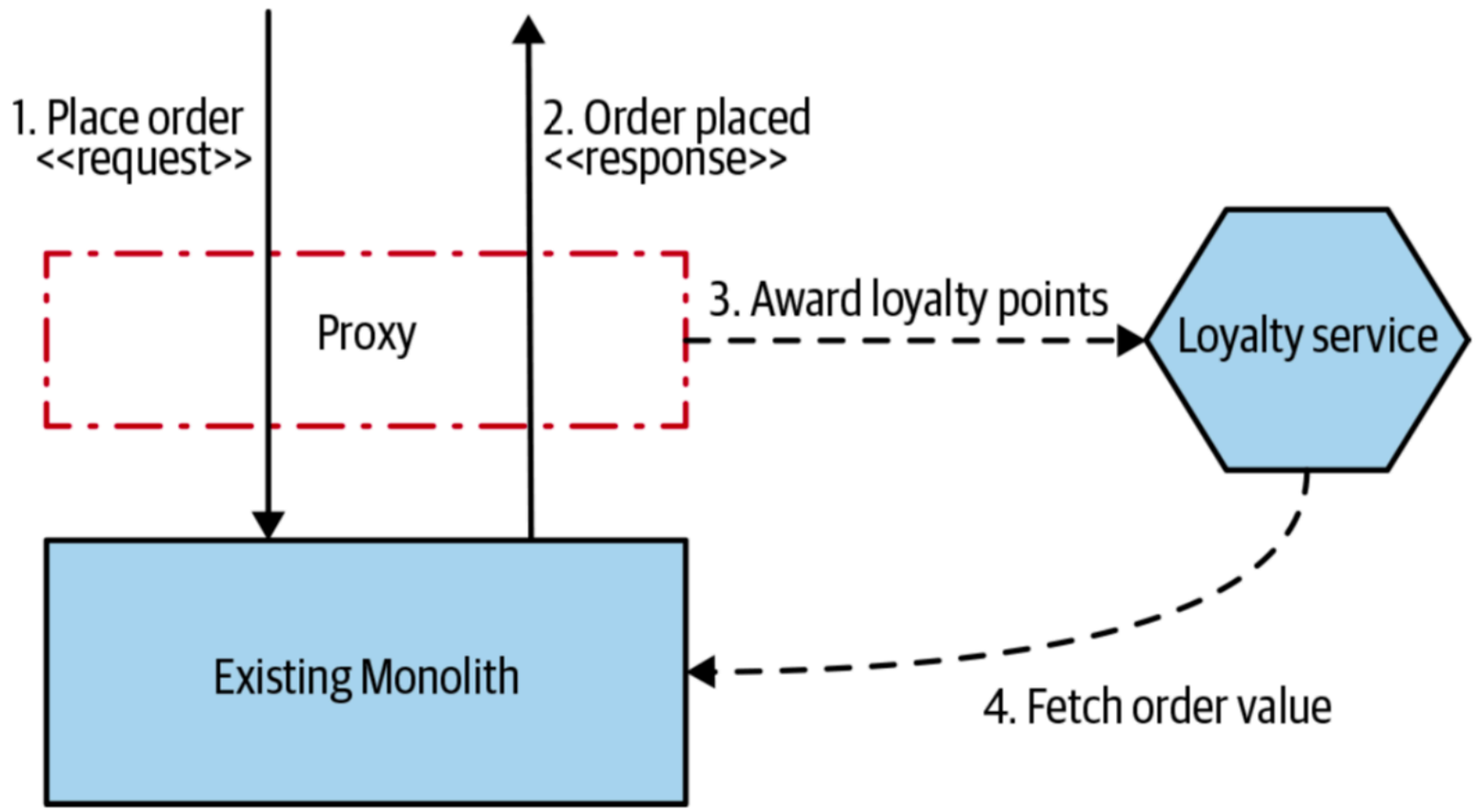
Strangler pattern



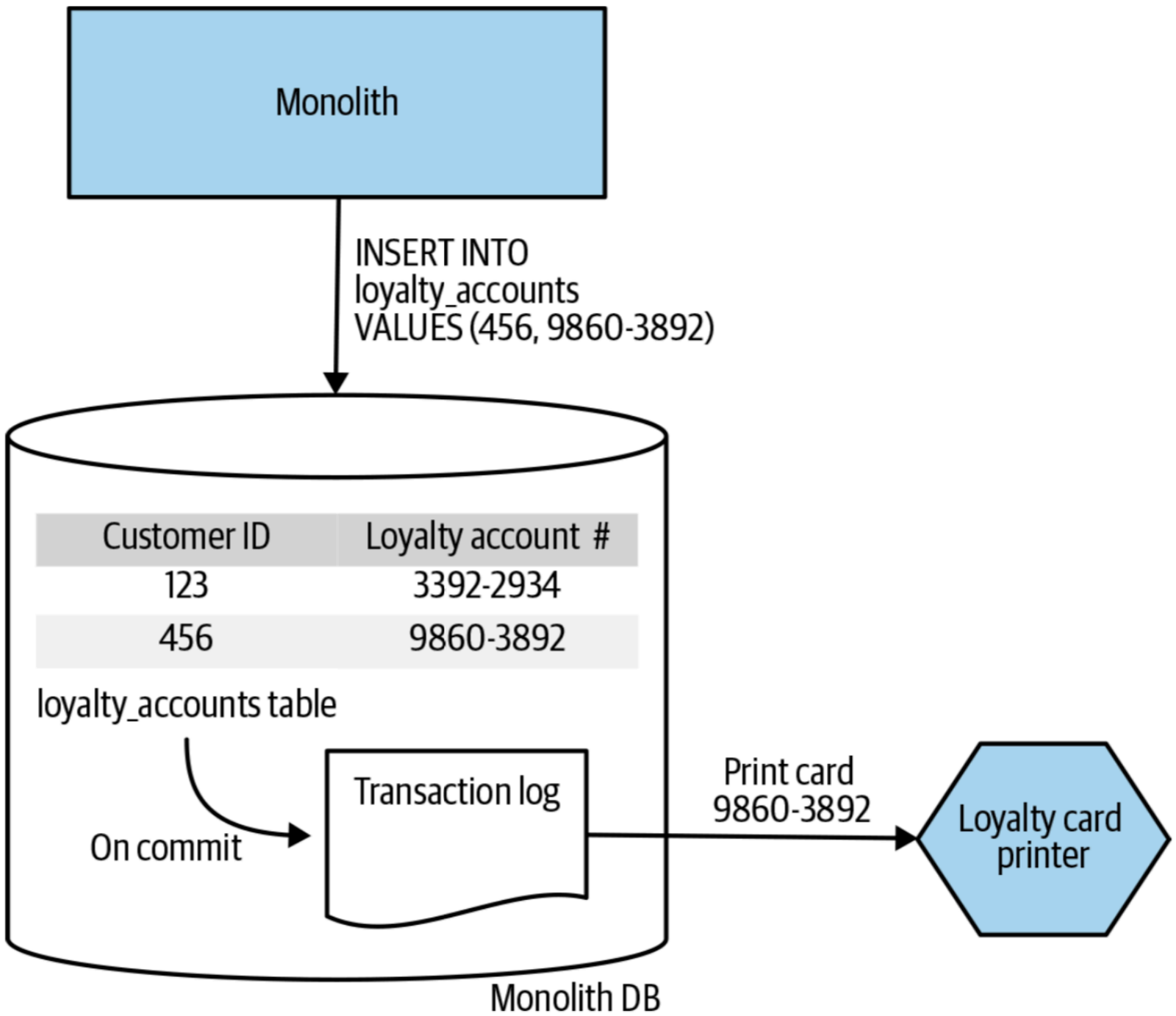
Parallel Run



Decorate pattern

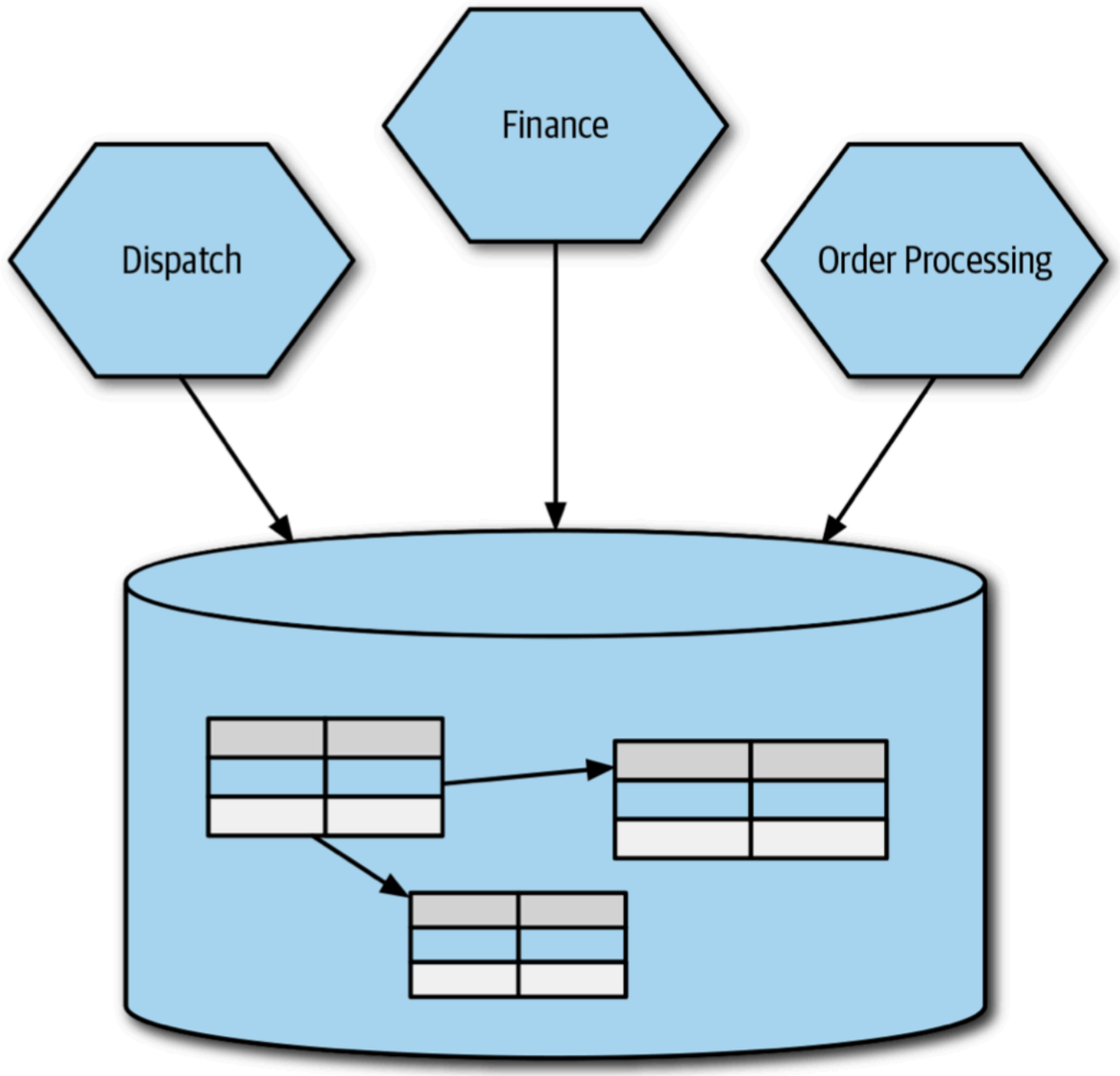


Change data capture

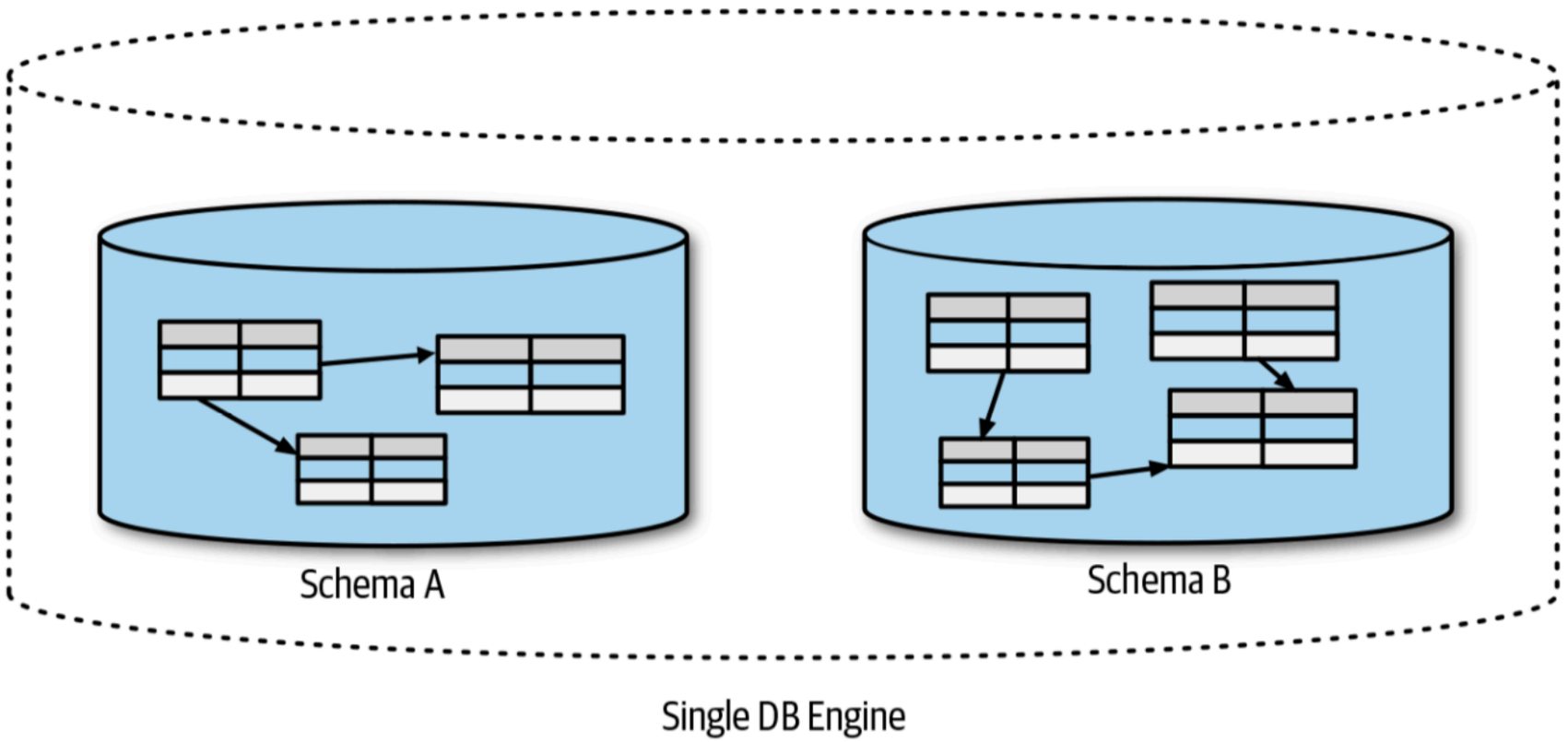


Decompose database

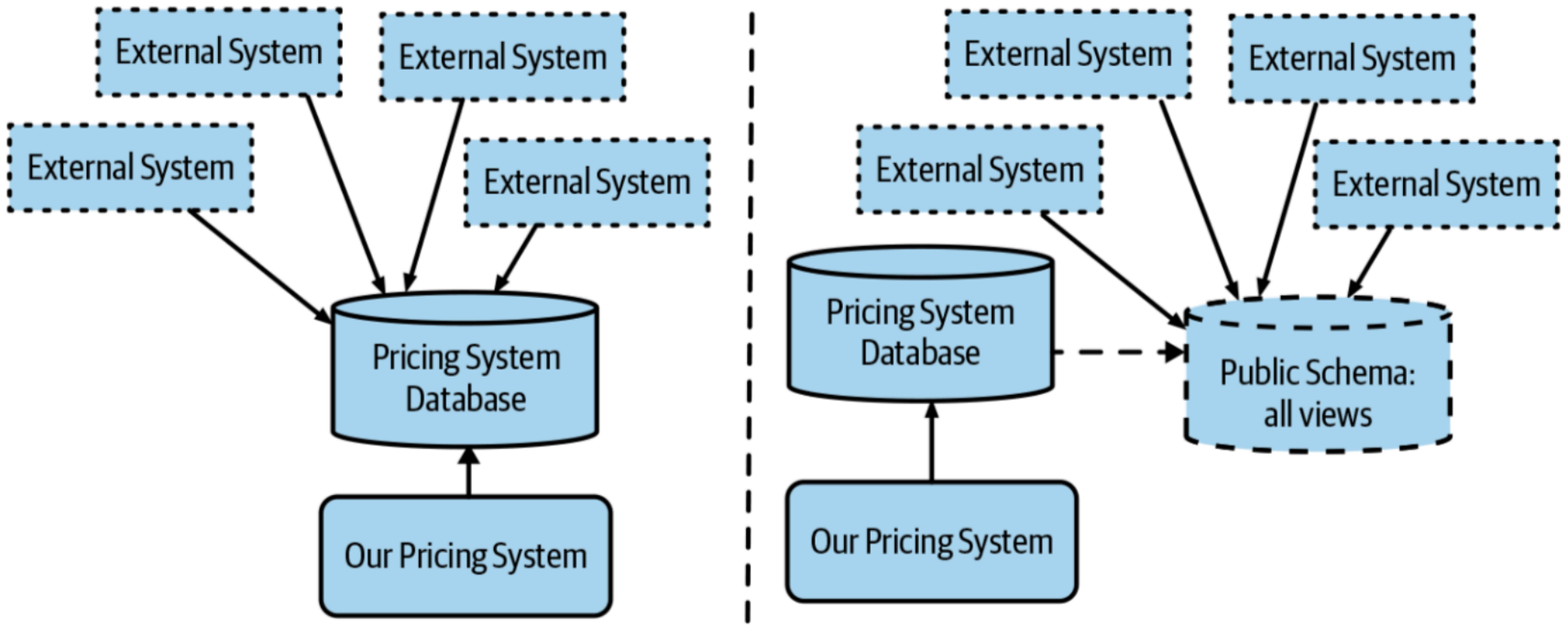
Shared database



Single database, multiple schema



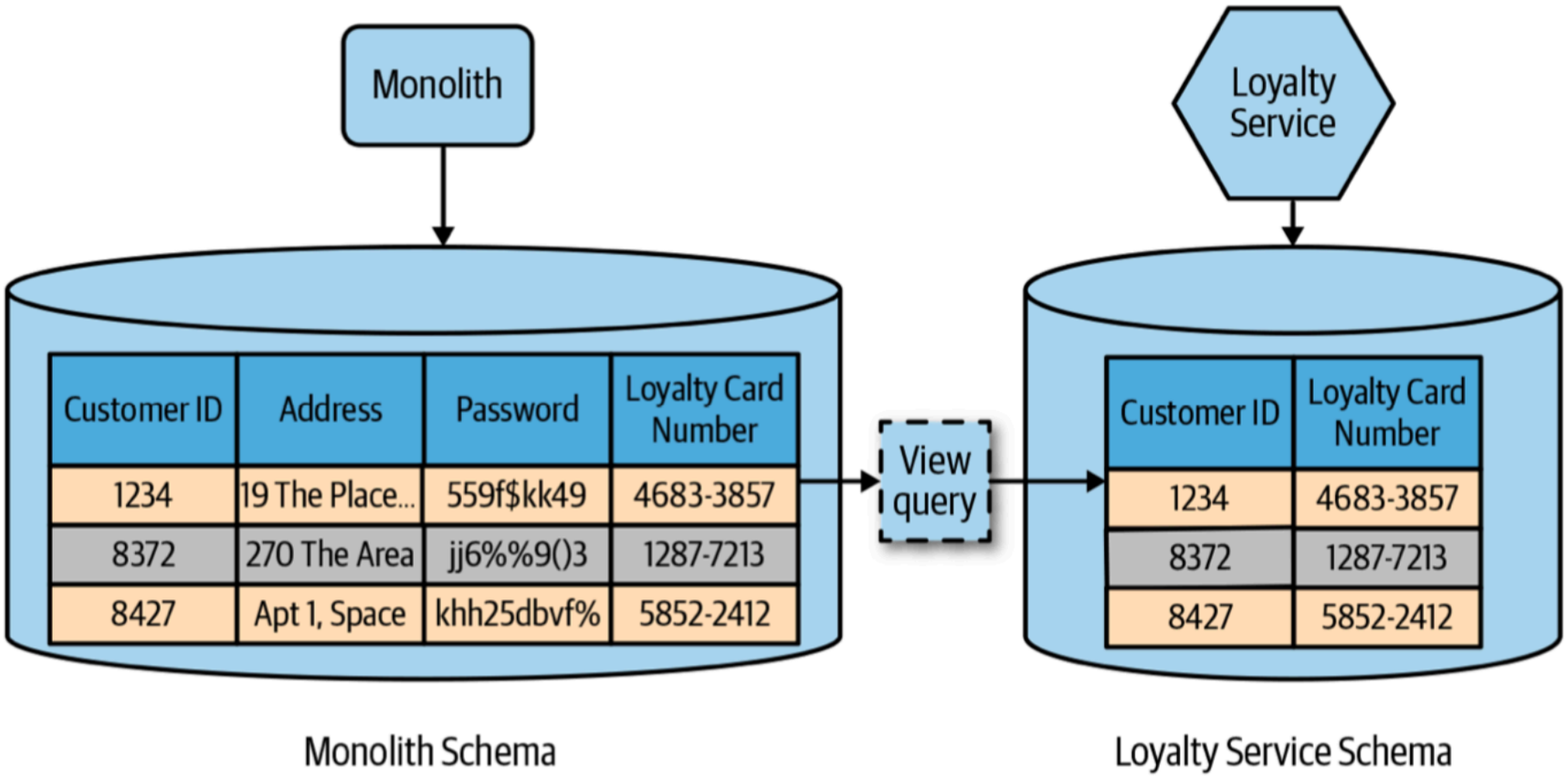
Database views



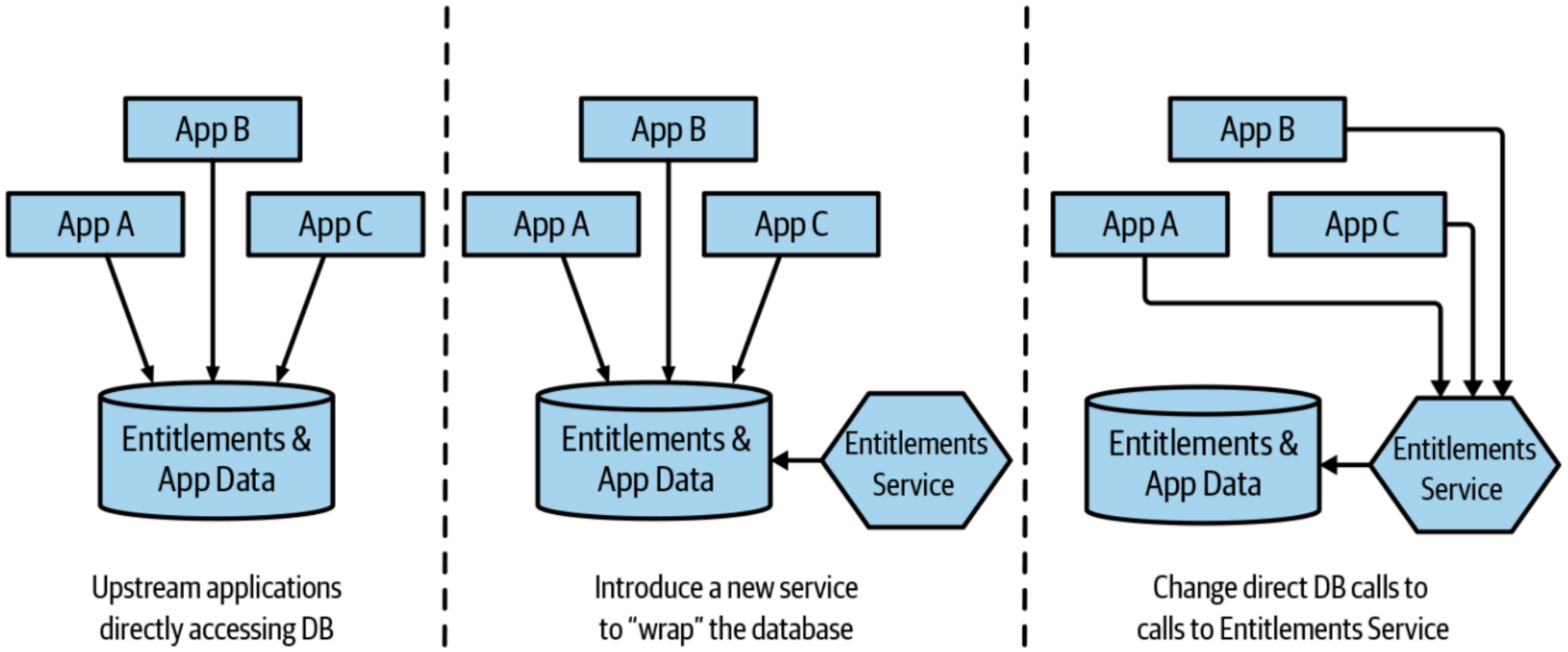
Before: External systems have direct DB access

After: External systems are redirected to read data from views, allowing the schema for the pricing system to change

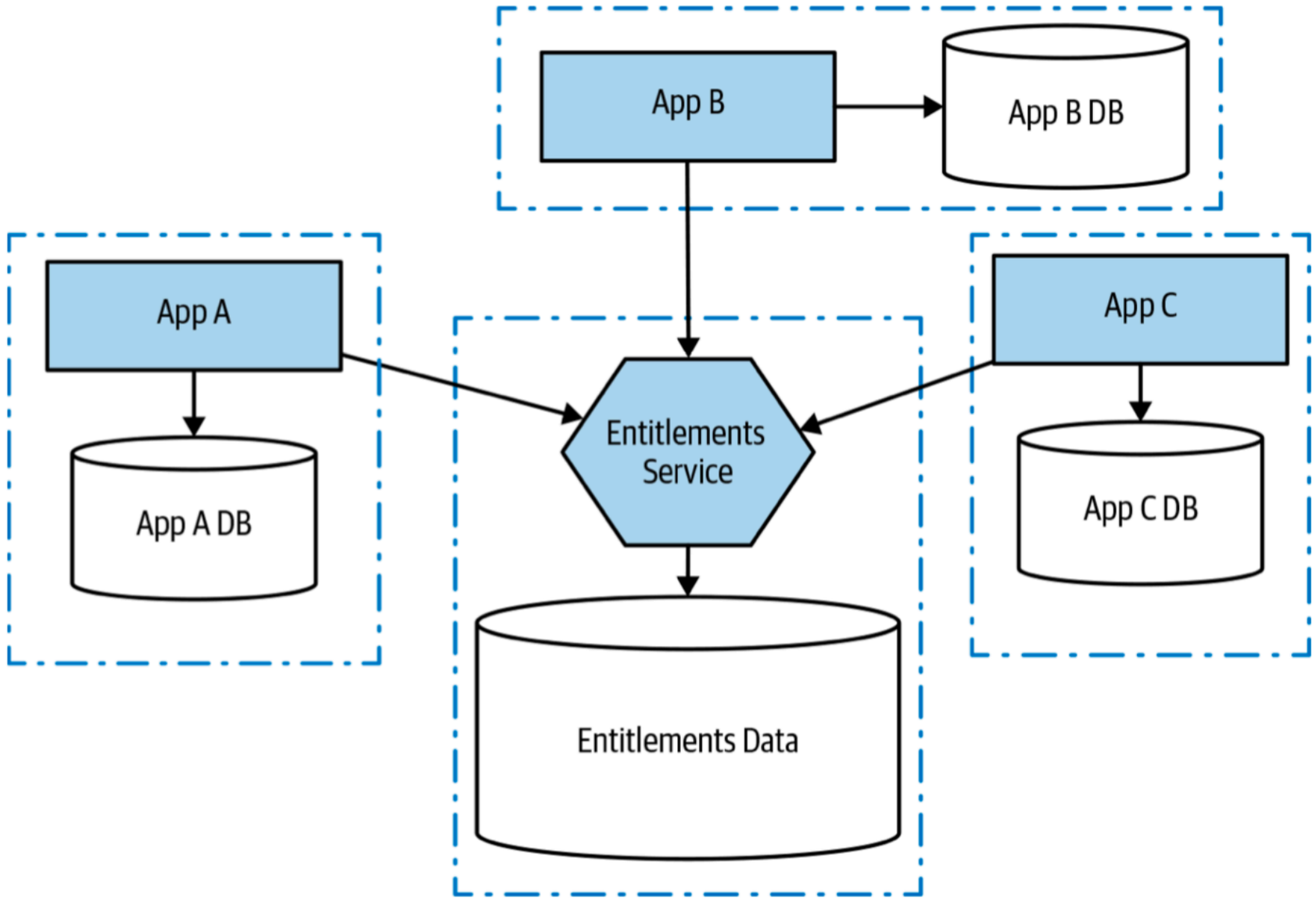
Database views



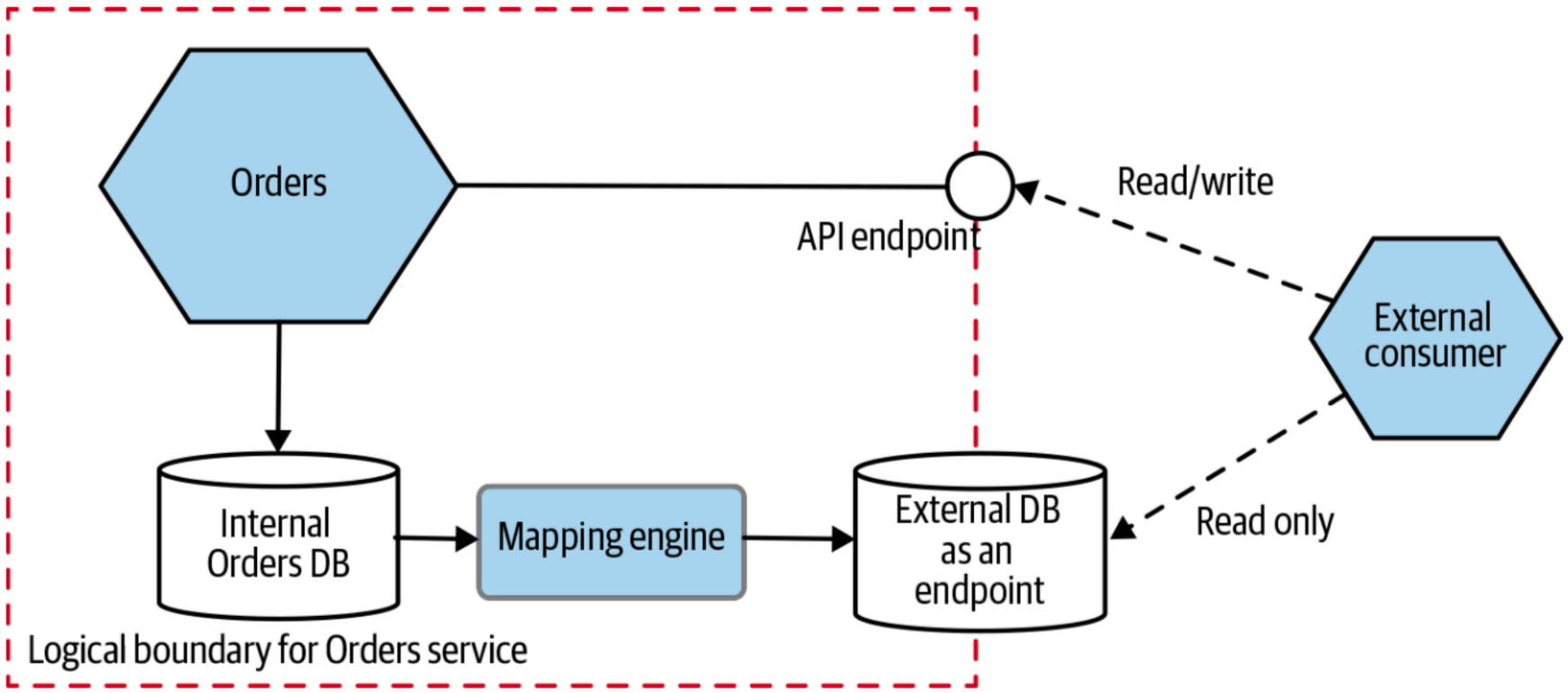
Service wrapping database



Service wrapping database

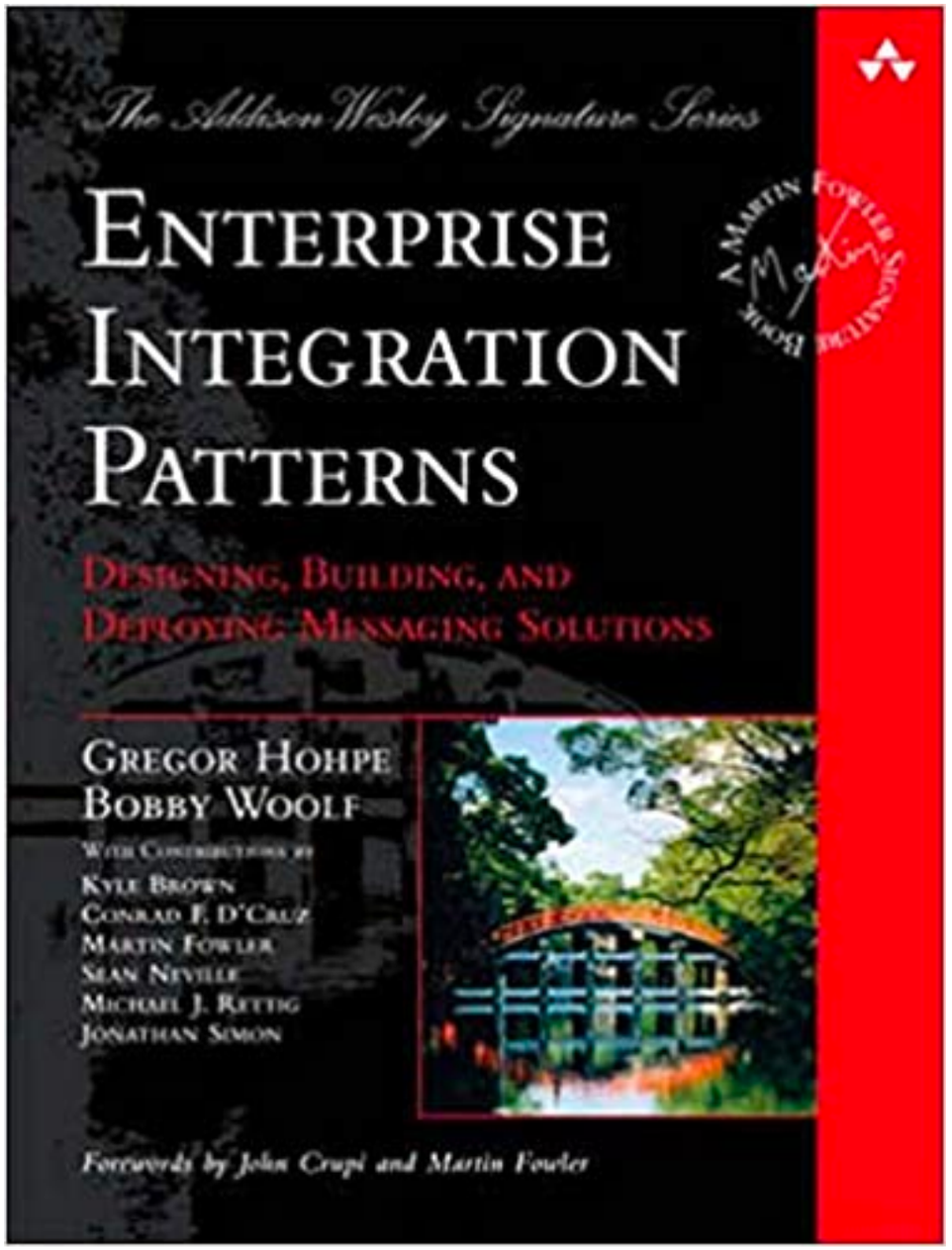


Database-as-a-Service



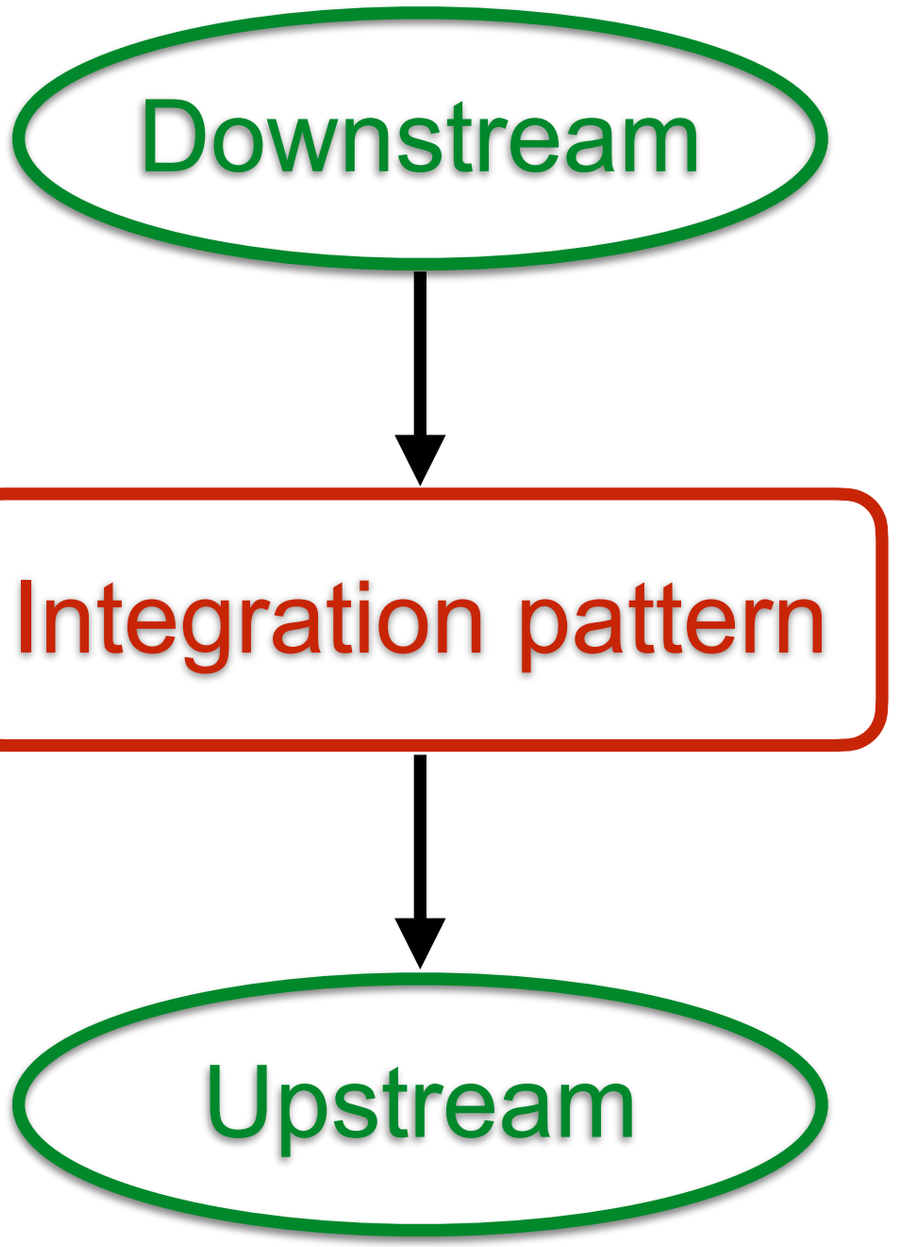
Integration patterns

Integration patterns

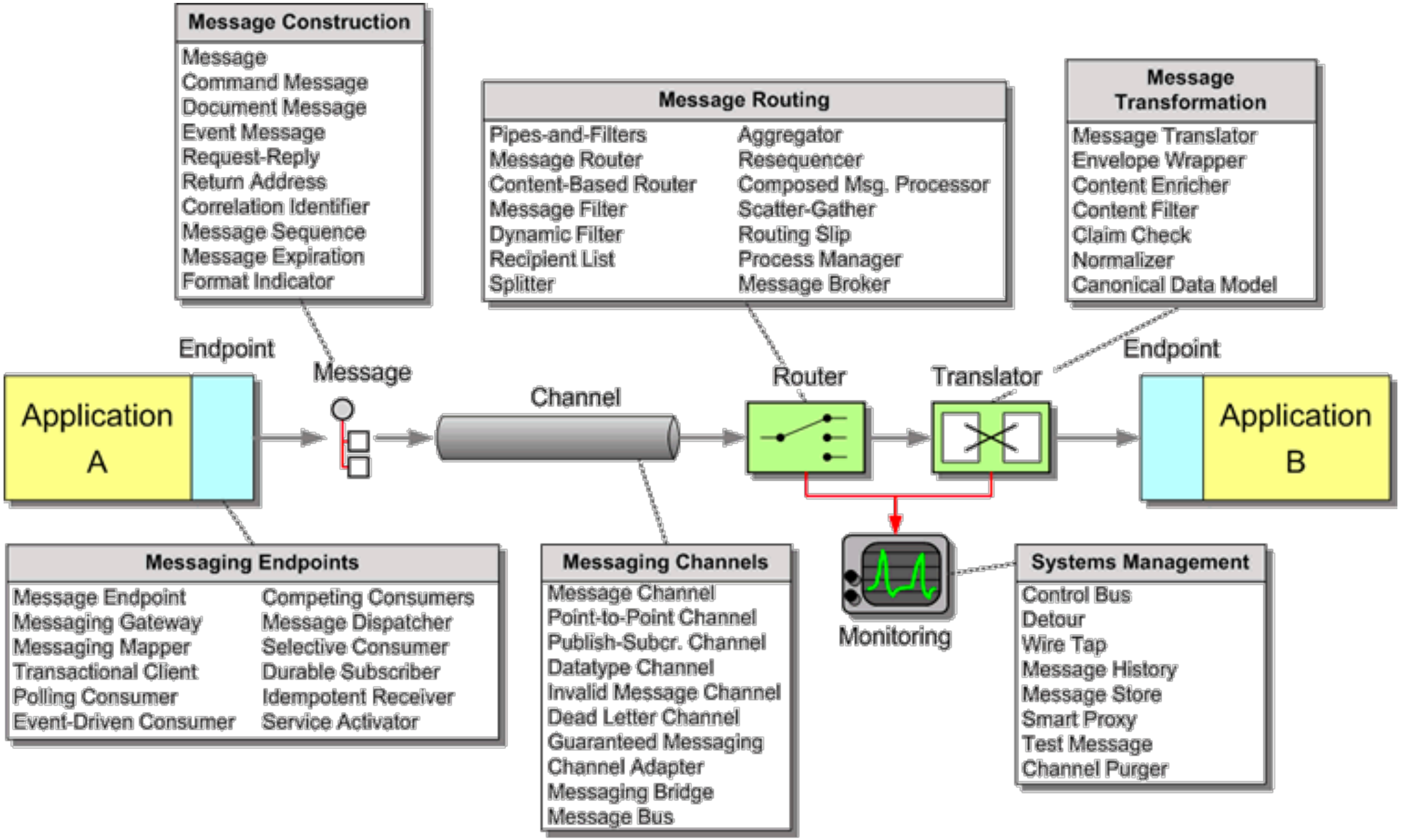


https://www.enterpriseintegrationpatterns.com/patterns/messaging/

Integration patterns (context map)



65 patterns !!



https://www.enterpriseintegrationpatterns.com/patterns/messaging/

Patterns

Partnership
Shared kernel
Customer-Supplier
Conformist
Anti-Corruption layer
Open-Host service
Publish language
Separate ways

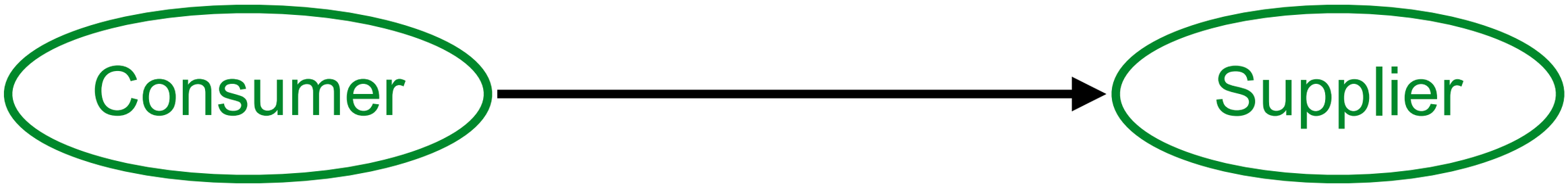
Problems ?

Applications were tightly-coupled
Applications were large monolith
Applications were written in a single language
Messaging systems were big and complex

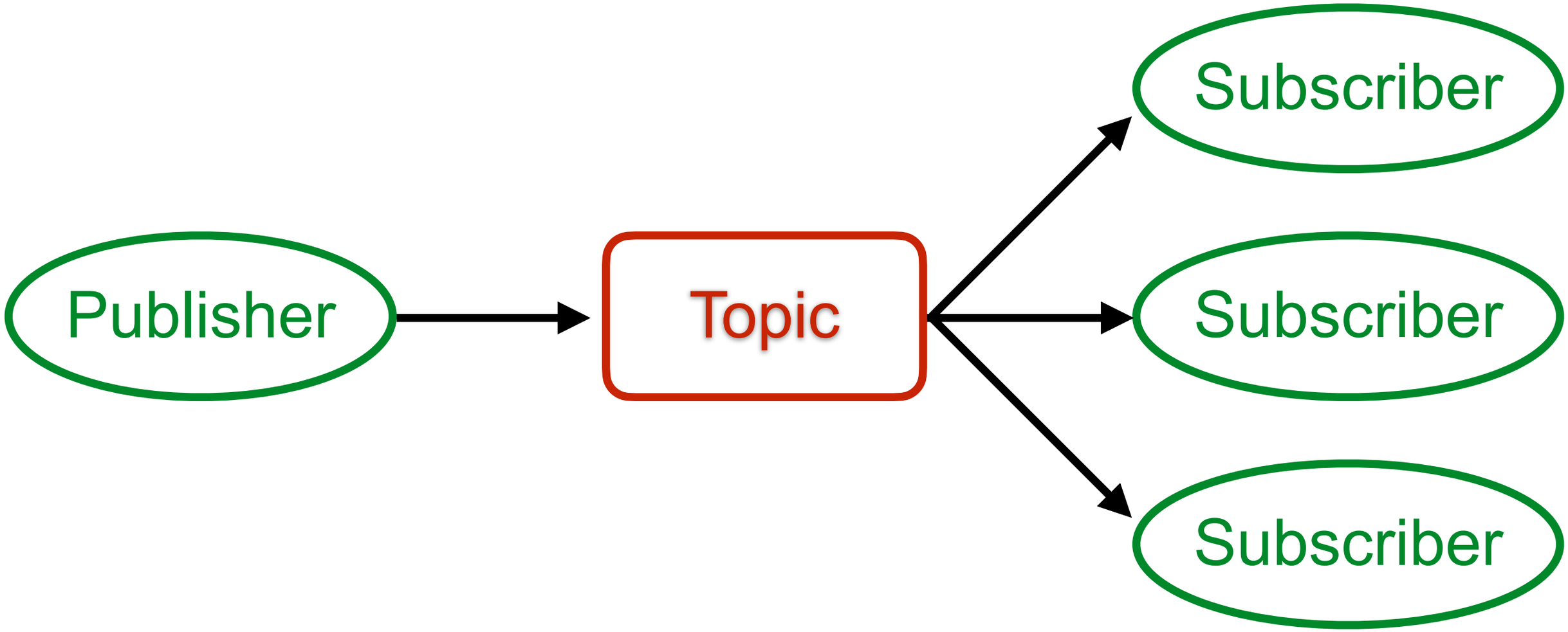
Messaging

Publish-Subscribe
Queue
Request-Reply

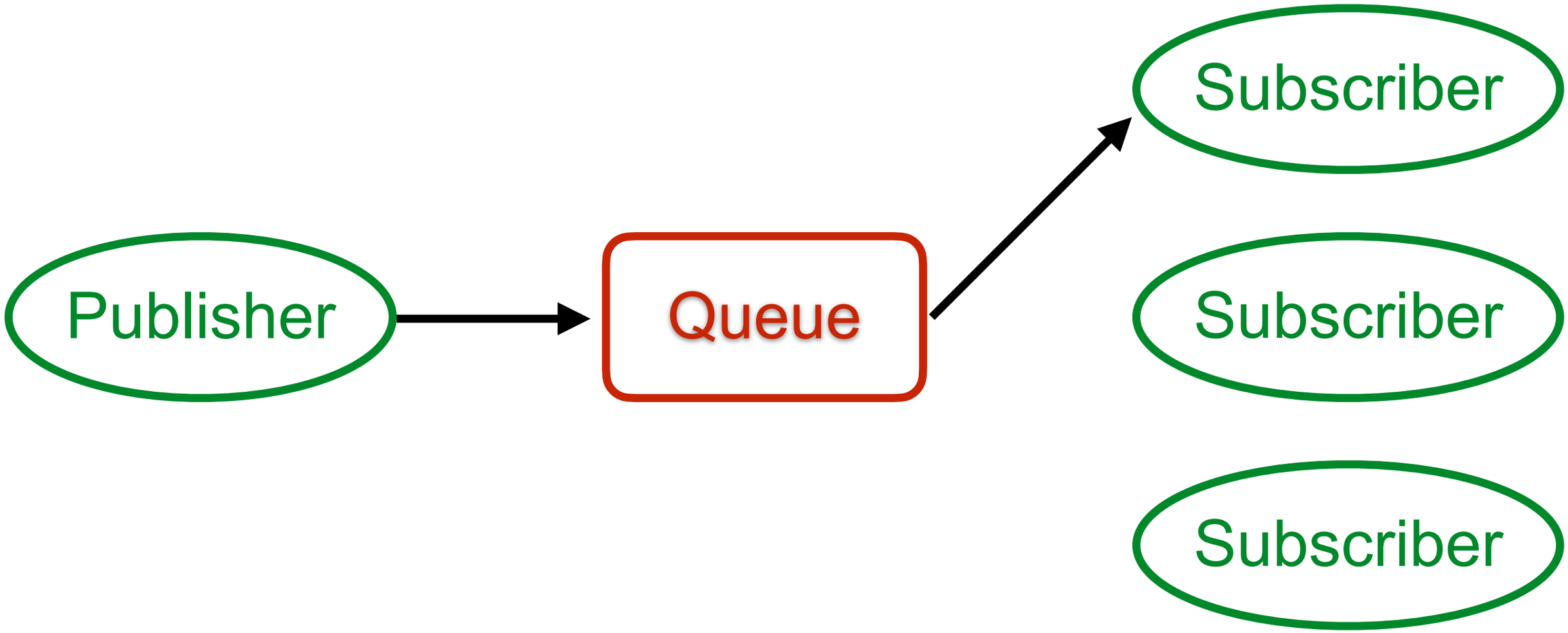
Consumer/supplier



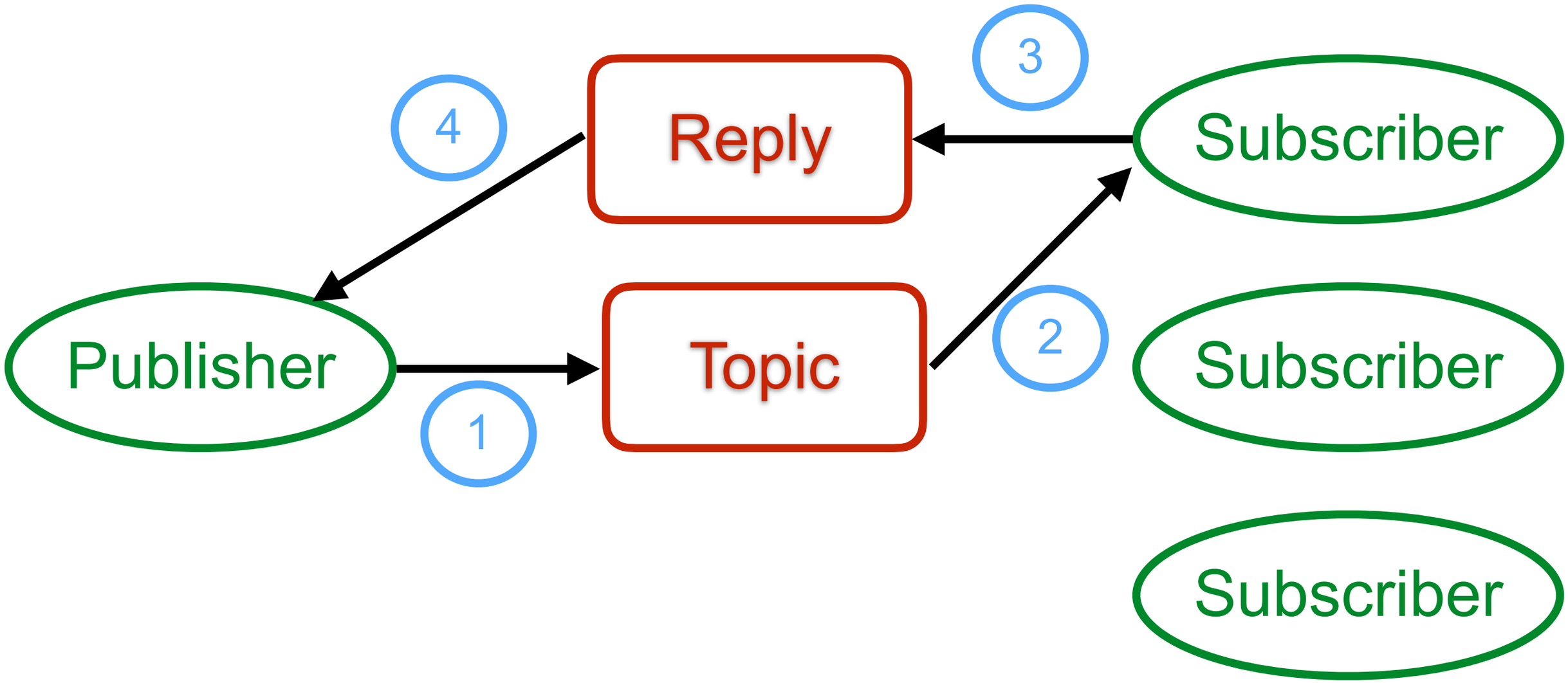
Publish-Subscribe



Queue

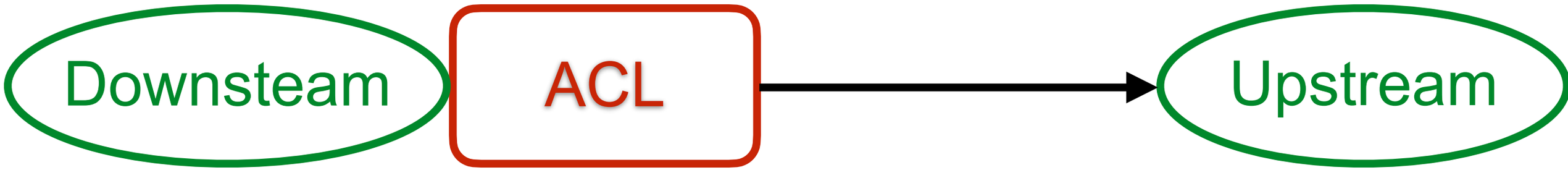


Request-Reply

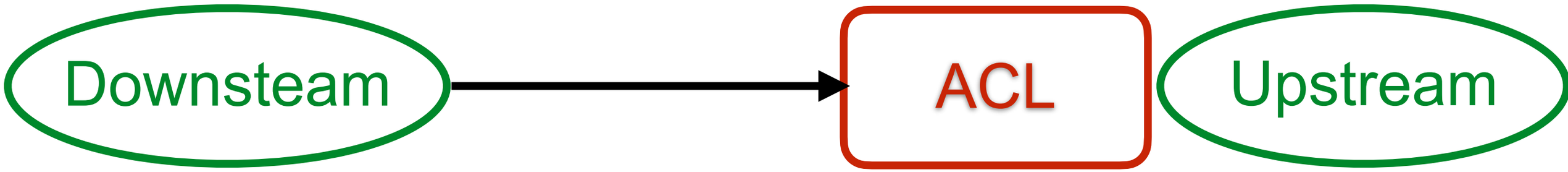


ACL (Anti-Corrupttion Layer)

ACL (Anti-Corrupttion Layer)



ACL (Anti-Corrupttion Layer)



Separate Way

What are good patterns ?

Microservices provide an opportunity to re-evaluate the way we think about communication between services

Anti-pattern ?

Anti-pattern ?

Problem => Bad solution

Anti-pattern ?

Problem => ~~Bad~~ solution

Anti-pattern ?

Problem => Inappropriate solution