MONOLITHS

TO MICROSERVICES

Sam Newman

Splitting Out Services

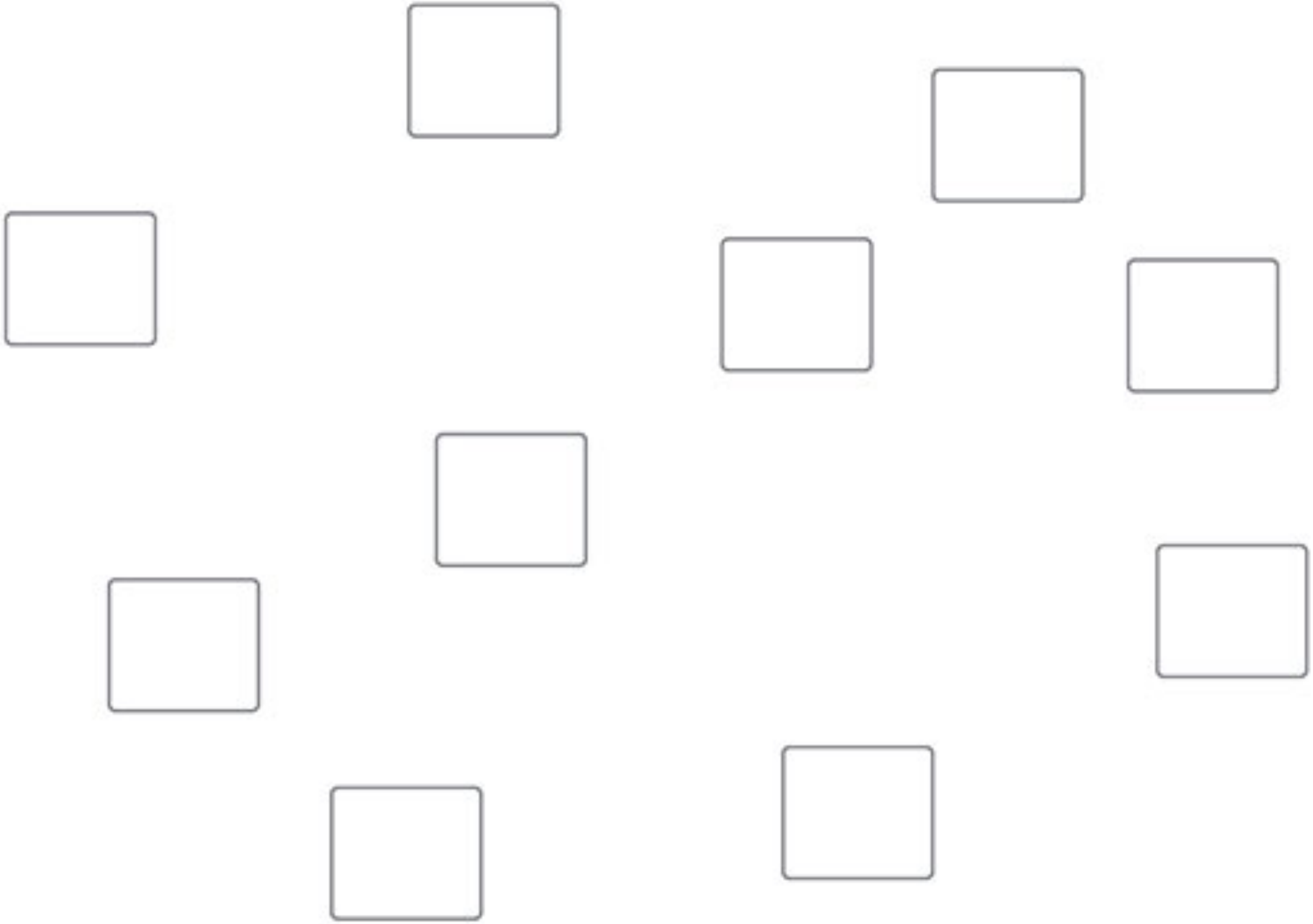
Overview

Where to start

Modelling services

Incremental change

DB Integration & Refactoring



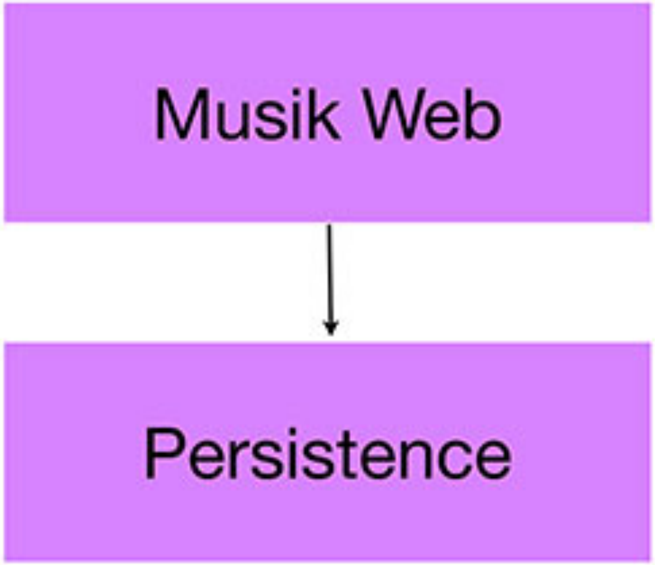
Where to start?

What makes a good service?

High Cohesion

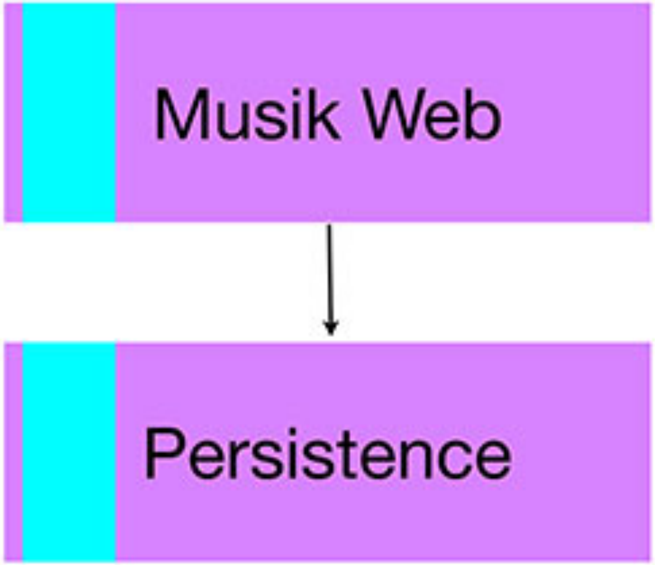
Loose Coupling

Musik Web



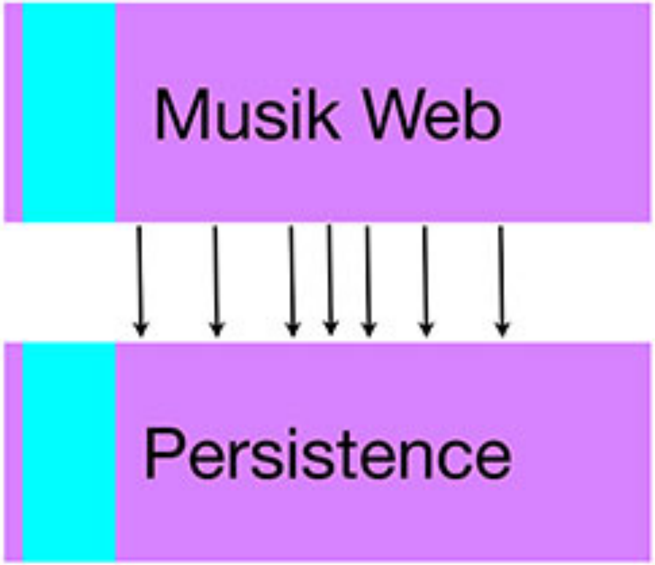
Musik Web

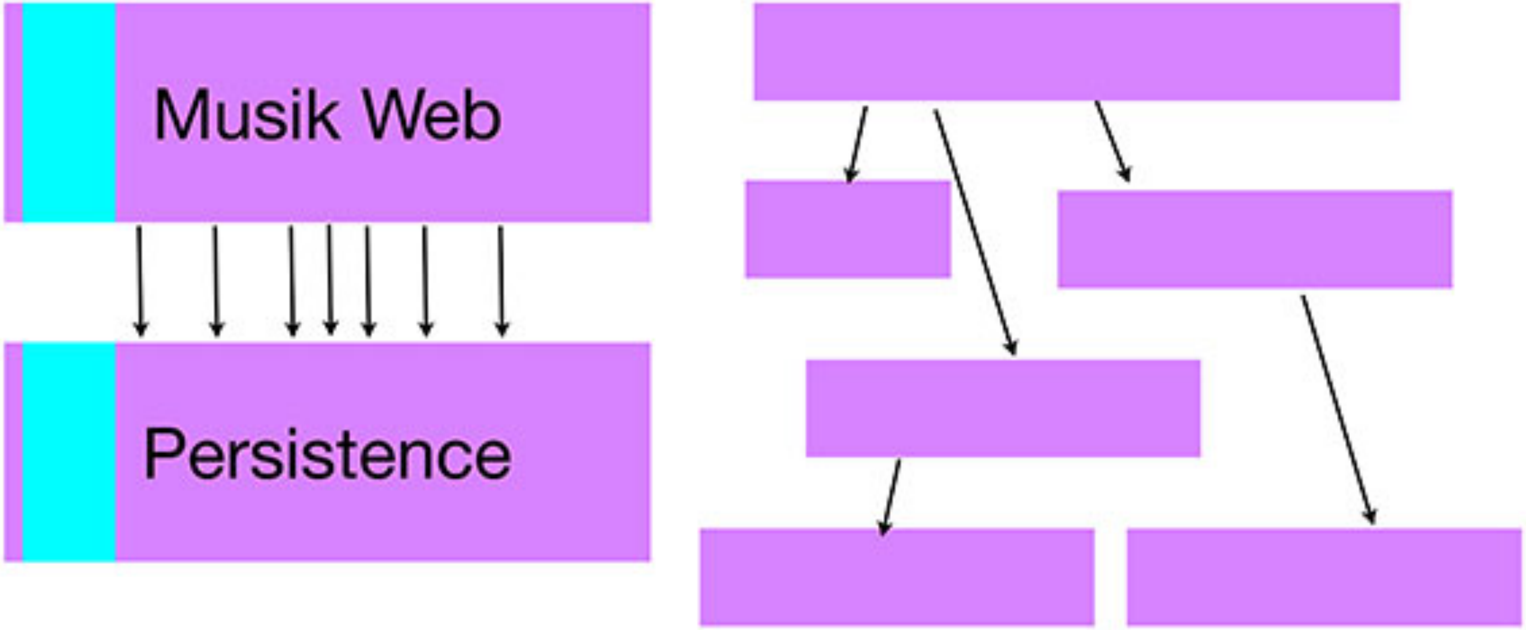
Persistence



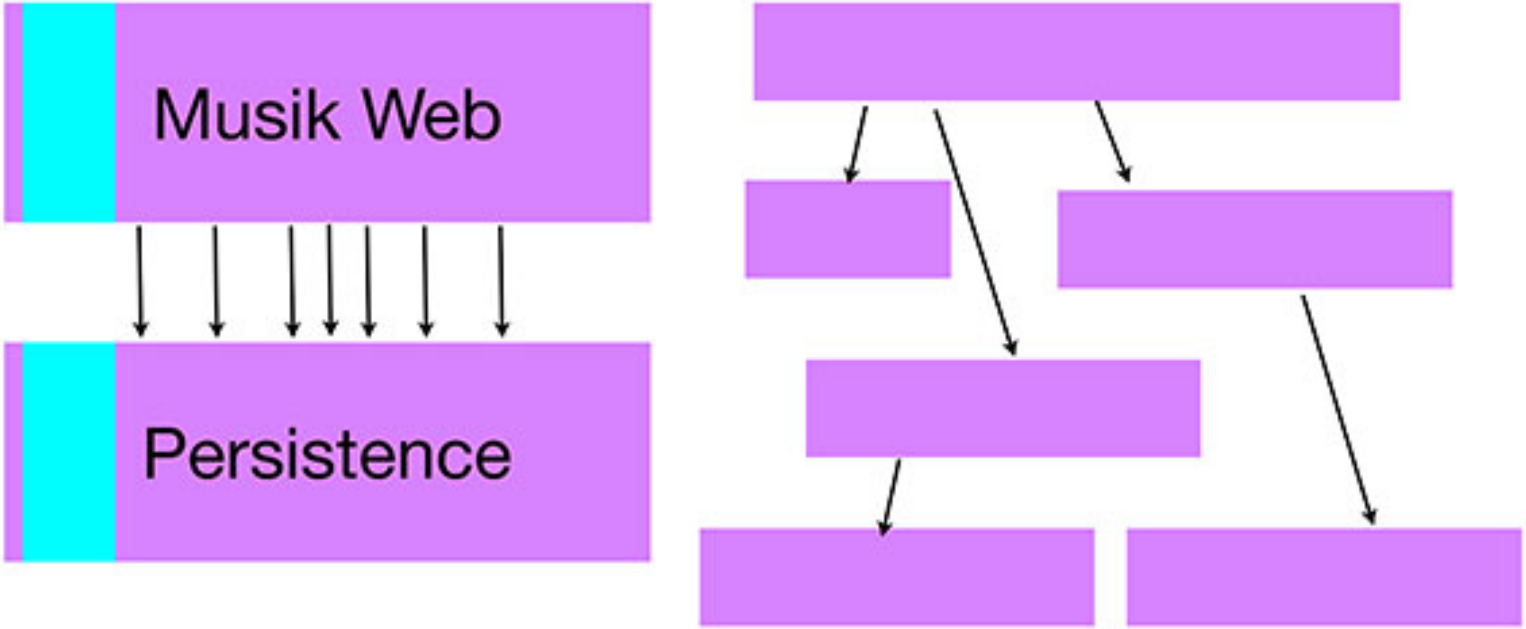
Musik Web

Persistence


Musik Web
Persistence

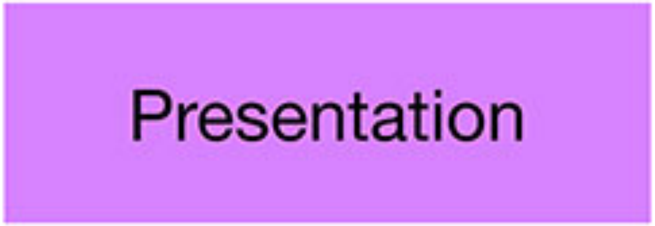

Musik Web
Persistence

ONION ARCHITECTURE



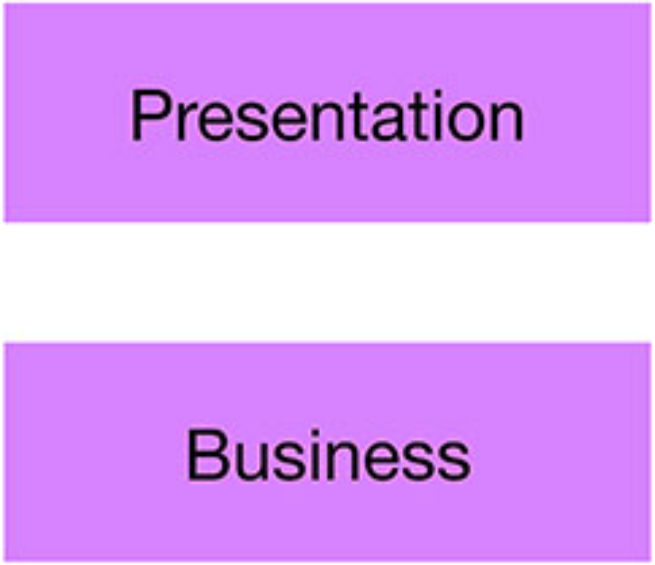
3 TIRED ARCHITECTURE

3 TIRED ARCHITECTURE



Presentation

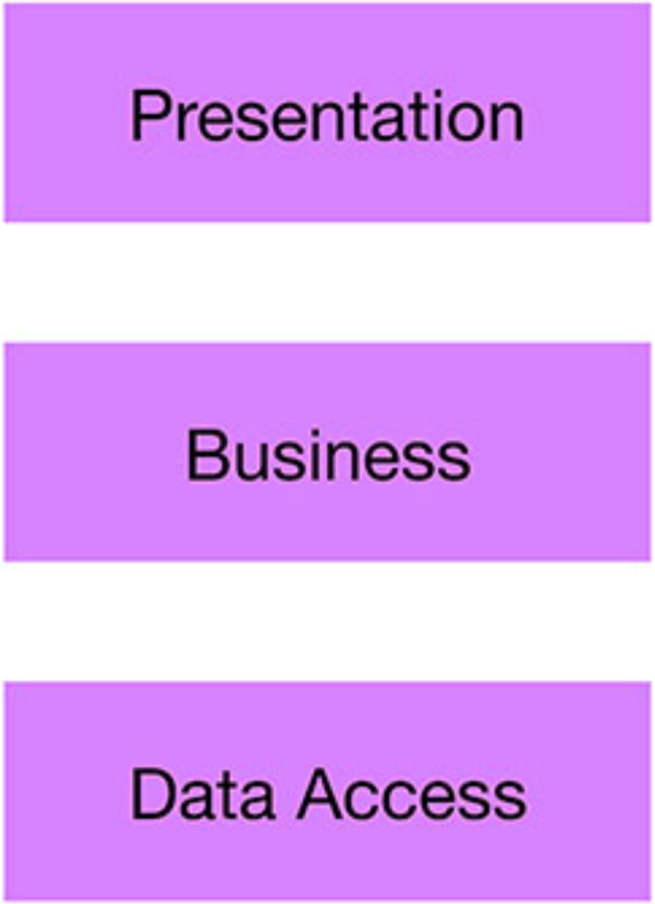
3 TIRED ARCHITECTURE



Presentation

Business

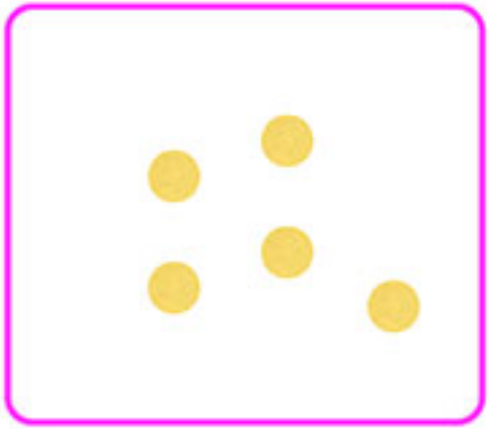
3 TIRED ARCHITECTURE

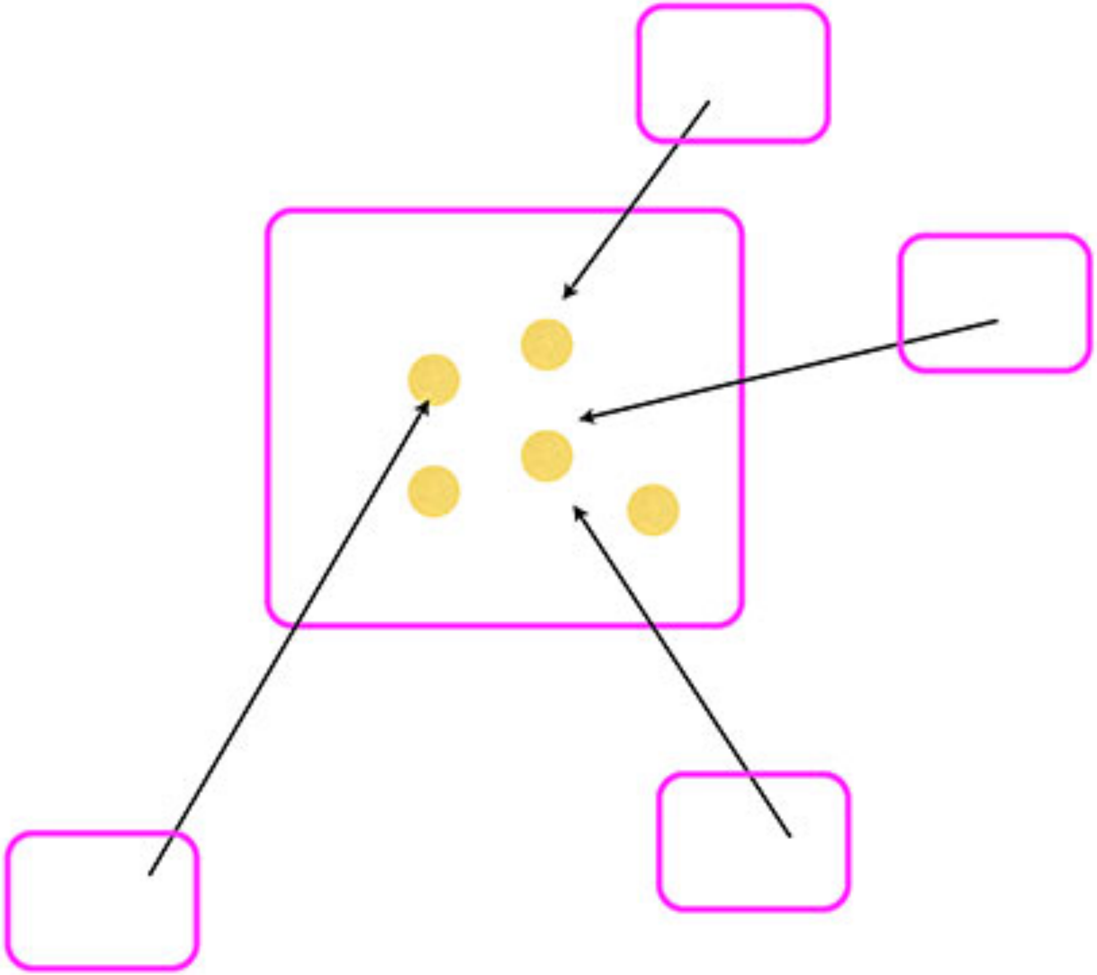


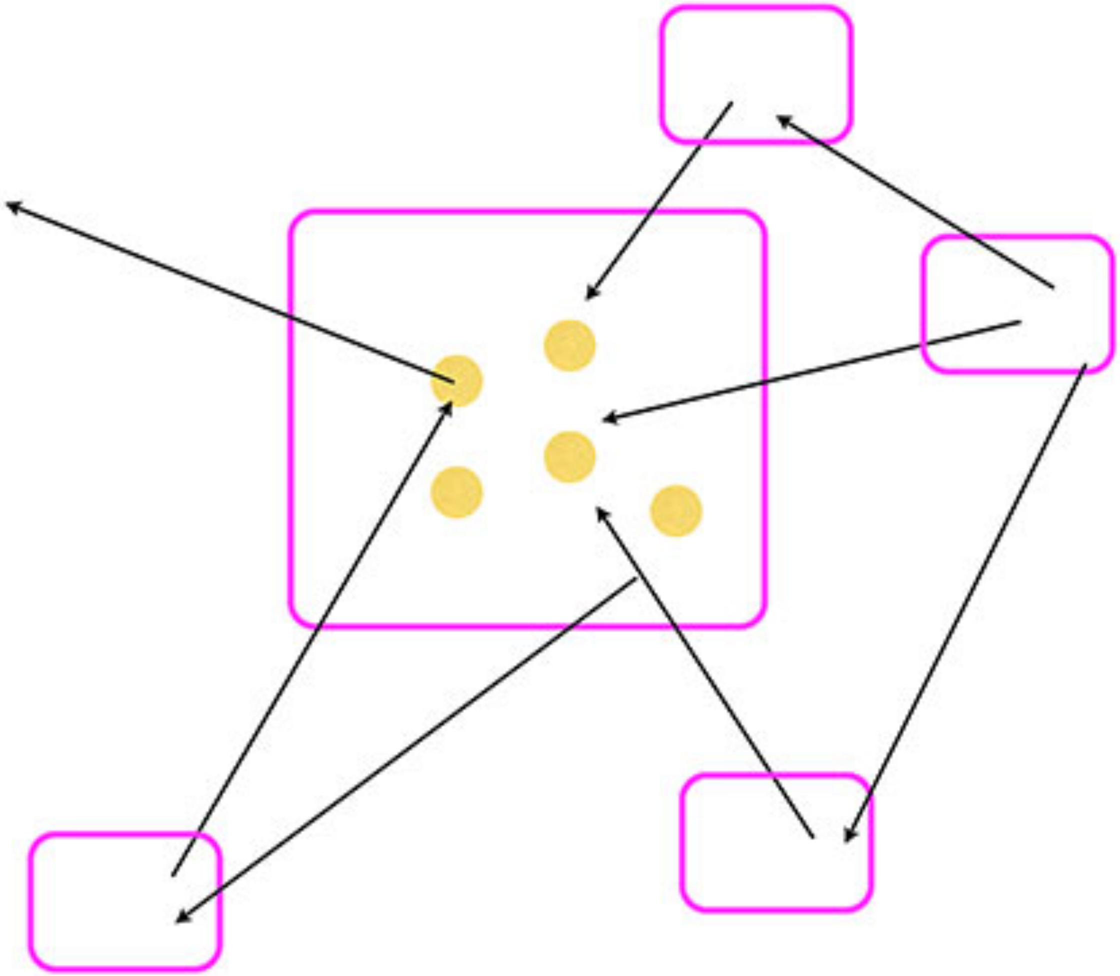
Presentation

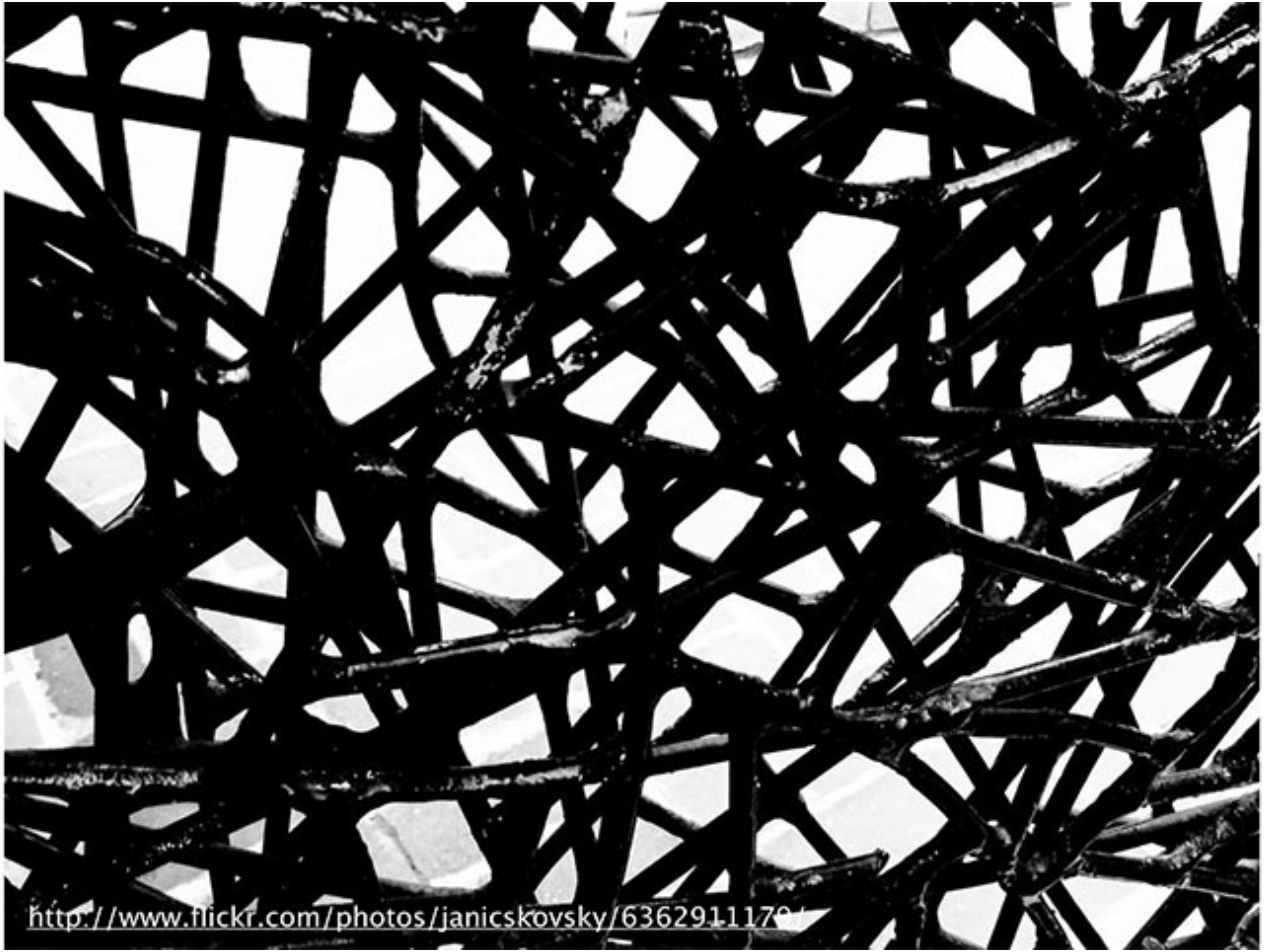
Business

Data Access

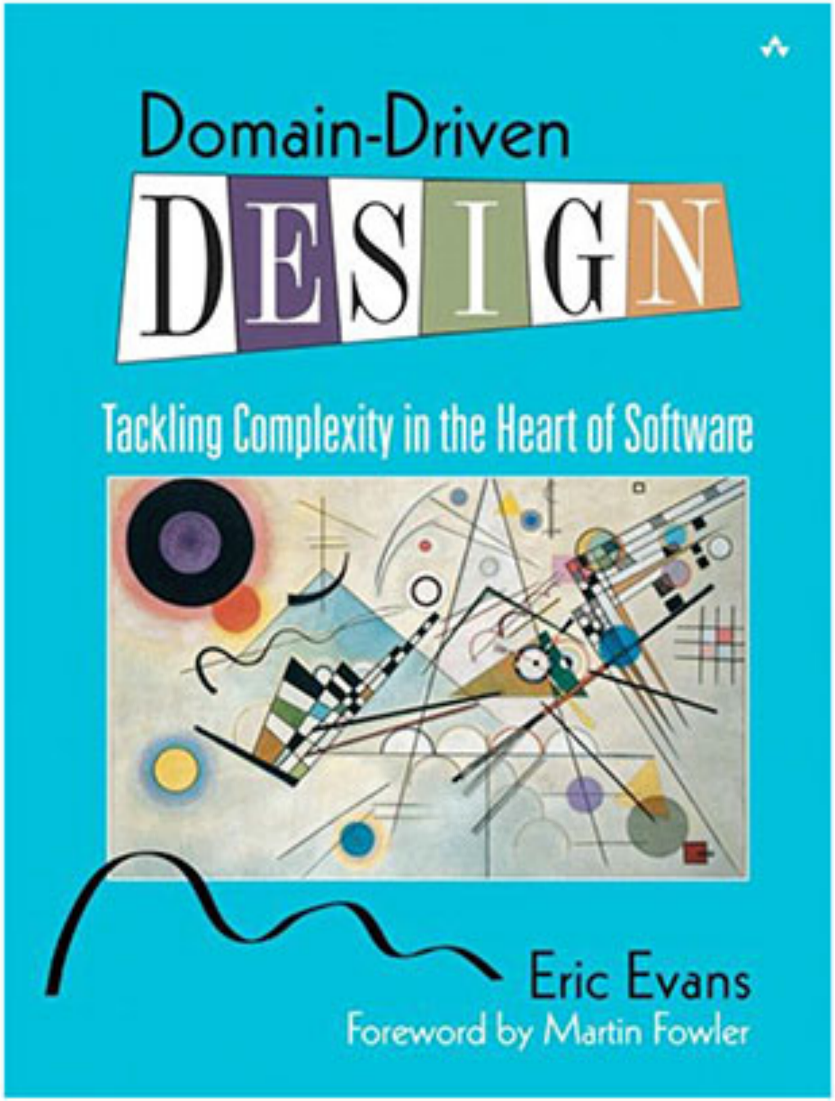








http://www.flickr.com/photos/janicskovsky/6362911179/



Domain-Driven DESIGN

Tackling Complexity in the Heart of Software

Eric Evans

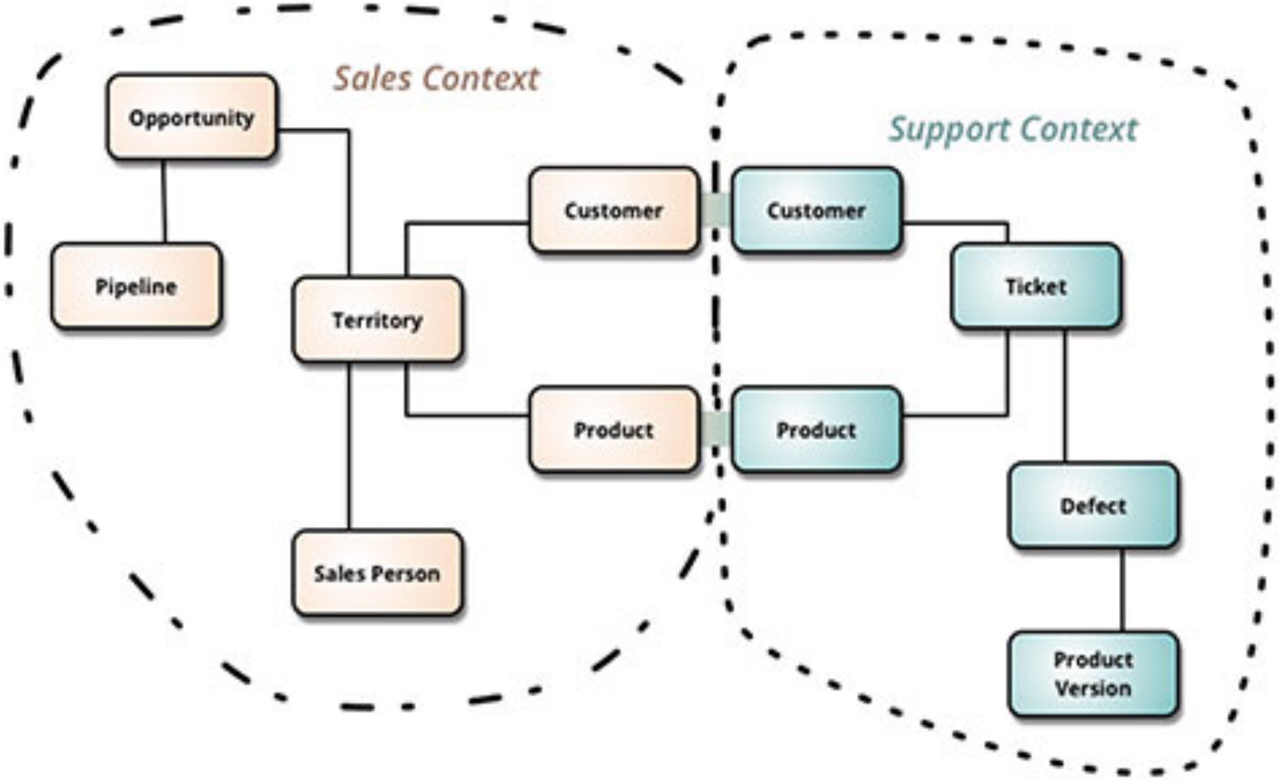
Foreword by Martin Fowler

Bounded Context

"The delimited applicability of a particular model. BOUNDING CONTEXTS gives team members a clear and shared understanding of what has to be consistent and what can develop independently."

“A specific responsibility enforced by explicit boundaries”

http://www.sapiensworks.com/blog/post/2012/04/17/DDD-The-Bounded-Context-Explained.aspx



http://martinfowler.com/bliki/images/boundedContext/sketch.png

Capabilities?

https://www.flickr.com/photos/glimeend/5692801655/

Register a new customer

Register a new customer

Oil Change

Register a new customer

Oil Change

Emission Test

Register a new customer

Oil Change

Emission Test

Replace Tire

Register a new customer

Oil Change

Emission Test

Replace Tire

Order parts

Register a new customer

Oil Change

Emission Test

Replace Tire

Order parts

Send an invoice

Register a new customer

Oil Change

Emission Test

Replace Tire

Order parts

Send an invoice

Make a repair

Register a new customer

Oil Change

Emission Test

Replace Tire

Contact a customer

Order parts

Send an invoice

Make a repair

Register a new customer

Contact a customer

Send an invoice

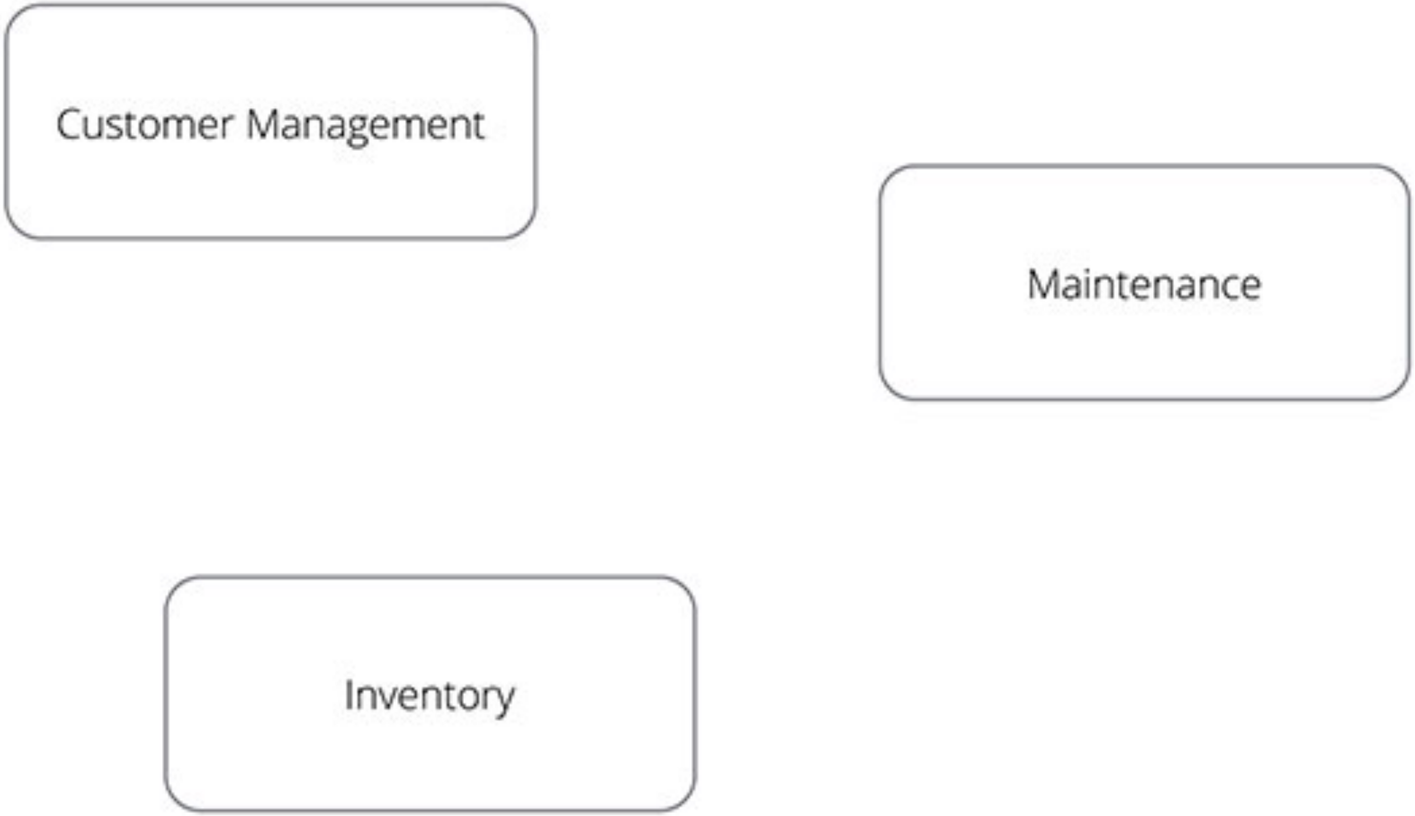
Order parts

Oil Change

Make a repair

Replace Tire

Emission Test

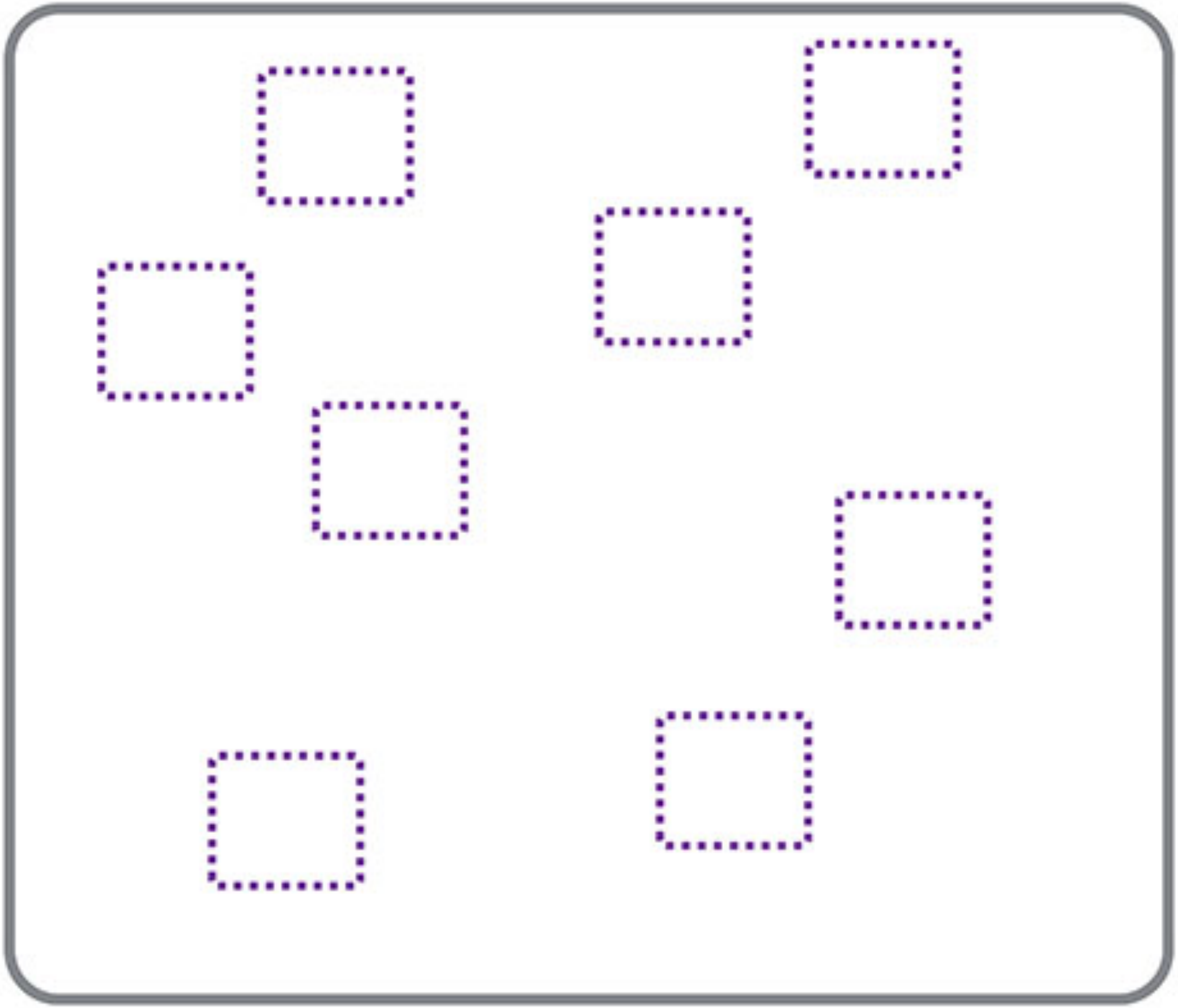


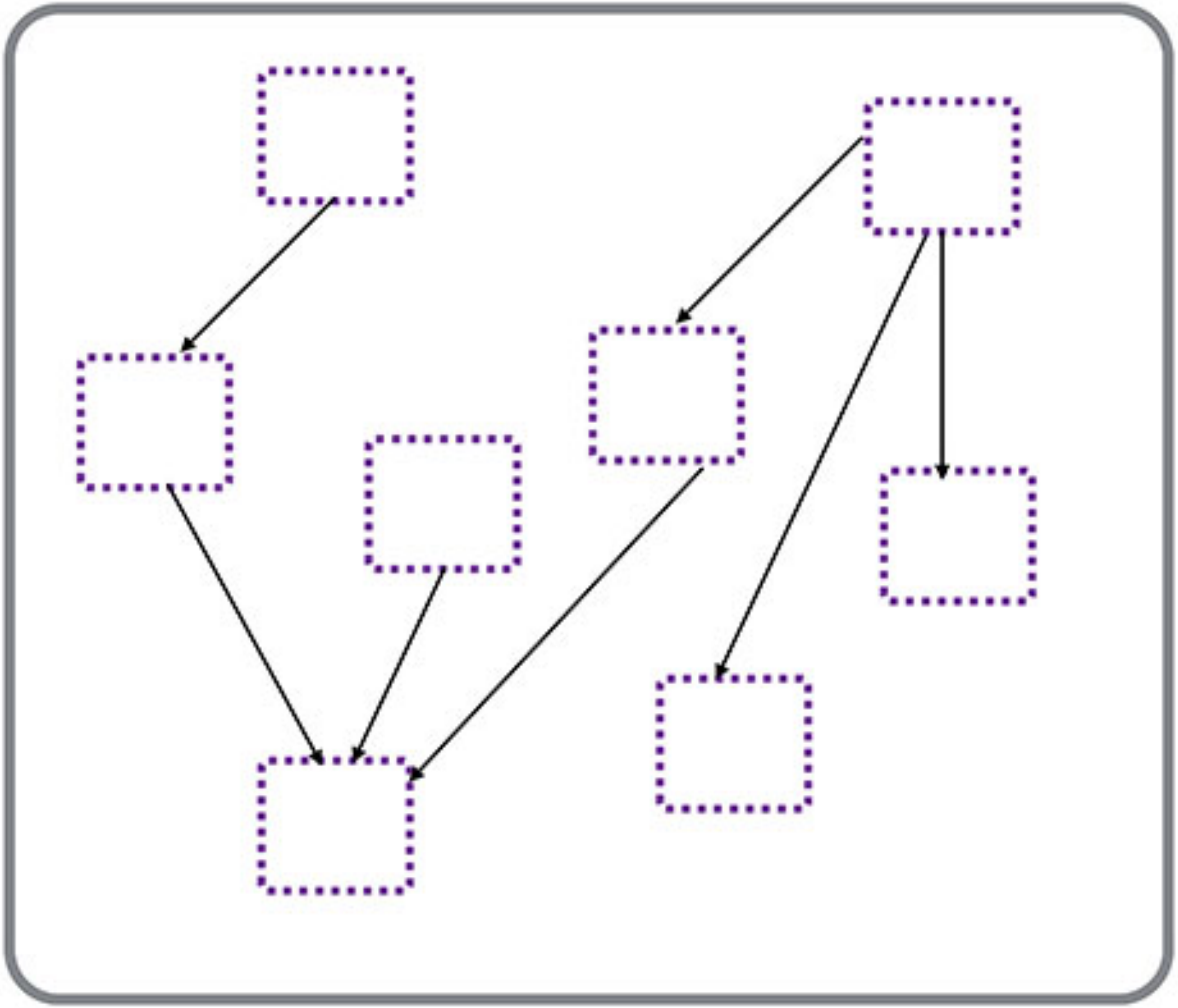
Customer Management

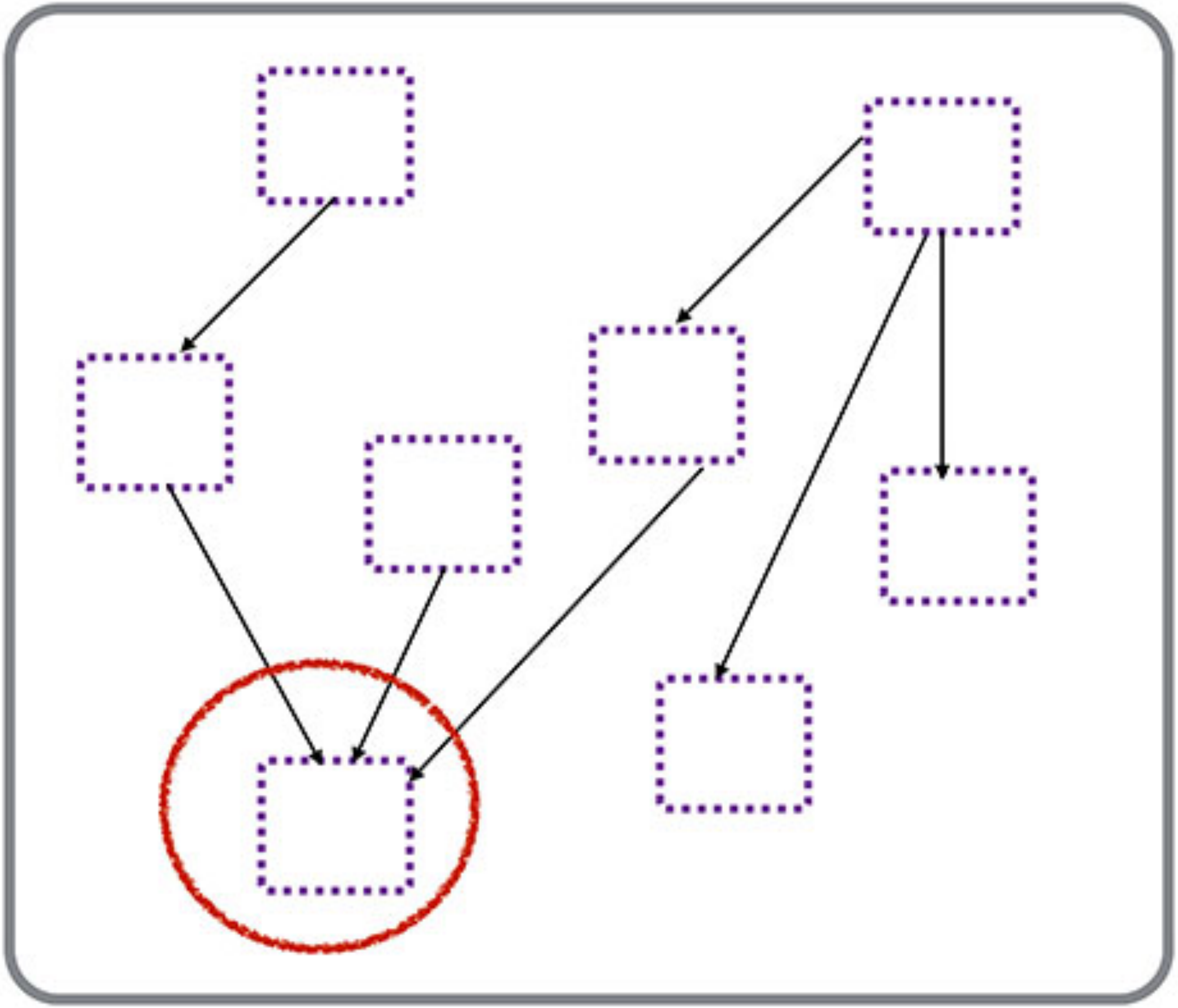
Maintenance

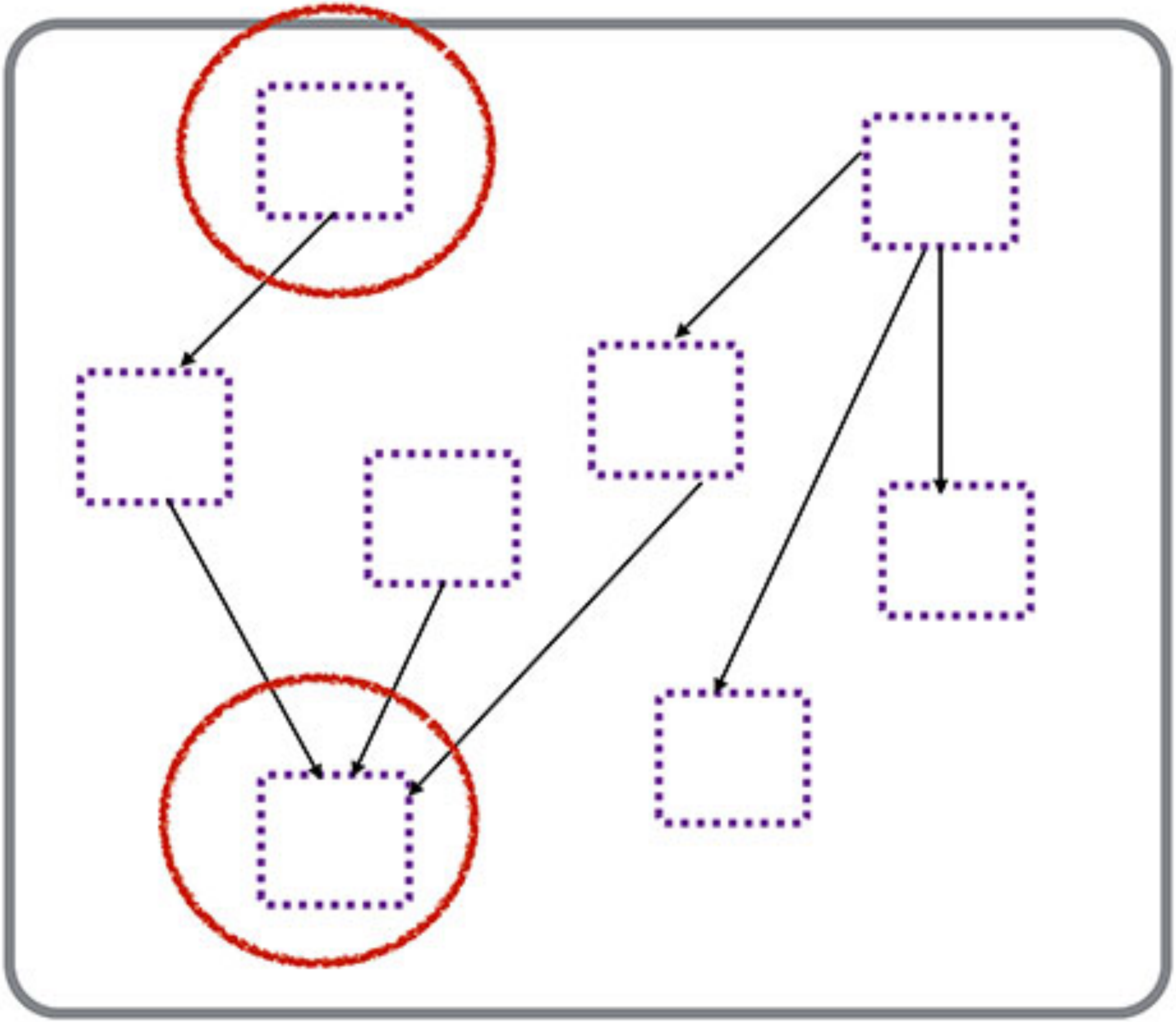
Inventory

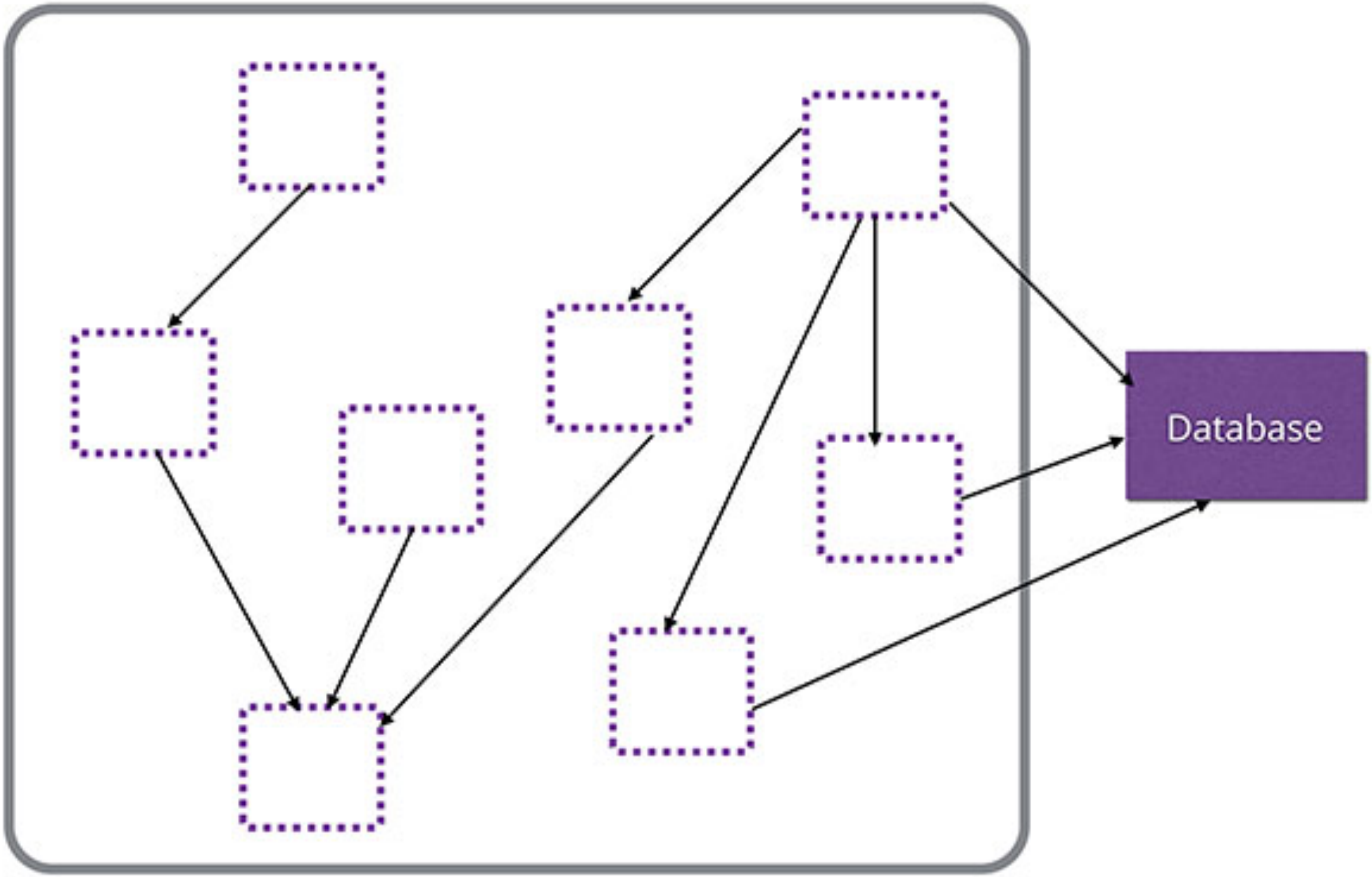
Remember The Goal!

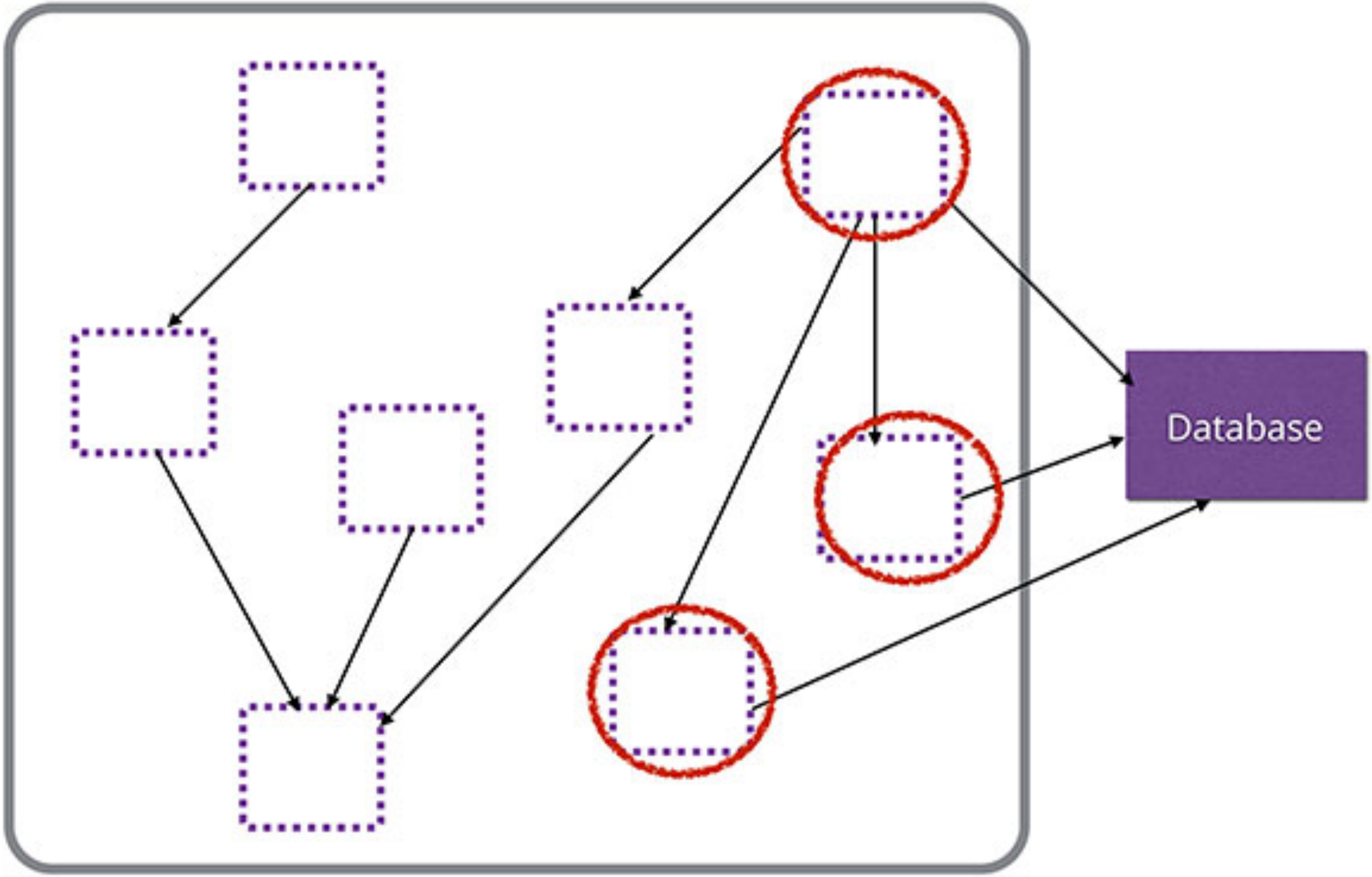


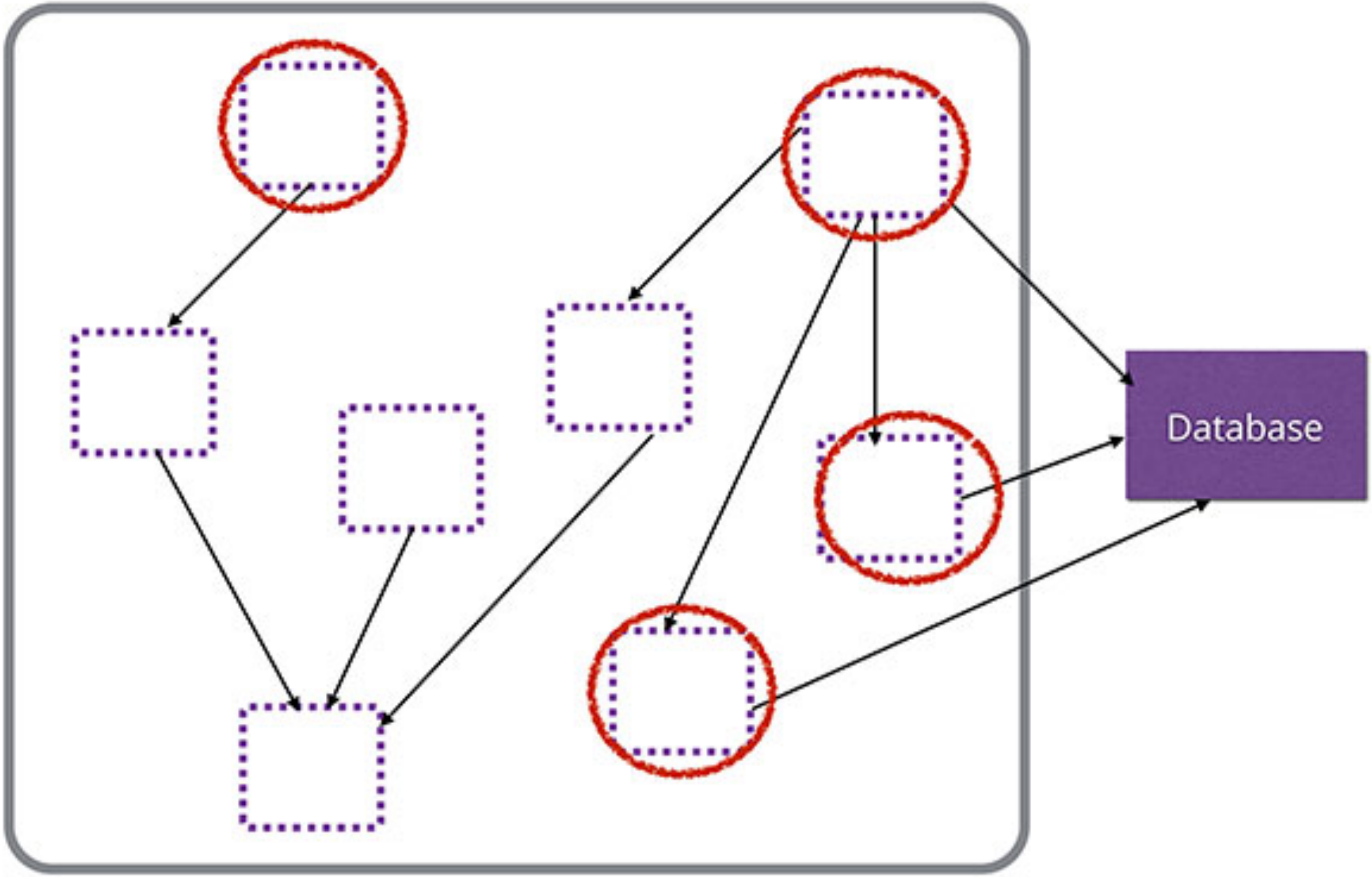







Database


Database


Database

So all things being equal, look for:

So all things being equal, look for:

modules with few inbound dependencies

So all things being equal, look for:

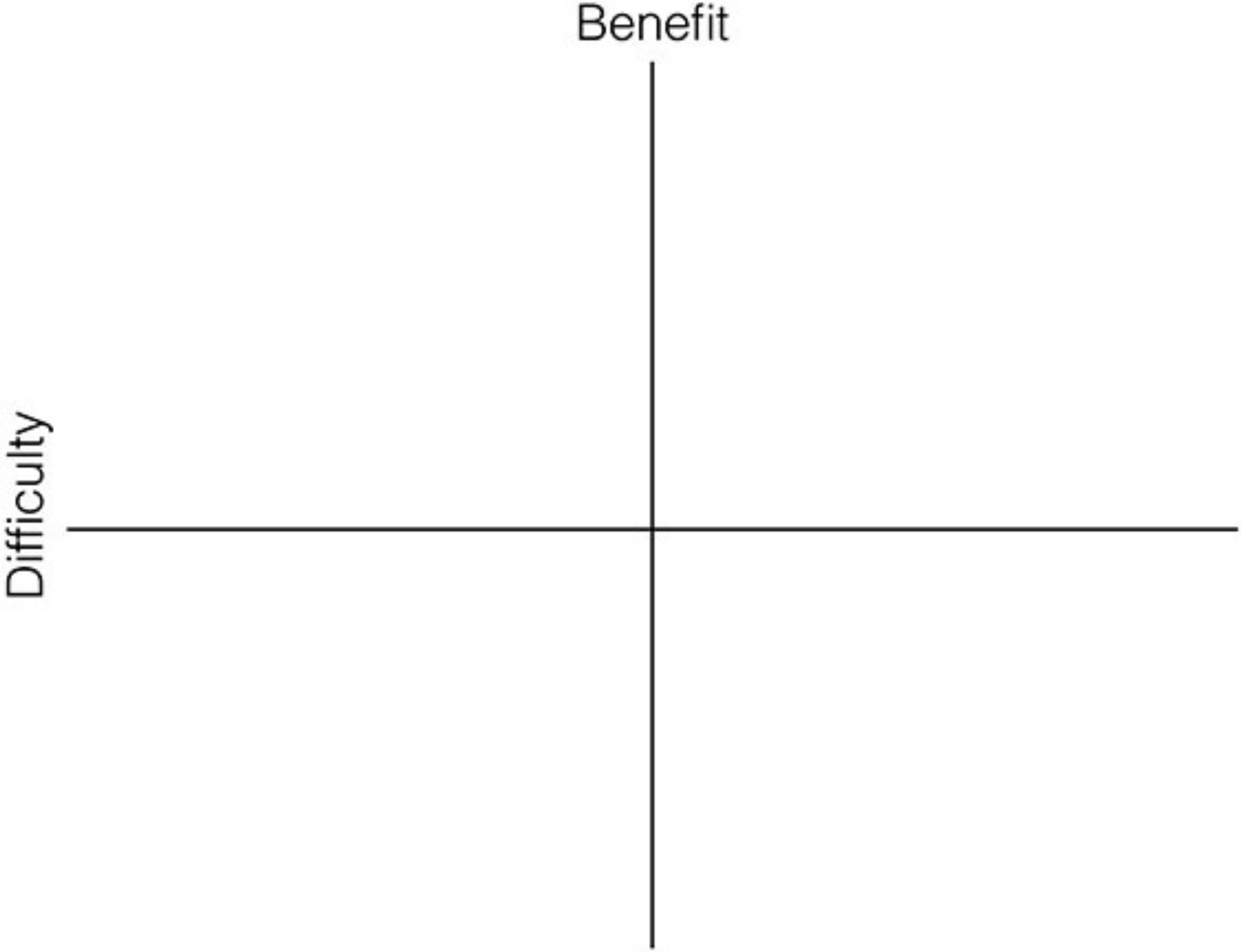
modules with few inbound dependencies

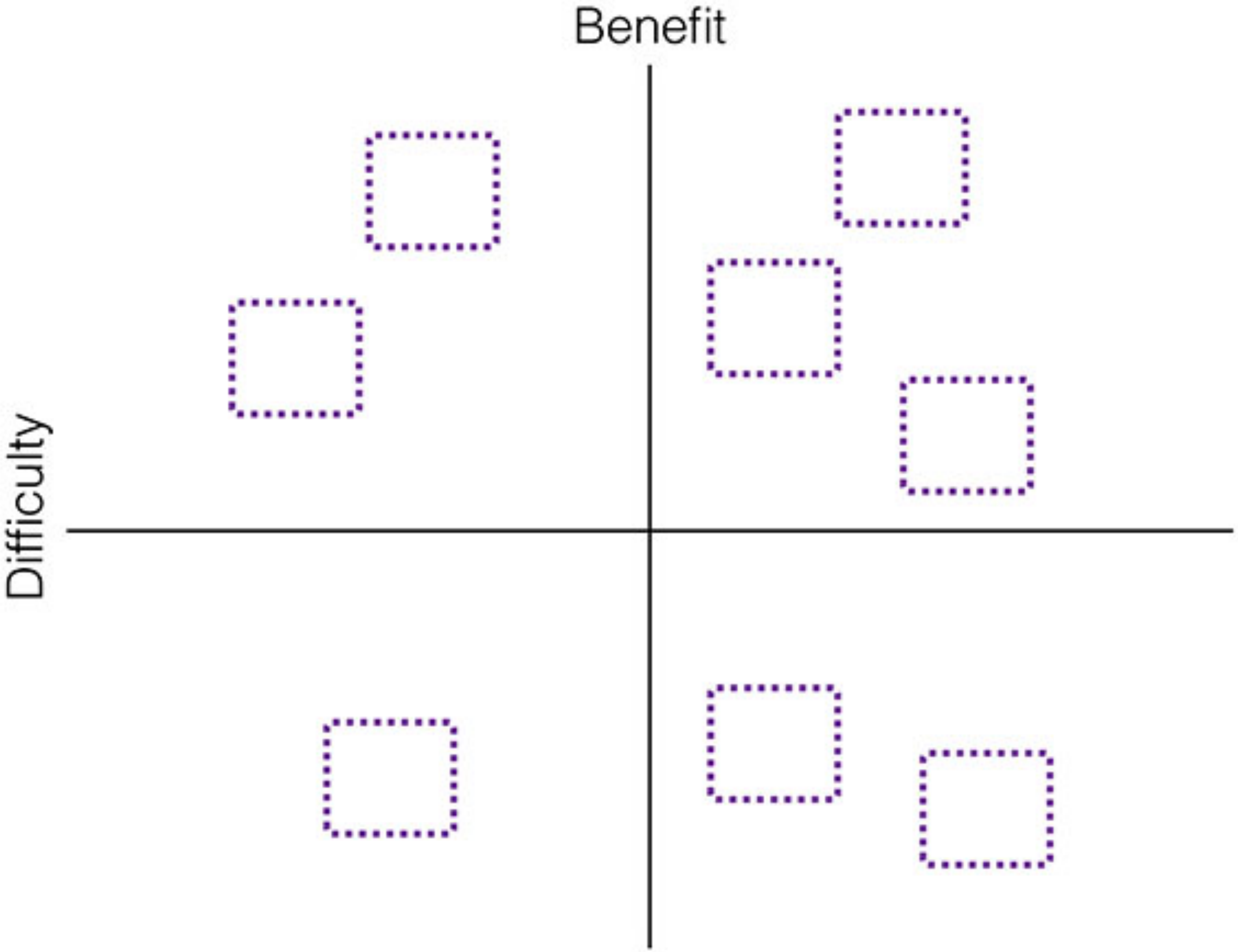
which can be stateless candidates

Ease Of Decompositon

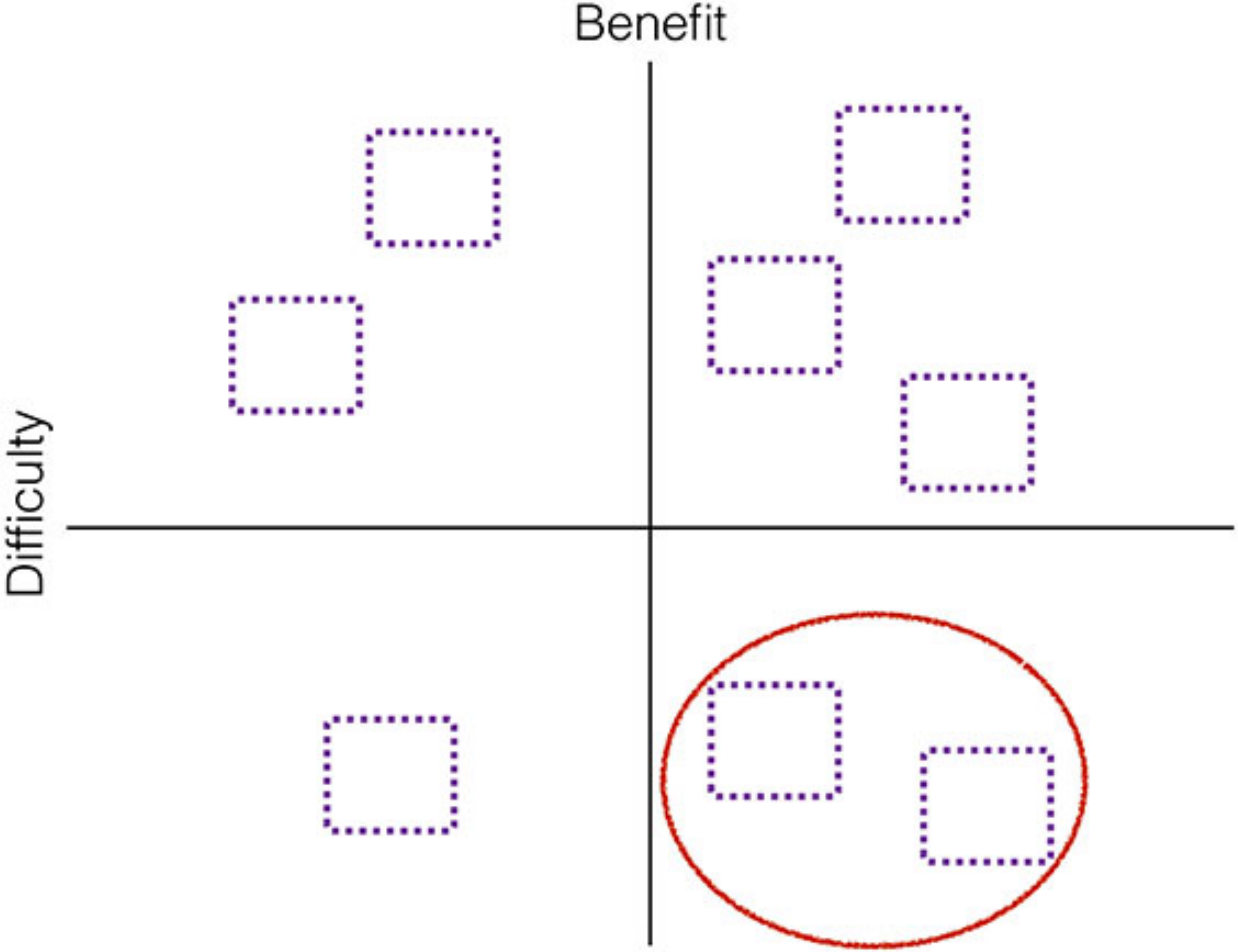
vs

Benefits Of Decompositon


Benefit
Difficulty



Benefit
Difficulty


Benefit
Difficulty

Incremental Change

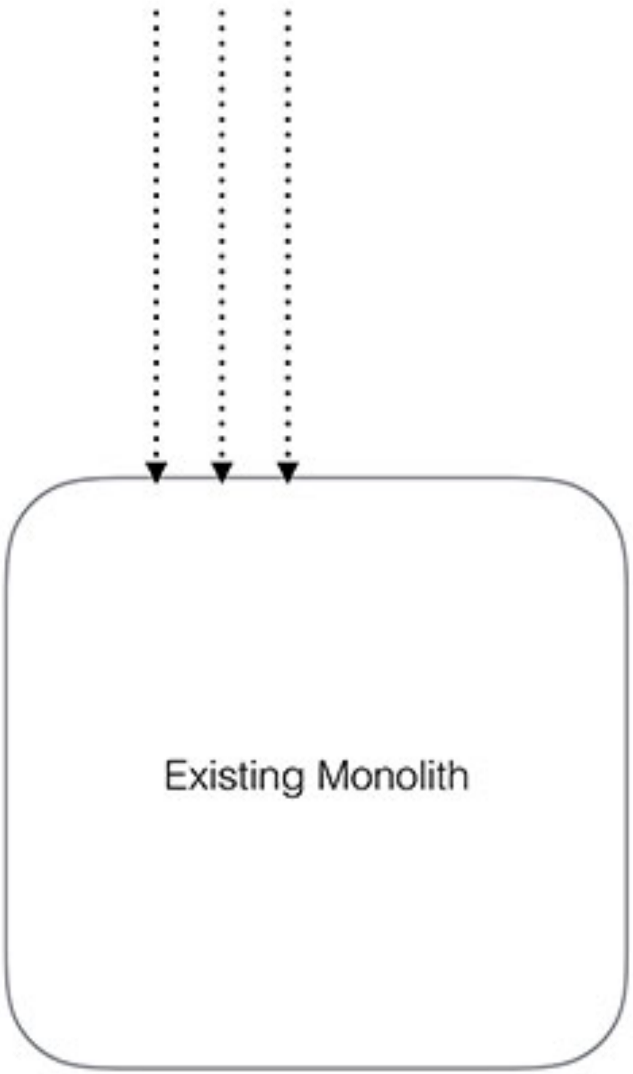
Enter the strangler pattern

Enter the strangler pattern

Great pattern, terrible name

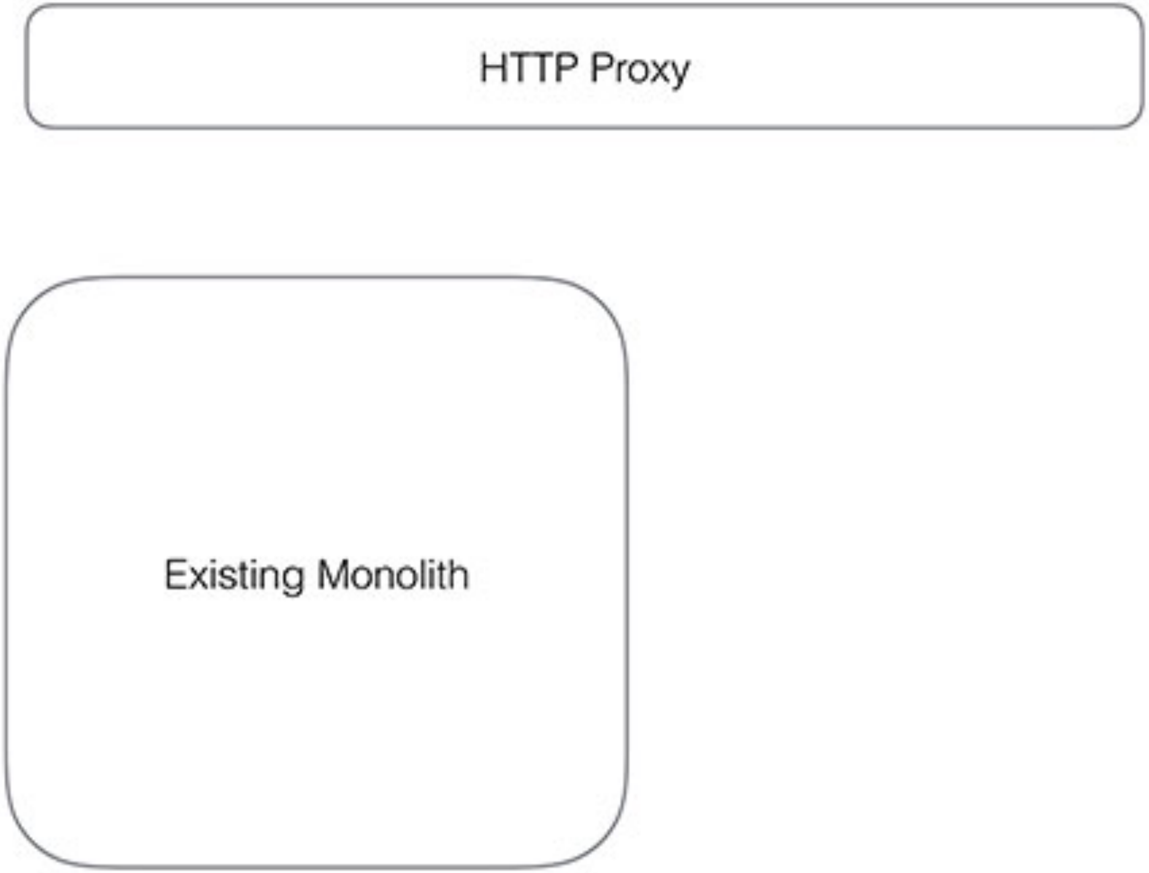
https://www.flickr.com/photos/cynren/16012074217/

HTTP PROXY

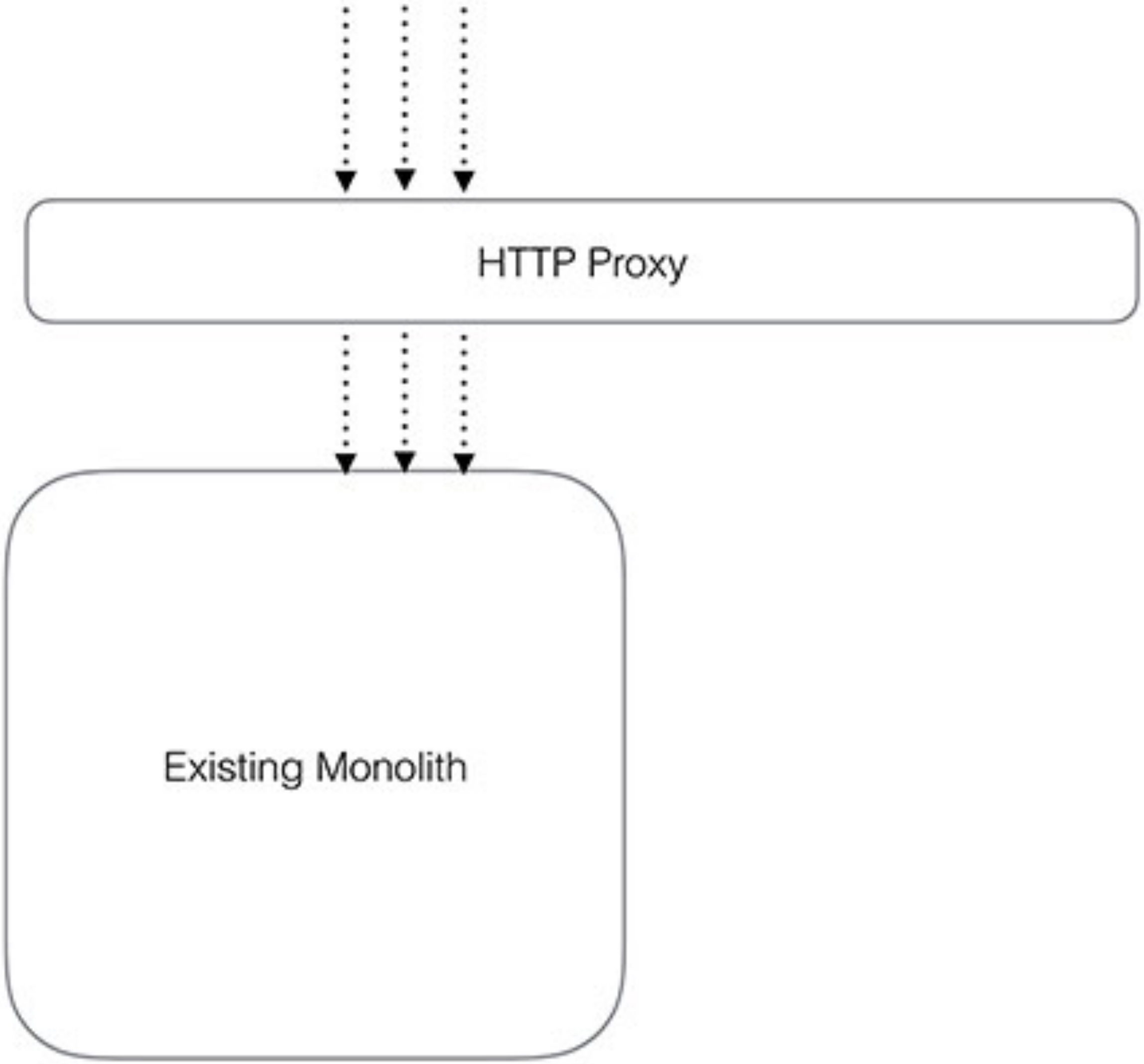


Existing Monolith

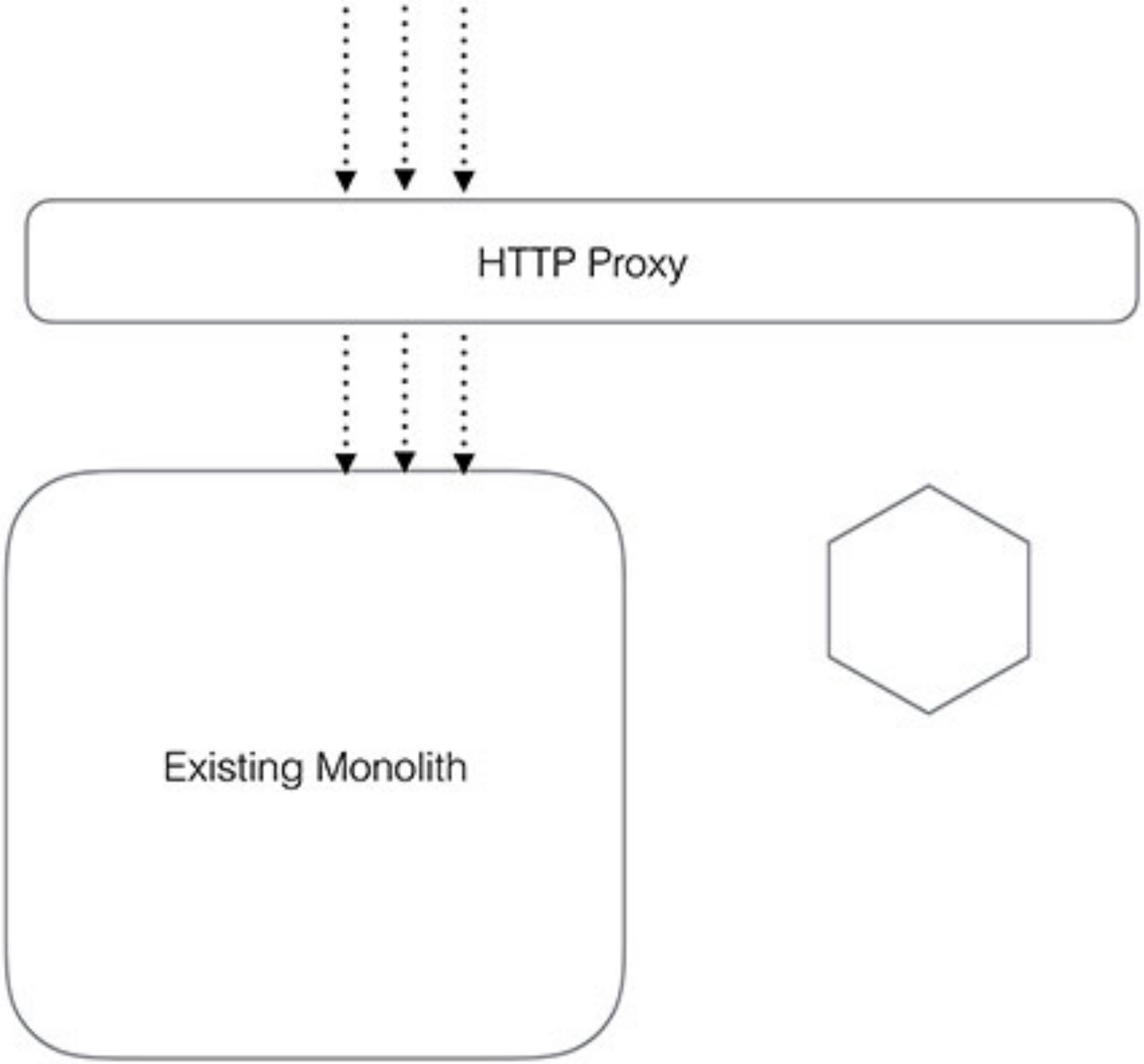
HTTP PROXY



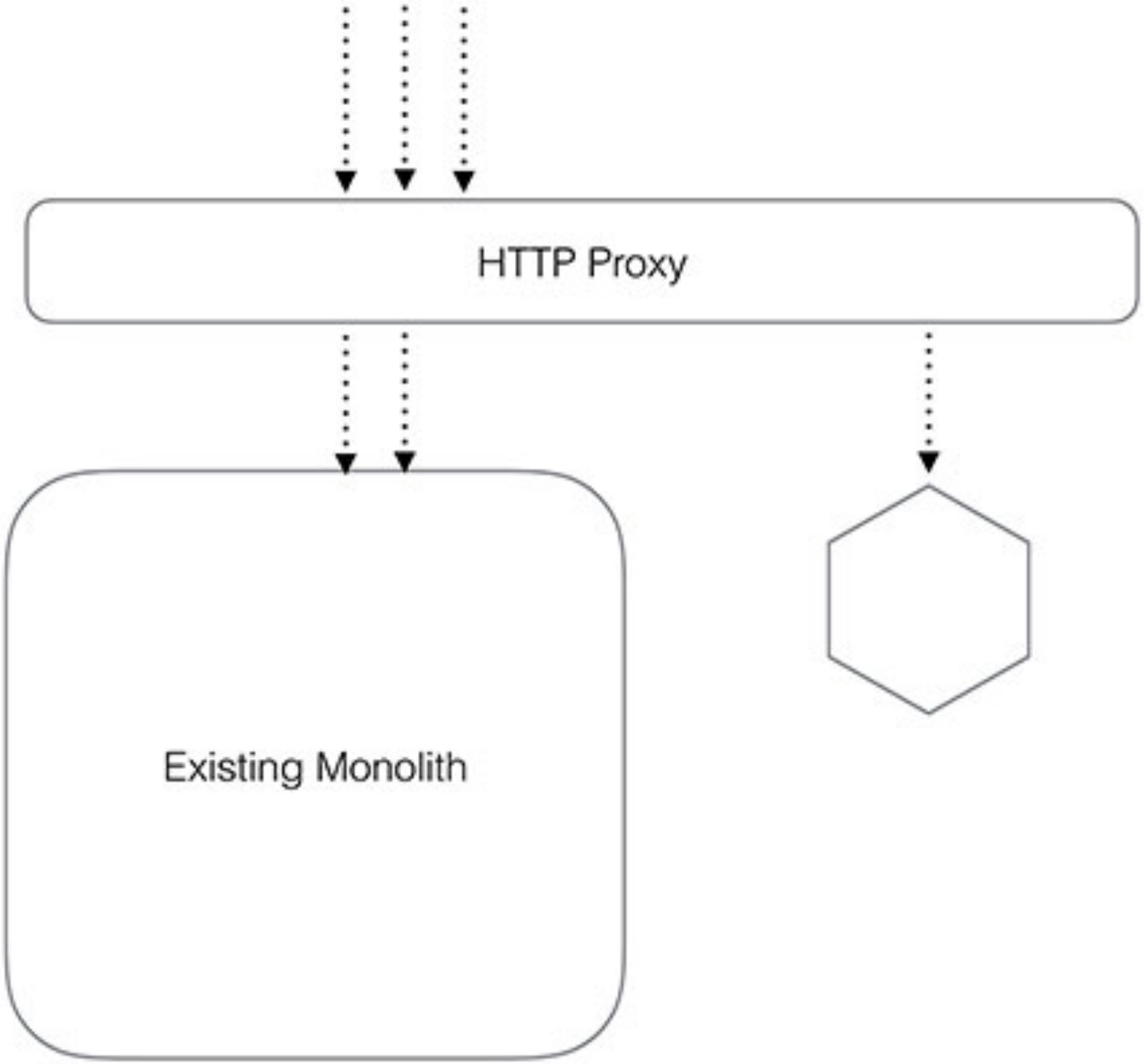
HTTP PROXY



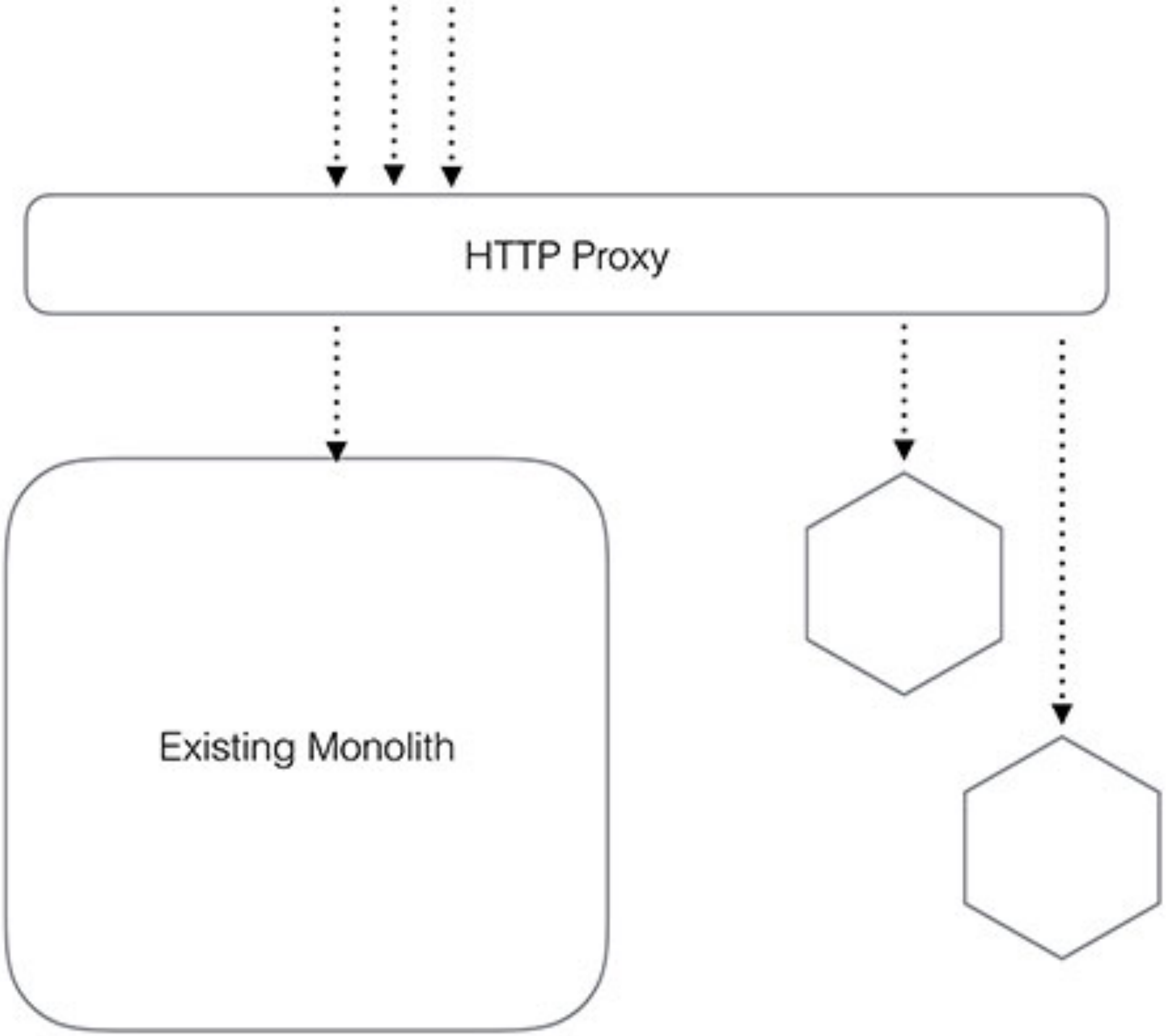
HTTP PROXY



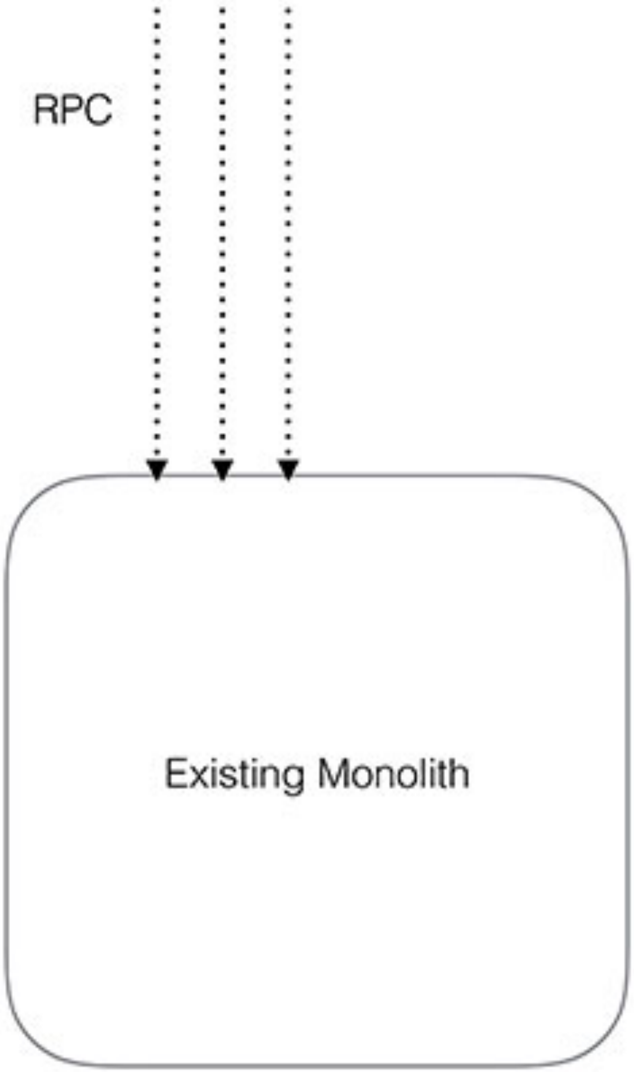
HTTP PROXY



HTTP PROXY



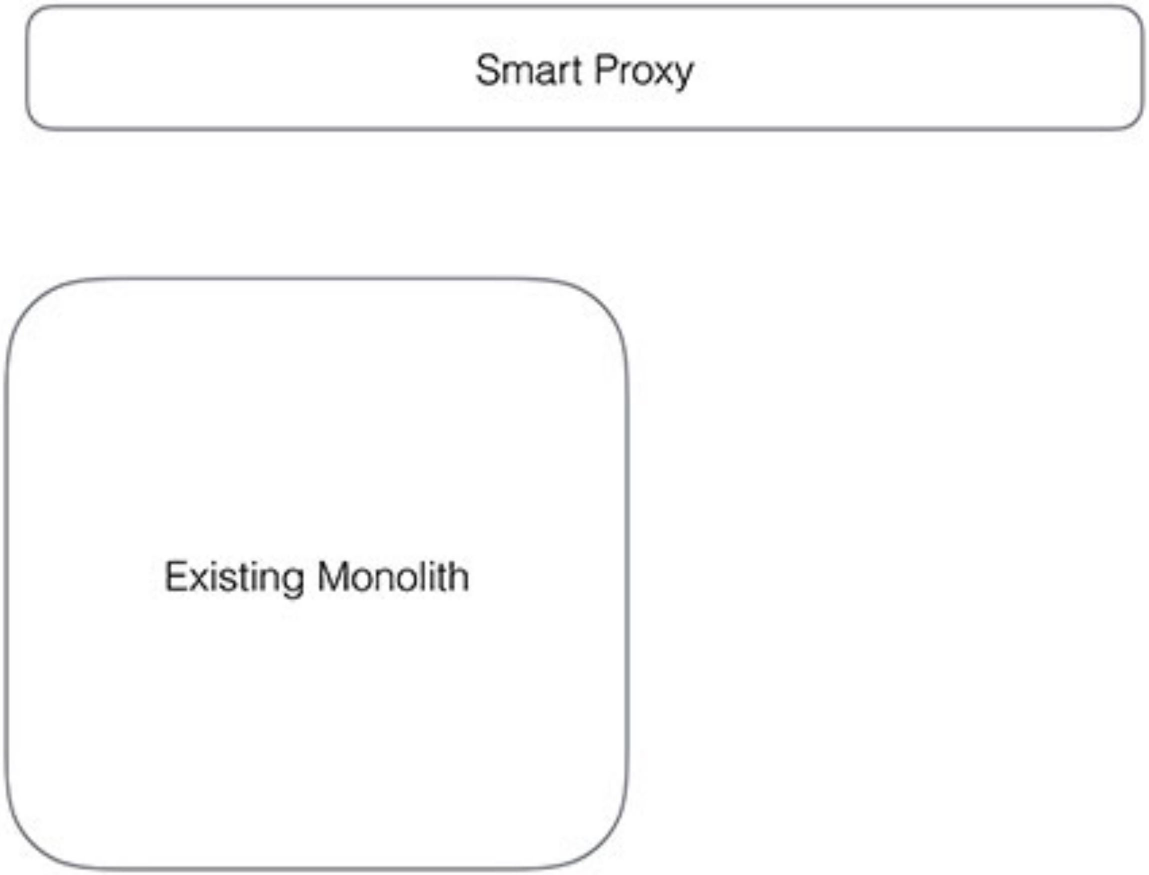
PROTOCOL CHANGING PROXY



RPC

Existing Monolith

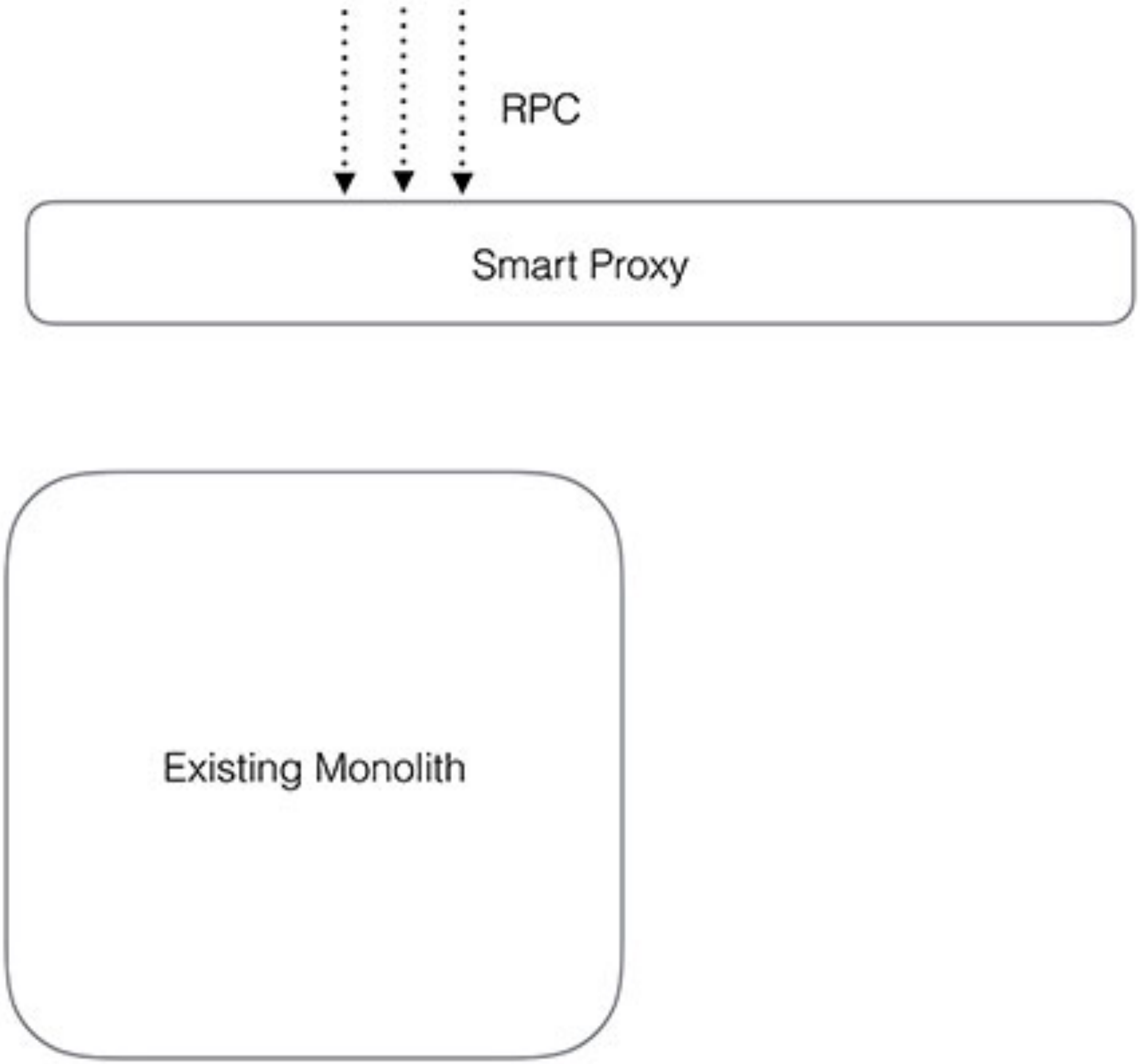
PROTOCOL CHANGING PROXY



Smart Proxy

Existing Monolith

PROTOCOL CHANGING PROXY

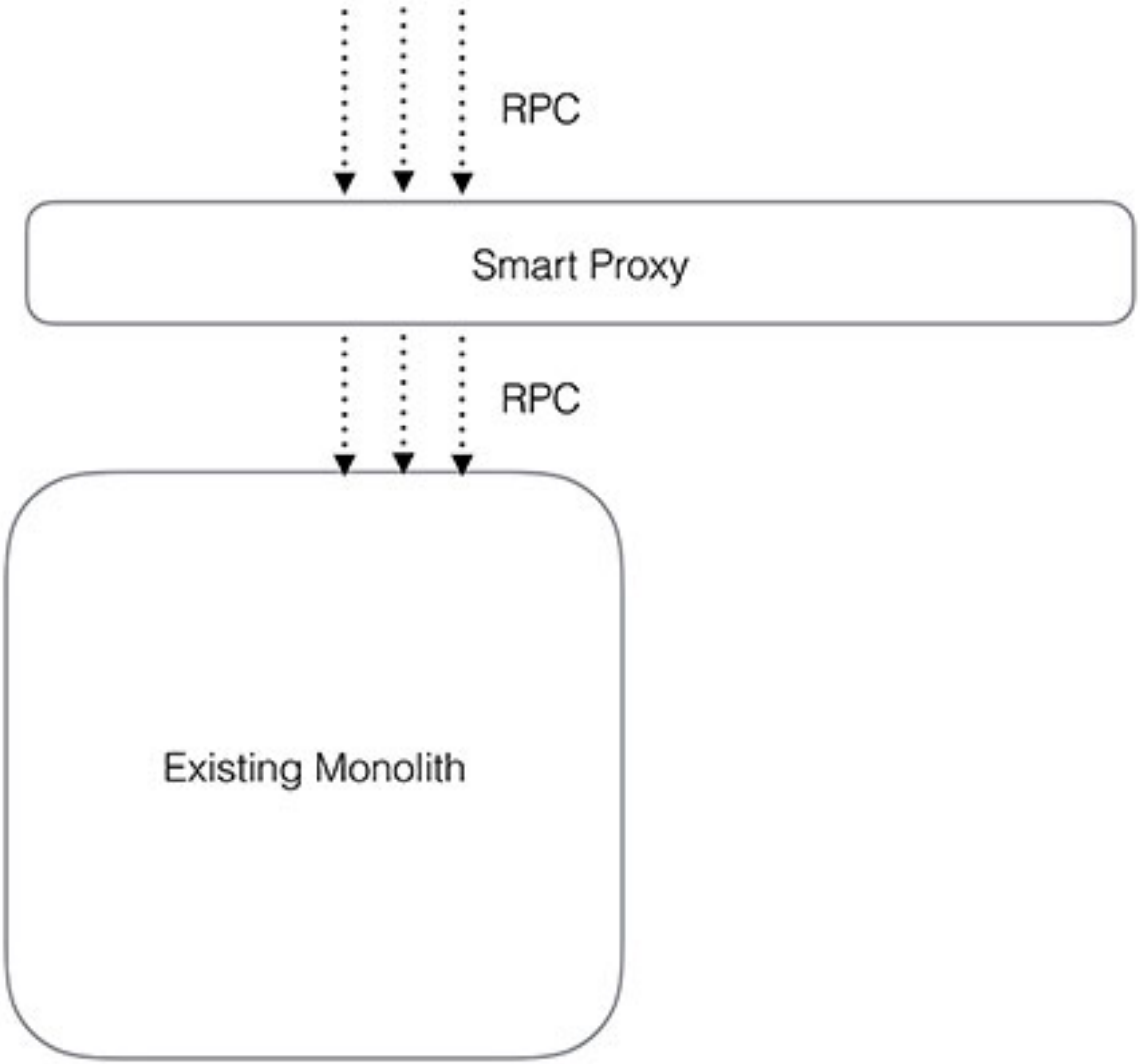


RPC

Smart Proxy

Existing Monolith

PROTOCOL CHANGING PROXY



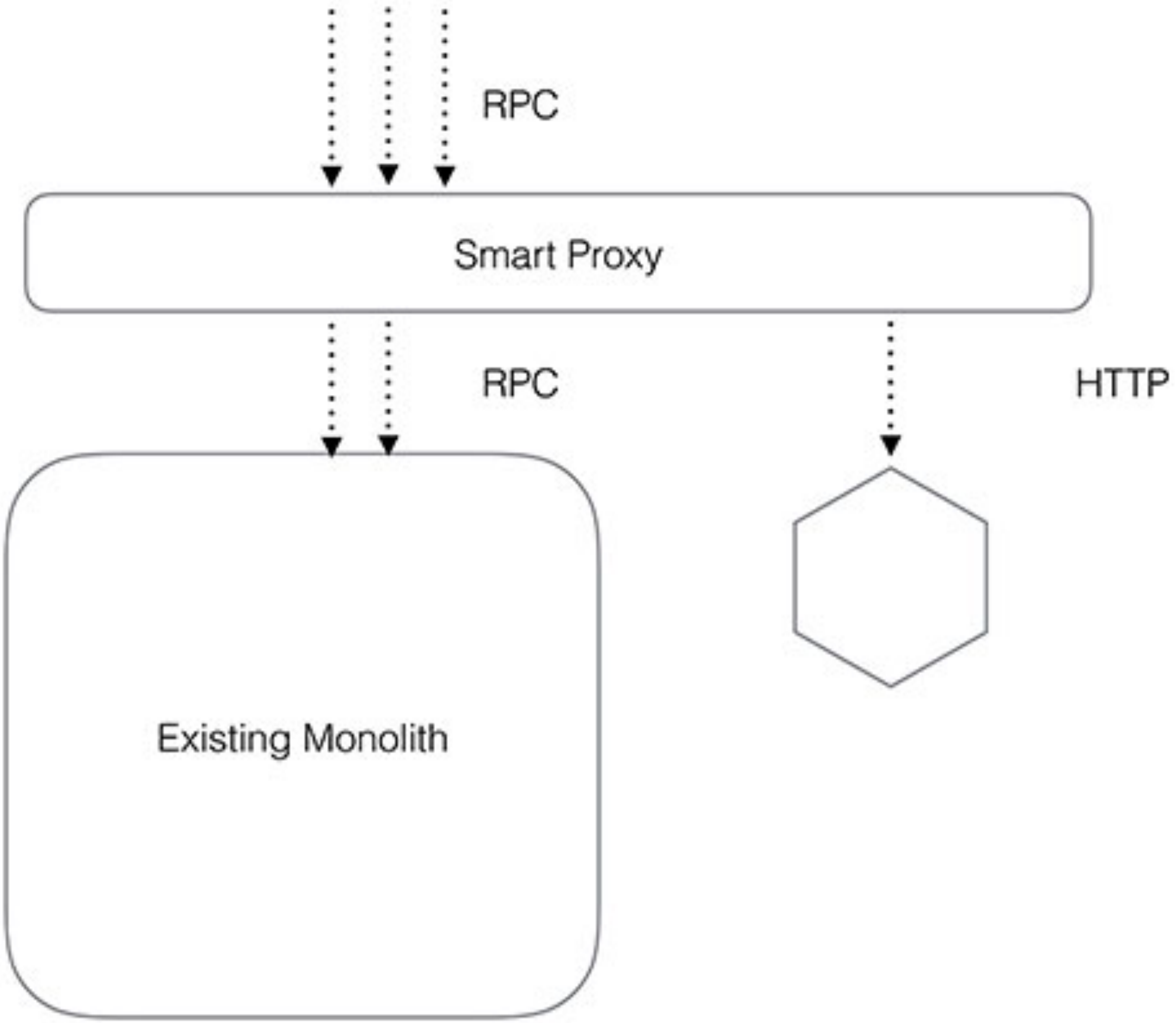
RPC

Smart Proxy

RPC

Existing Monolith

PROTOCOL CHANGING PROXY



RPC

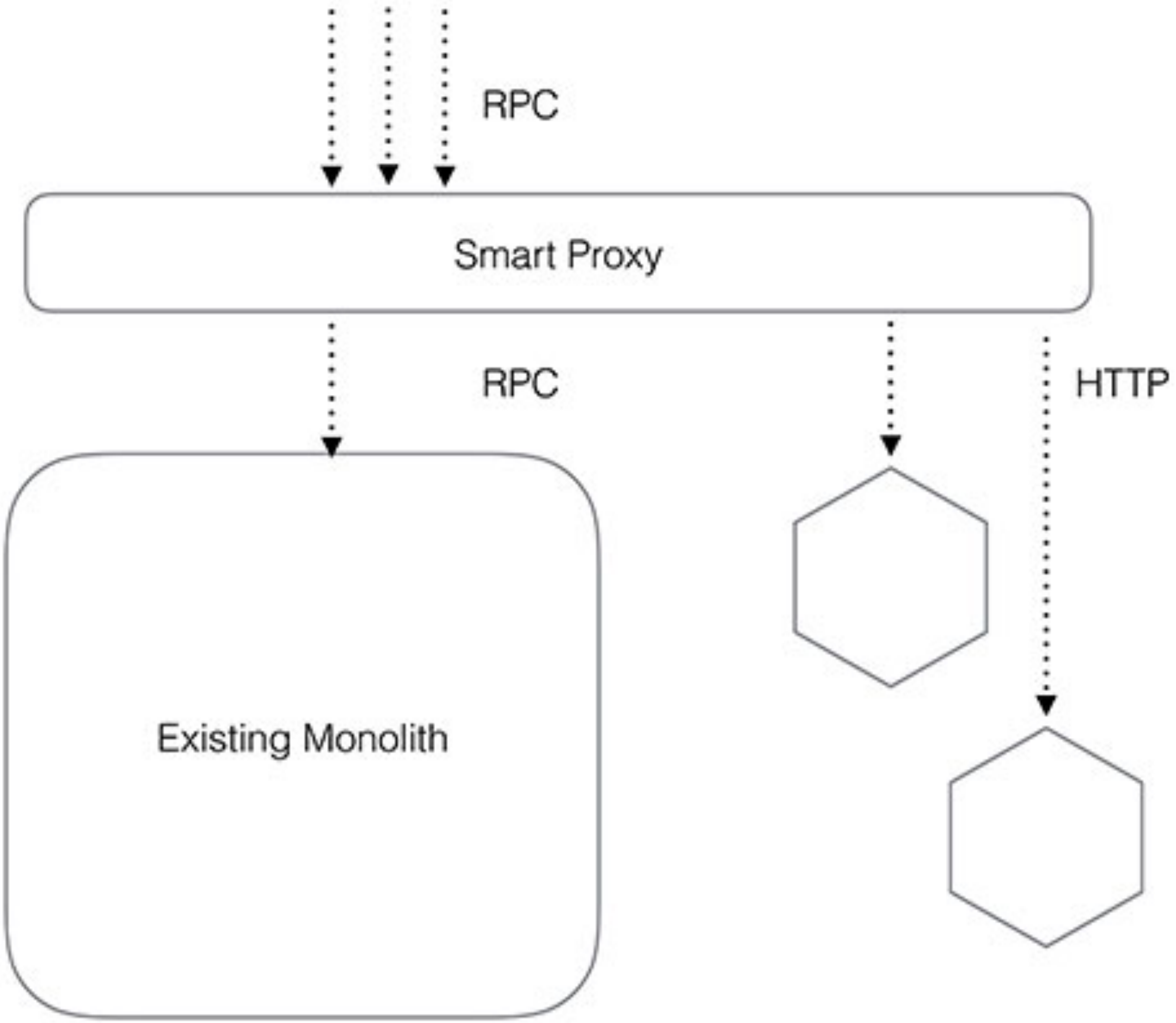
Smart Proxy

RPC

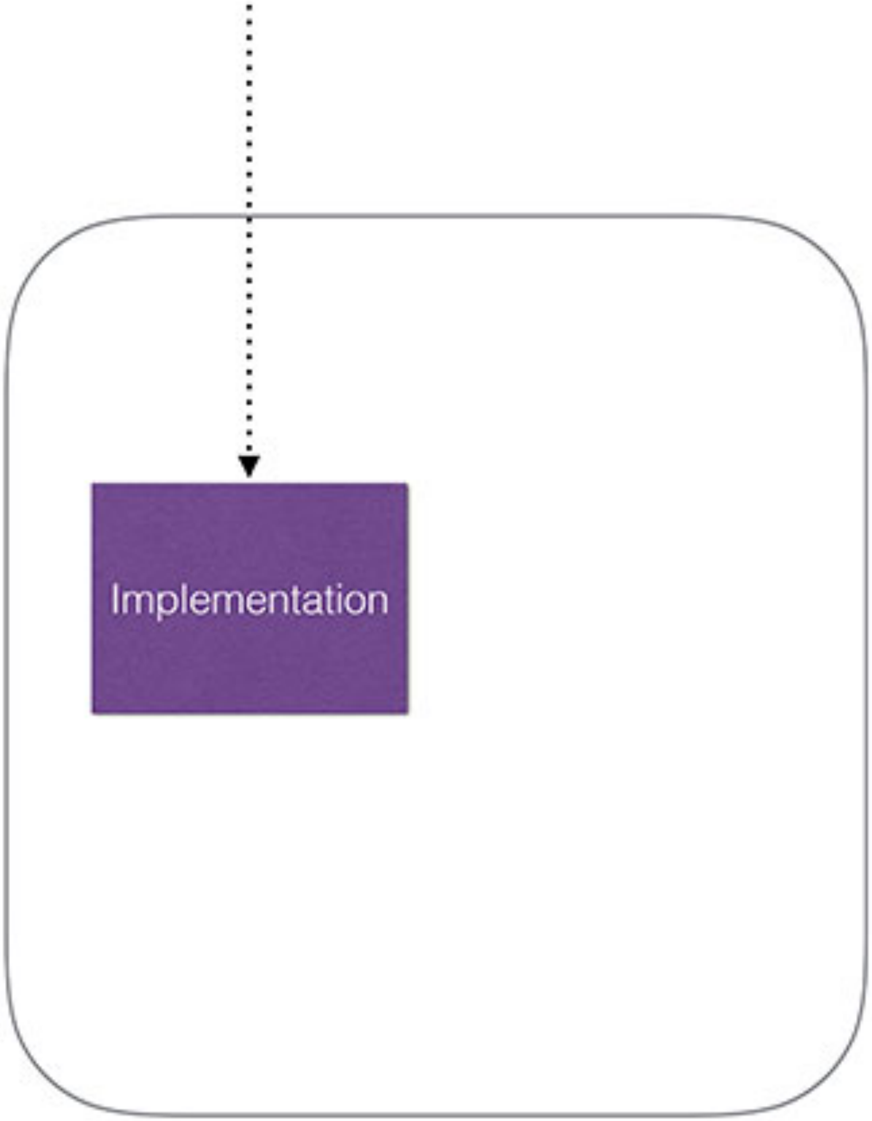
HTTP

Existing Monolith

PROTOCOL CHANGING PROXY

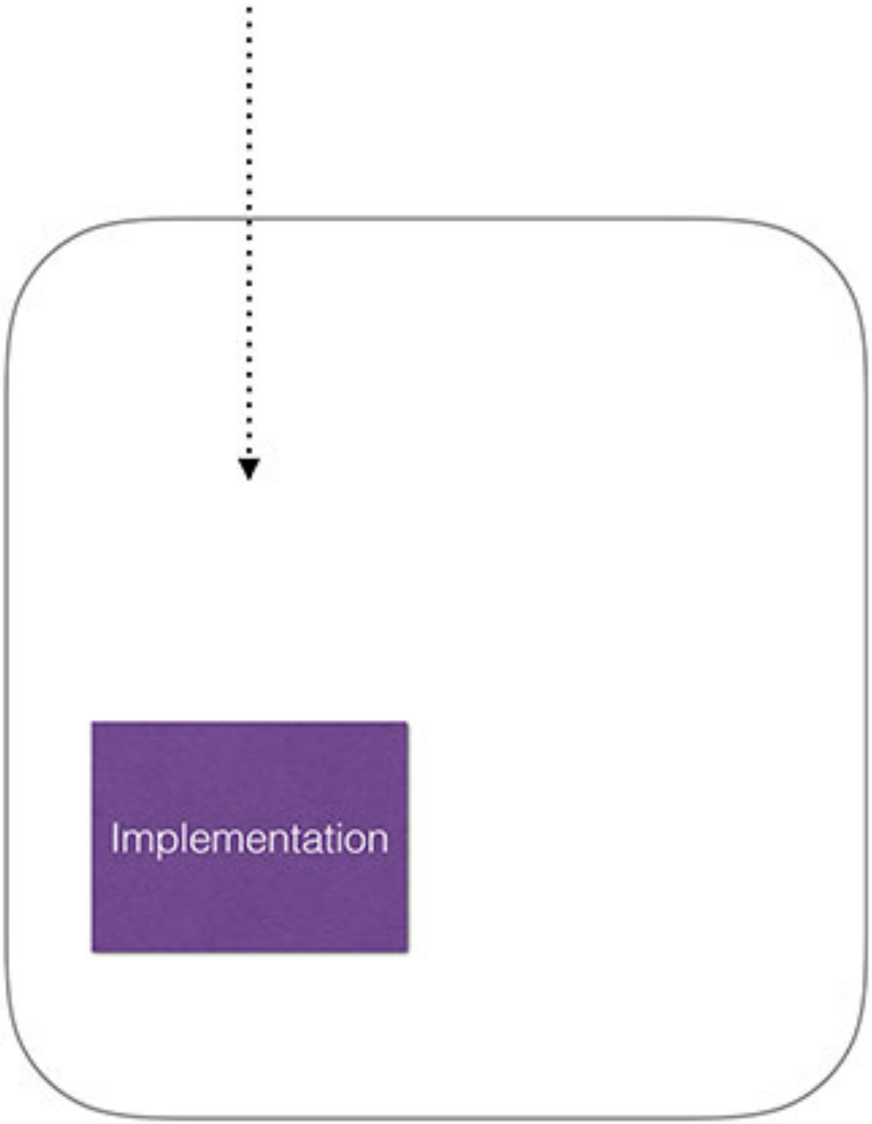


INTERNAL ABSTRACTION

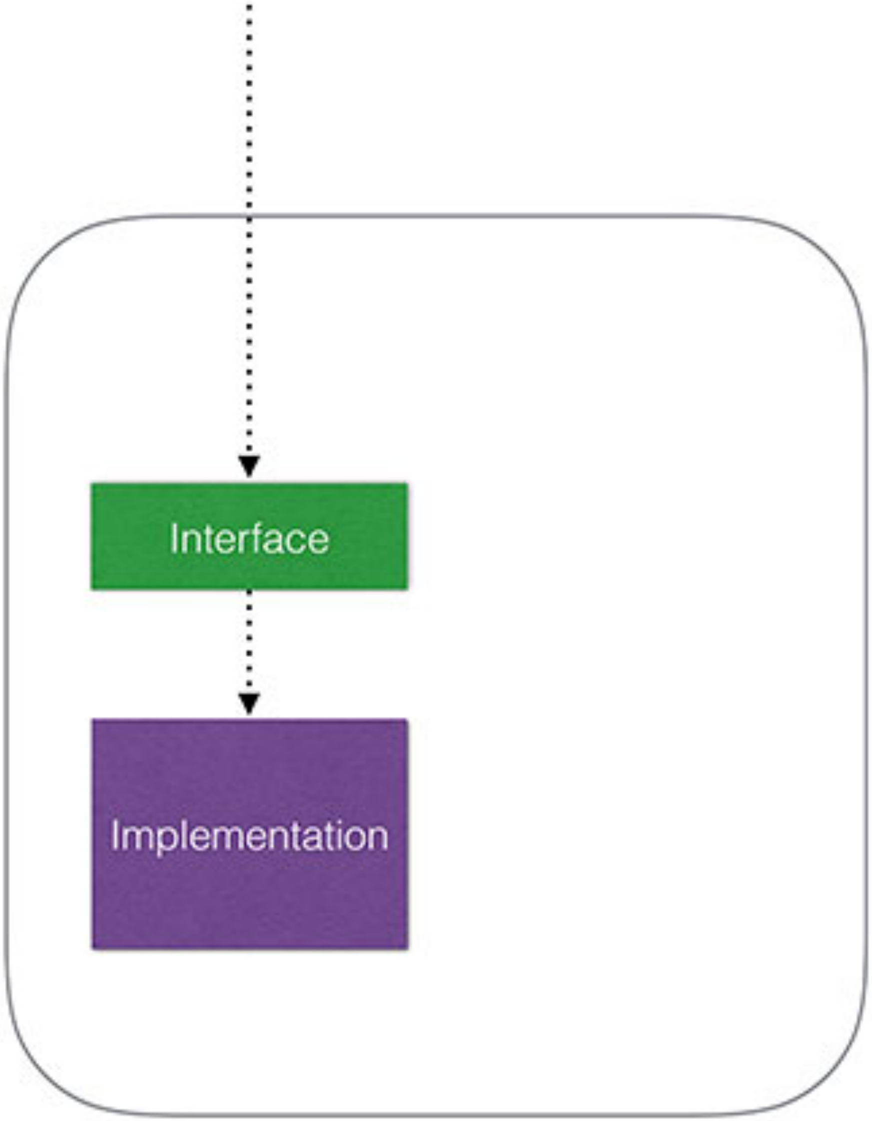


Implementation

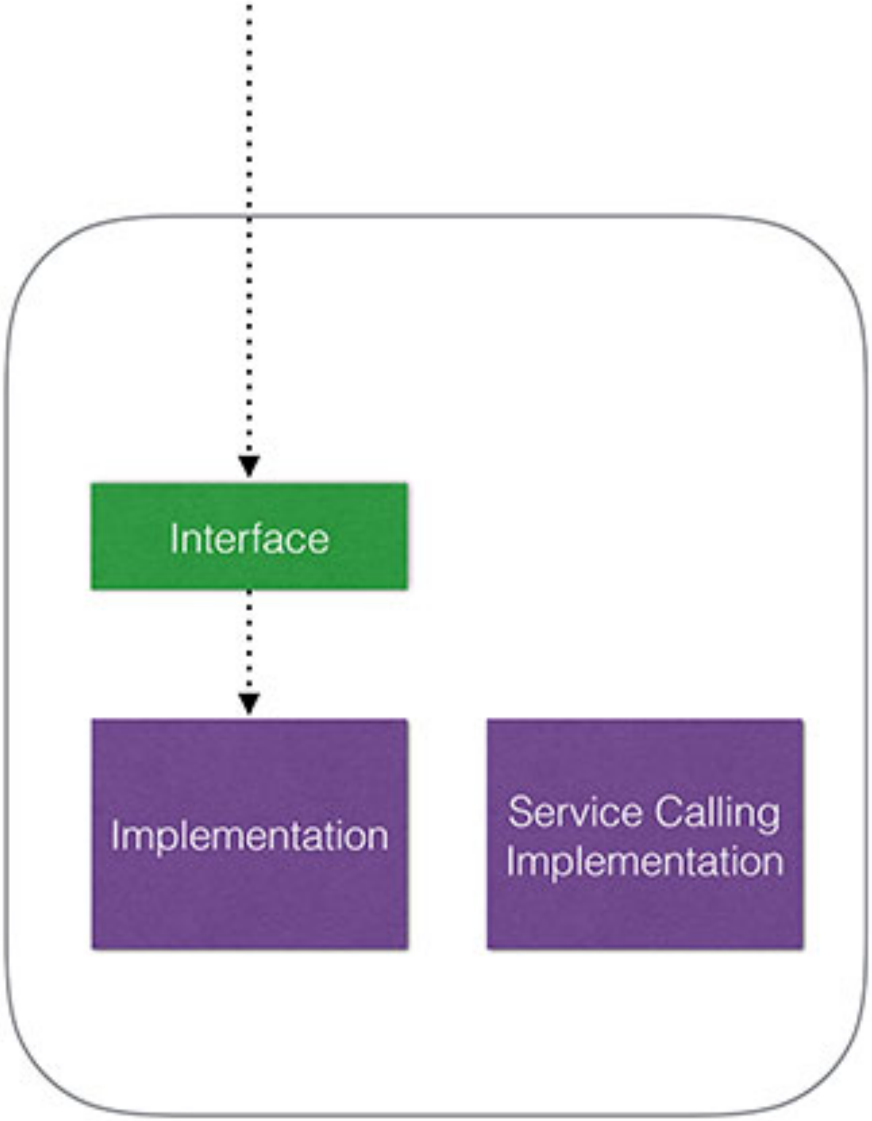
INTERNAL ABSTRACTION



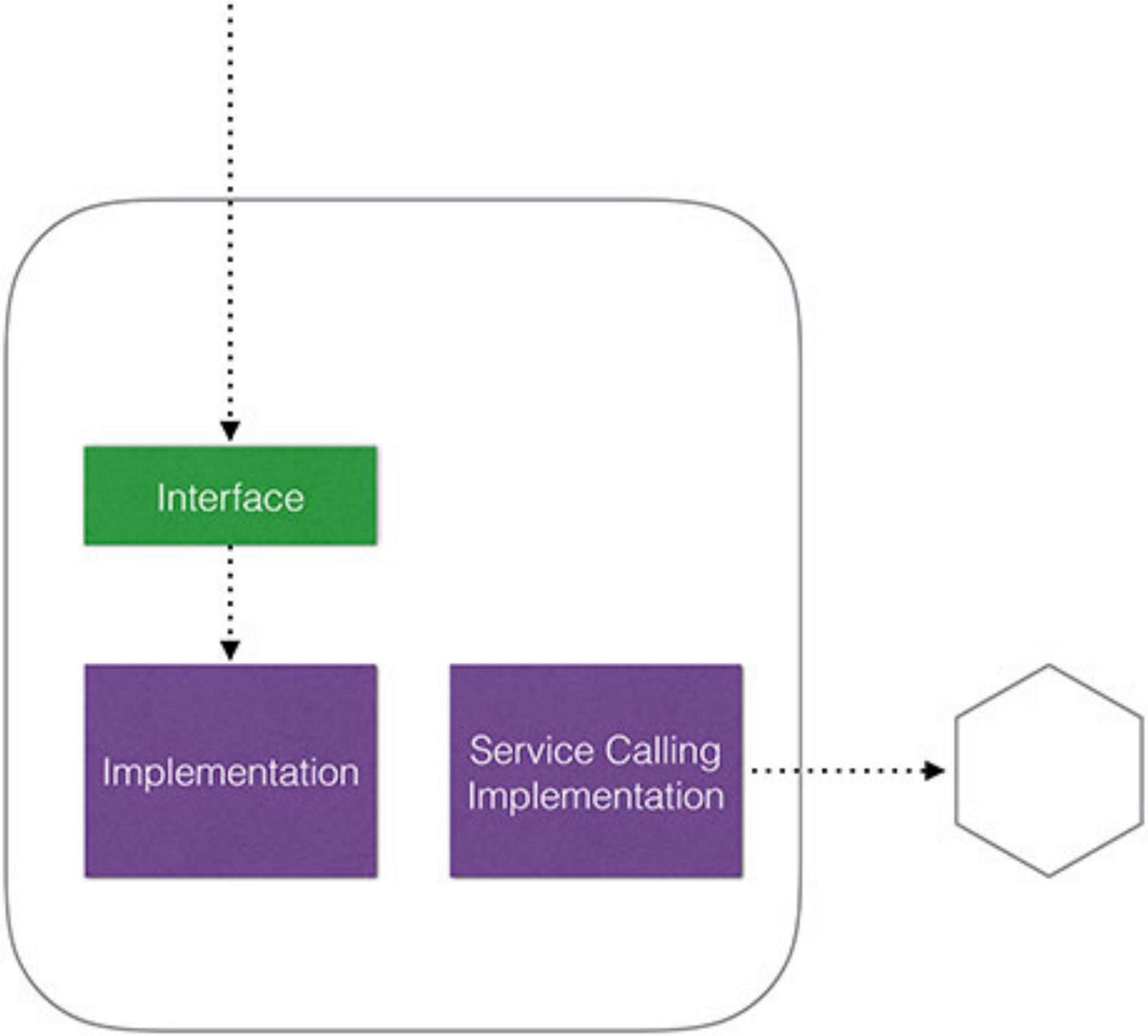
INTERNAL ABSTRACTION



INTERNAL ABSTRACTION



INTERNAL ABSTRACTION

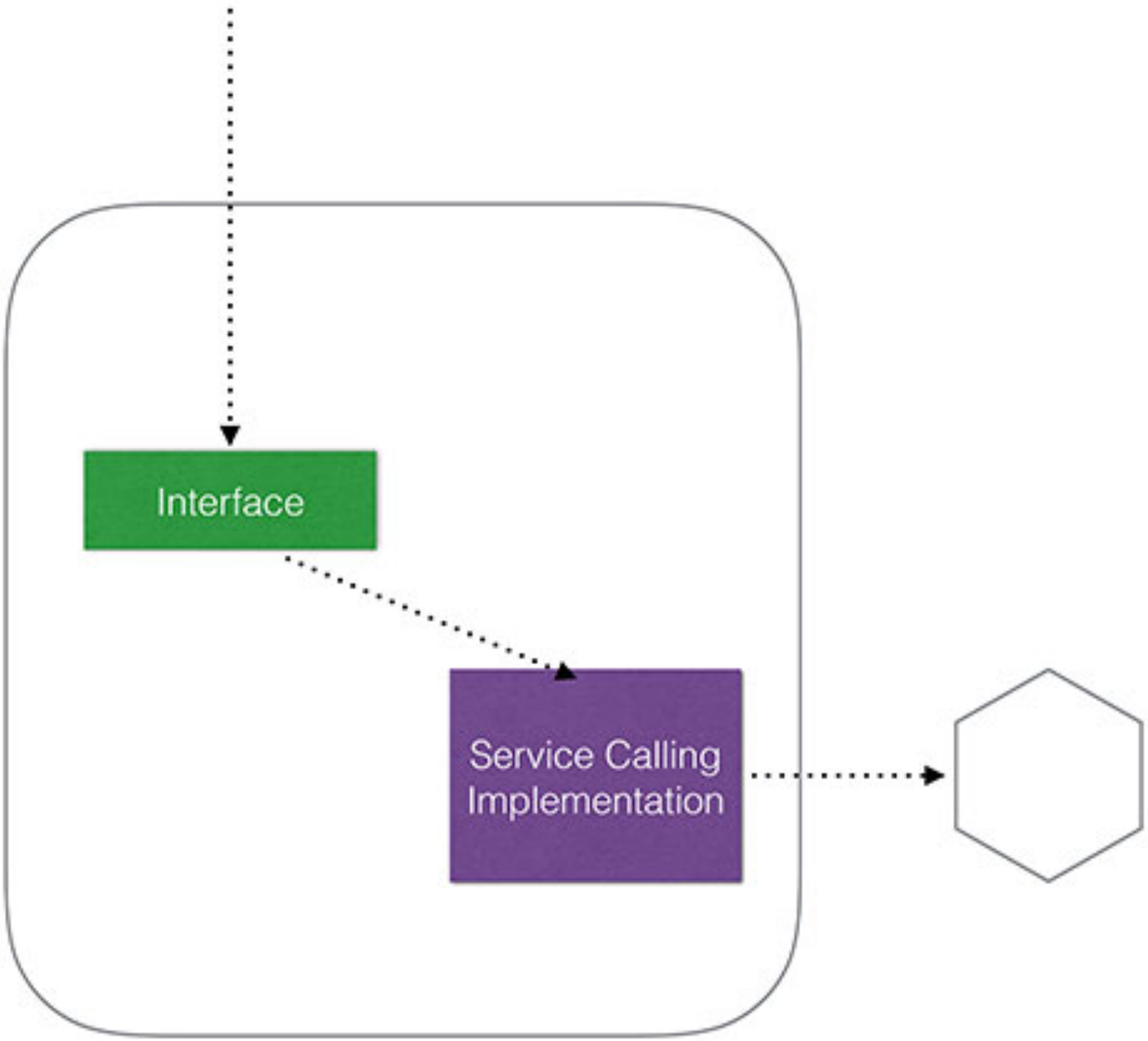


Interface

Implementation

Service Calling Implementation

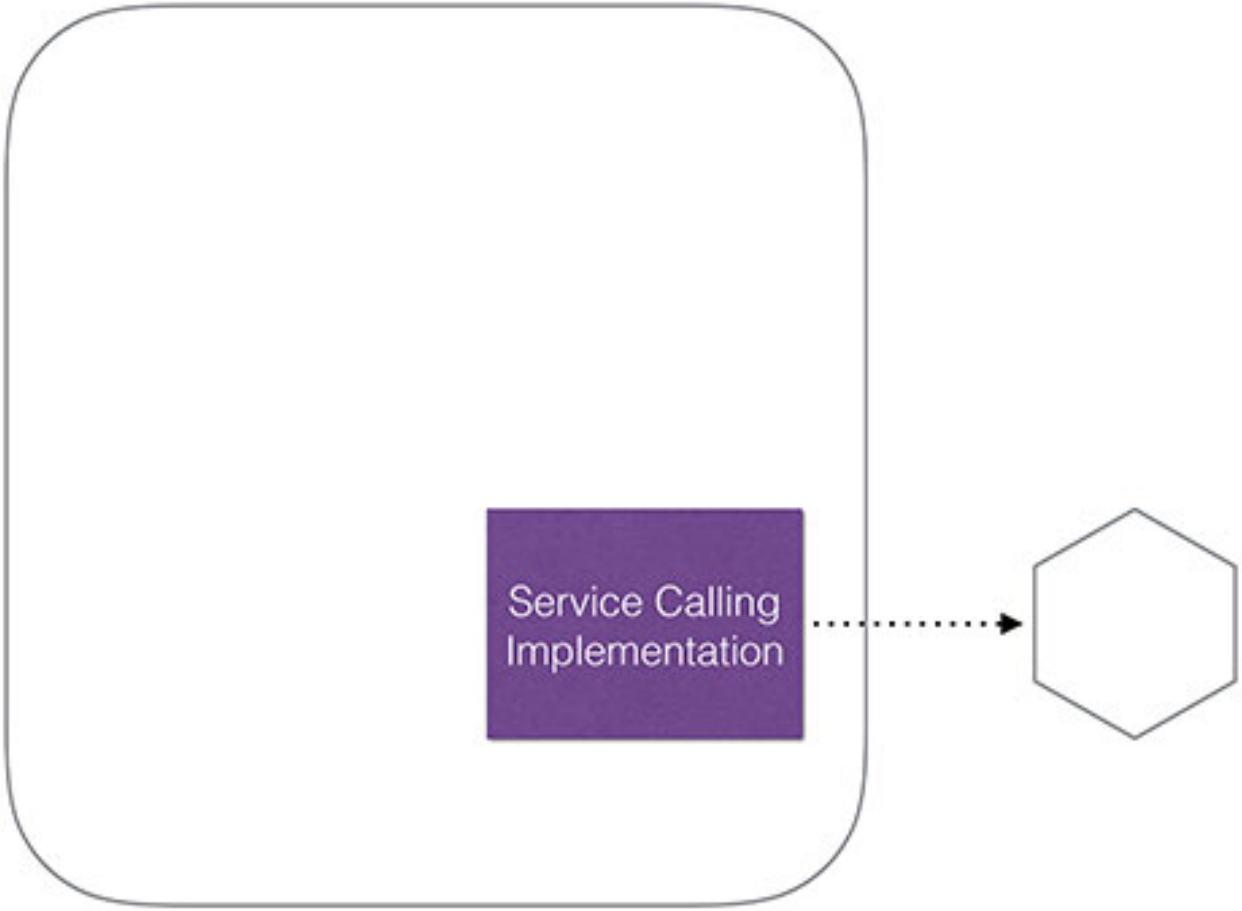
INTERNAL ABSTRACTION



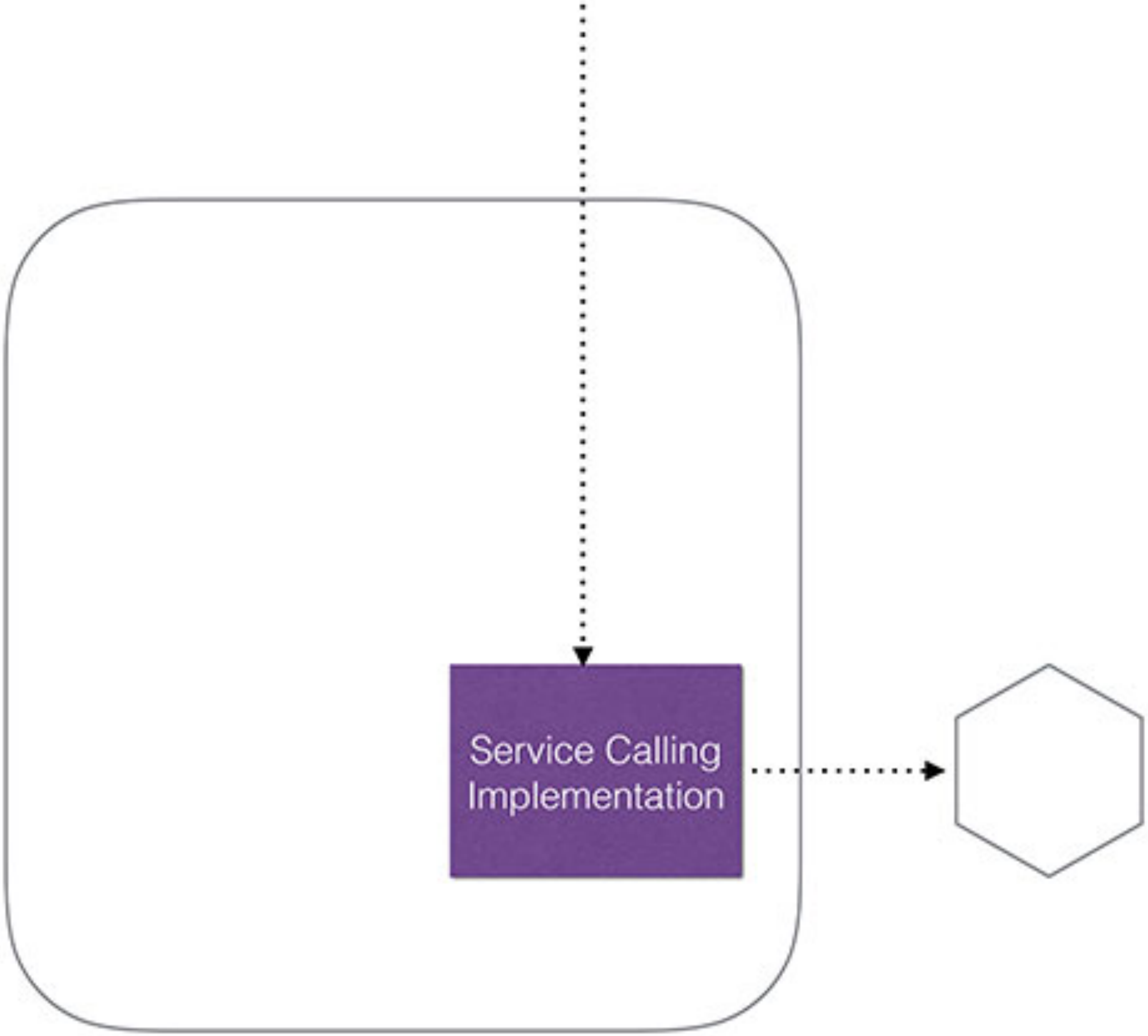
Interface

Service Calling Implementation

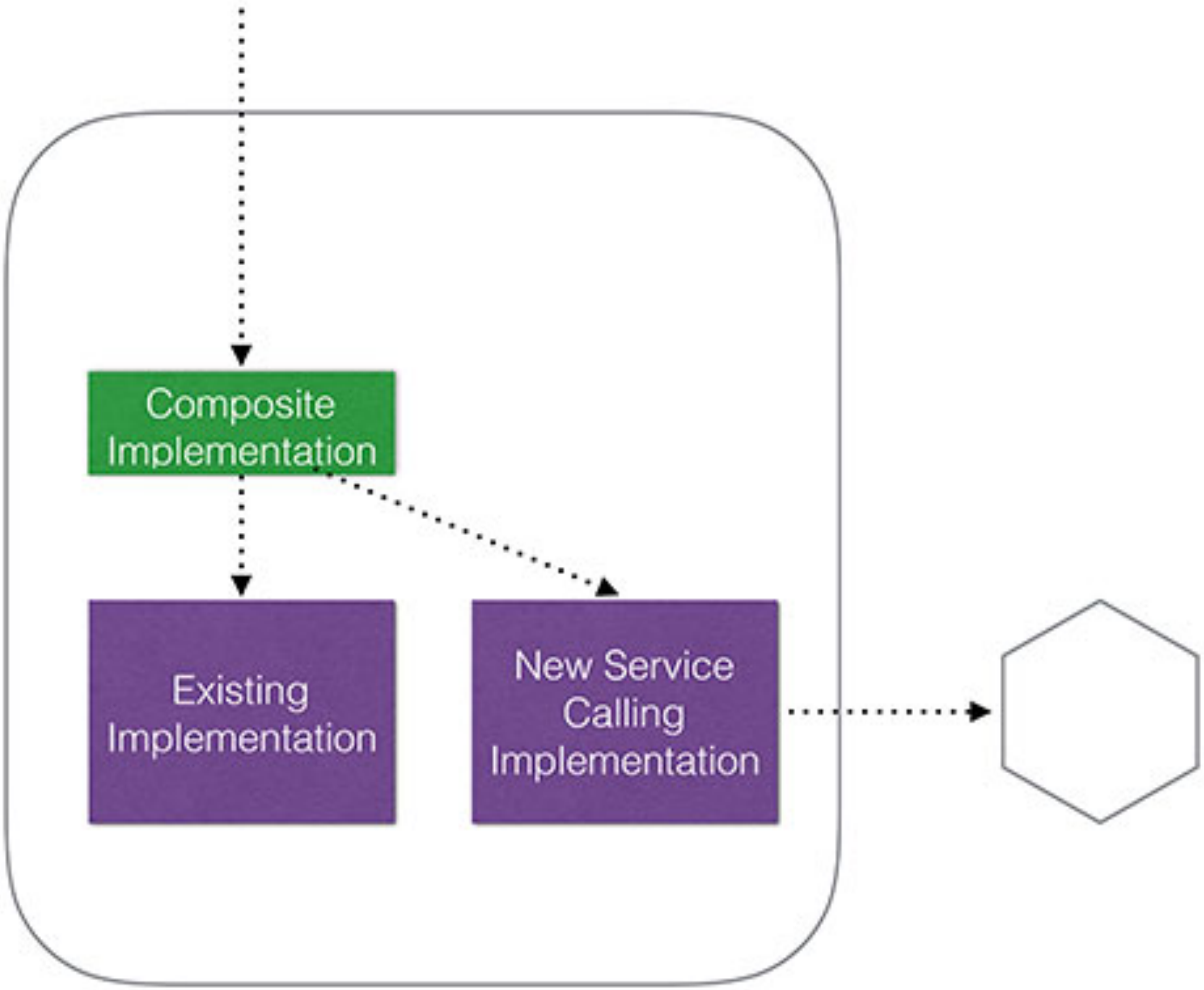
INTERNAL ABSTRACTION



INTERNAL ABSTRACTION



CO-EXISTING INTERNAL ABSTRACTIONS

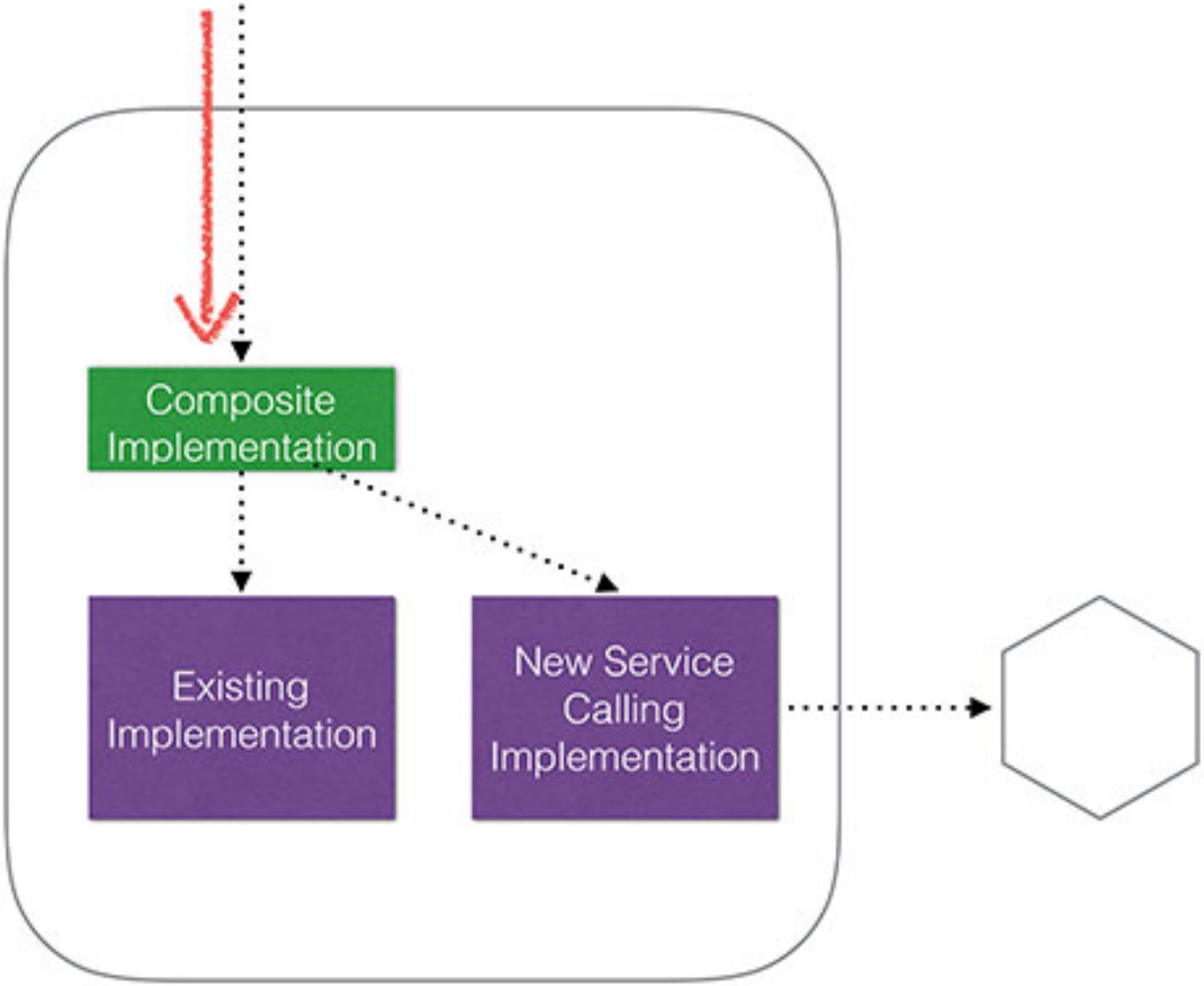


Composite Implementation

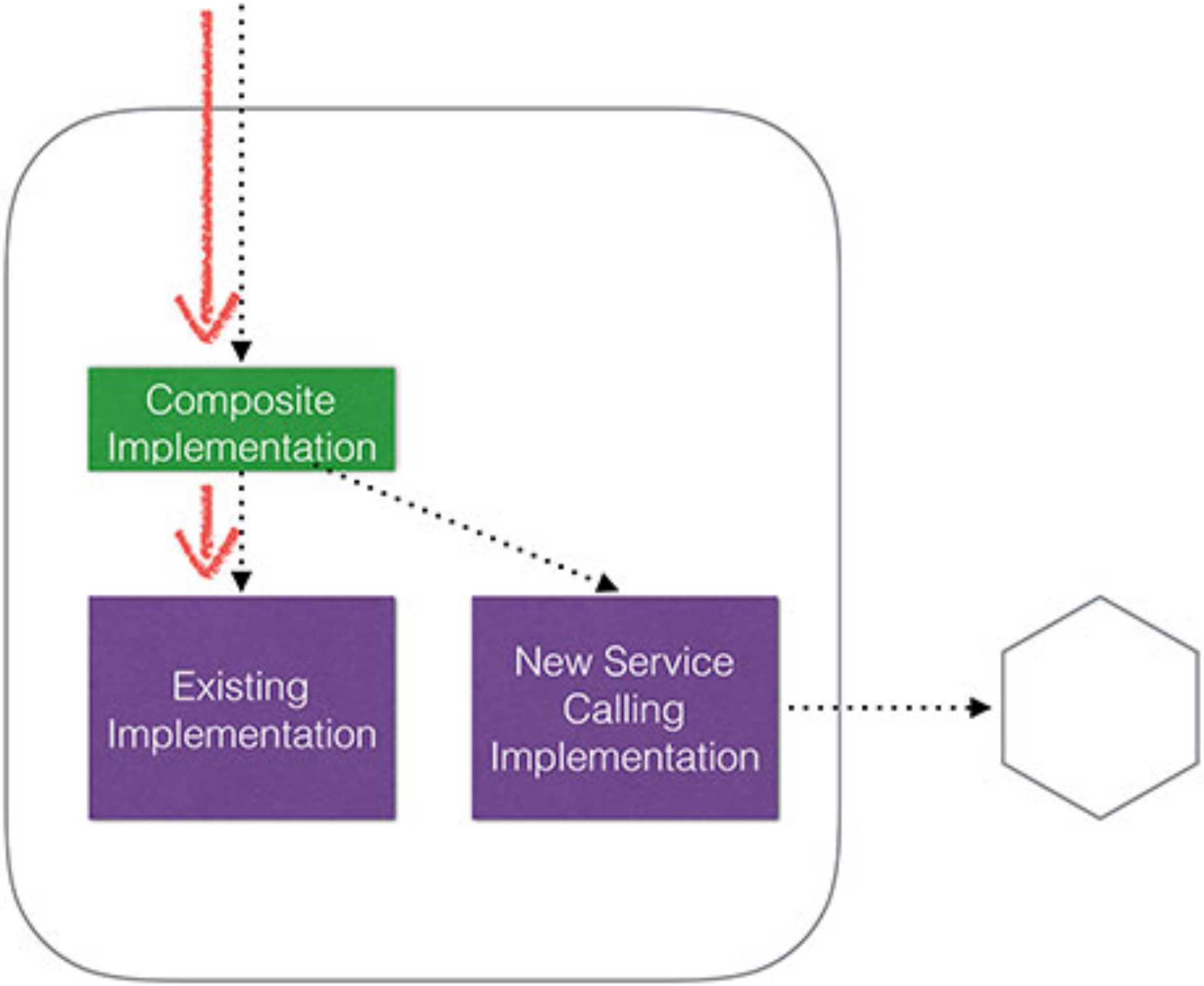
Existing Implementation

New Service Calling Implementation

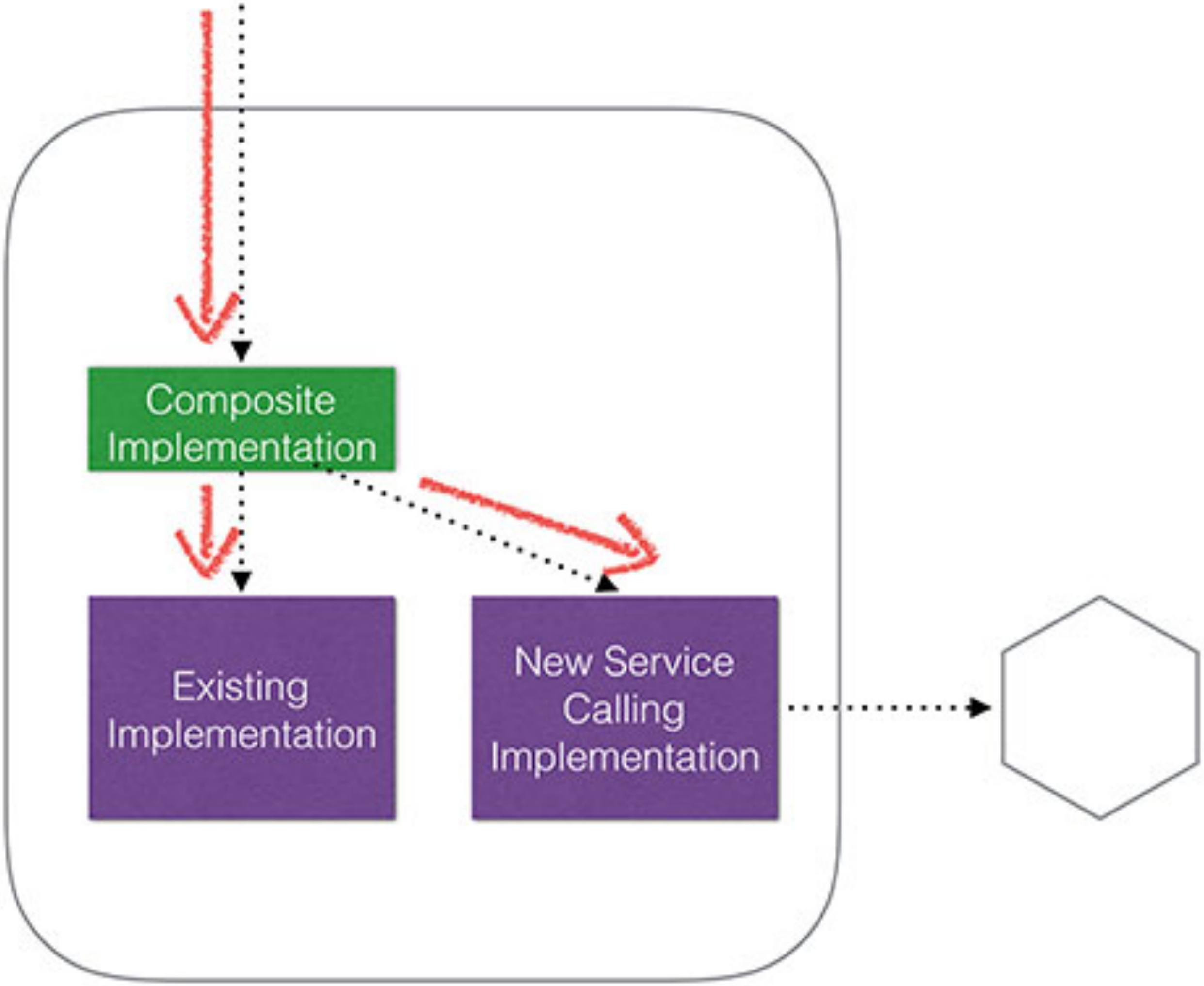
CO-EXISTING INTERNAL ABSTRACTIONS



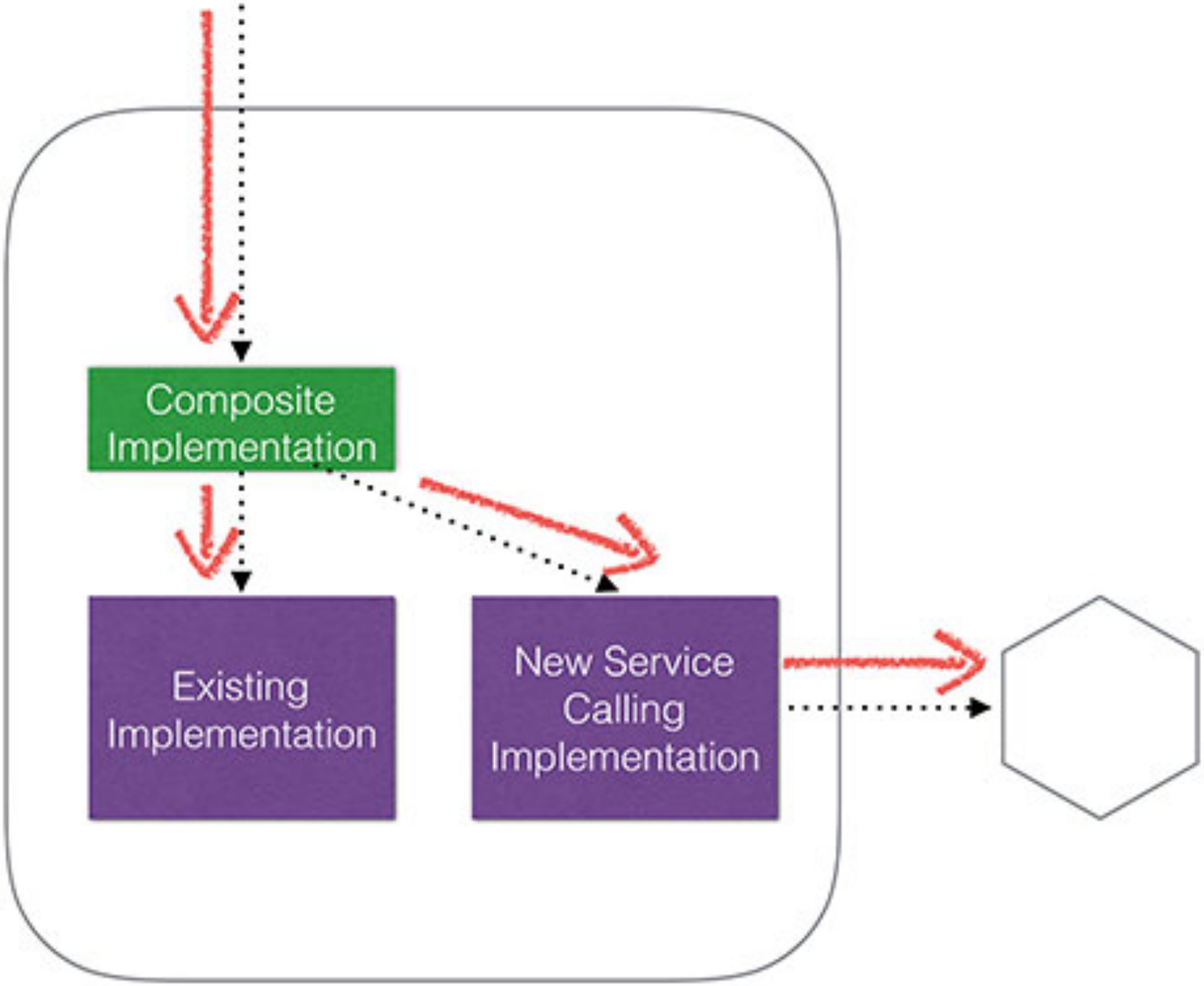
CO-EXISTING INTERNAL ABSTRACTIONS



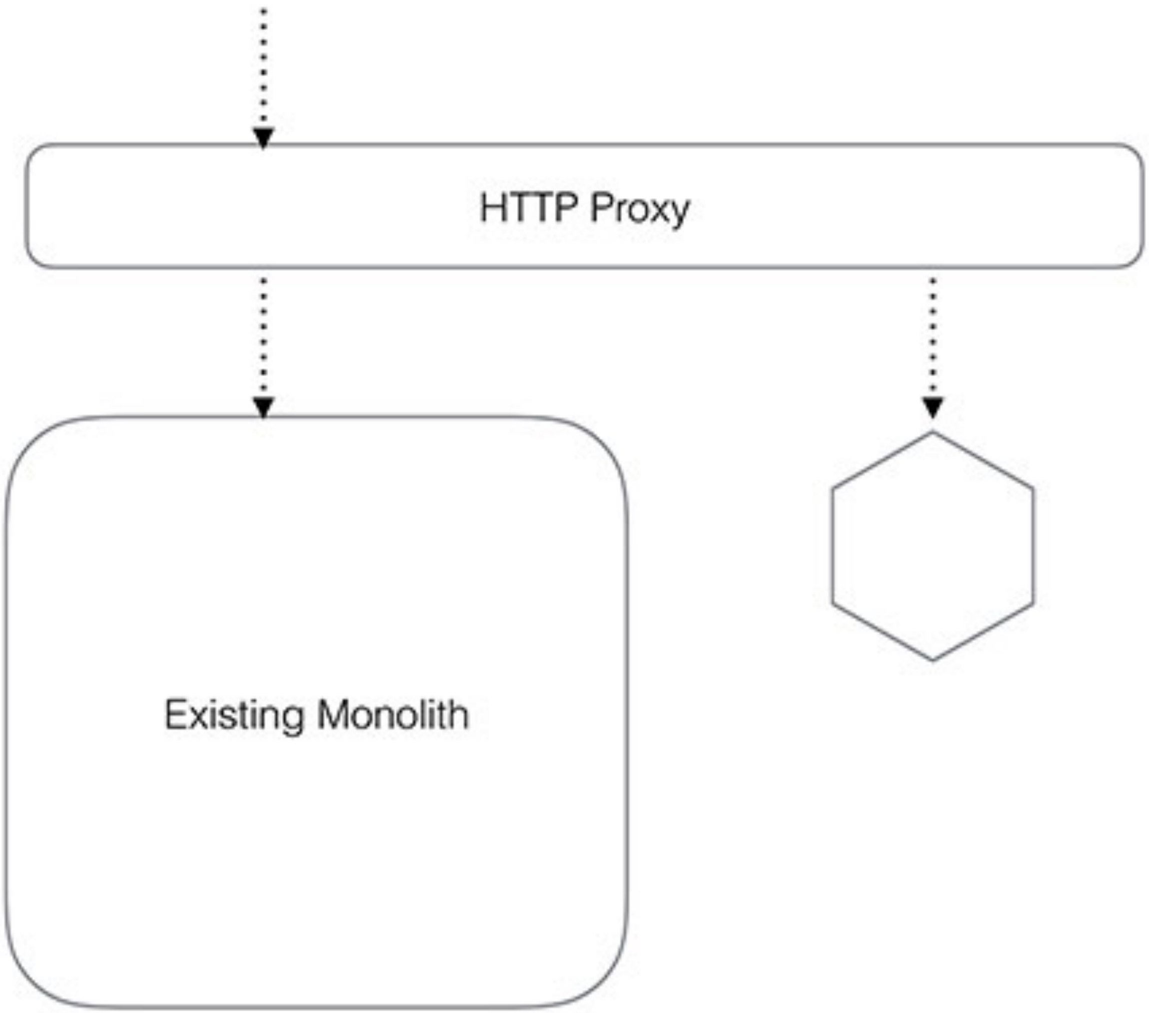
CO-EXISTING INTERNAL ABSTRACTIONS



CO-EXISTING INTERNAL ABSTRACTIONS



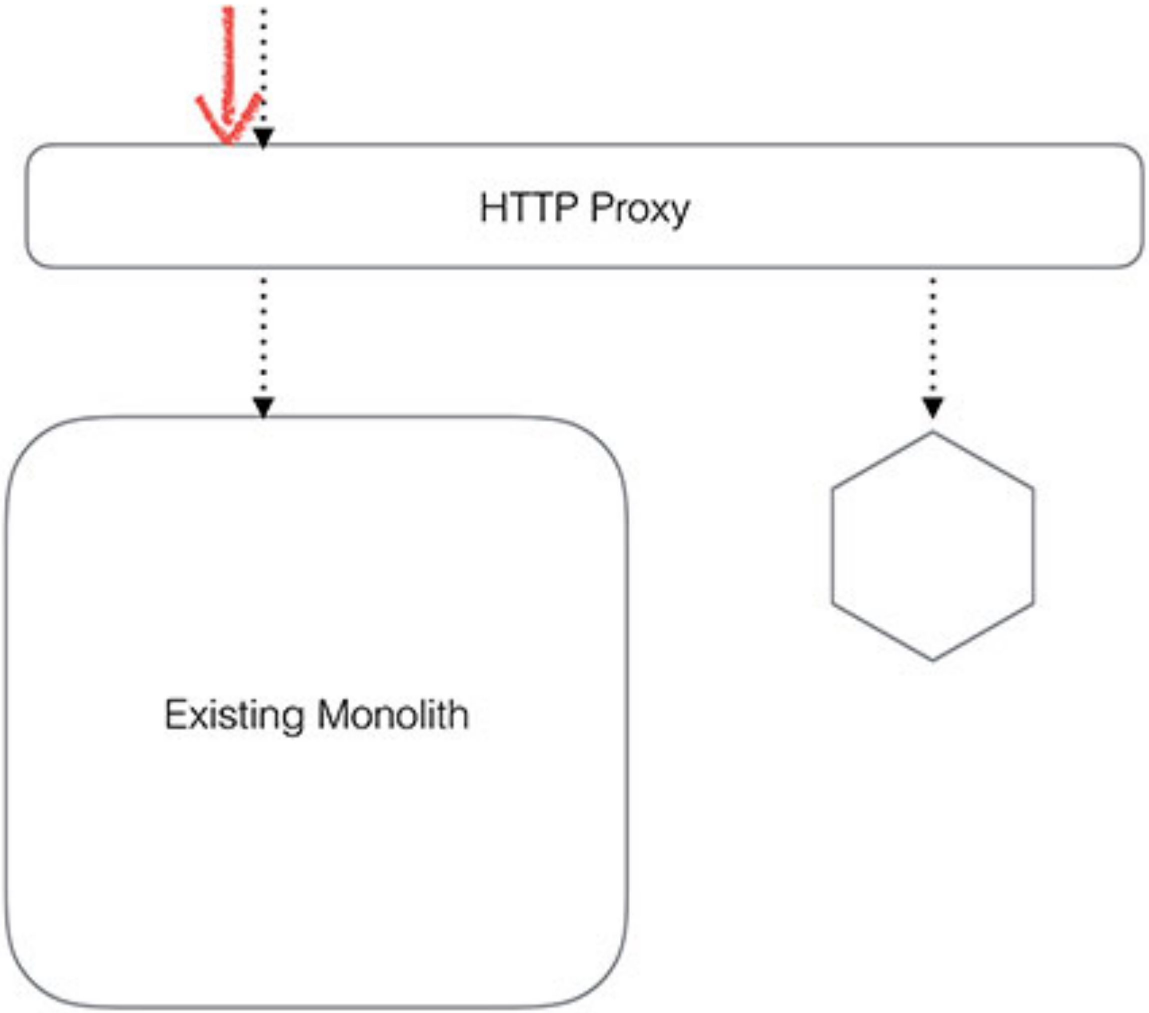
CO-EXISTING PROXY IMPLEMENTATIONS



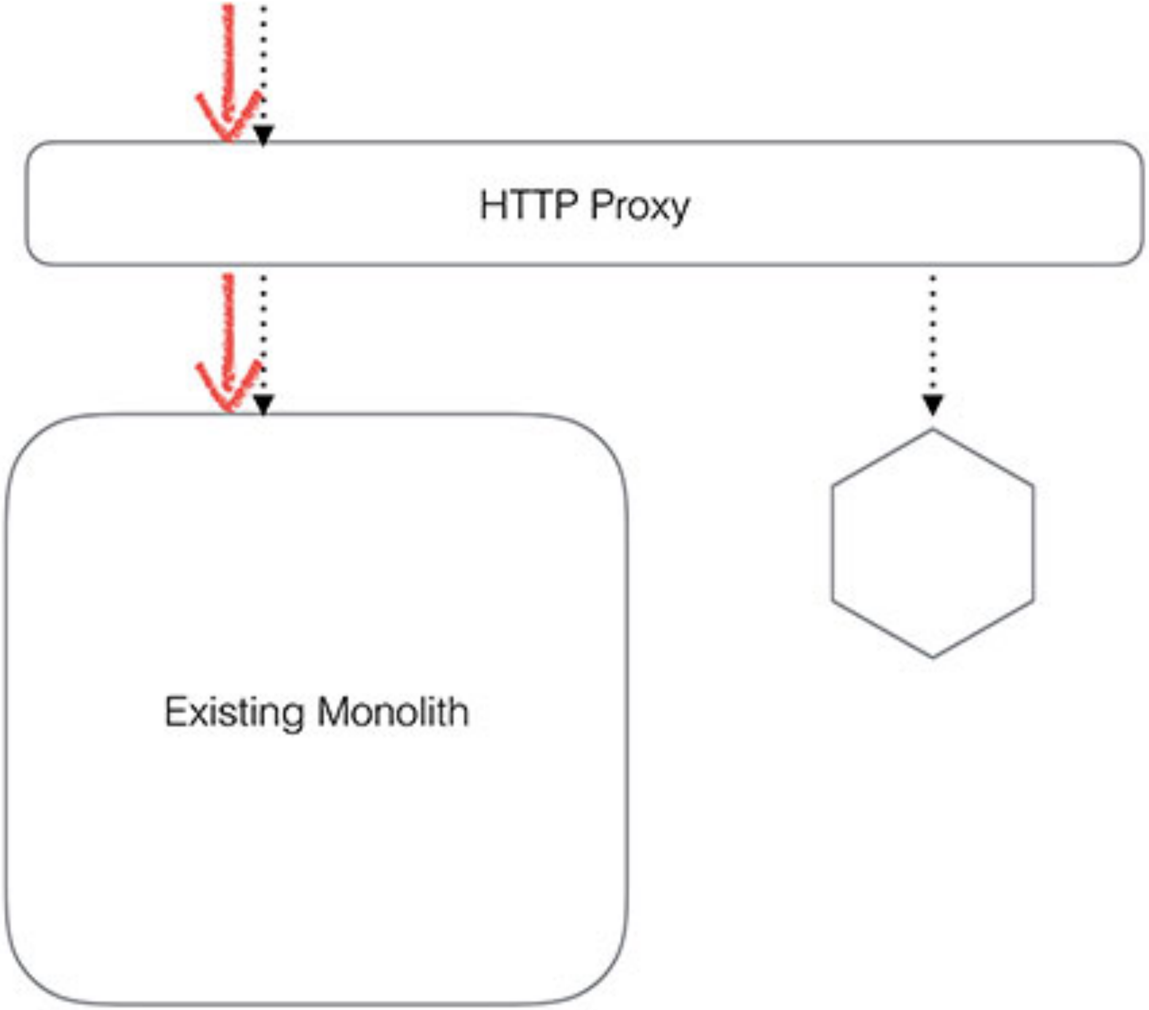
HTTP Proxy

Existing Monolith

CO-EXISTING PROXY IMPLEMENTATIONS



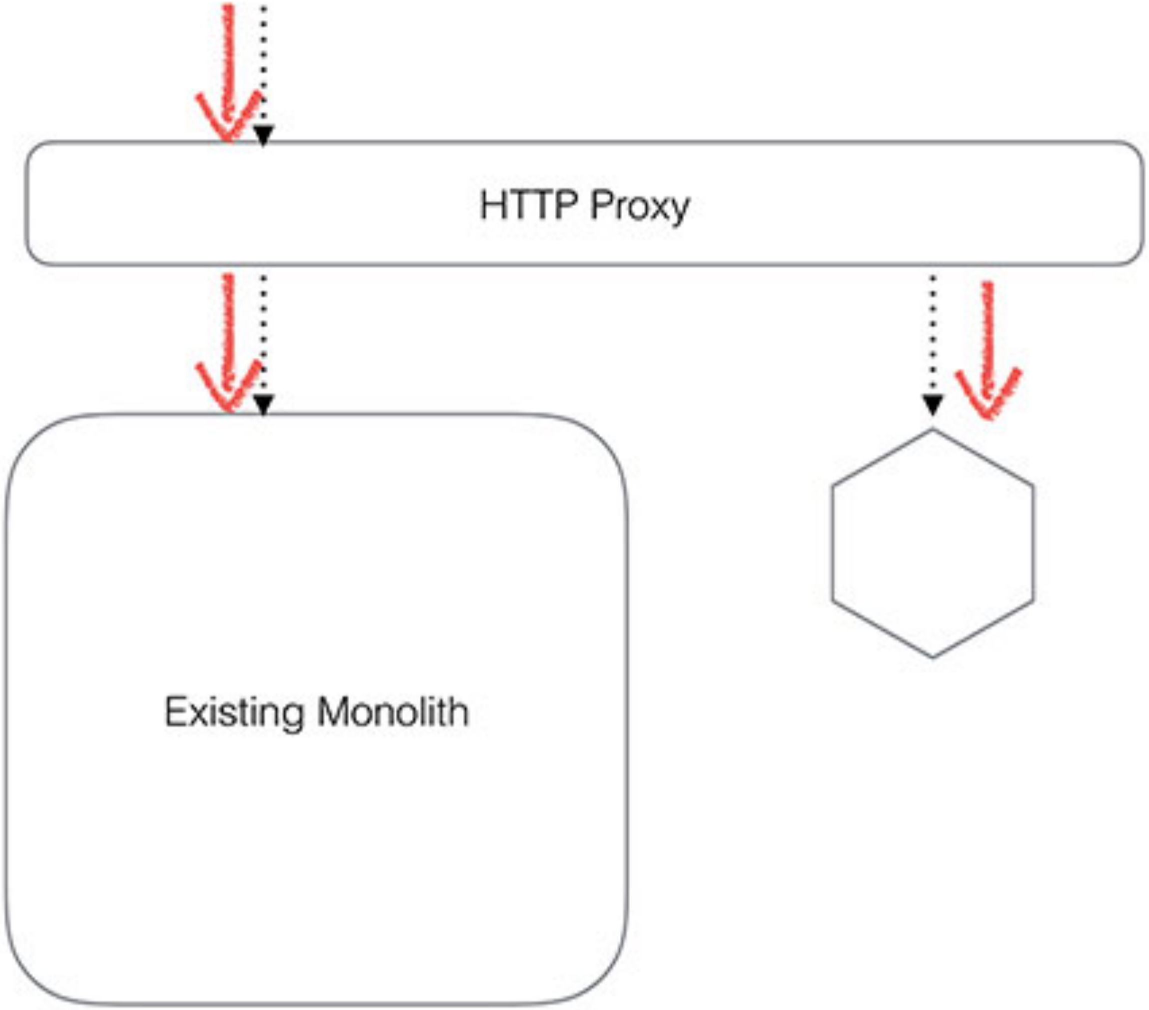
CO-EXISTING PROXY IMPLEMENTATIONS



HTTP Proxy

Existing Monolith

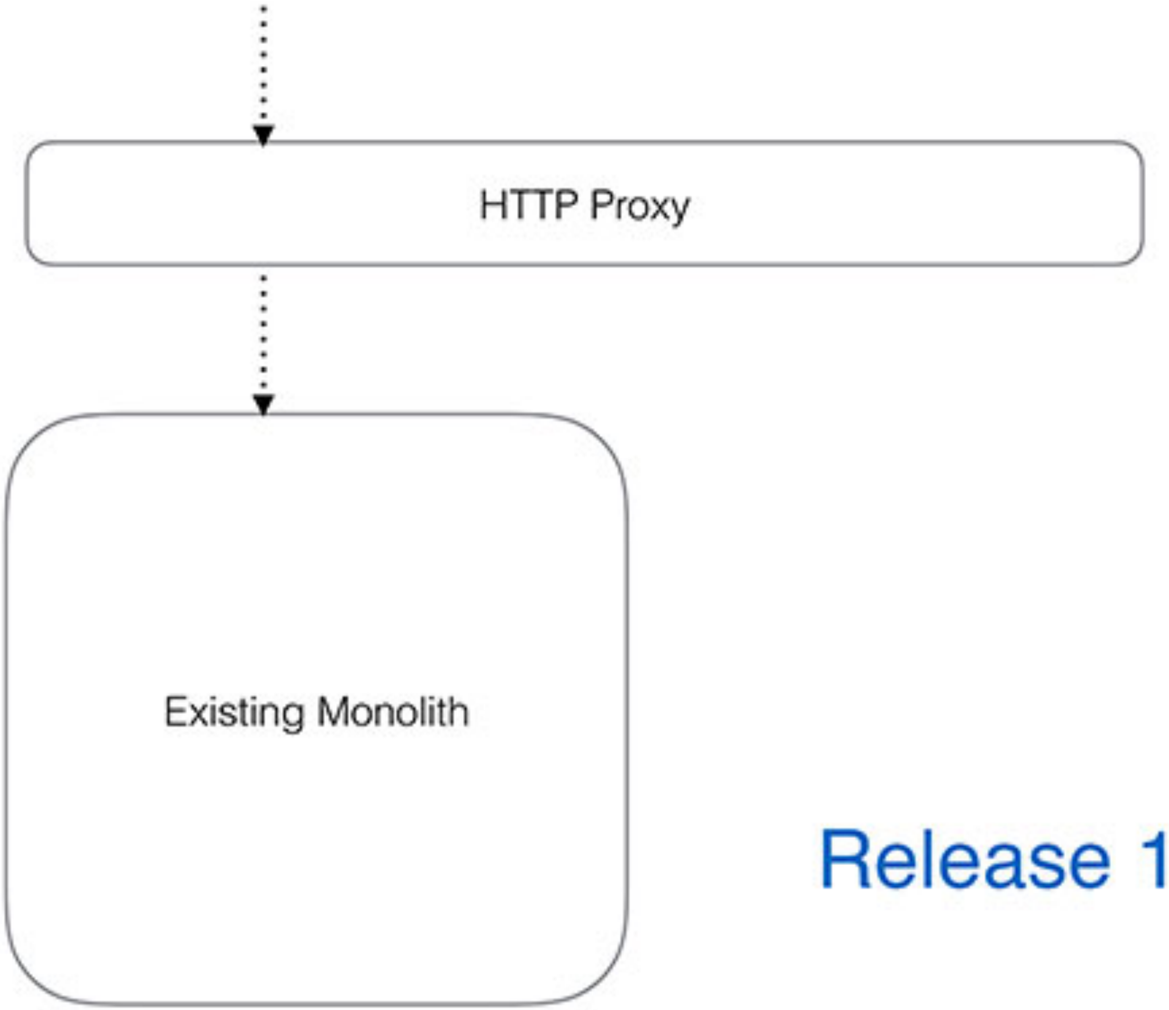
CO-EXISTING PROXY IMPLEMENTATIONS



DARK LAUNCHING

Existing Monolith

DARK LAUNCHING

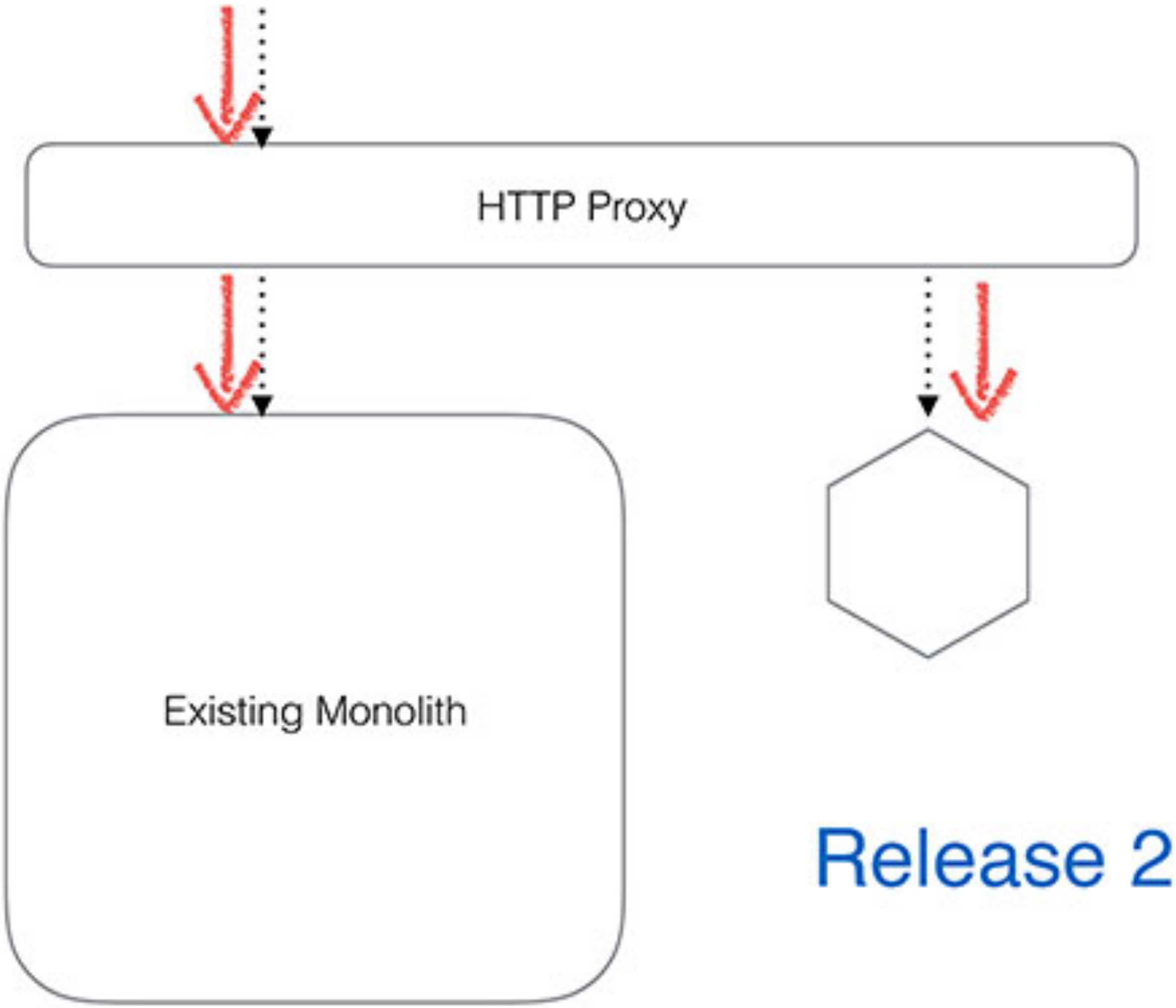


HTTP Proxy

Existing Monolith

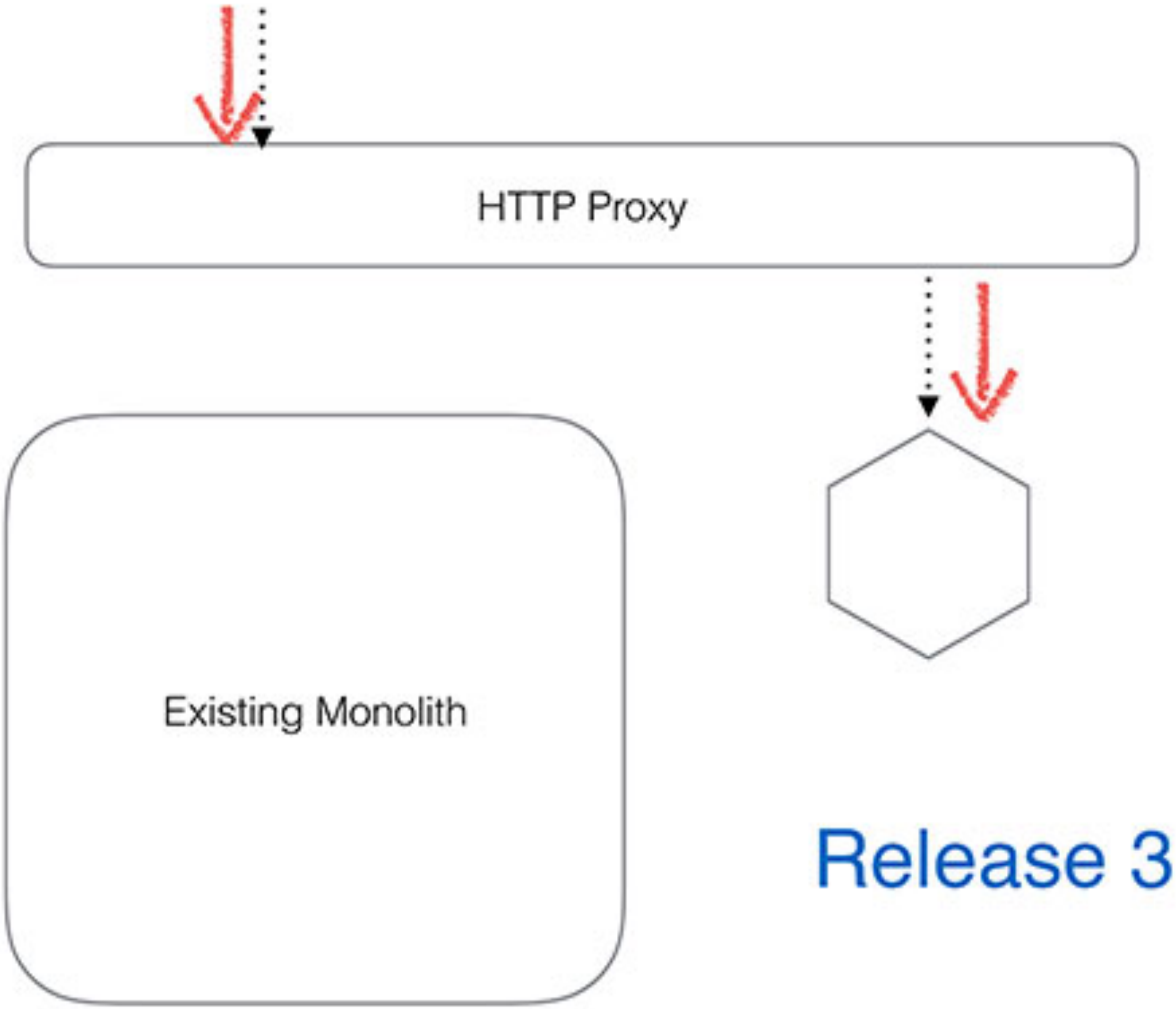
Release 1

DARK LAUNCHING



Release 2

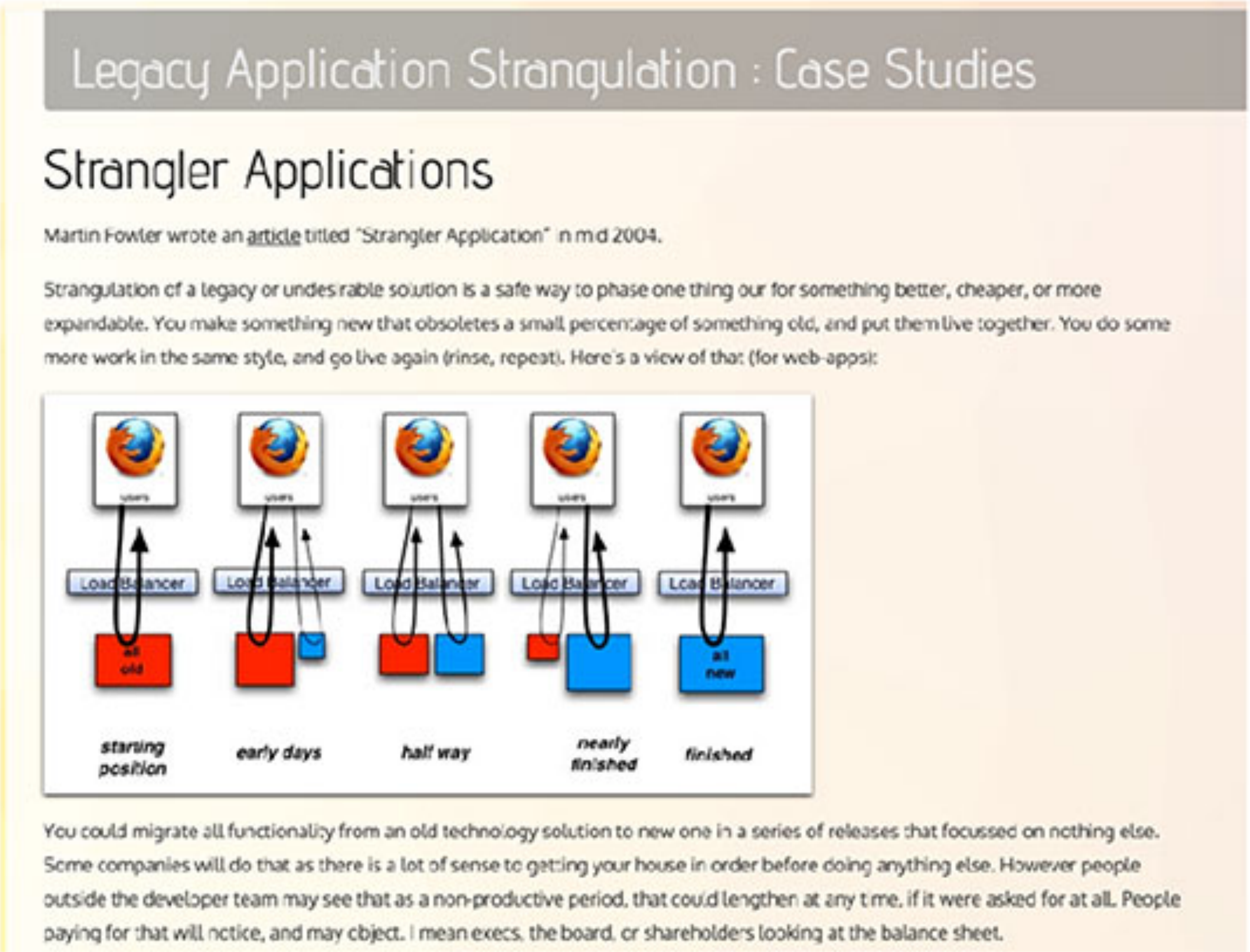
DARK LAUNCHING



Release 3

Call both implementations & compare

Divert proportion of traffic to test new implementation (canary release)



Legacy Application Strangulation : Case Studies

Strangler Applications

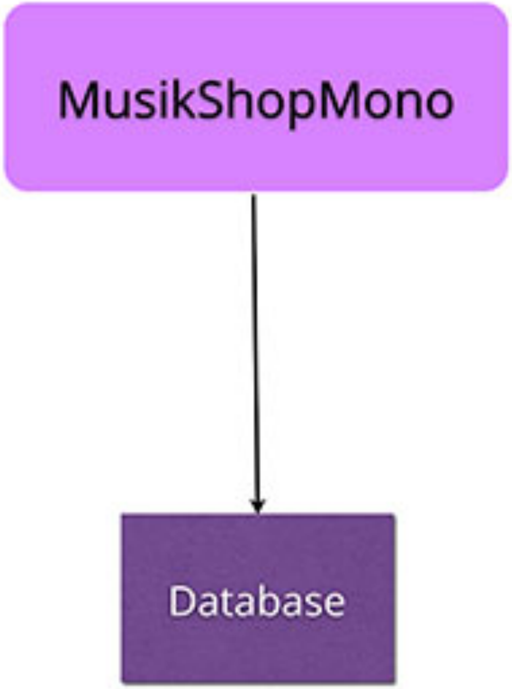
Martin Fowler wrote an article titled "Strangler Application" in mid 2004.

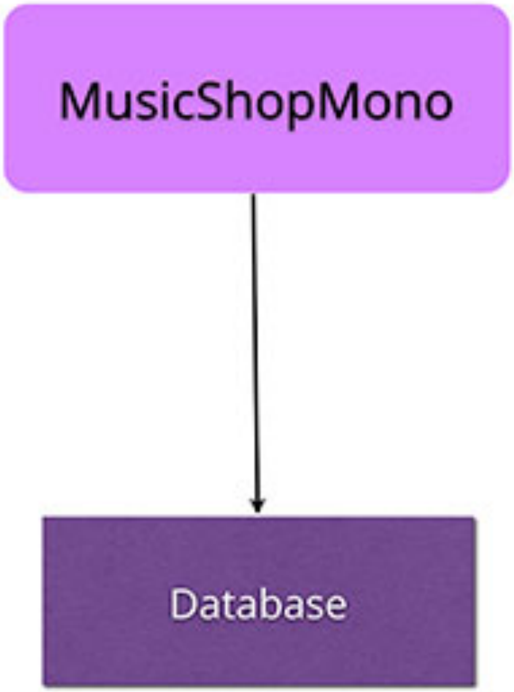
Strangulation of a legacy or undesirable solution is a safe way to phase one thing our for something better, cheaper, or more expandable. You make something new that obsoletes a small percentage of something old, and put them live together. You do some more work in the same style, and go live again (rinse, repeat). Here's a view of that (for web-apps):

You could migrate all functionality from an old technology solution to new one in a series of releases that focussed on nothing else. Some companies will do that as there is a lot of sense to getting your house in order before doing anything else. However people outside the developer team may see that as a non-productive period, that could lengthen at any time, if it were asked for at all. People paying for that will notice, and may object. I mean execs, the board, or shareholders looking at the balance sheet.

http://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/

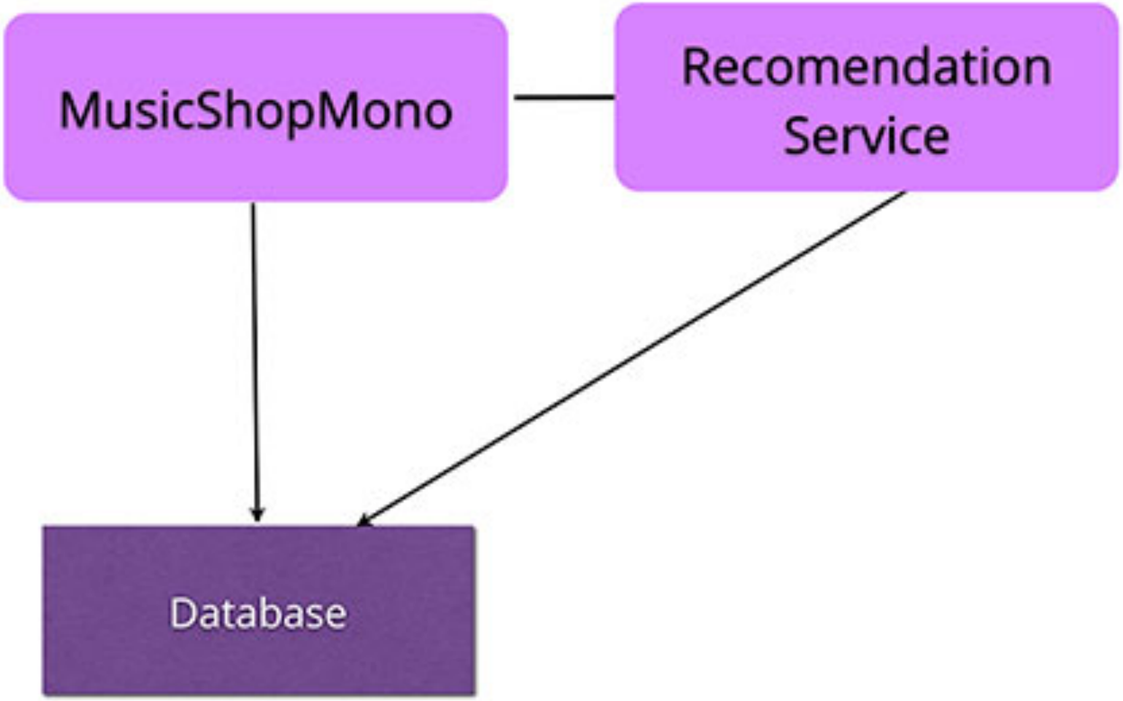
Databases

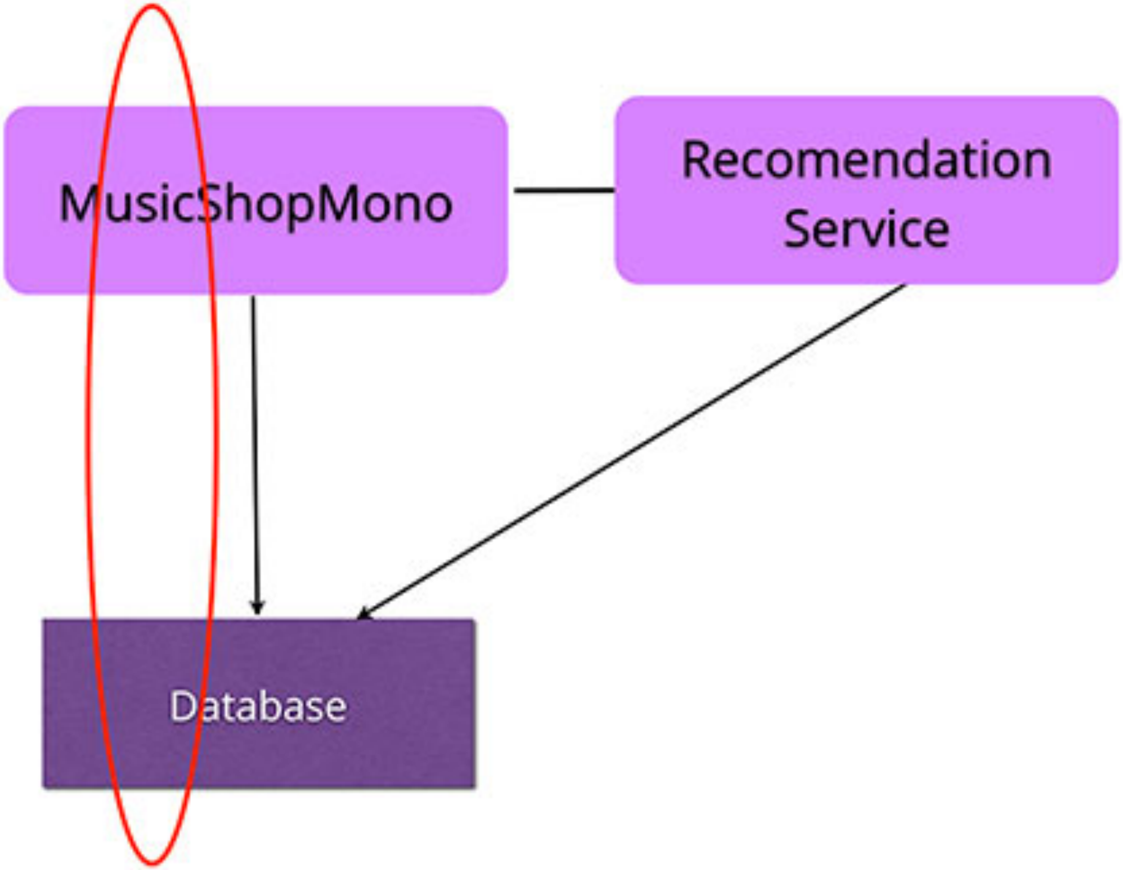

MusikShopMono
Database



MusicShopMono

Database

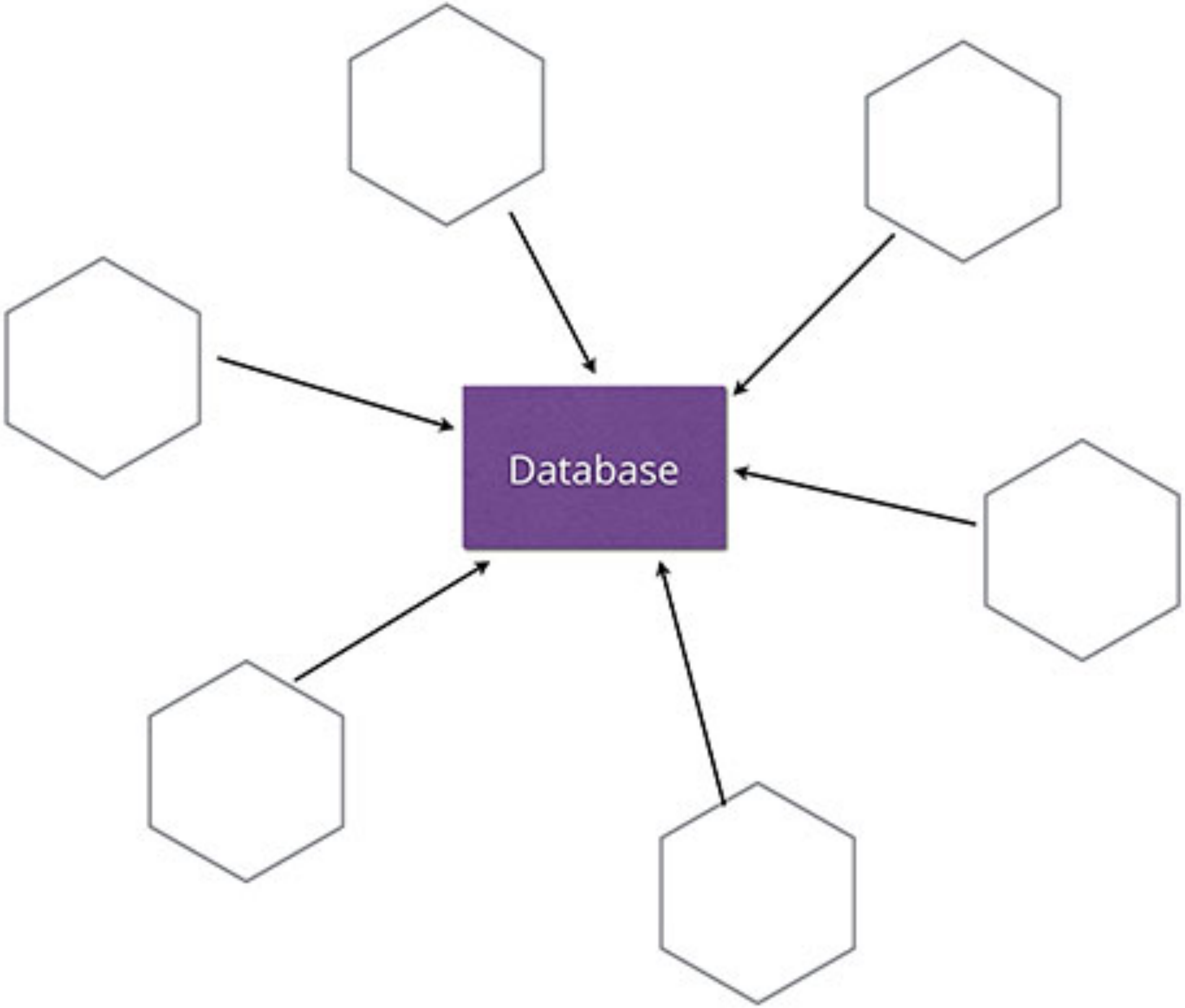

MusicShopMono
Recomendation Service
Database

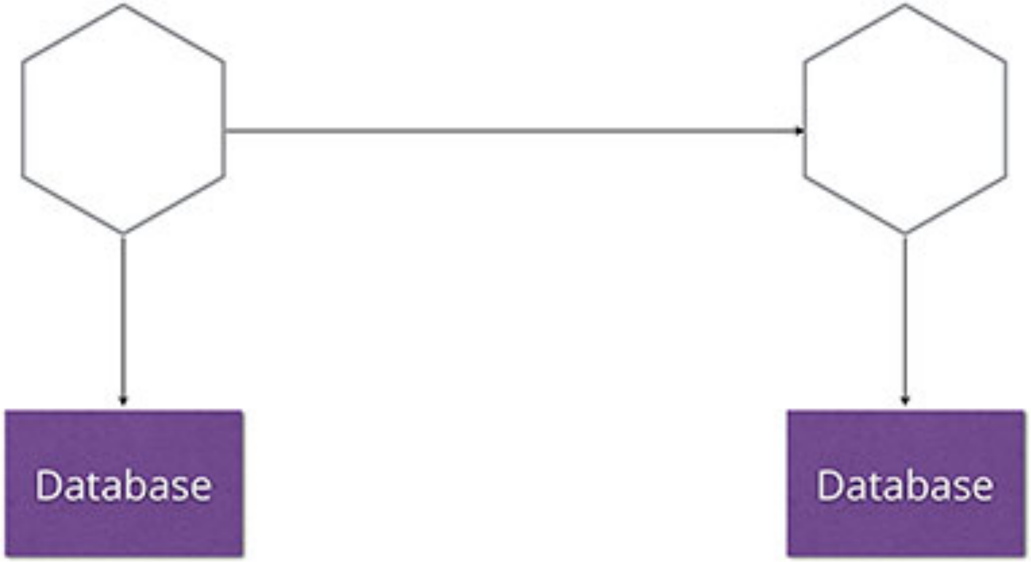


MusicShopMono

Recomendation Service

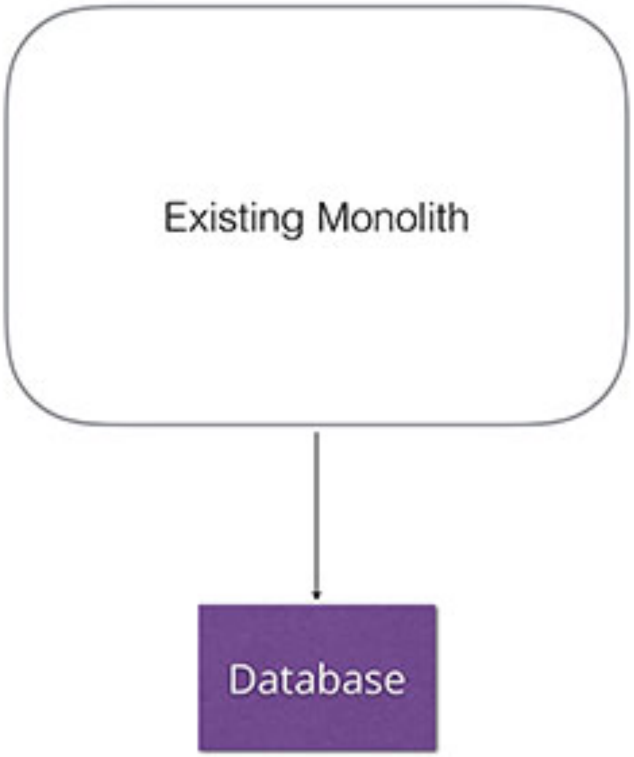
Database


Database



Database
Database

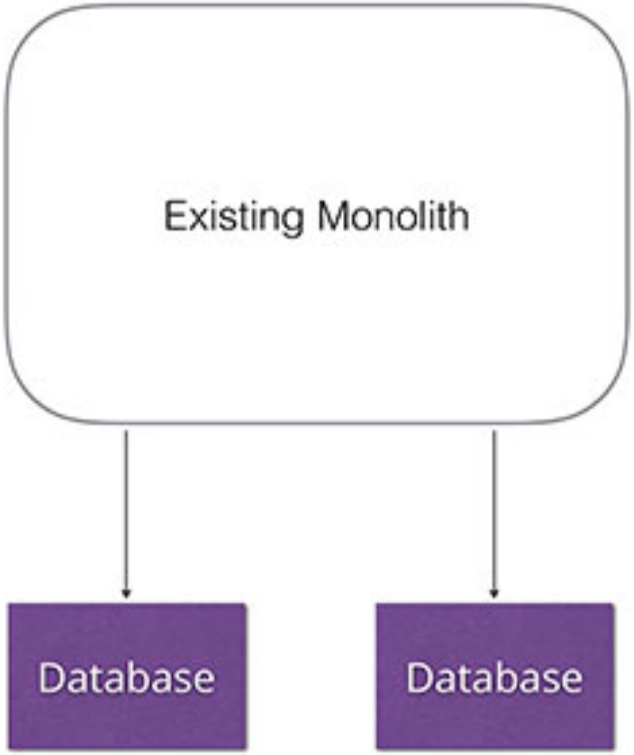
SPLIT DATABASES FIRST



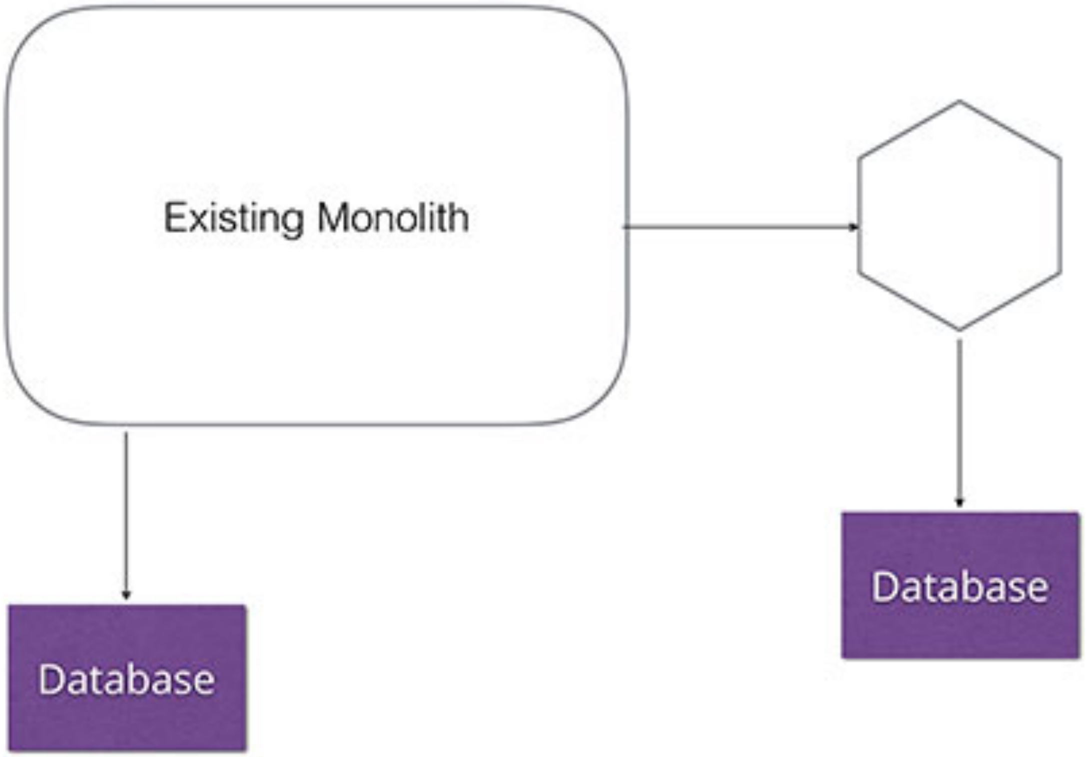
Existing Monolith

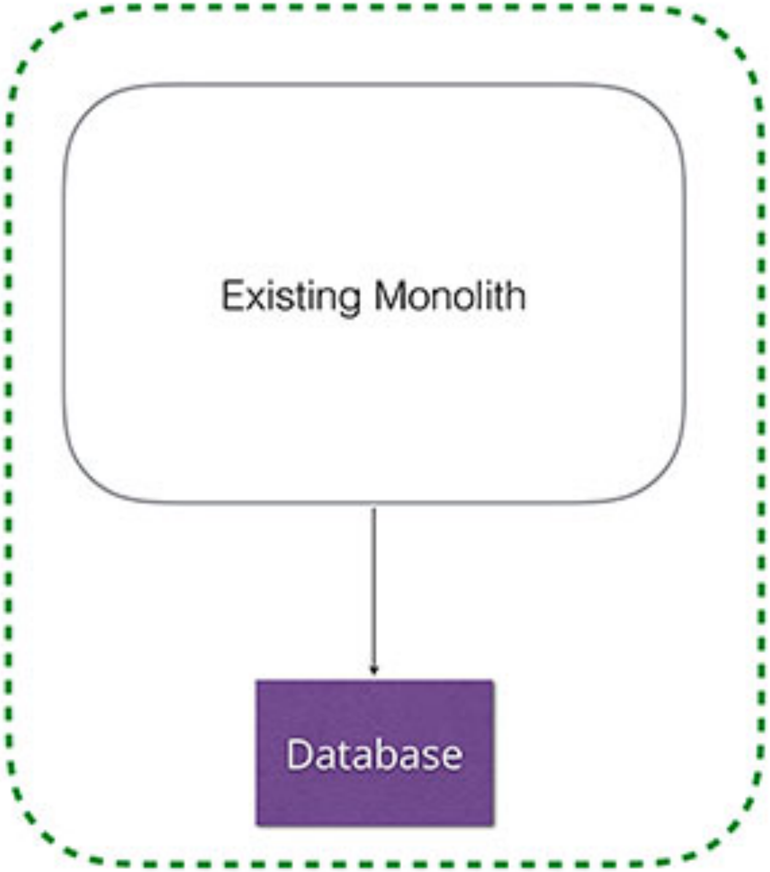
Database

SPLIT DATABASES FIRST



SPLIT DATABASES FIRST

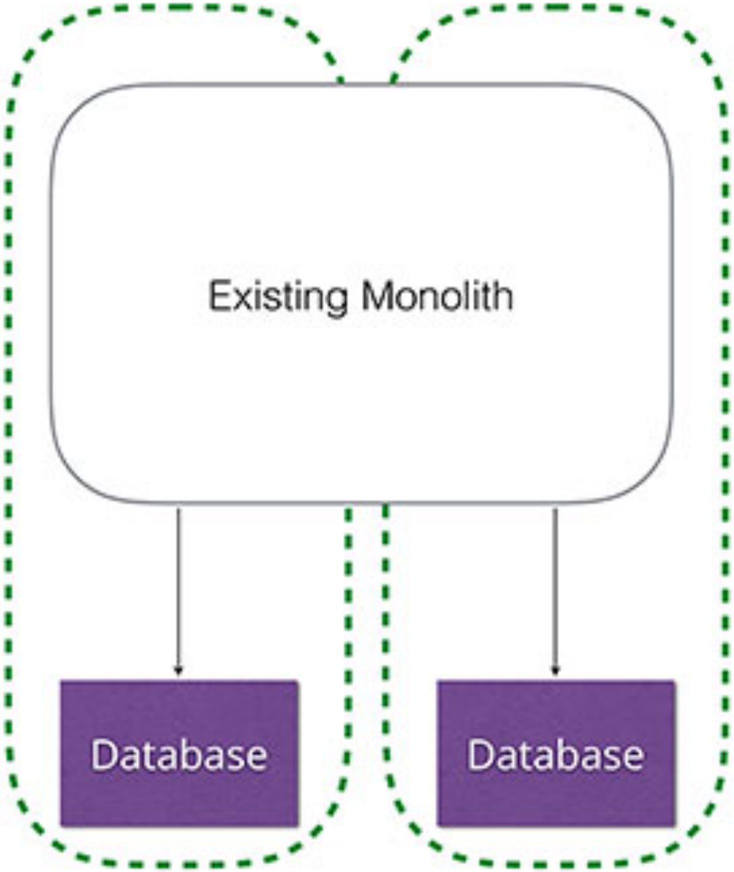




One Transactional Boundary

Single Database Call

Joins done in the DB



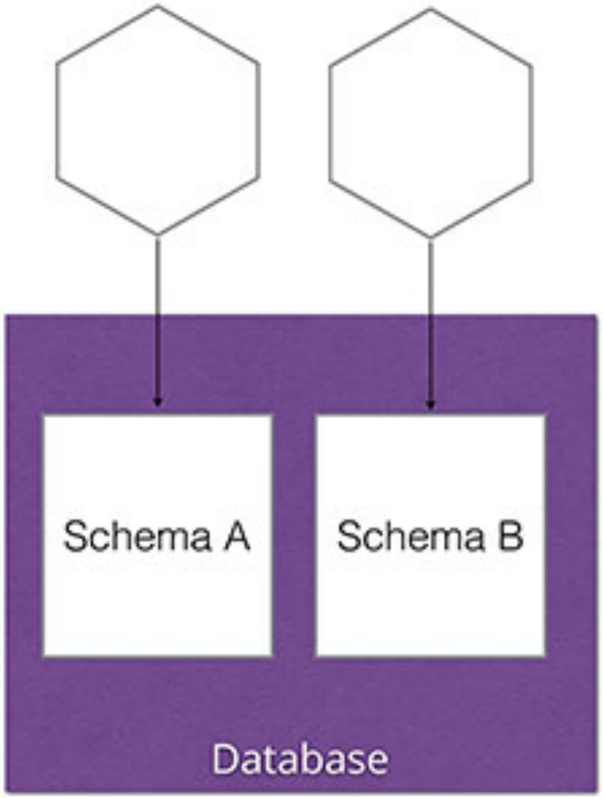
Broke transactional integrity

Two database calls

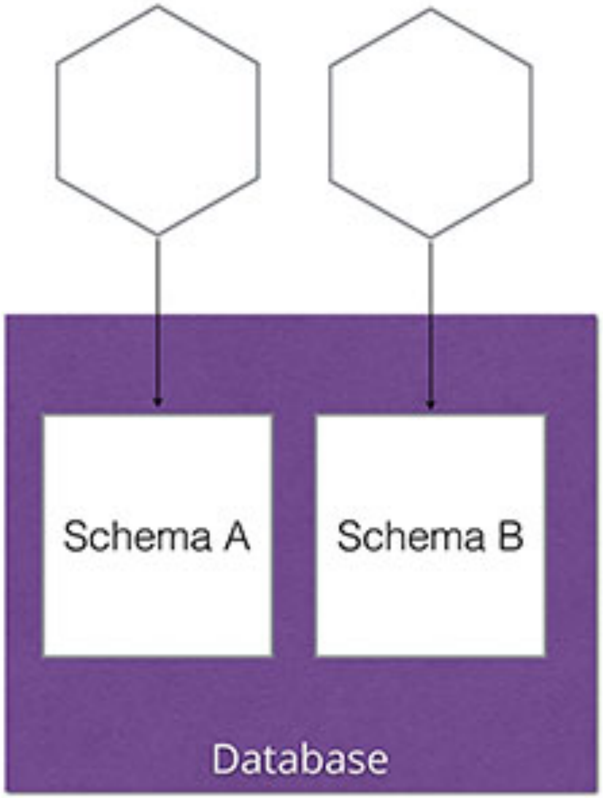
Some joins now done in application code

Understand the performance and transactional integrity issues early

SEPARATE SCHEMAS, SAME UNDERLYING DB DEPLOYMENT

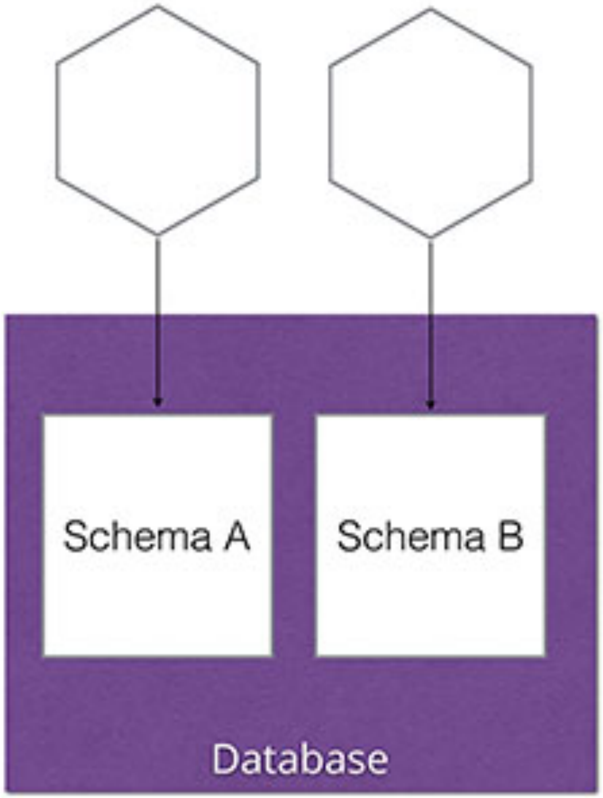


SEPARATE SCHEMAS, SAME UNDERLYING DB DEPLOYMENT



Logically decoupled

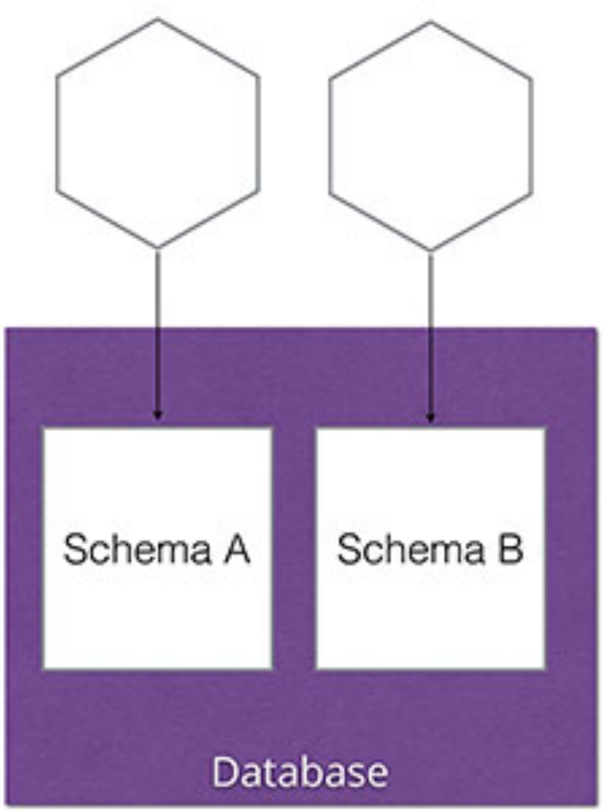
SEPARATE SCHEMAS, SAME UNDERLYING DB DEPLOYMENT



Logically decoupled

Little or no extra infrastructure required

SEPARATE SCHEMAS, SAME UNDERLYING DB DEPLOYMENT

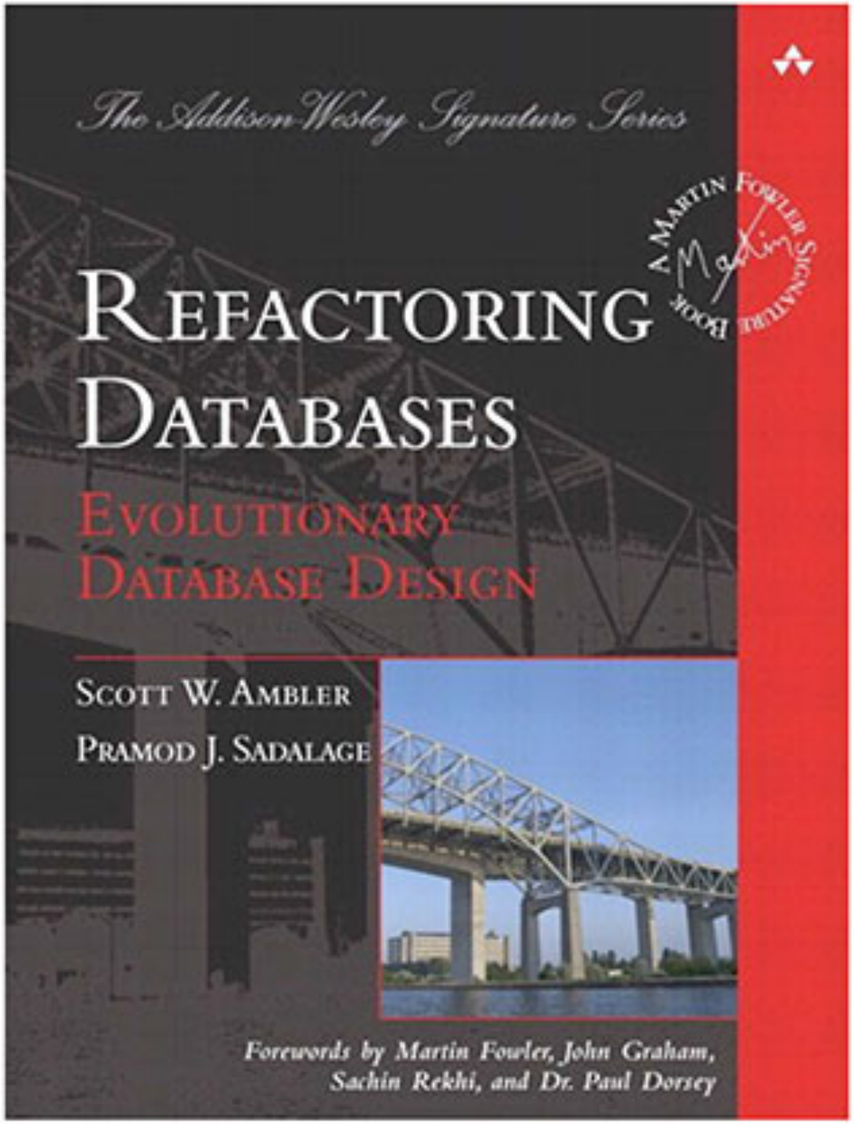


Logically decoupled

Little or no extra infrastructure required

Potential single point of failure

DB Refactoring Patterns



The Addison-Wesley Signature Series

A Martin Fowler Signature Book

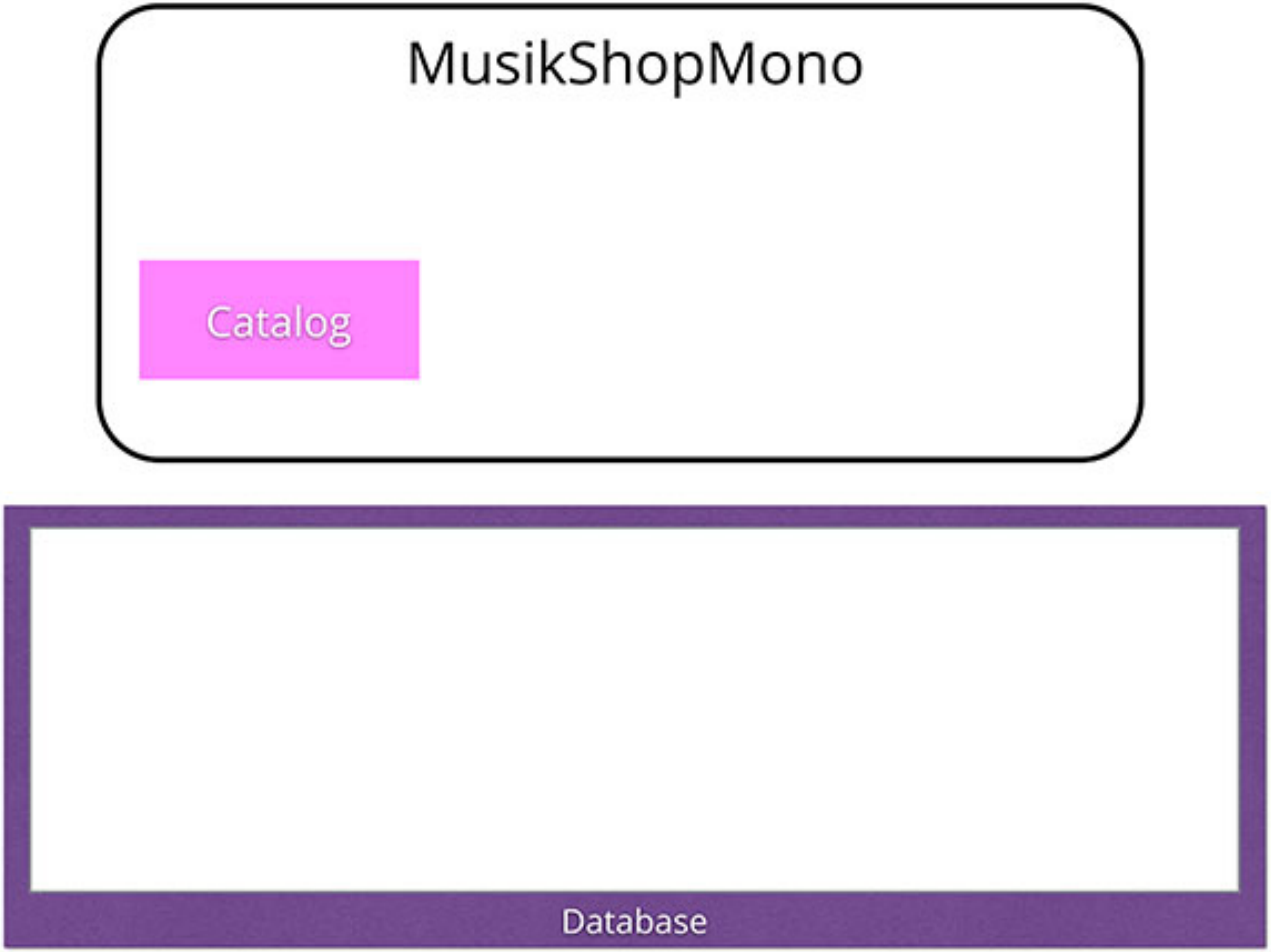
REFACTORING DATABASES

EVOLUTIONARY DATABASE DESIGN

SCOTT W. AMBLER

PRAMOD J. SADALAGE

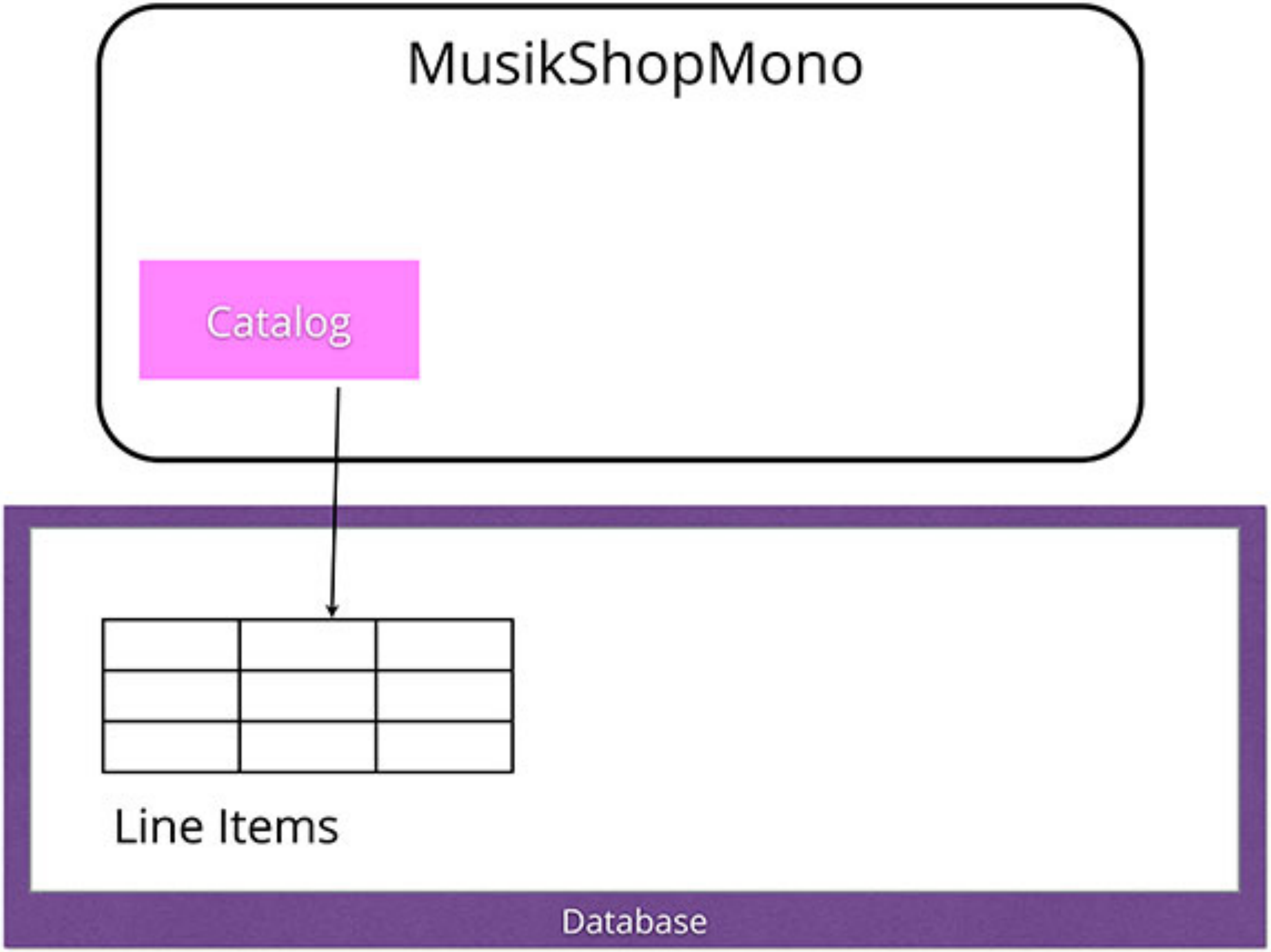
Forewords by Martin Fowler, John Graham, Sachin Rekhi, and Dr. Paul Dorsey



MusikShopMono

Catalog

Database

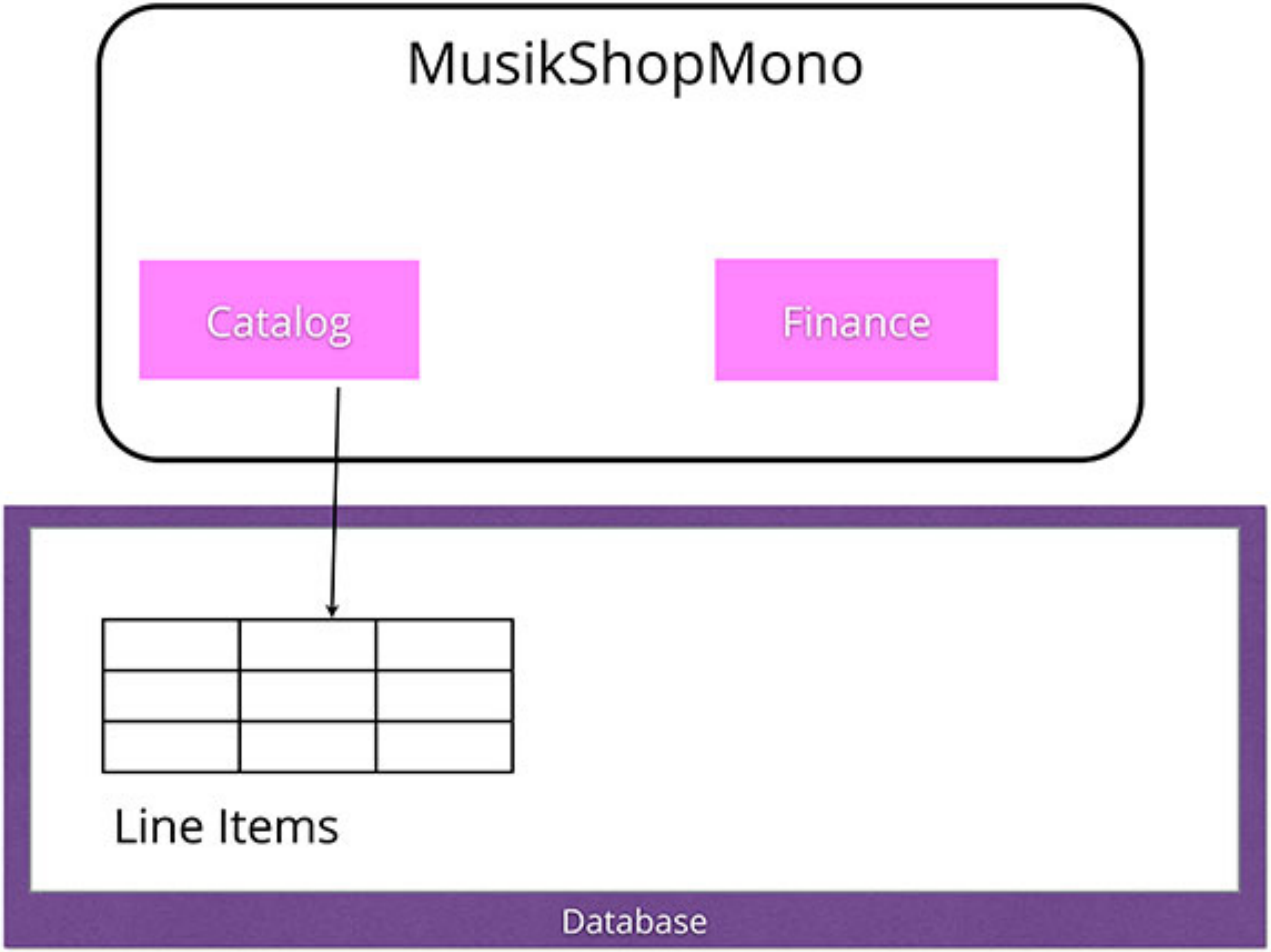


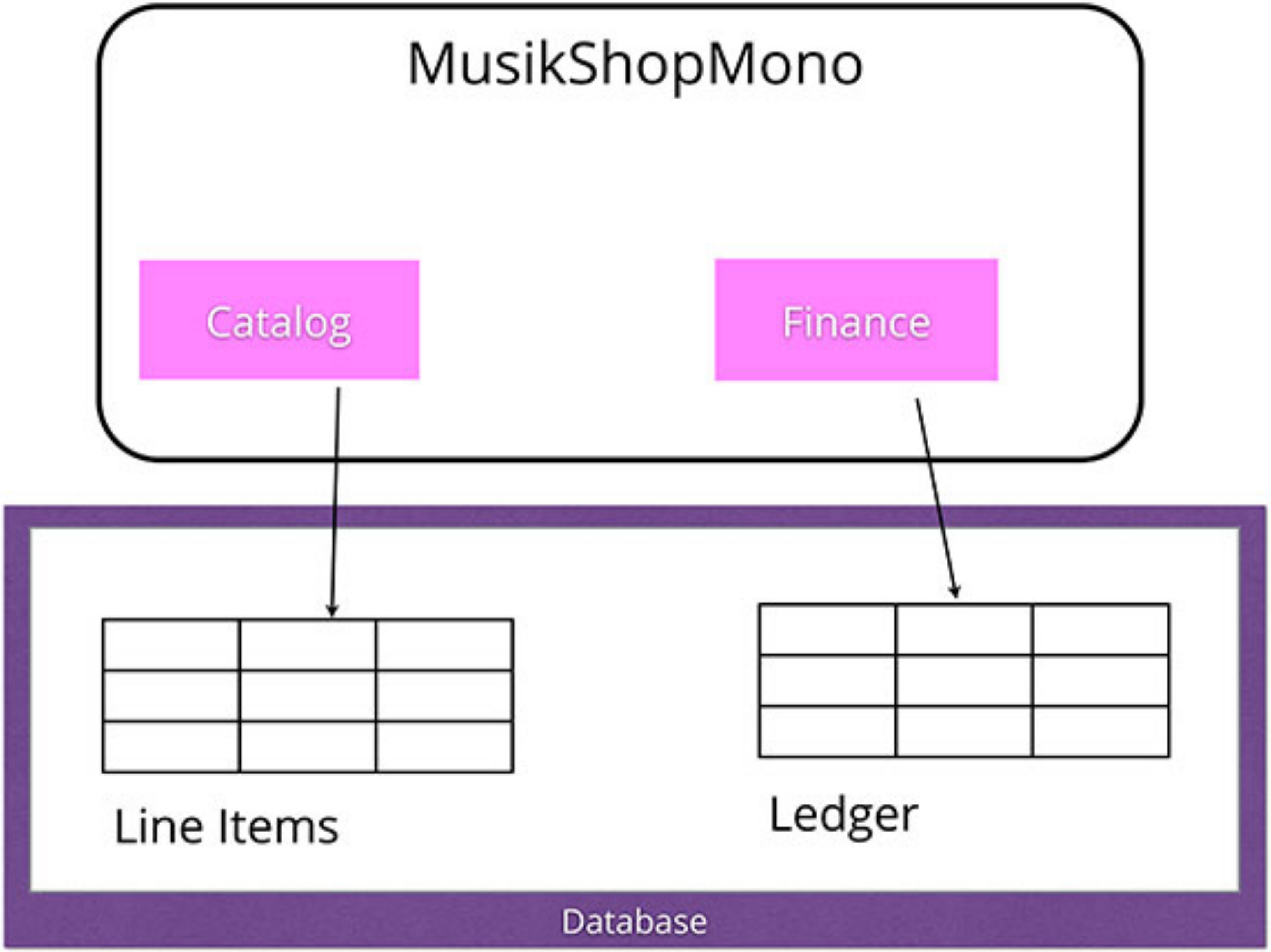
MusikShopMono

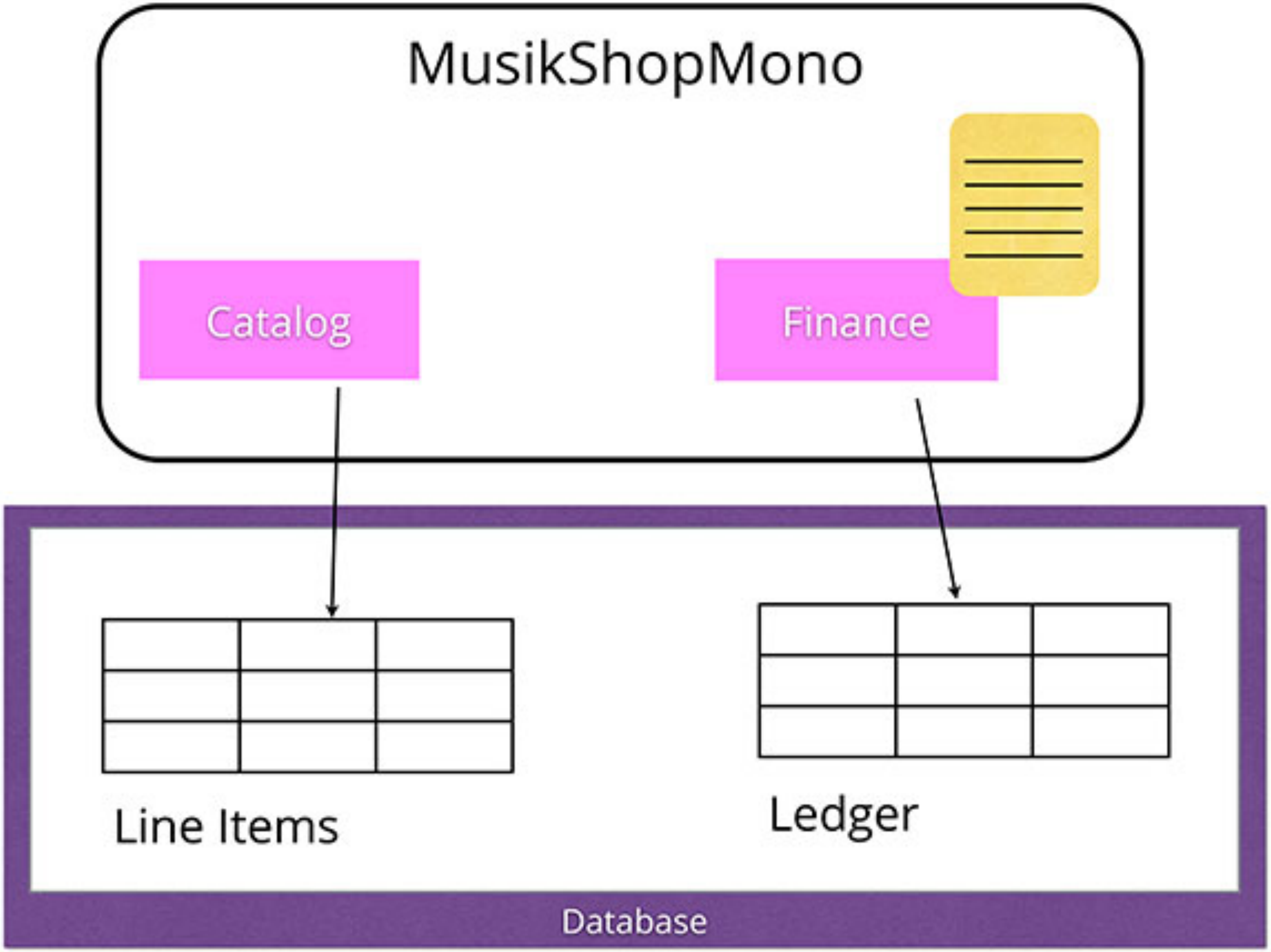
Catalog

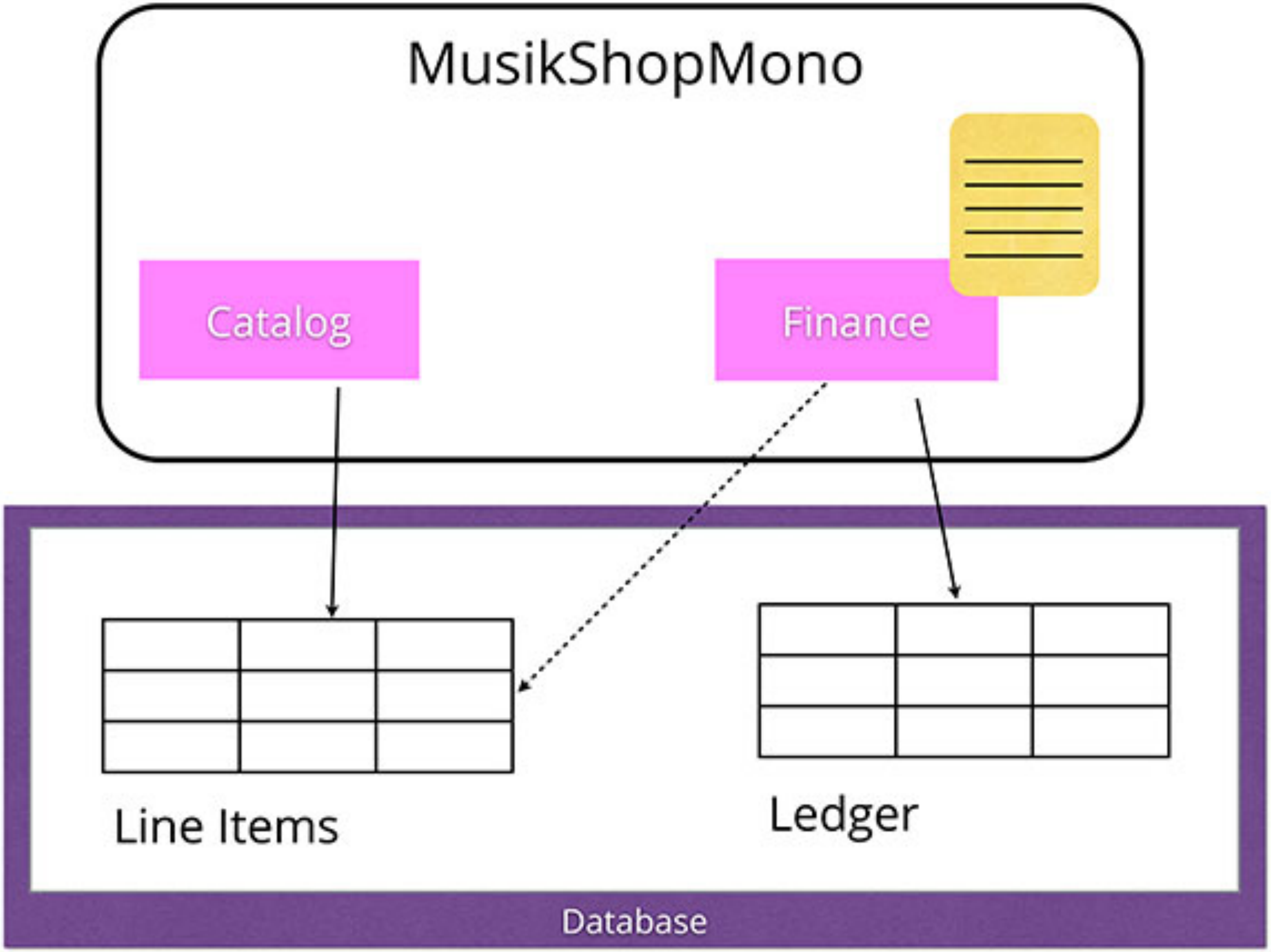
Line Items

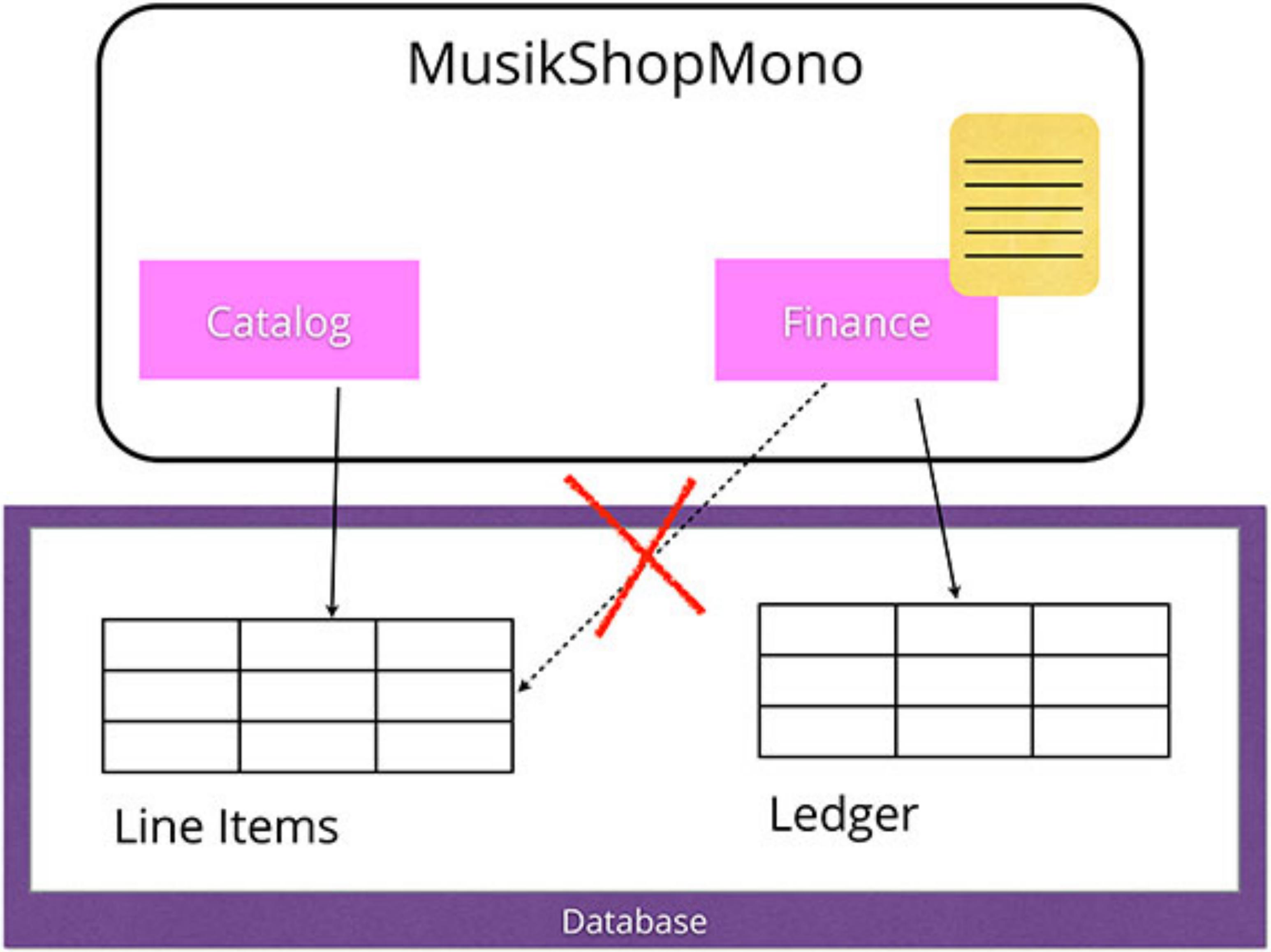
Database

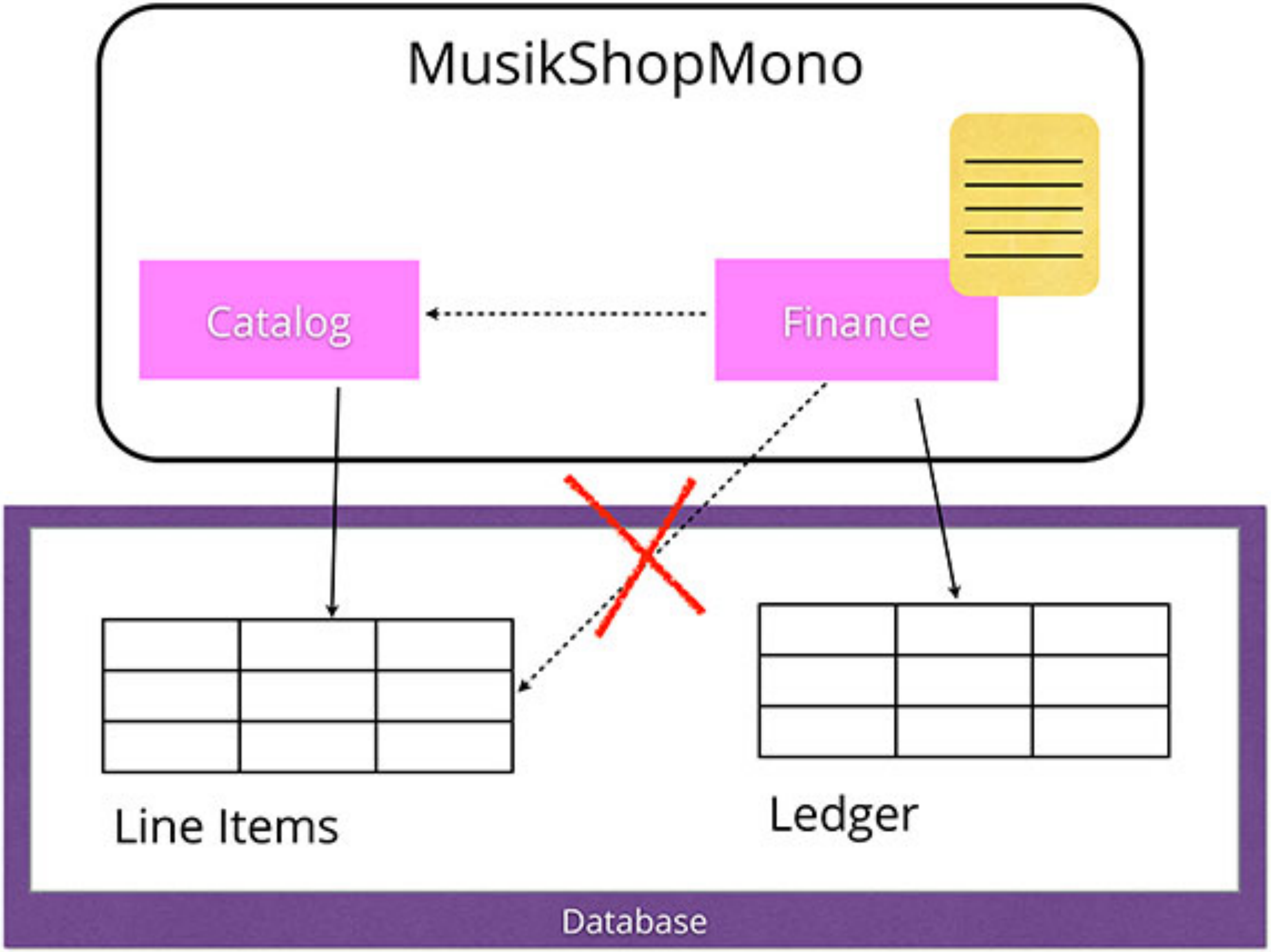

MusikShopMono
Catalog
Finance
Line Items
Database

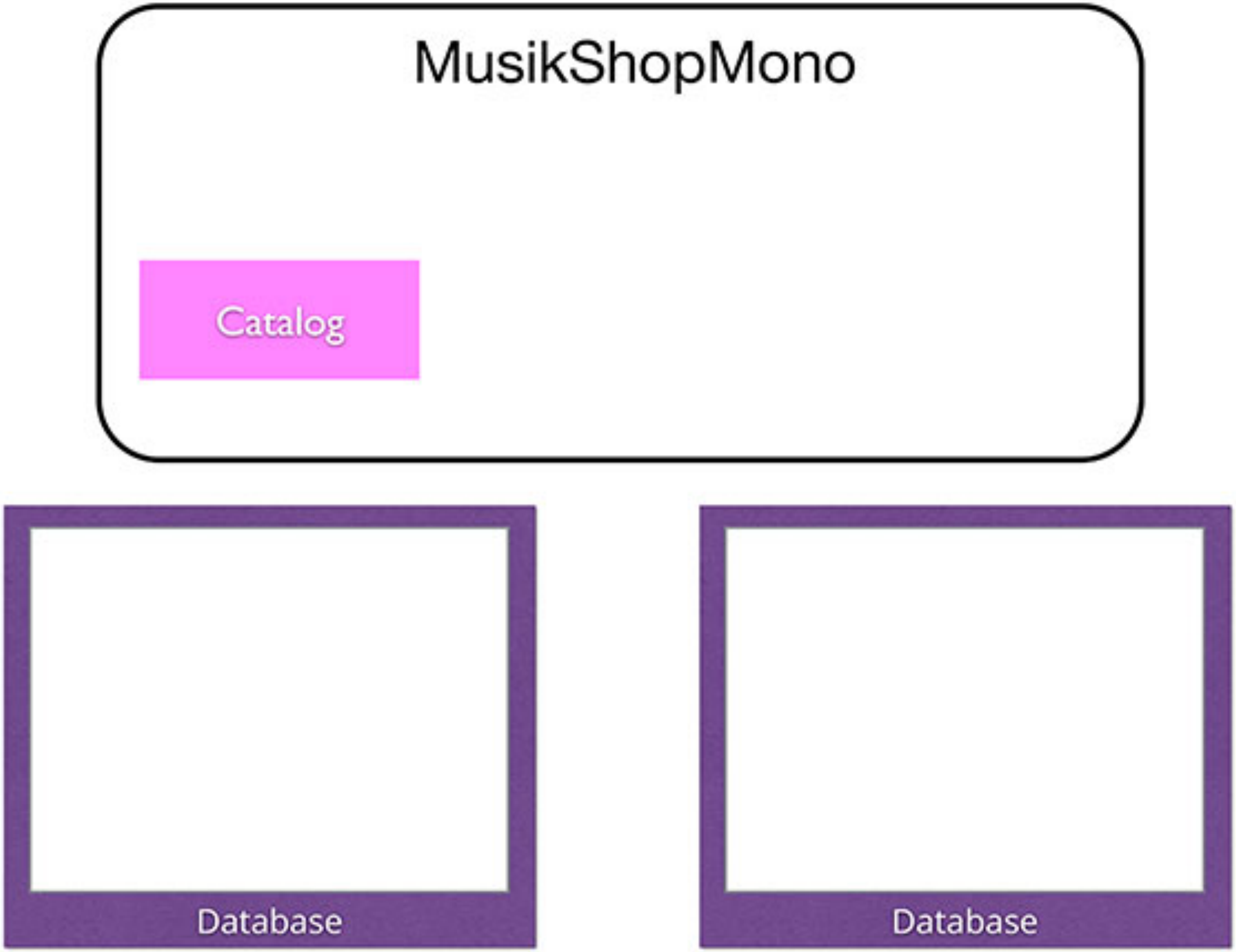

MusikShopMono
Catalog
Finance
Line Items
Ledger
Database


MusikShopMono
Catalog
Finance
Line Items
Ledger
Database


MusikShopMono
Catalog
Finance
Line Items
Ledger
Database


MusikShopMono
Catalog
Finance
Line Items
Ledger
Database


MusikShopMono
Catalog
Finance
Line Items
Ledger
Database

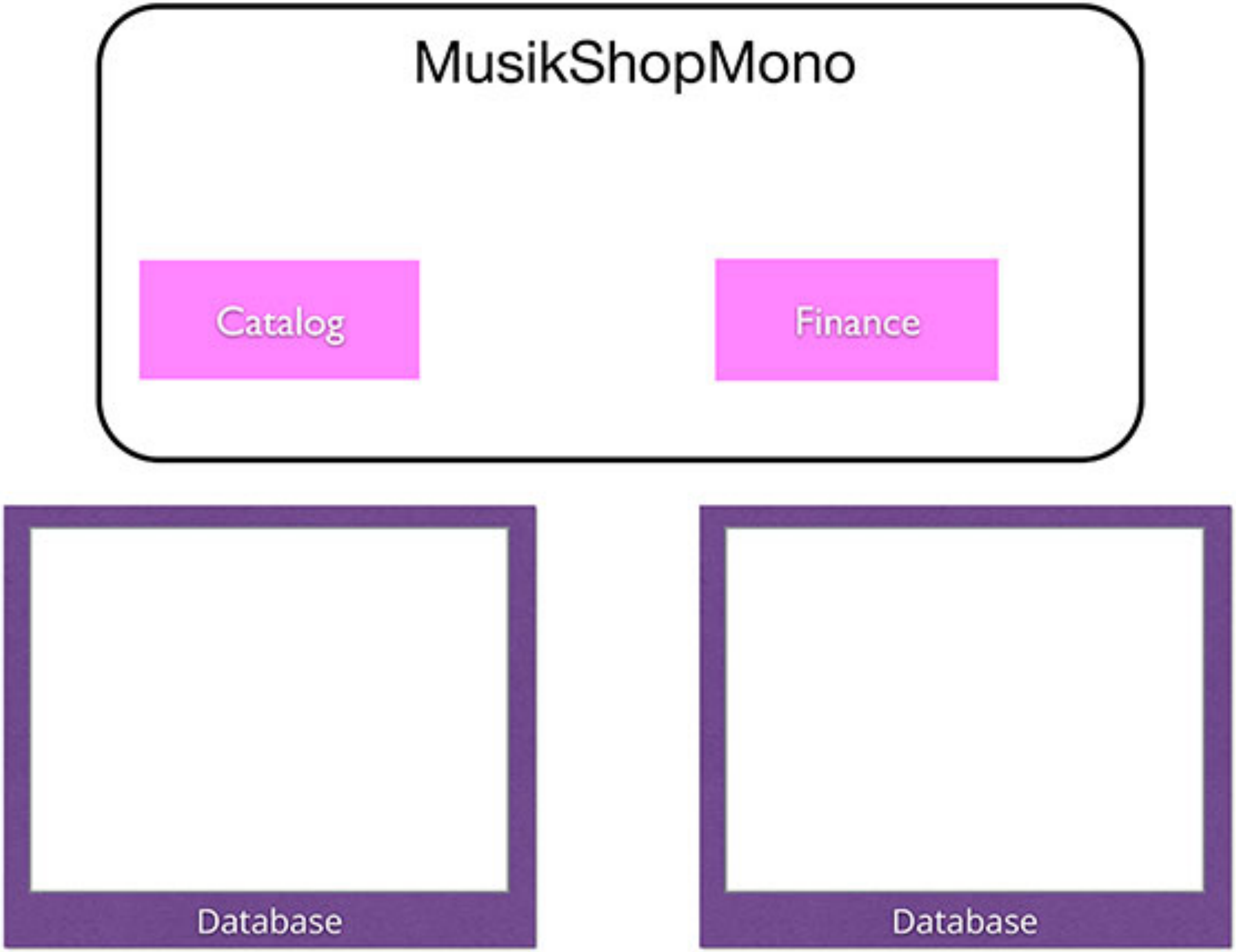


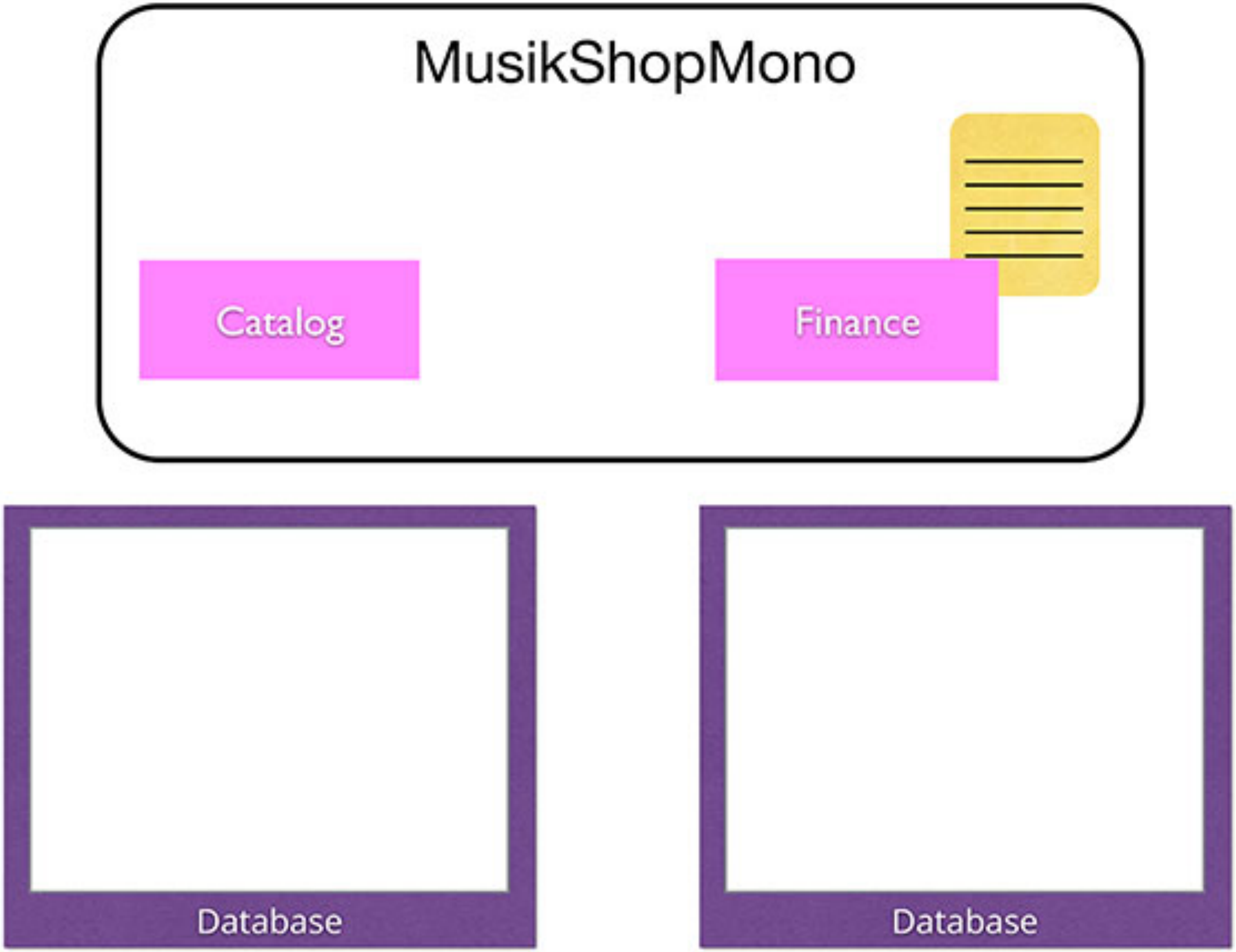
MusikShopMono

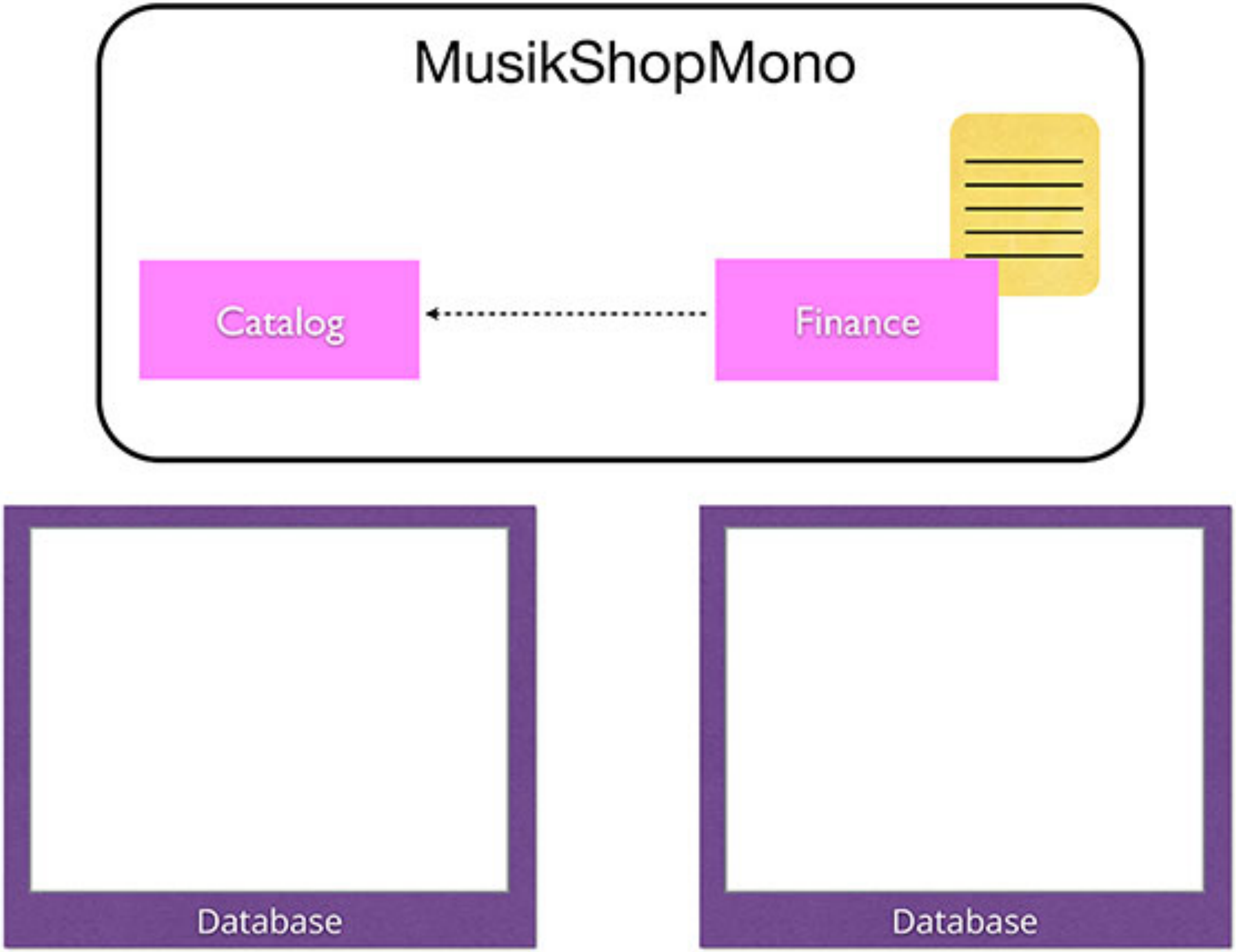
Catalog

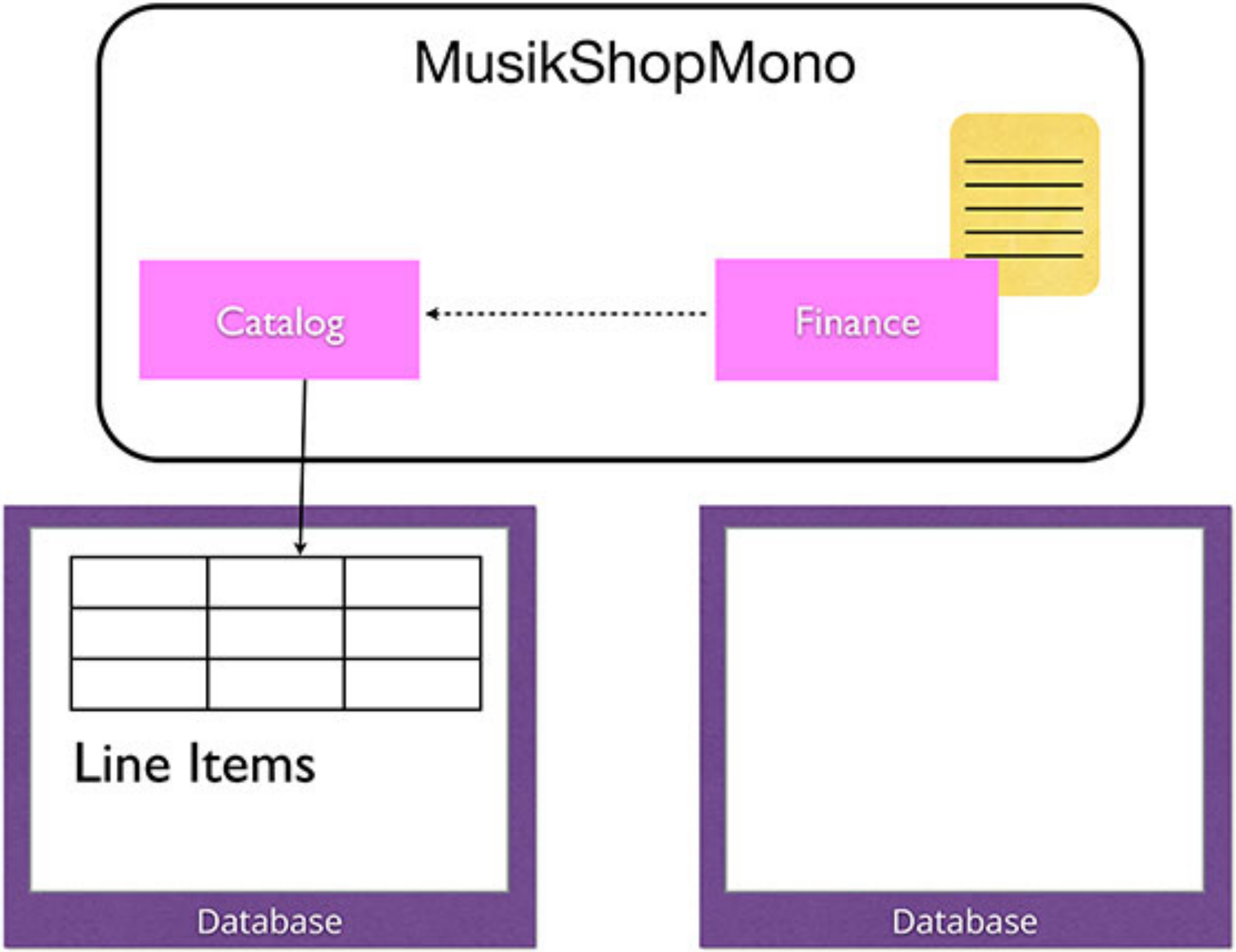
Database

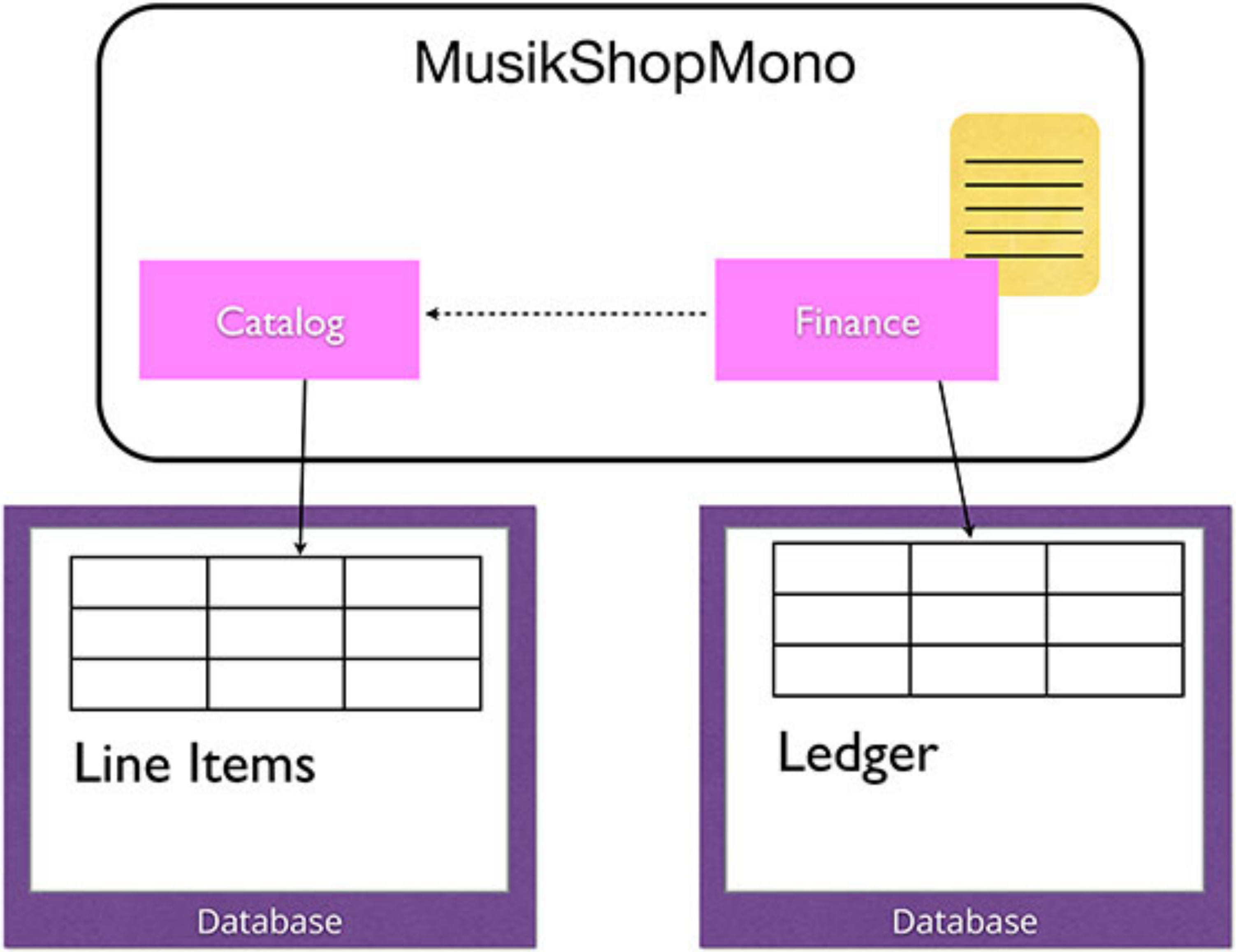
Database


MusikShopMono
Catalog
Finance
Database
Database

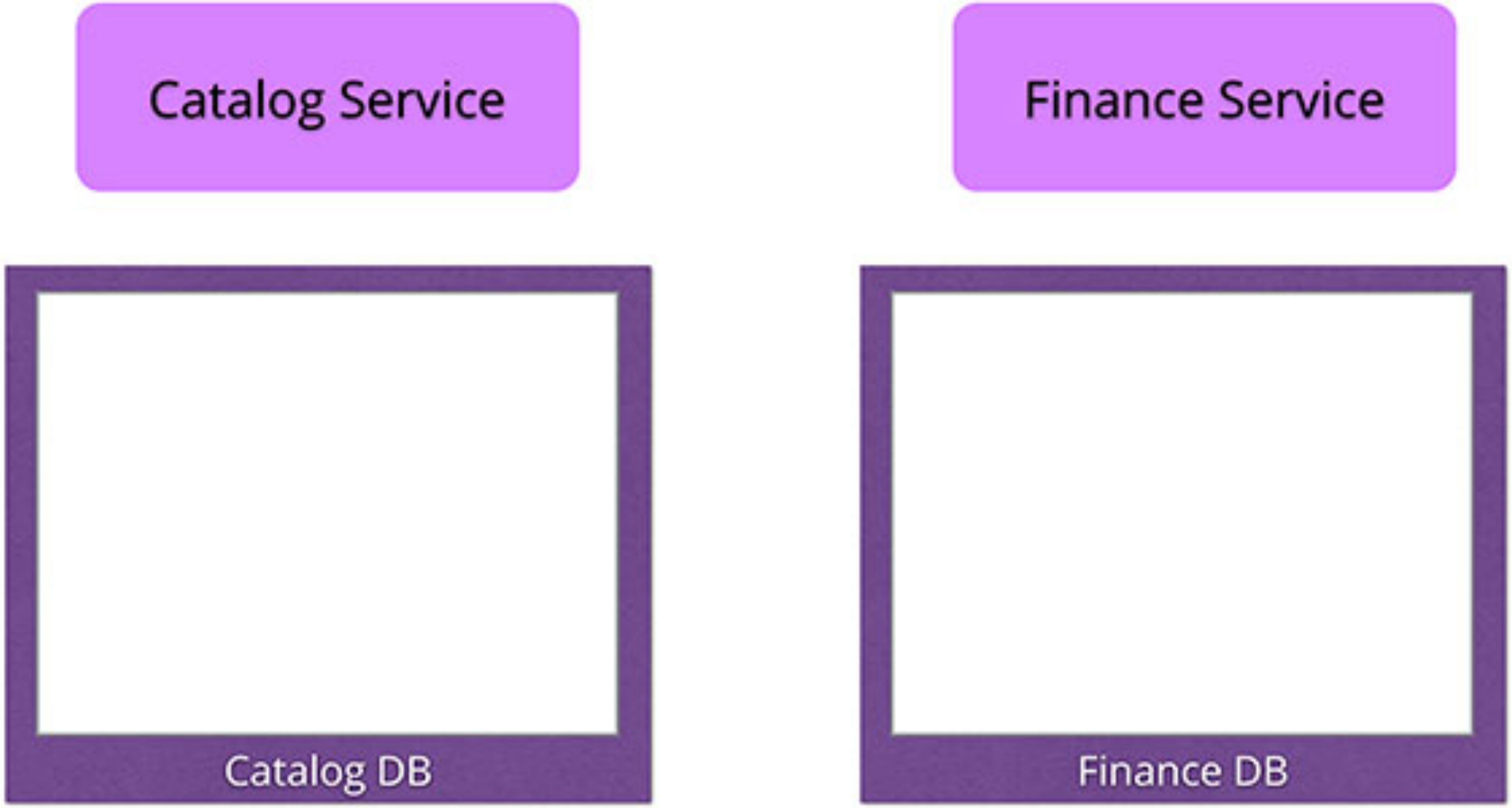

MusikShopMono
Catalog
Finance
Database
Database


MusikShopMono
Catalog
Finance
Database
Database


MusikShopMono
Catalog
Finance
Line Items
Database
Database


MusikShopMono
Catalog
Finance
Line Items
Database
Ledger
Database

MusikShop System



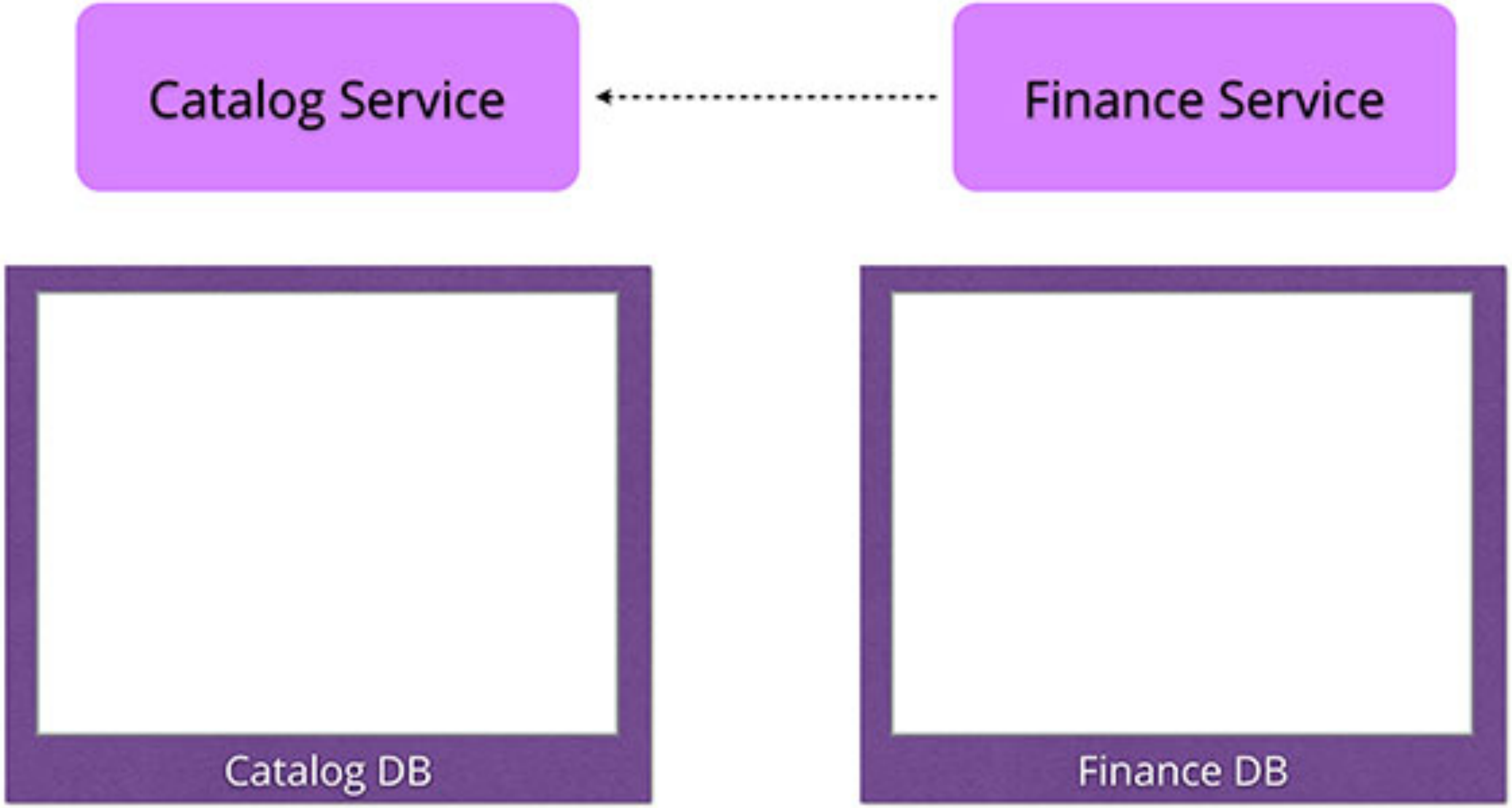
Catalog Service

Finance Service

Catalog DB

Finance DB

MusikShop System

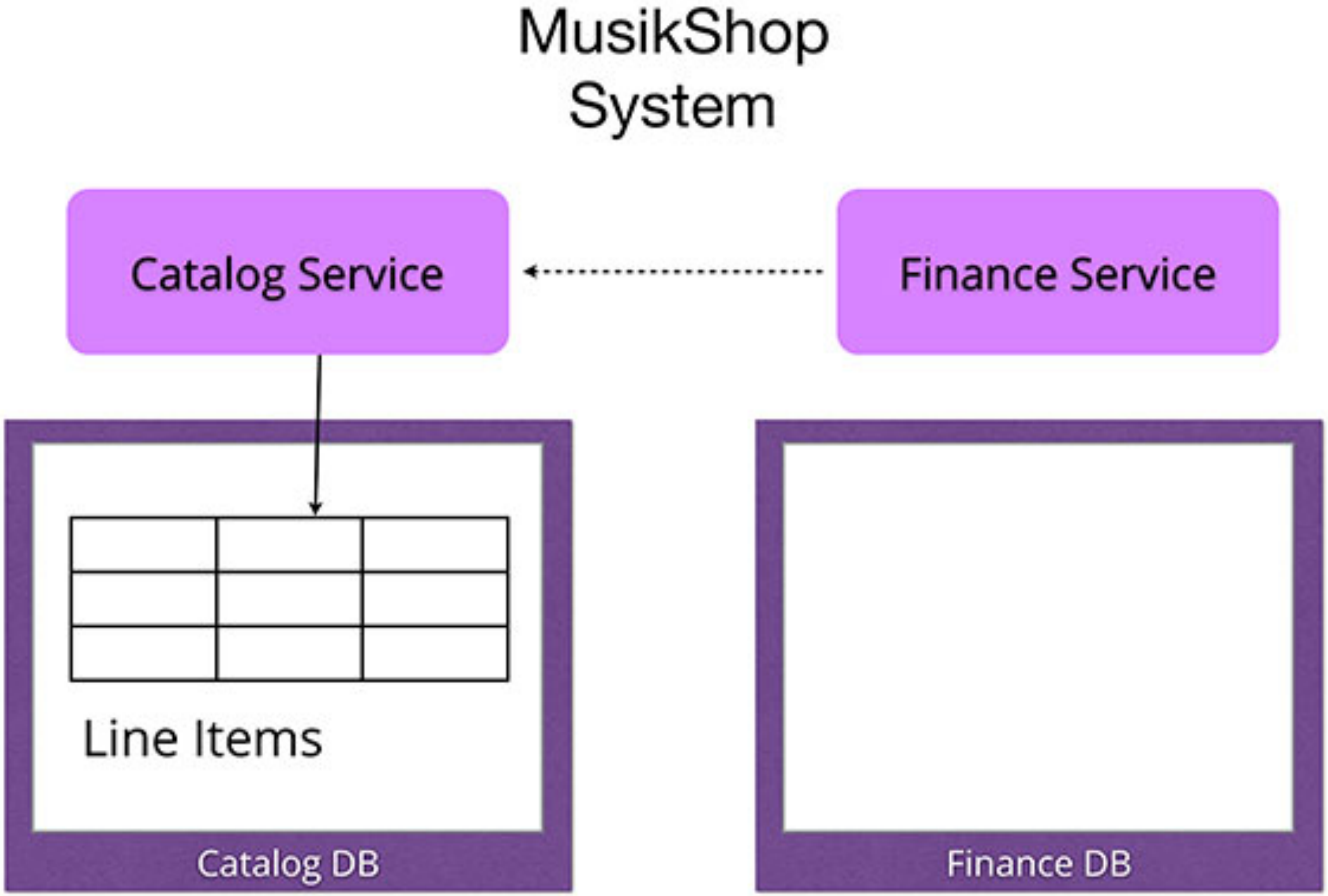


Catalog Service

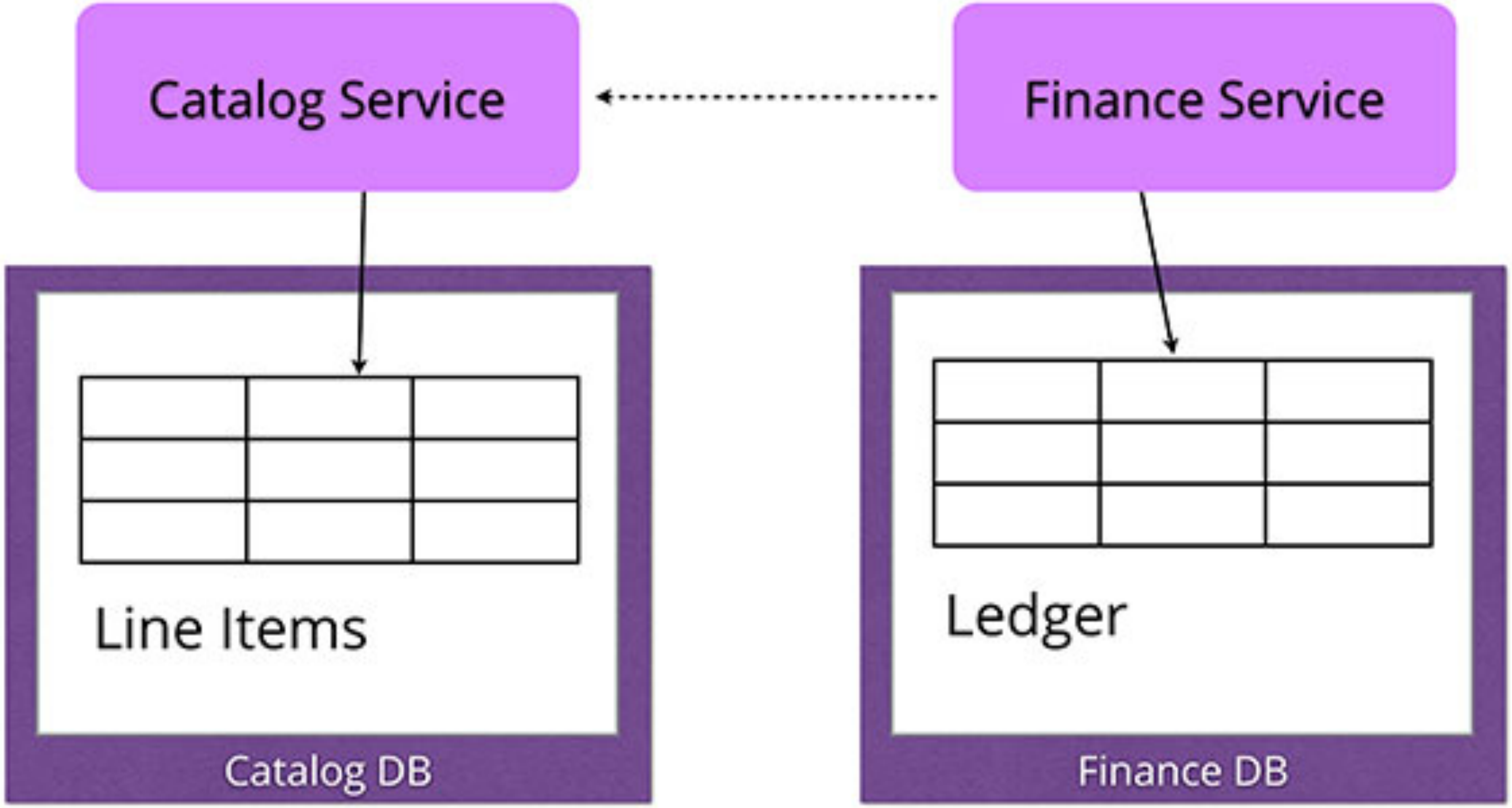
Finance Service

Catalog DB

Finance DB


MusikShop
System
Catalog Service
Finance Service
Line Items
Catalog DB
Finance DB

MusikShop System



Catalog Service

Finance Service

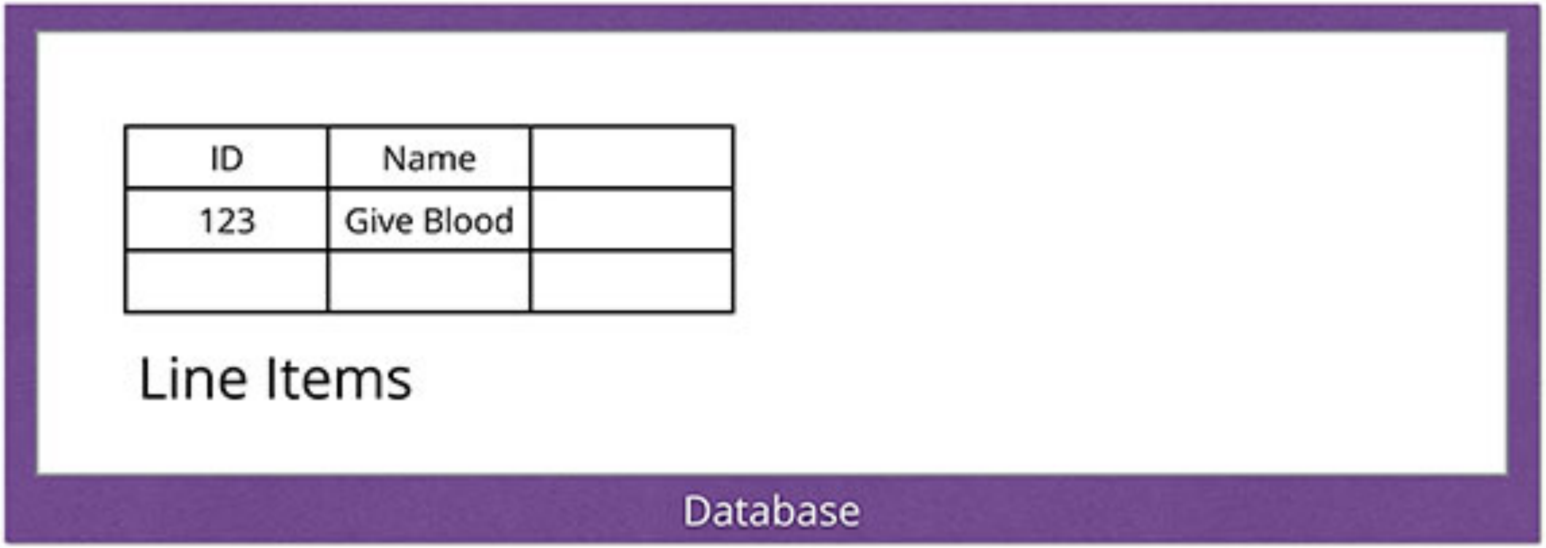
Line Items

Ledger

Catalog DB

Finance DB

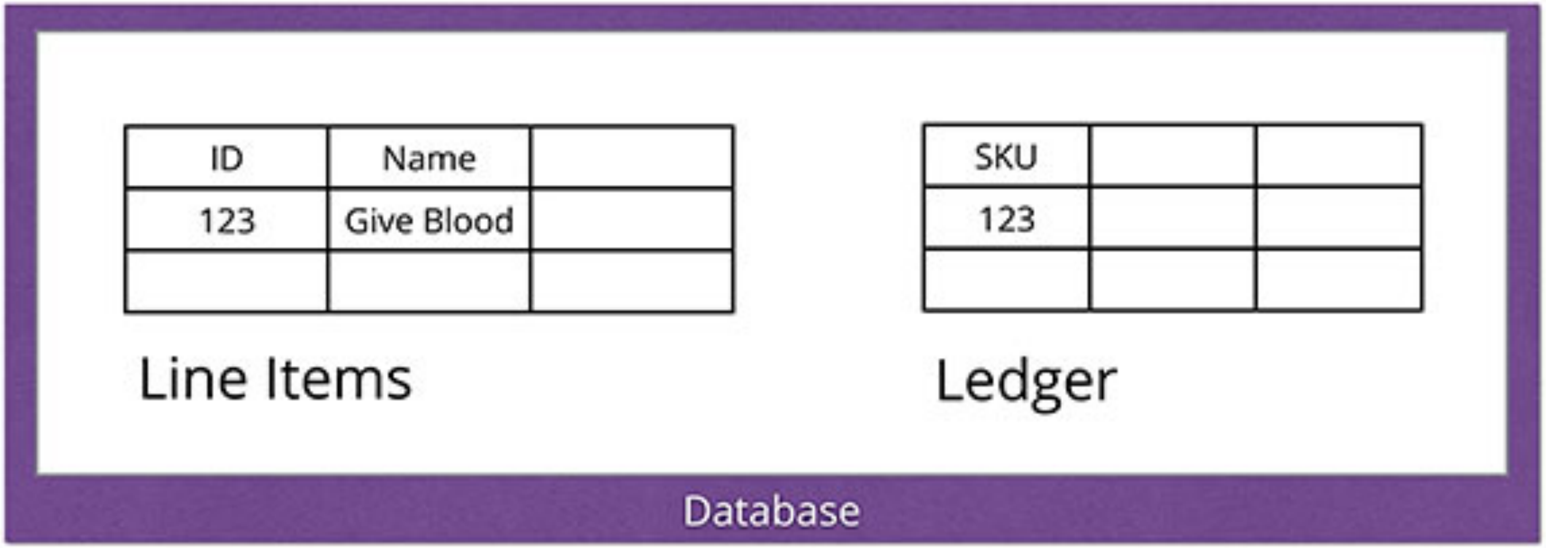
Database



ID	Name	
123	Give Blood	

Line Items

Database

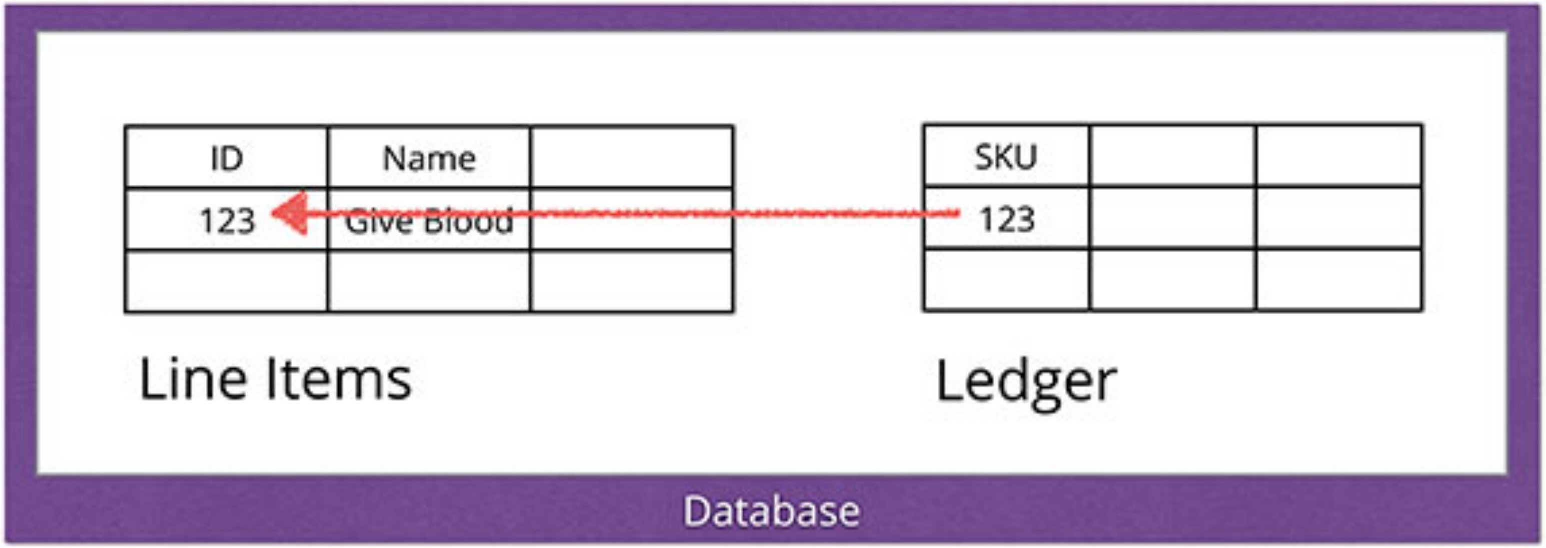


ID	Name	
123	Give Blood	

Line Items

SKU		
123		

Ledger



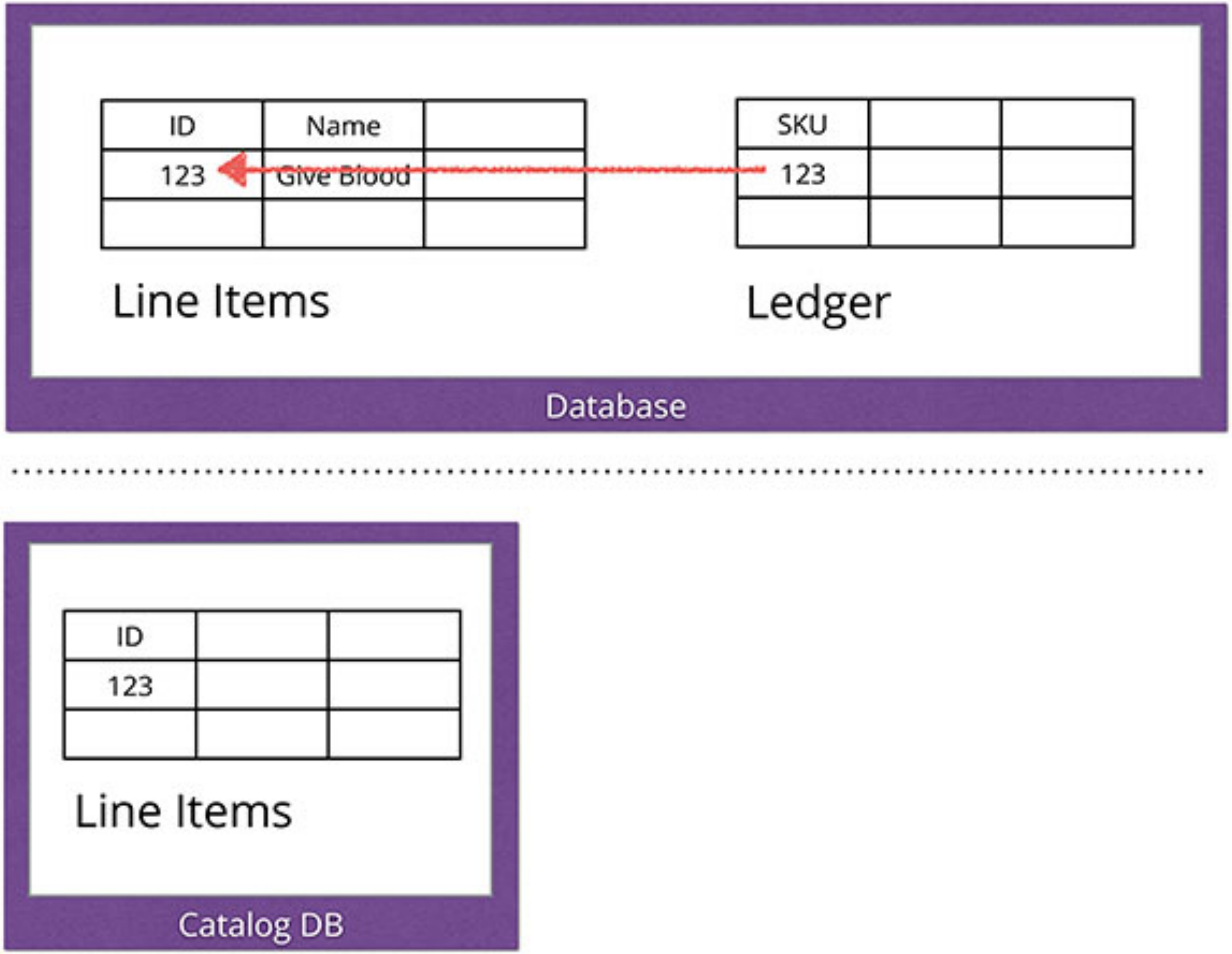
ID	Name	
123	Give Blood	

Line Items

SKU		
123		

Ledger

Database



ID	Name	
123	Give Blood	

Line Items

SKU		
123		

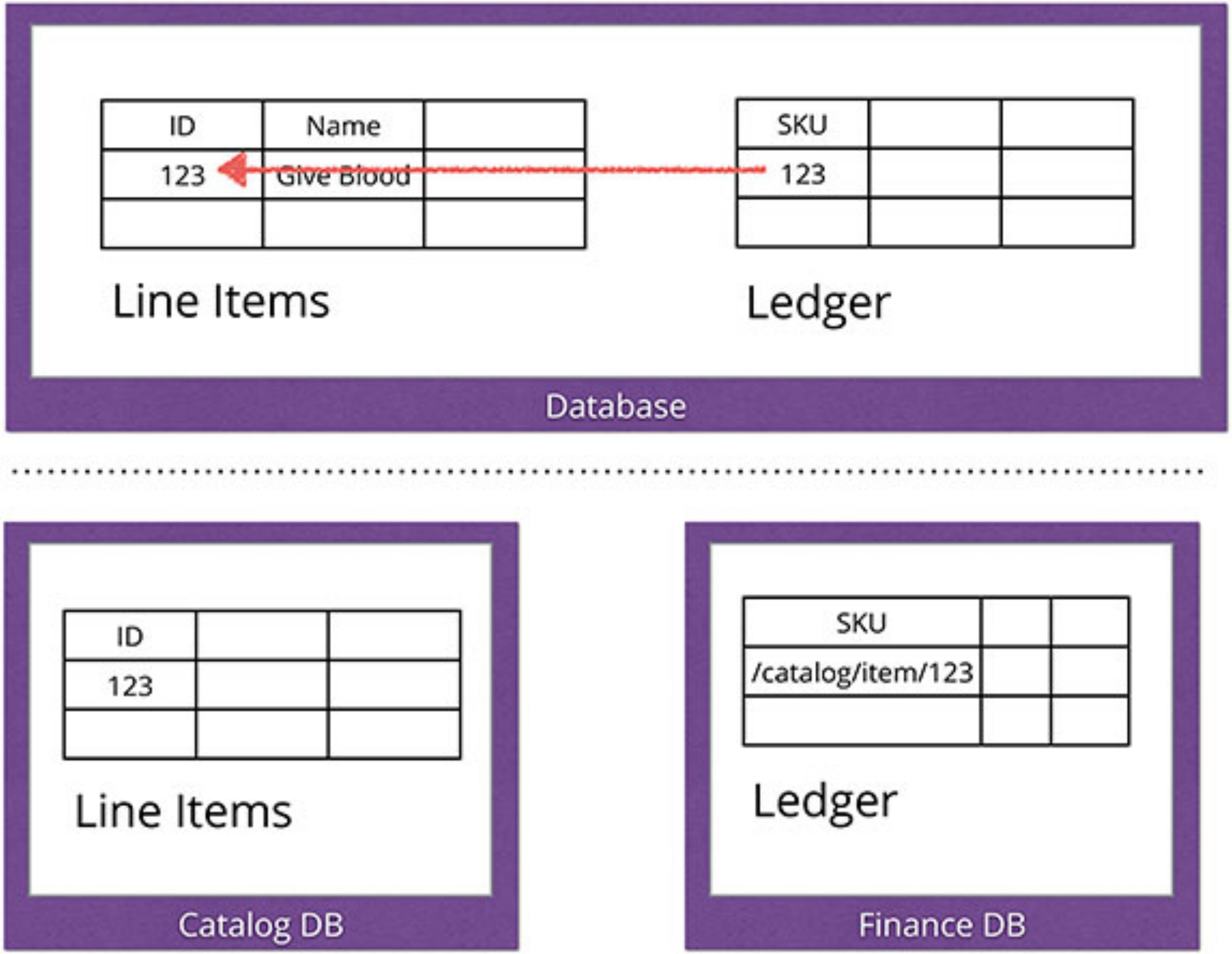
Ledger

Database

ID		
123		

Line Items

Catalog DB



ID	Name	
123	Give Blood	

Line Items

SKU		
123		

Ledger

Database

ID		
123		

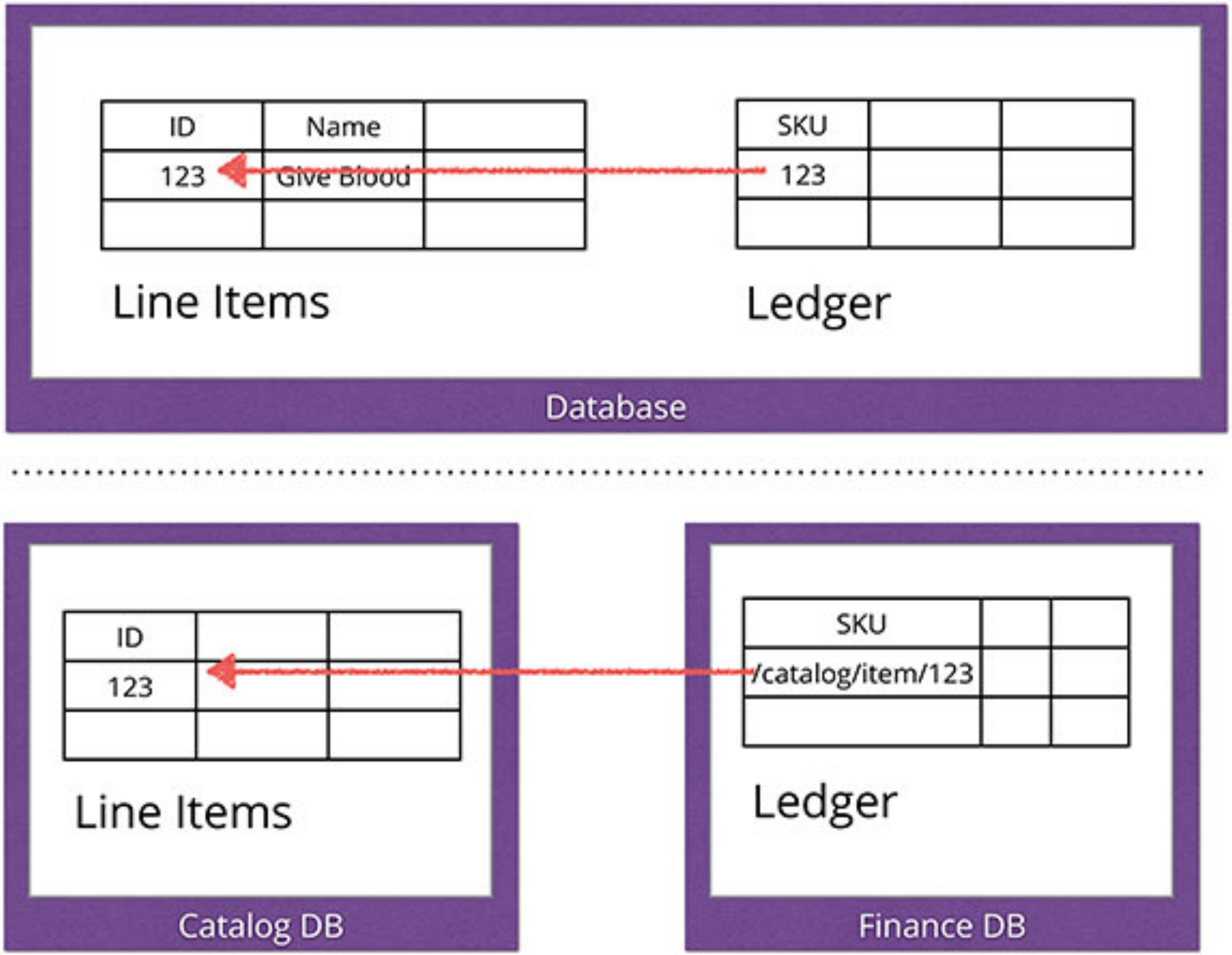
Line Items

Catalog DB

SKU		
/catalog/item/123		

Ledger

Finance DB



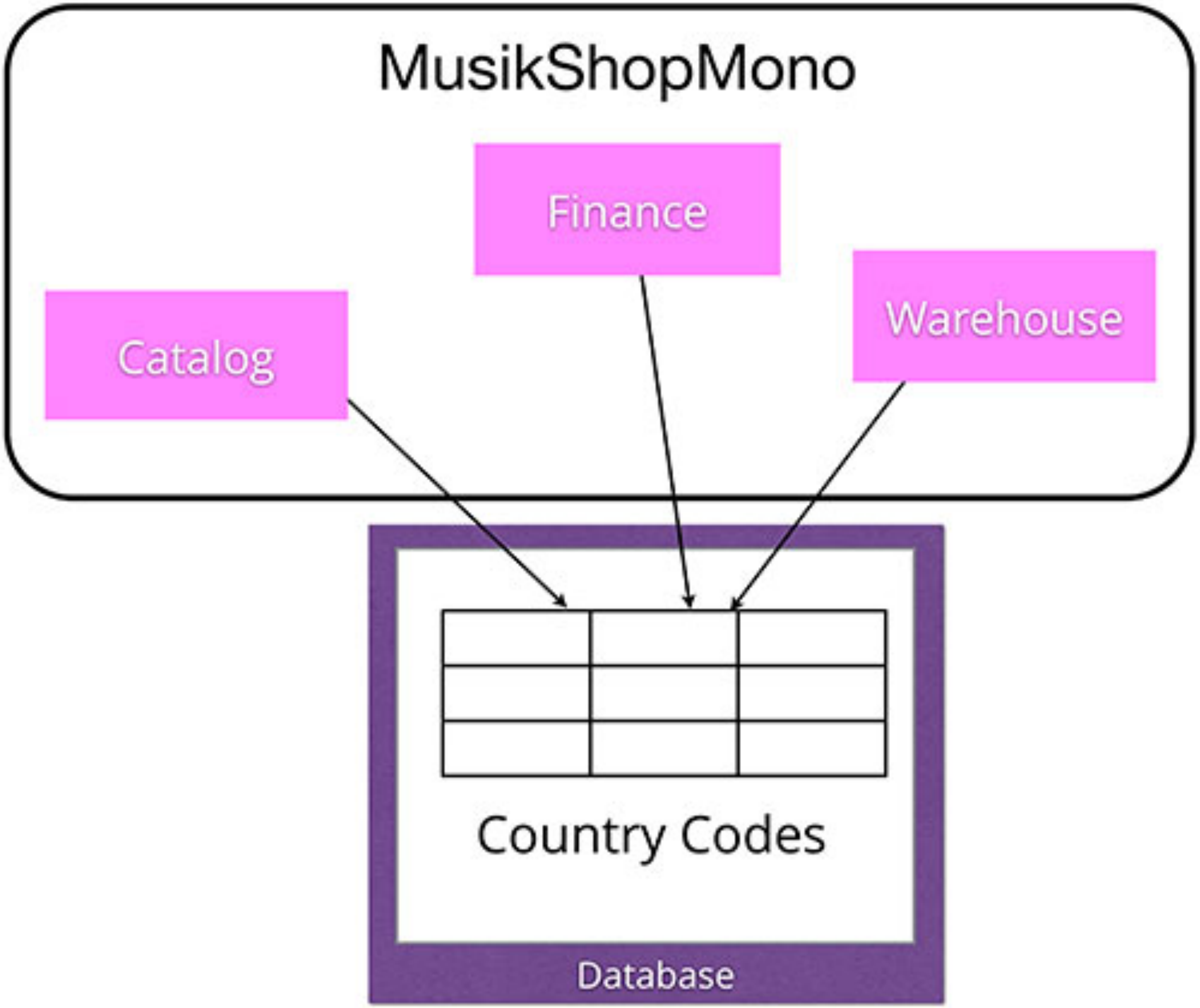
ID	Name	
123	Give Blood	

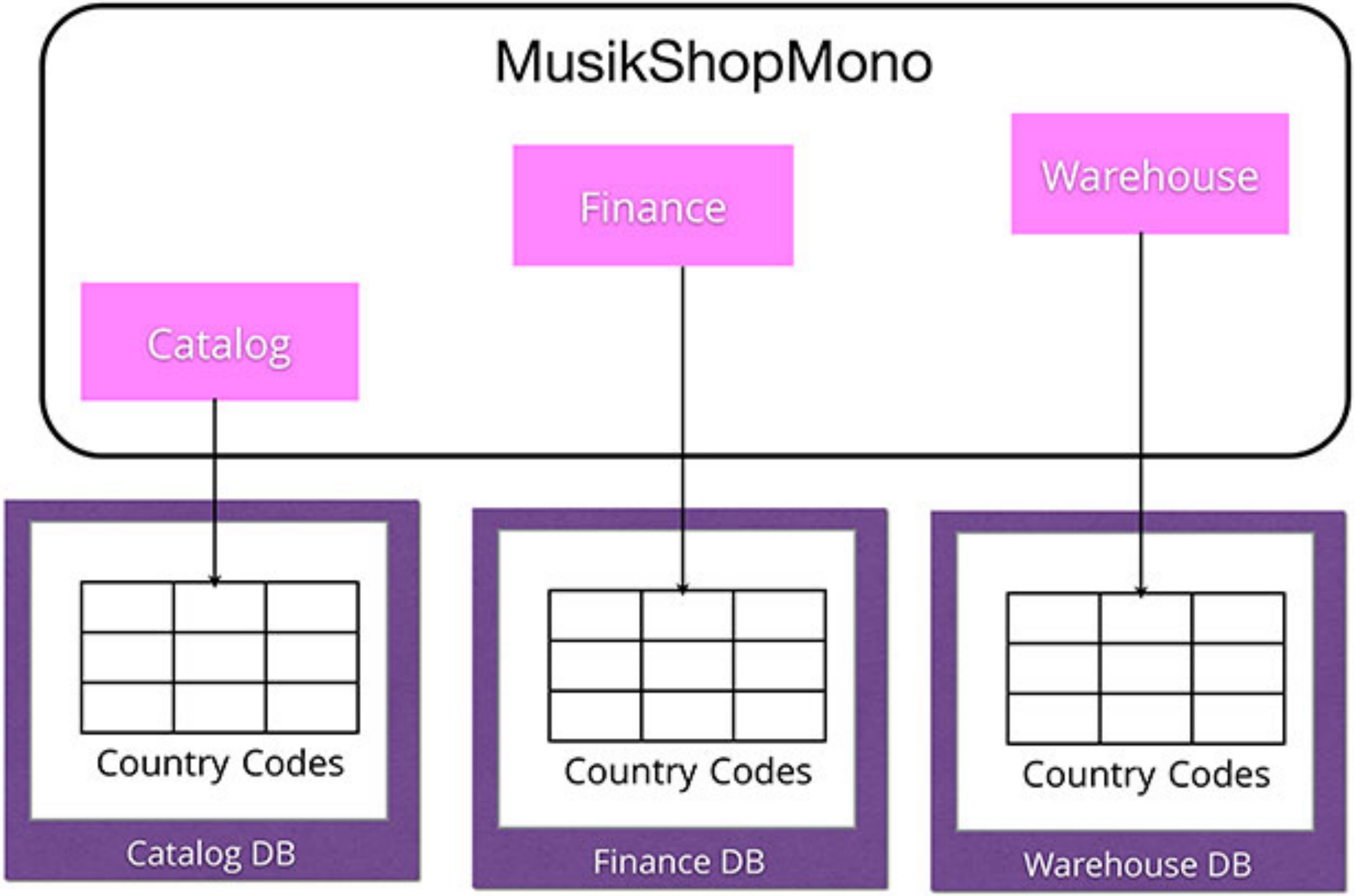
SKU		
123		

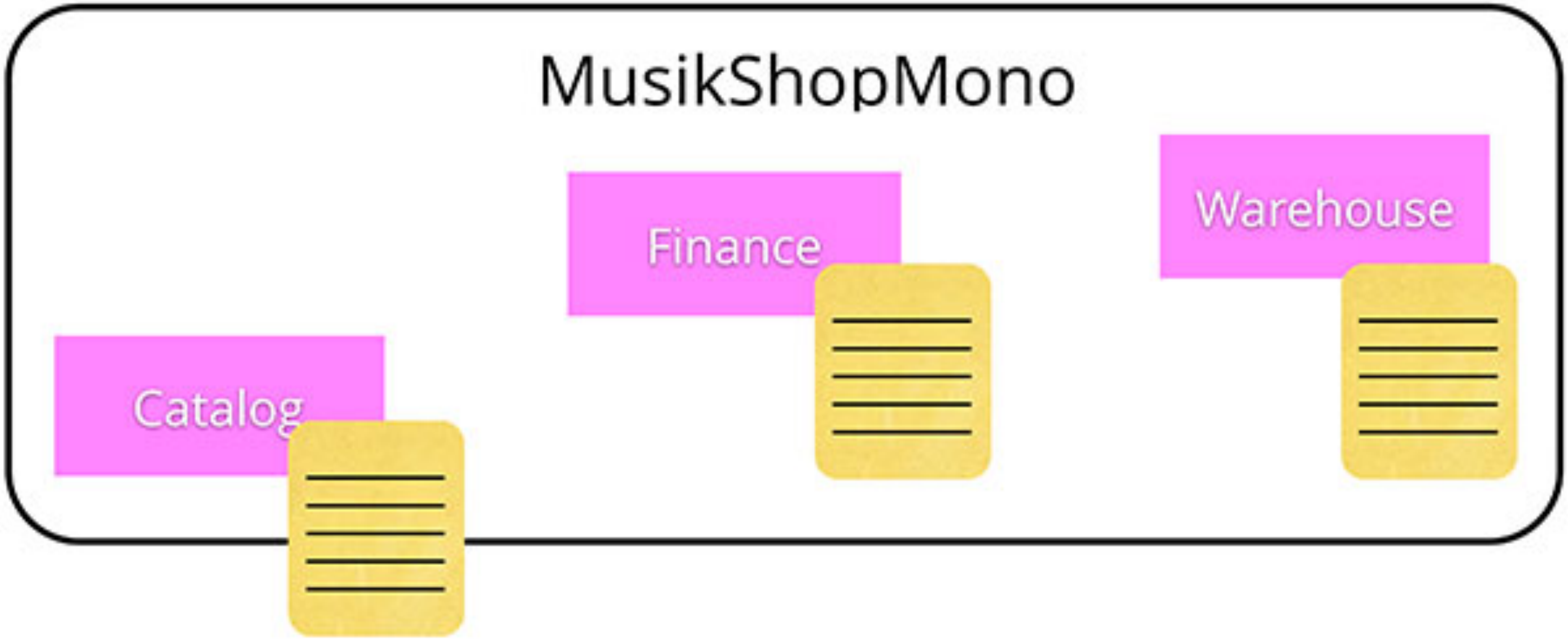
ID		
123		

SKU		
/catalog/item/123		

Breaking foreign key relationships can introduce inconsistency in your system


MusikShopMono
Finance
Warehouse
Catalog
Country Codes
Database


MusikShopMono
Finance
Warehouse
Catalog
Country Codes
Country Codes
Country Codes
Catalog DB
Finance DB
Warehouse DB

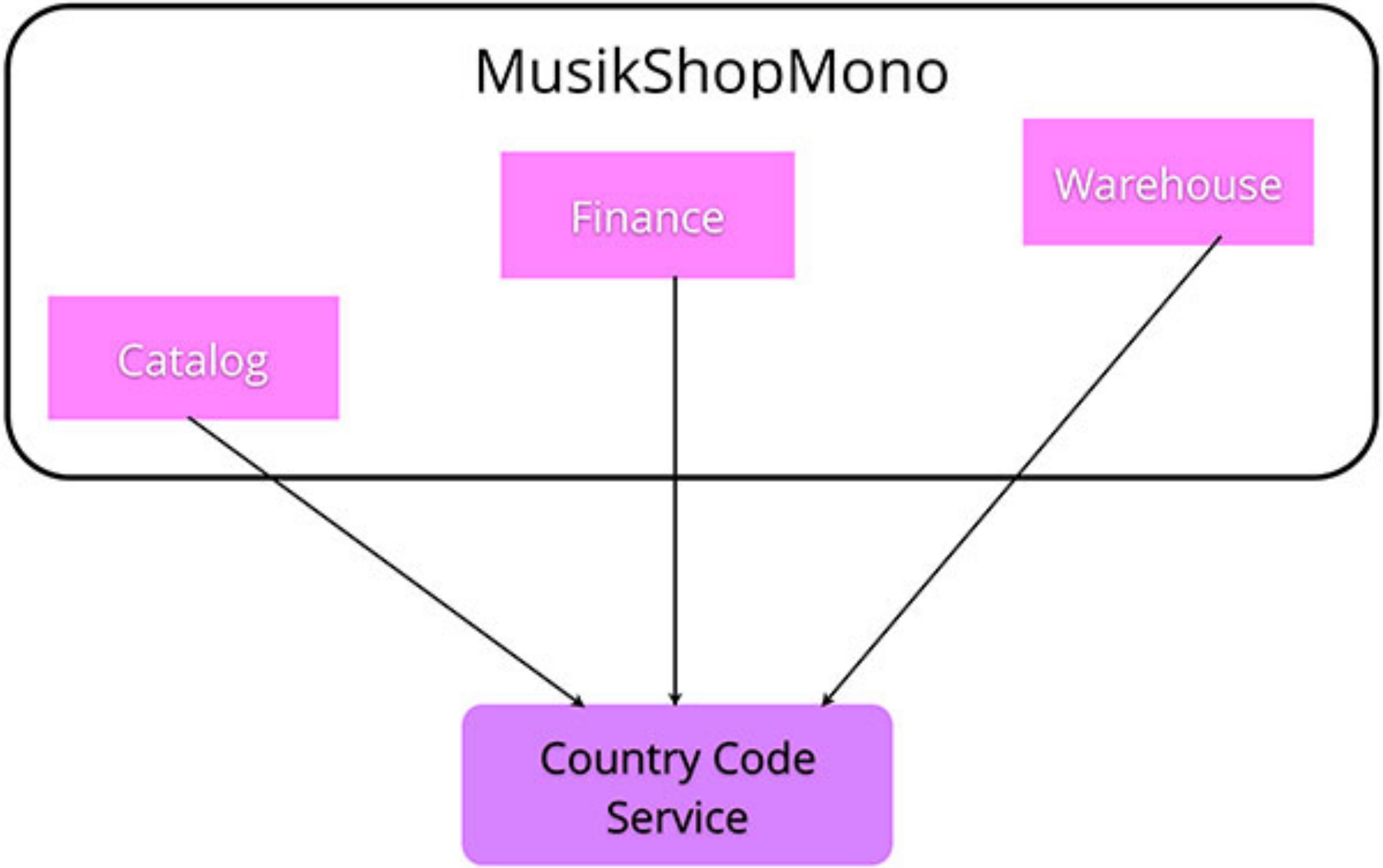


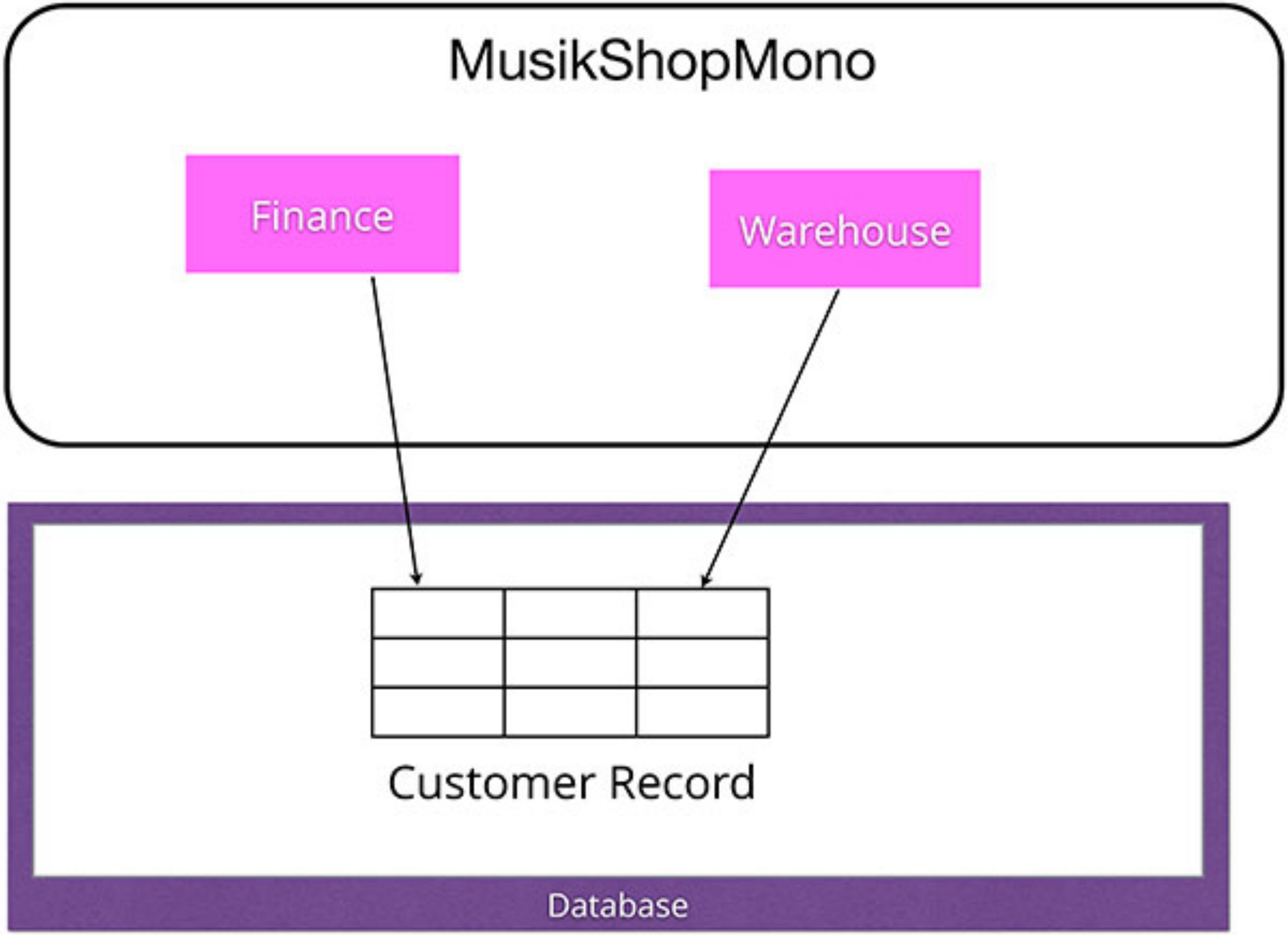
MusikShopMono

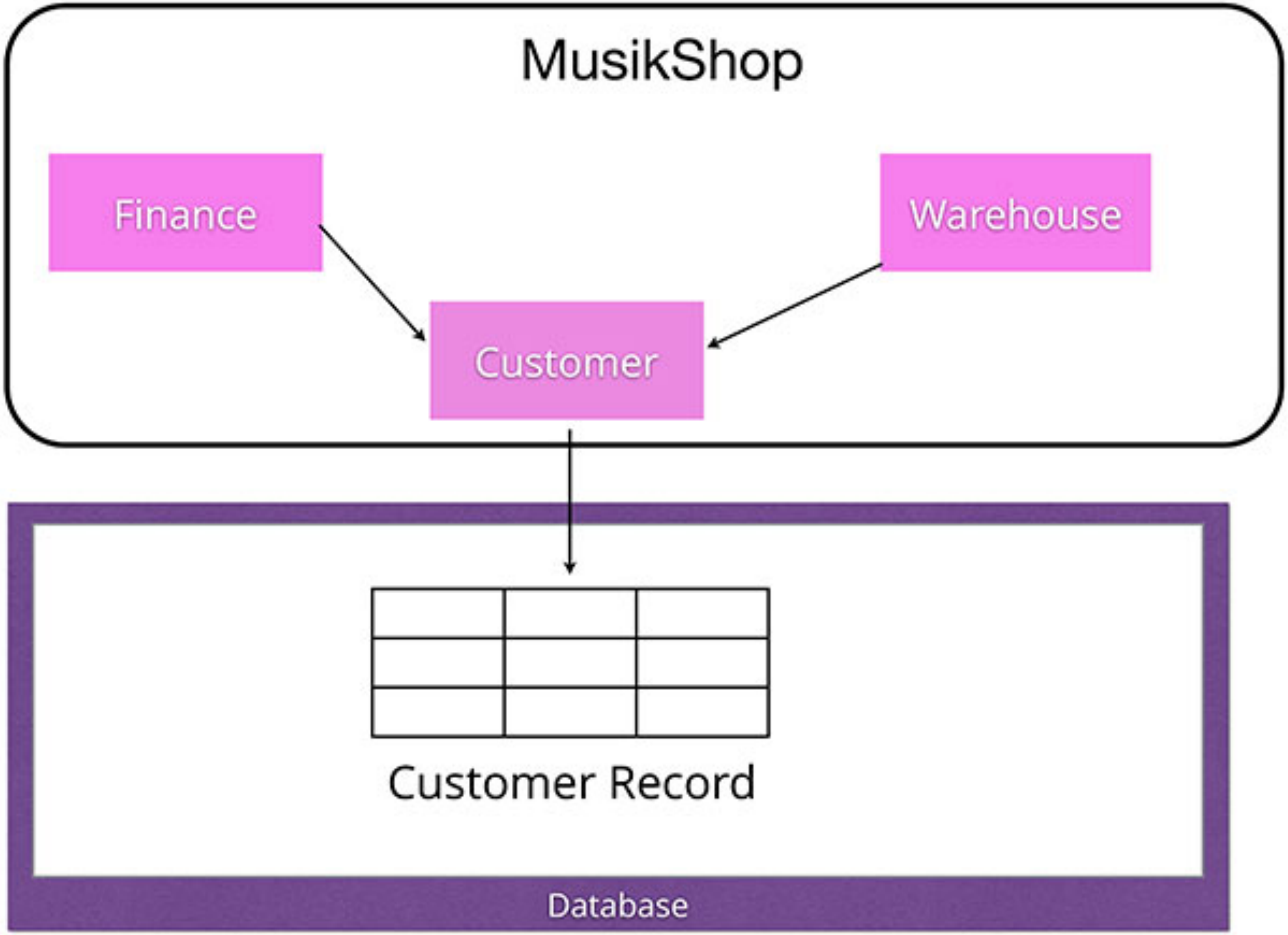
Finance

Warehouse

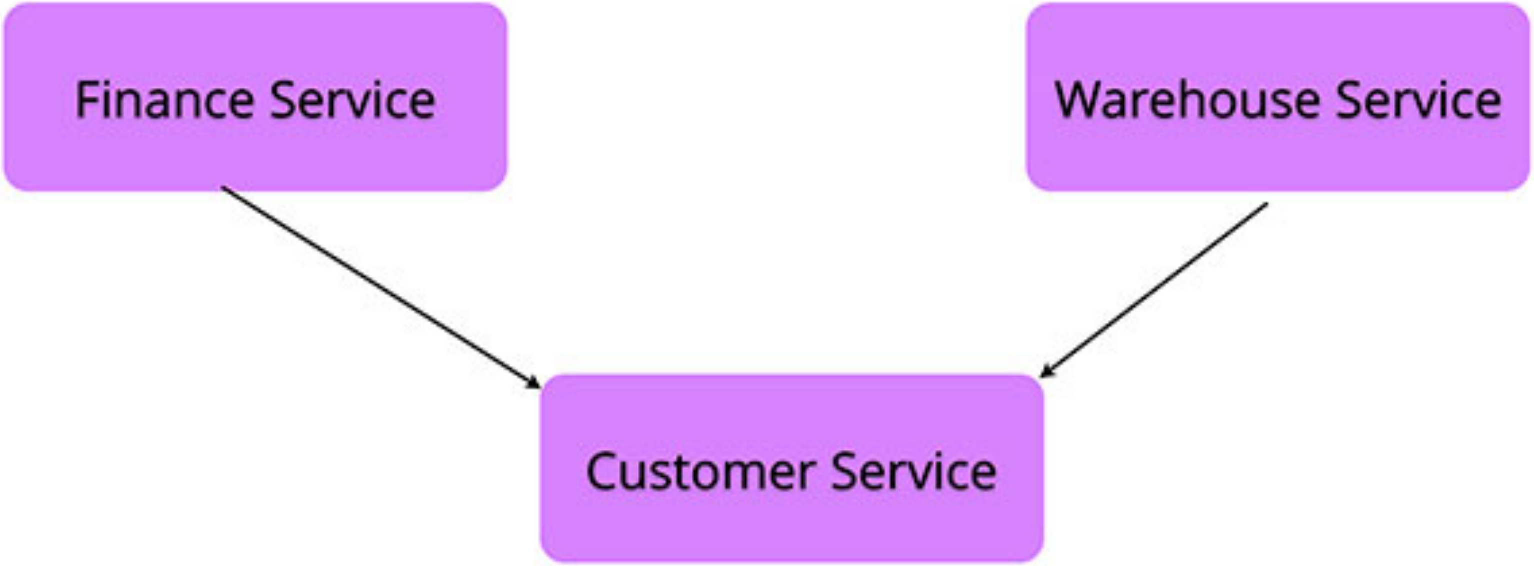
Catalog


MusikShopMono
Finance
Warehouse
Catalog
Country Code Service


MusikShopMono
Finance
Warehouse
Customer Record
Database


MusikShop
Finance
Warehouse
Customer
Customer Record
Database

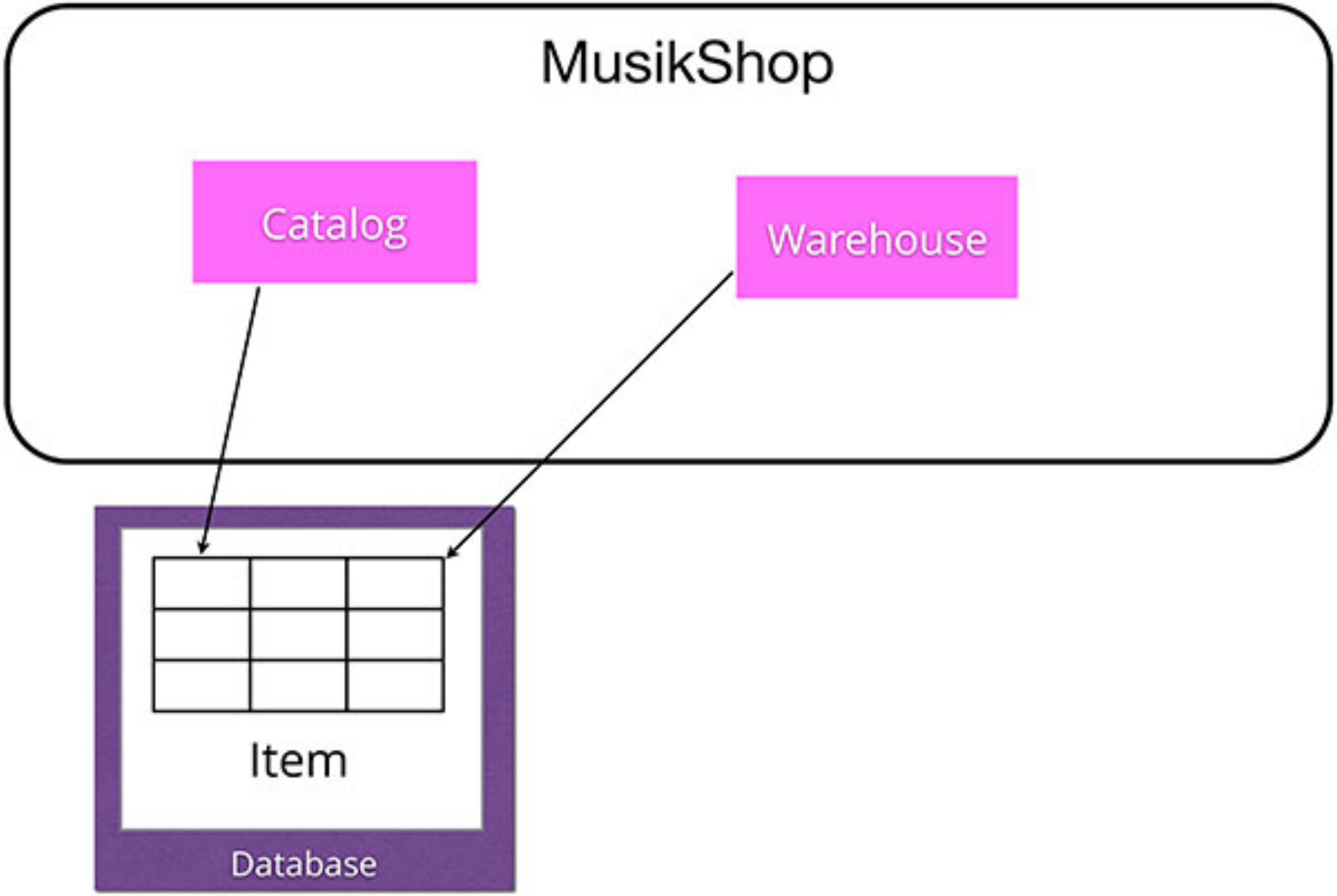
MusikShop System



Finance Service

Warehouse Service

Customer Service



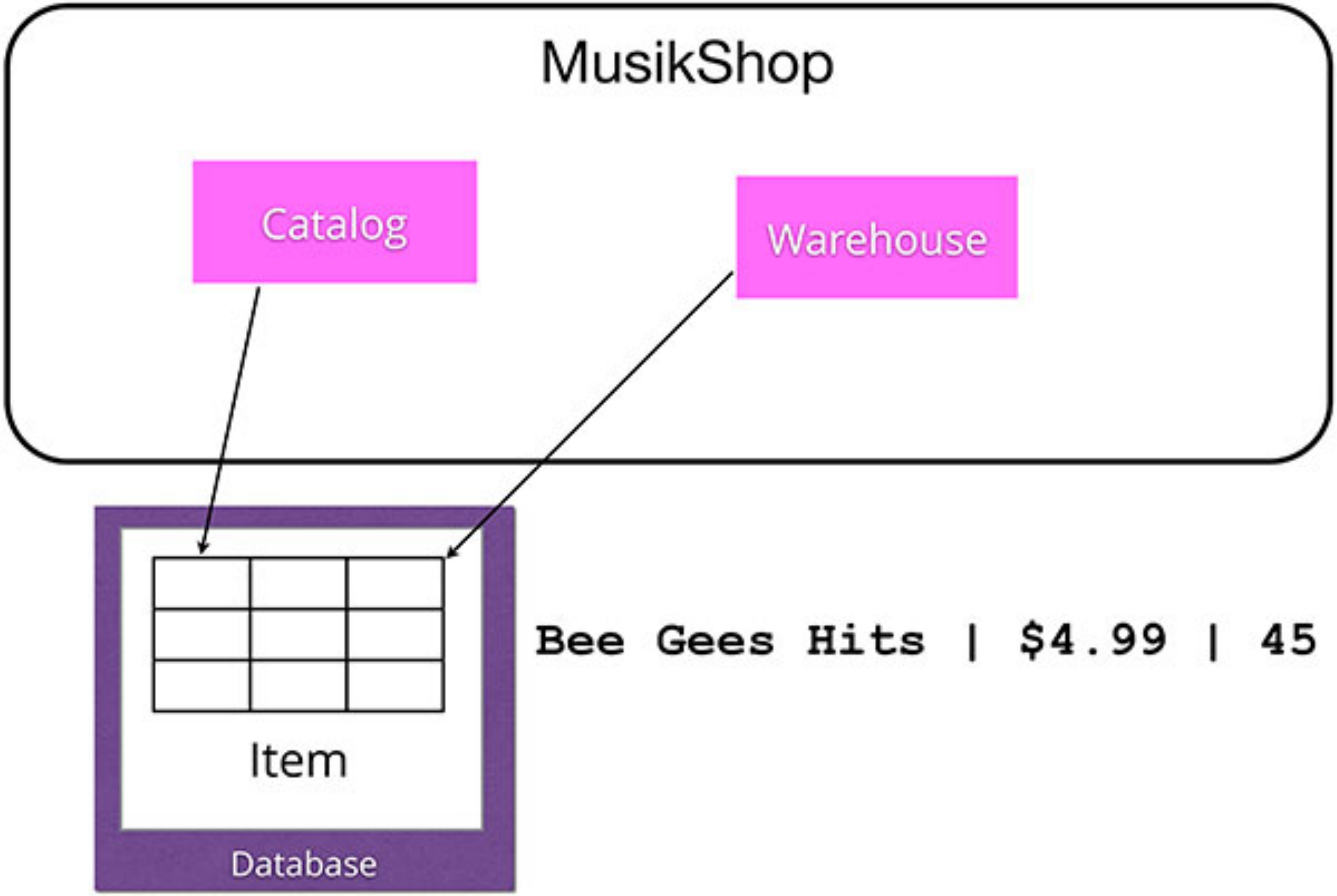
MusikShop

Catalog

Warehouse

Item

Database



MusikShop

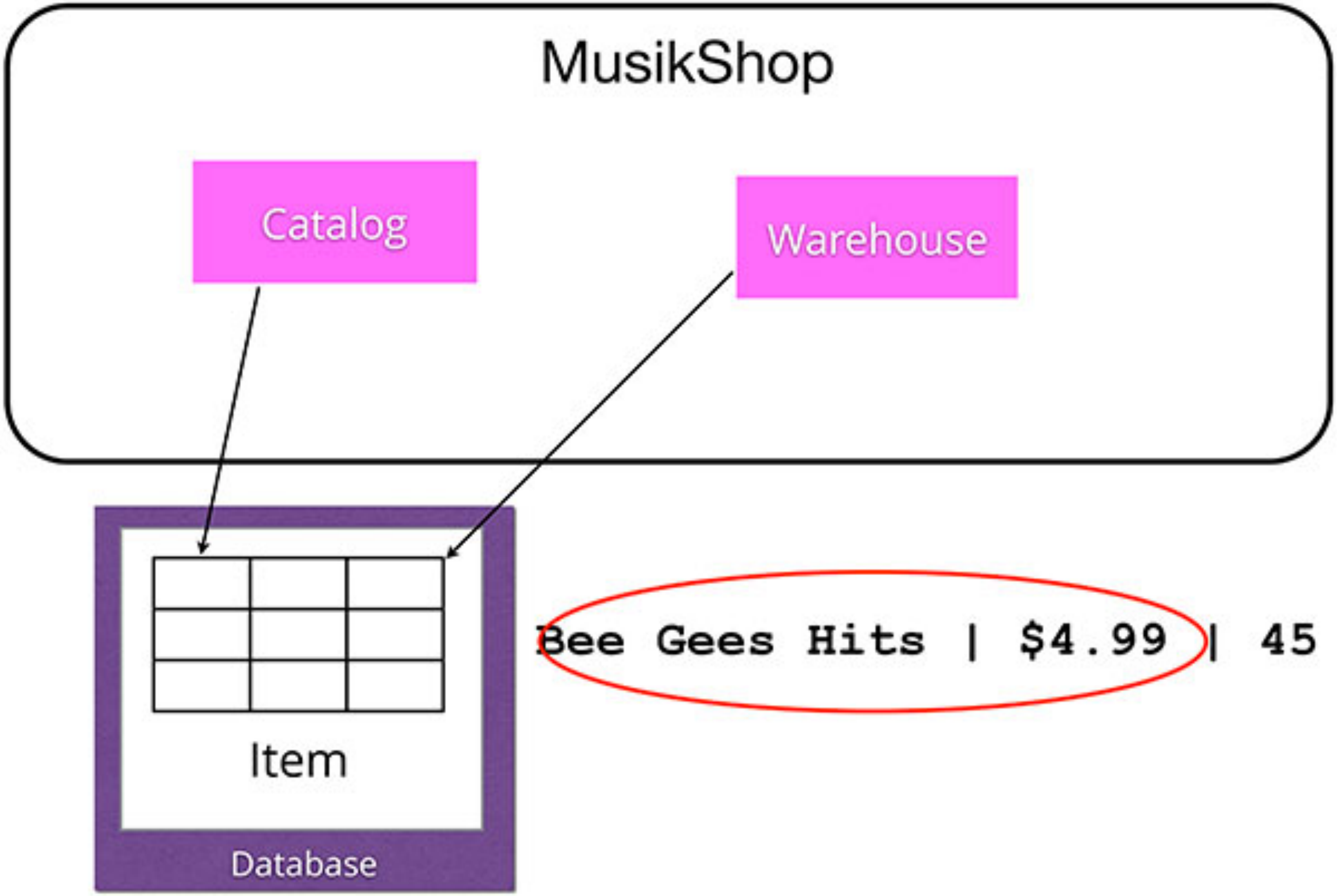
Catalog

Warehouse

Item

Database

Bee Gees Hits | $4.99 | 45


MusikShop

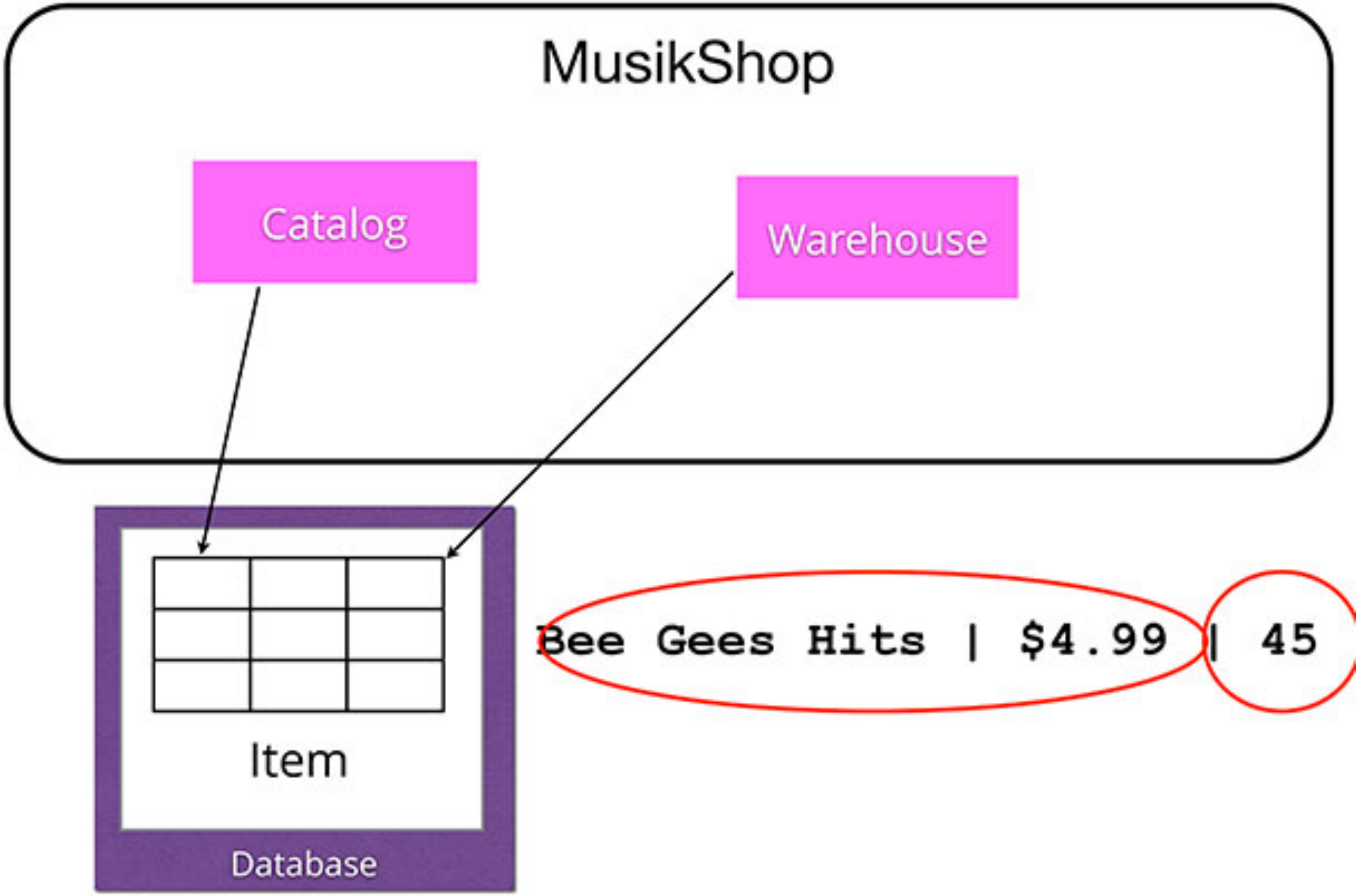
Catalog

Warehouse

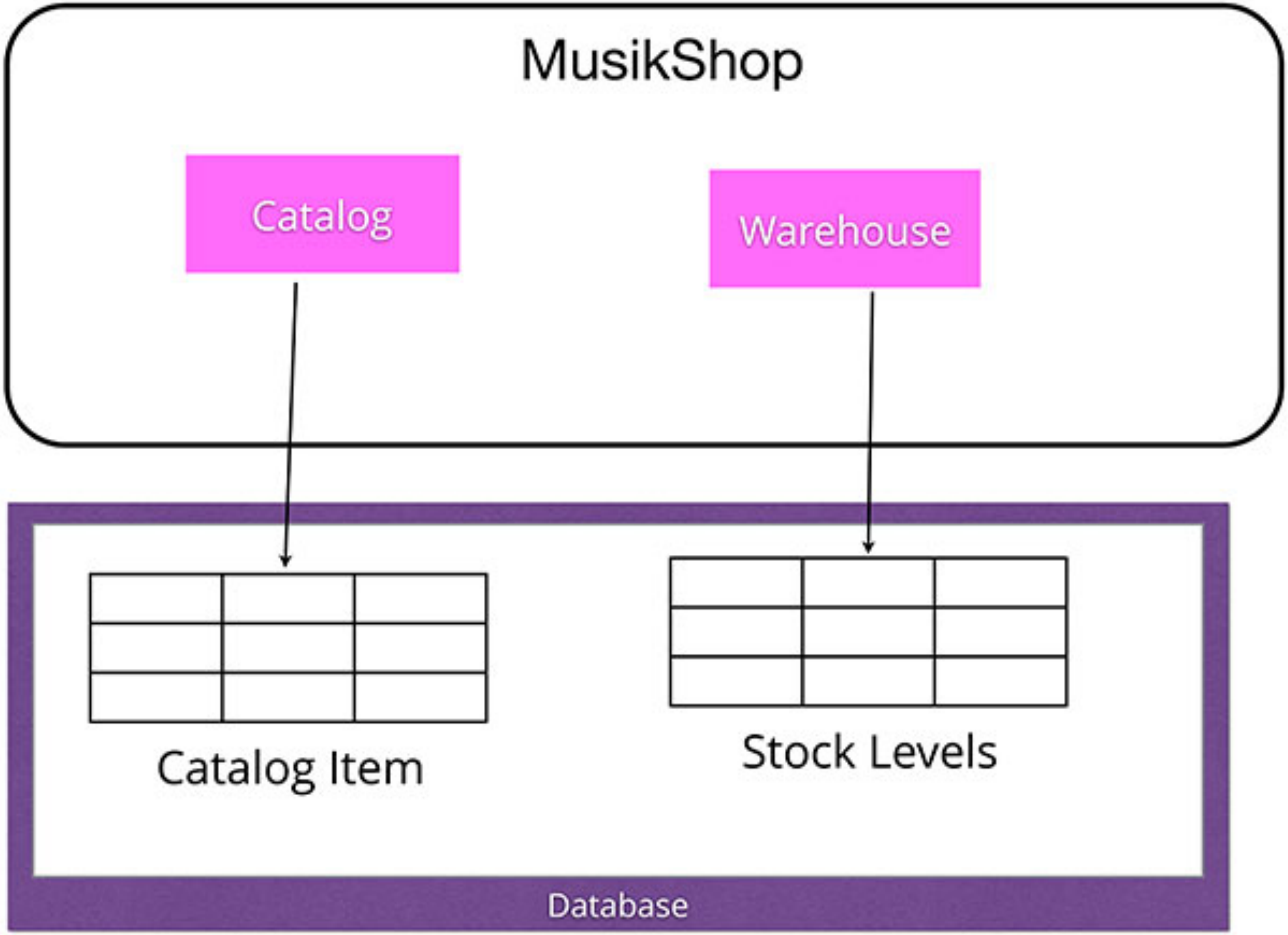
Item

Database

Bee Gees Hits | $4.99 | 45



Bee Gees Hits | $4.99 | 45


MusikShop
Catalog
Warehouse
Catalog Item
Stock Levels
Database

Summary

Where to start

Modelling services

Incremental change

DB Integration & Refactoring