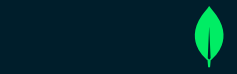
WELCOME

Understanding MongoDB and Document Modeling

Daniel Coupal
Senior Staff Developer Advocate
Strategic Accounts



If you had to design a modern database from scratch …

Objects friendly to code

If you had to design a modern database from scratch …

If you had to design a modern database from scratch …

Objects friendly to code

Schema easy to change

If you had to design a modern database from scratch …

Objects friendly to code

Schema easy to change

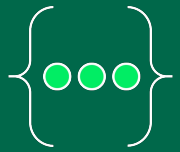
Security

If you had to design a modern database from scratch …

Objects friendly to code

Schema easy to change

Security



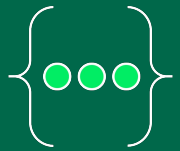
Partitioning

If you had to design a modern database from scratch …

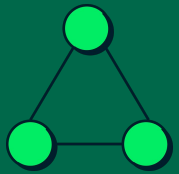
Objects friendly to code

Schema easy to change

Security



Partitioning



Replication

Agenda

1. The Great Migration
   Why do enterprises choose NoSQL?

Agenda

1. The Great Migration
   Why do enterprises choose NoSQL?

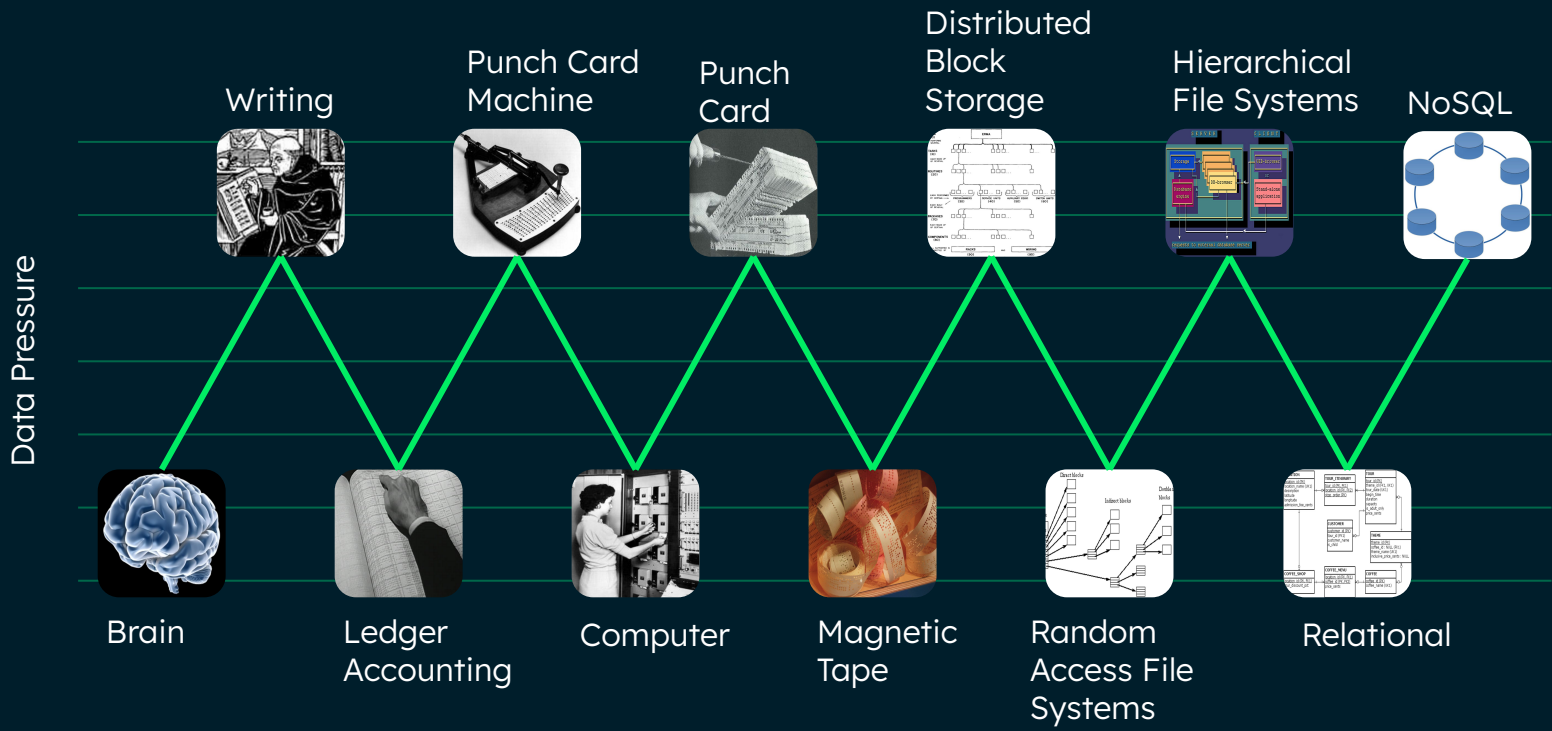
2. Methodology to model for MongoDB
   How designing for NoSQL differs from RDBMS?

Agenda

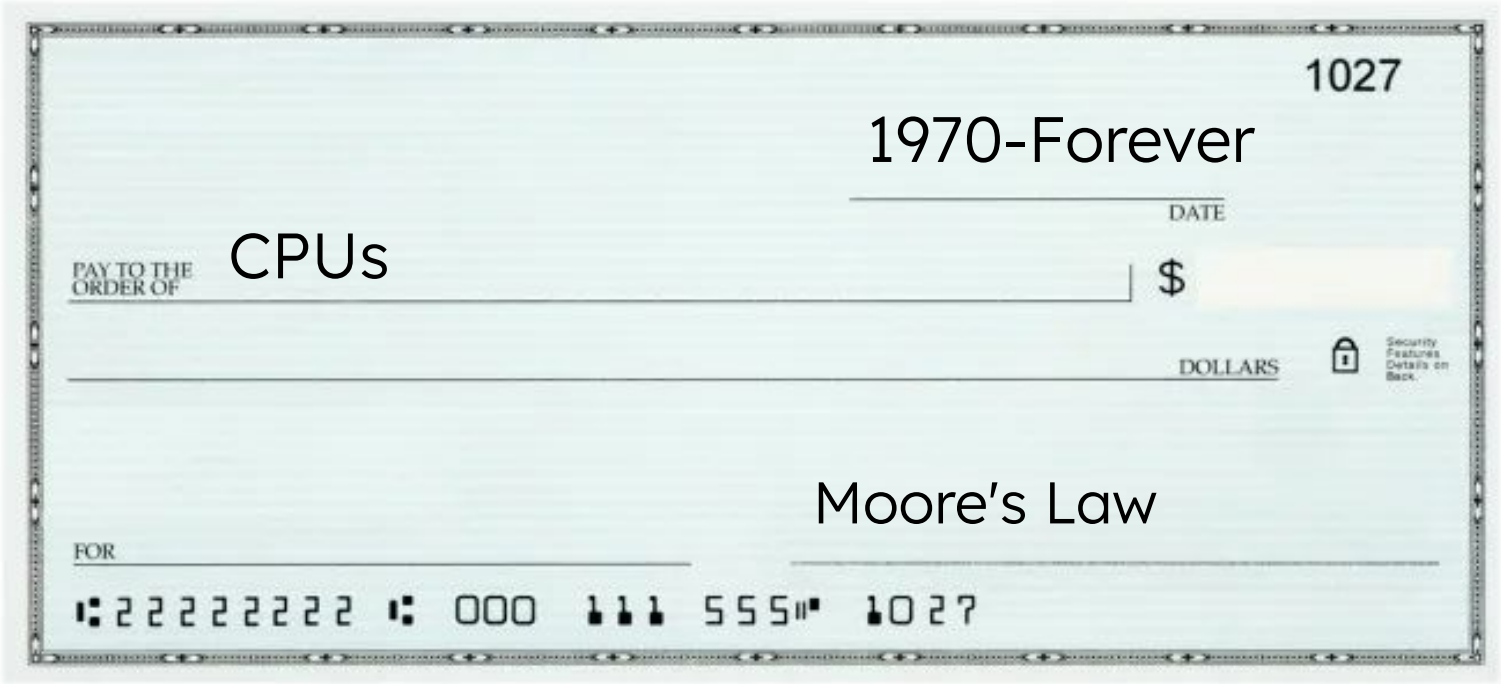
1. The Great Migration
   Why do enterprises choose NoSQL?
2. Methodology to model for MongoDB
   How designing for NoSQL differs from RDBMS?
3. Executing the Plan
   What to do next?

The Great Migration: Why do enterprises choose NoSQL?

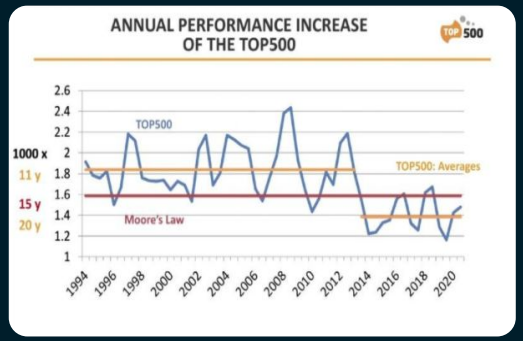
Timeline of Database Technology

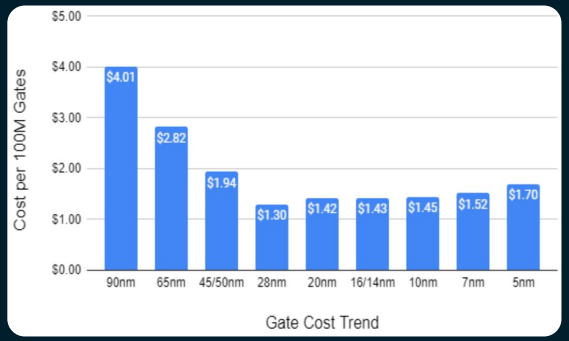


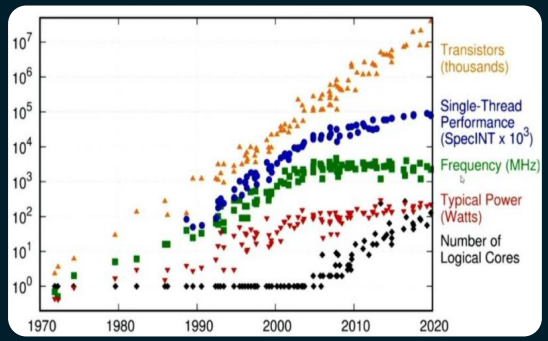
Moore's Law paid the bill



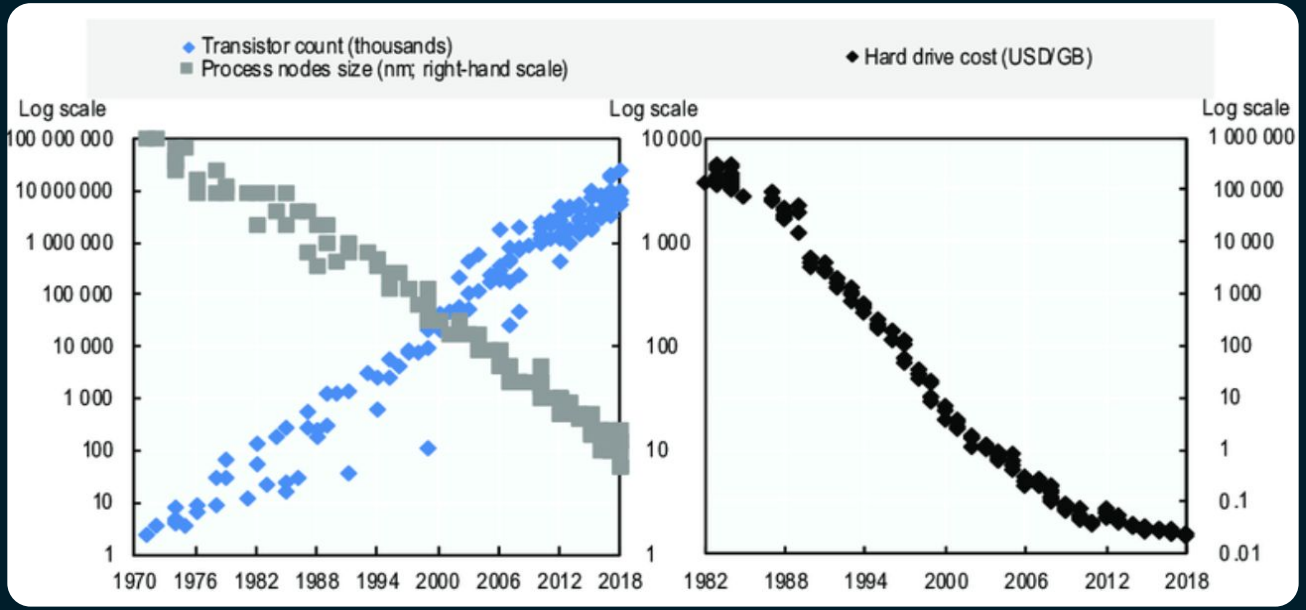
CPU performance is flattening



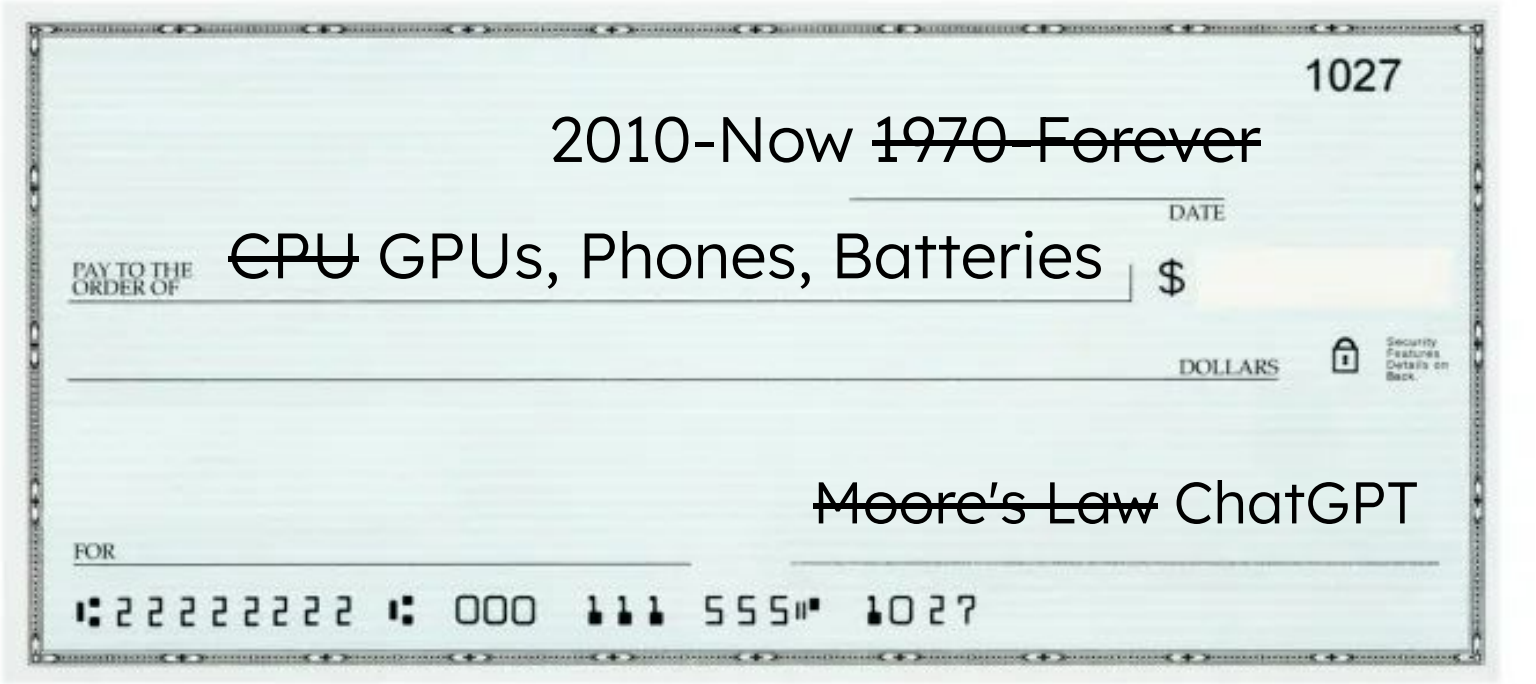




Cost of storage continues to fall

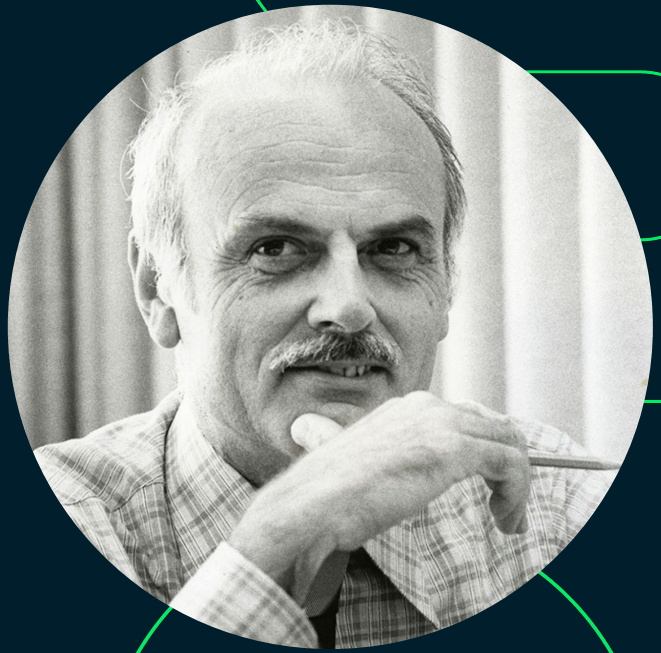


Moore's Law paid the bill ... not anymore for CPU



It’s not new thinking…

A Relational Model of Data for Large Shared Data Banks



EDGAR FRANK "TED" CODD

19 August 1923 — 18 April 2003

> If the strong redundancies in the named set are directly reflected in strong redundancies in the stored set (or if other strong redundancies are introduced into the stored set), then, generally speaking, **extra storage space and update time are consumed** with a **potential drop in query time** for some queries and in load on the central processing units.

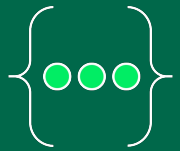
EDGAR FRANK "TED" CODD ON DENORMALIZATION

If you had to design a modern database from scratch …

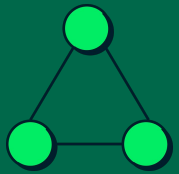
Objects friendly to code

Schema easy to change

Security



Partitioning



Replication

MONGODB

SQL vs NoSQL?

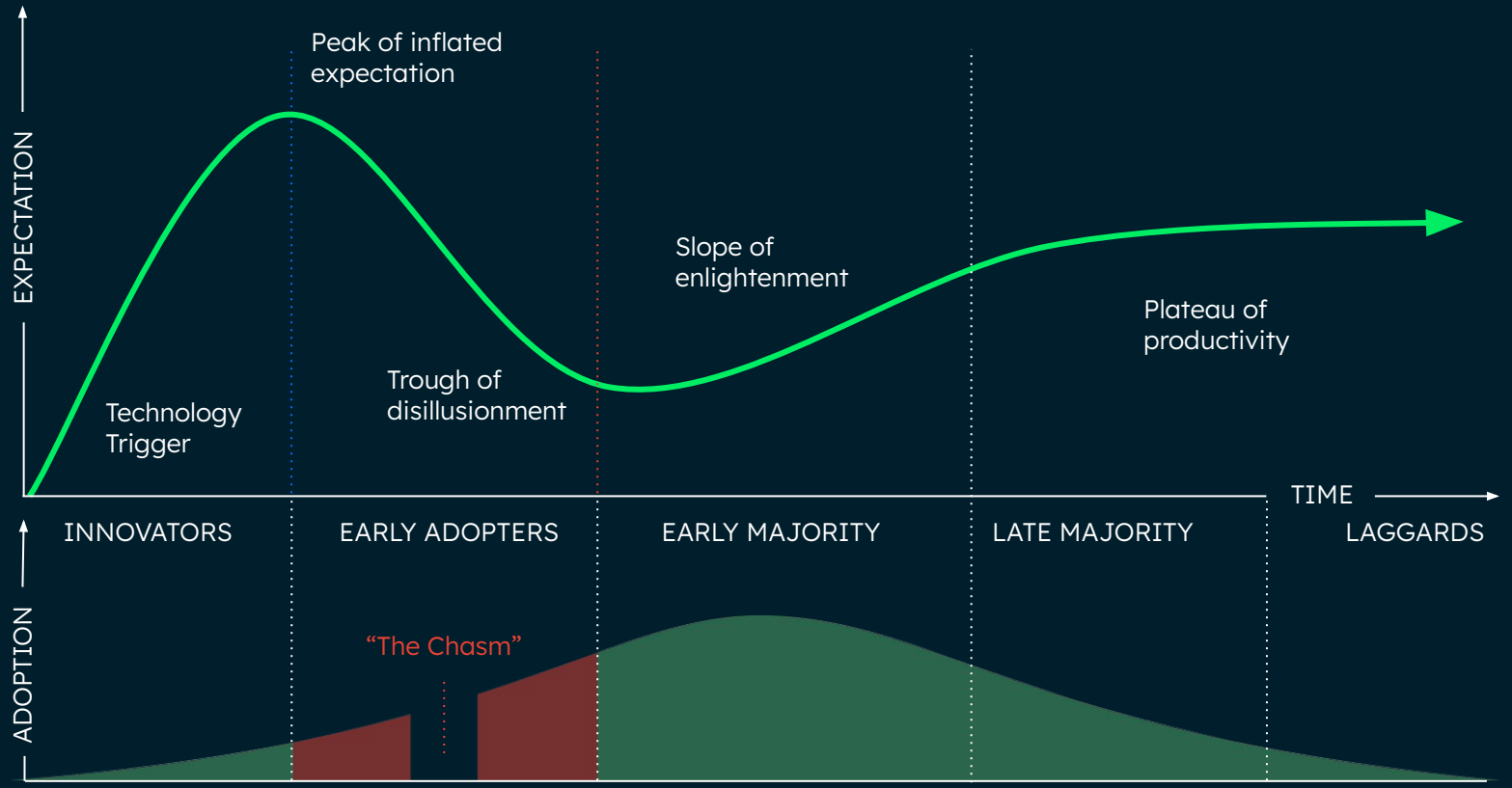
SQL	NoSQL
Optimized for storage	Optimized for compute
Normalized/relational	De-normalized/hierarchical
Ad hoc queries	Instantiated views
Scale vertically	Scale vertically and horizontally
Good for OLAP	Built for OLTP at scale

MONGODB

What is Best?

SQL	NoSQL
Optimized for storage	Optimized for compute ✓
Normalized/relational	De-normalized/hierarchical
Ad hoc queries	Instantiated views
Scale vertically	Scale vertically and horizontally ✓
Good for OLAP	Built for OLTP at scale

Technology Adoption & the Hype Curve



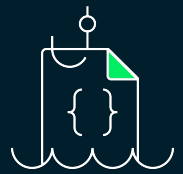
It’s all about relationships and joins

Social network

Document management

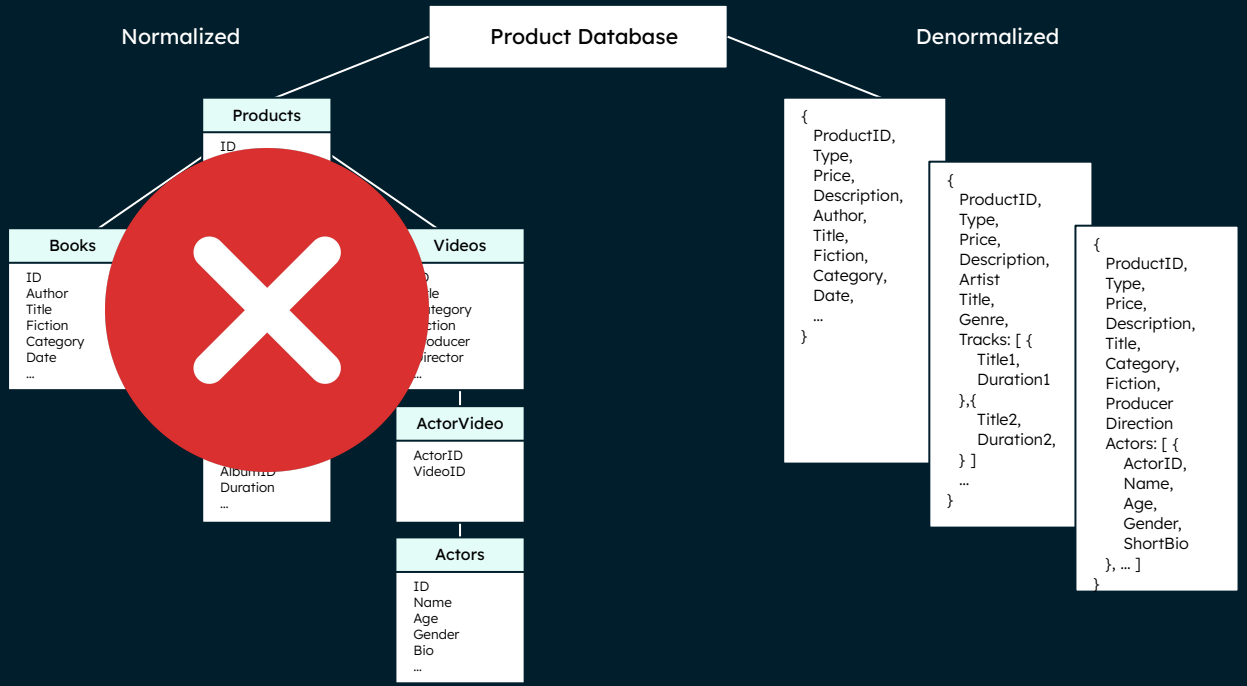
Process control

IT monitoring



Data lake

SQL vs. NoSQL Design



Documents

A row by any other name…

Data in an RDBMS is stored as rows and columns in tables.

CustomerId	CustomerName	Address_Num	Address_St	Address_Town	Address_Area	Address_Code
1325142	Mark Bolan	15	Green road	London	Middlesex	E2 7OF

We could visualise that row as CSV

CustomerId	CustomerName	Address_Num	Address_St	Address_Town	Address_Area	Address_Code
1325142	Mark Bolan	15	Green road	London	Middlesex	E2 7OF

```
CustomerId,CustomerName,Address_Num,Address_St,Address_Town,Adress_Area,Address_Code
1325142,"Mark Bolan",15,"Green road","London","Middlesex","E2 7OF"
```

We could also visualise that row as JSON - that doesn't make it JSON

CustomerId	CustomerName	Address_Num	Address_St	Address_Town	Address_Area	Address_Code
1325142	Mark Bolan	15	Green road	London	Middlesex	E2 7OF

```
{
  "CustomerId": 1325142,
  "CustomerName": "Mark Bolan",
  "Address_Num": 15,
  "Address_St": "Green road",
  "Address_Town": "London",
  "Address_Area": "Middlesex",
  "Address_Code": "E2 7OF"
}
```

In MongoDB Document is also a field type, like Date, Integer or Double => #1 Embedded Documents

CustomerId	CustomerName	Address				
		Num	St	Town	Area	Code
1325142	Mark Bolan	15	Green road	London	Middlesex	E2 7OF

```
{
  "CustomerId": 1325142,
  "CustomerName": "Mark Bolan",
  "Address" : {
              "Num": 15,
              "St": "Green road",
              "Town": "London",
              "Area": "Middlesex",
              "Code": "E2 7OF"
             }
}
```

Where you have multiple values in an RDBMS it needs two tables

CustomerId	Name	CustomerSince
1325142	"Mark Bolan"	"1978-01-03"
1276191	"Sarah O'Brien"	"1996-05-08"
3416713	"Lee Wan"	"2023-03-16"

CustomerId	Phone Type	Phone Number
1325142	Cellphone	077652441661
3416713	Office	01698427887
1325142	Office	02034558982
1276191	Cellphone	07979554414
3416713	Cellphone	07623481631
1325142	Cellphone	07713165342

Using Arrays we co-locate rows from two tables in storage and cache

CustomerId	Name	CustomerSince	Phone (Array)	
			Type	Number
1325142	"Mark Bolan"	"1978-01-03"	Cellphone	077652441661
			Office	02034558982
			Cellphone	07713165342
1276191	"Sarah O'Brien"	"1996-05-08"	Cellphone	07979554414
3416713	"Lee Wan"	"2023-03-16"	Cellphone	07623481631
			Office	01698427887

Using Arrays we co-locate in storage and cache

=> #2 Arrays (representation of a 1-many or many-many relationship)

CustomerId	Name	CustomerSince	Phone (Array)	
			Type	Number
1325142	"Mark Bolan"	"1978-01-03"	Cellphone	077652441661
			Office	02034558982
			Cellphone	07713165342

```
{ "CustomerId": 1325142,
  "Name": "Mark Bolan",
  "CustomerSince": "1978-01-03T00:00:00Z",
  "Phone": [
             { "Type": "Cellphone", "Number" :077652441661 },
             { "Type": "Office", "Number" :02034558982 },
             { "Type": "Cellphone", "Number" :07713165342 },
   ]
}
```

Ad hoc “joins” in SQL

```
SELECT * FROM PRODUCTS INNER JOIN BOOKS ON
productId = productId
INNER JOIN ALBUMS ON productId = productId
INNER JOIN VIDEOS ON productId = productId
INNER JOIN ACTORS ON actorId = actorId
WHERE name = “Movie Title”
```

bookId	productId	author	publisher	IBSN-10
1	1	Mary Shelley	Bantam	553212478
2	4	Charlotte Bronte	Wordsworth	1853260207

albumId	productId	artist	producer	releaseDate
1	2	Dire Straits	Muff Winwood	10/7/78
2	5	Pink Floyd	Pink Floyd	3/1/73

videoId	productId	author	director	releaseDate
1	3	Ann Spielberg	Penny Marshall	6/5/88
2	6	Robert Rodat	Steven Spielberg	7/21/98

actorVideoId	gender	name	birthDate
1	M	Tom Hanks	7/9/56
2	F	Elizabeth Perkins	11/18/60
3	M	Robert Loggia	1/3/30

actorVideoId	videoId	actorId	character
1	1	1	Josh
2	2	1	Captain Miller
3	1	2	Susan
4	1	3	MacMillan

productId	name	type	price
1	Frankenstein	Book	11.99
2	Dire Straits	Album	17.49
3	Big	Video	14.99
4	Jane Eyre	Book	10.99
5	The Dark Side of the Moon	Album	17.49
6	Saving Private Ryan	Video	18.99

trackId	albumId	song	duration
1	1	Down to the Waterline	3:55
2	1	Water of Love	5:23
3	1	Setting Me Up	3:18
4	1	Six Blade Knife	4:10
5	1	Southbound Again	2:58
6	1	Sultans of Swing	5:47
7	1	In the Gallery	6:16
8	1	Wild West End	4:42
9	1	Lions	5:05
10	2	Speak to Me	1:13
11	2	Breathe	2:43
12	2	On the Run	3:36
13	2	Time	4:36
14	2	The Great Gig in the Sky	19:27
15	2	Money	6:23
16	2	Us and Them	7:49
17	2	Any Colour You Like	3:26
18	2	Brain Damage	3:49
19	2	Eclipse	2:03

Time Complexity: **O(log(N) + Nlog(M))+ Nlog(M))+ Nlog(M))**

Modeled “joins” in NoSQL

SELECT * WHERE _id = { or like: “Big” }

Time Complexity: O(log(N))

```
{
    "_id": "Big",
    "data": {
        "price": 14.99,
        "release": "1988-06-05",
        "roles": {
            "Josh": "Tom Hanks",
            "Susan": "Elizabeth Perkins",
            "MacMillan": "Robert Loggia"
        }
    },
    "target": [
        {"id": "Big"},
        {"id": "Penny Marshall", "type": "director"},
        {"id": "Ann Spielberg", "type": "writer"},
        {"id": "Tom Hanks", "type": "actor"},
        {"id": "Elizabeth Perkins", "type": "actor"},
        {"id": "Robert Loggia", "type": "actor"}
    ]
},
```

Bank Secrecy Act (BSA)

The Bank Secrecy Act (BSA), 31 USC 5311 et seq establishes program, recordkeeping and reporting requirements for national banks, federal savings associations, federal branches and agencies of foreign banks.

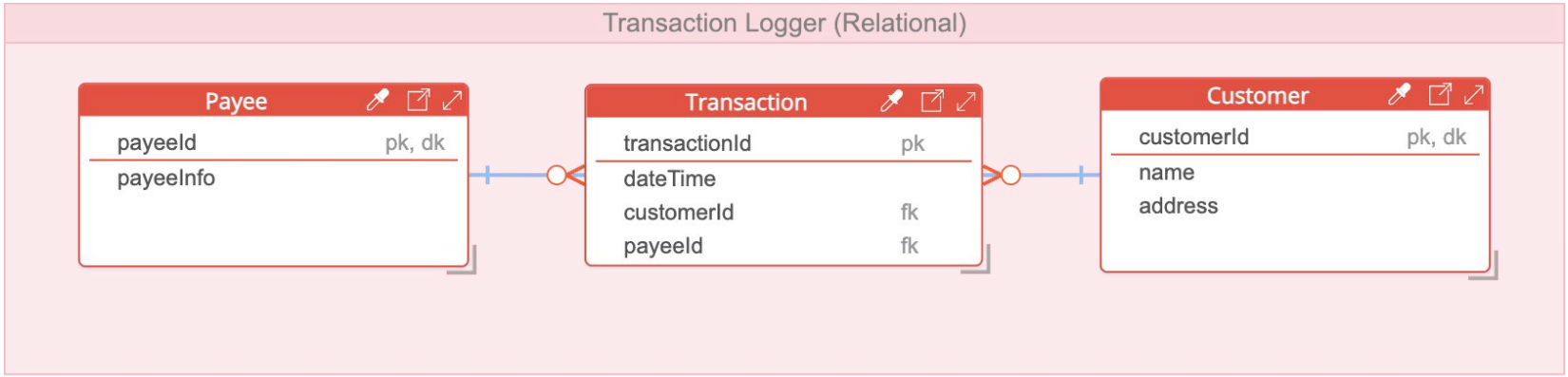
- This regulation requires every national bank to file a Suspicious Activity Report (SAR) when they **detect** certain known or **suspected violations** of federal law or **suspicious transactions** related to a money laundering activity or a violation of the BSA. A SAR filing is required for any potential crimes.

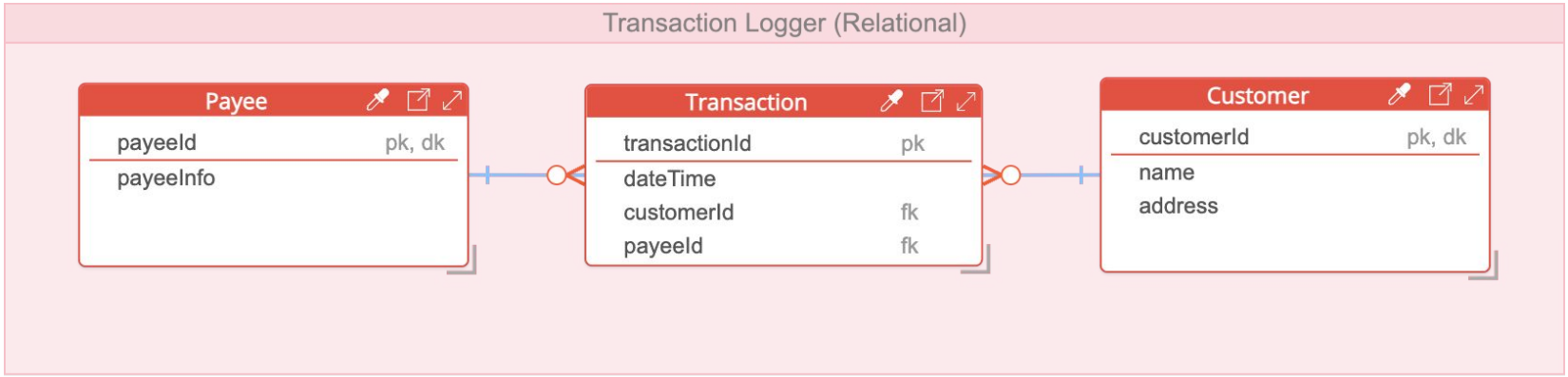
Bank Secrecy Act (BSA)

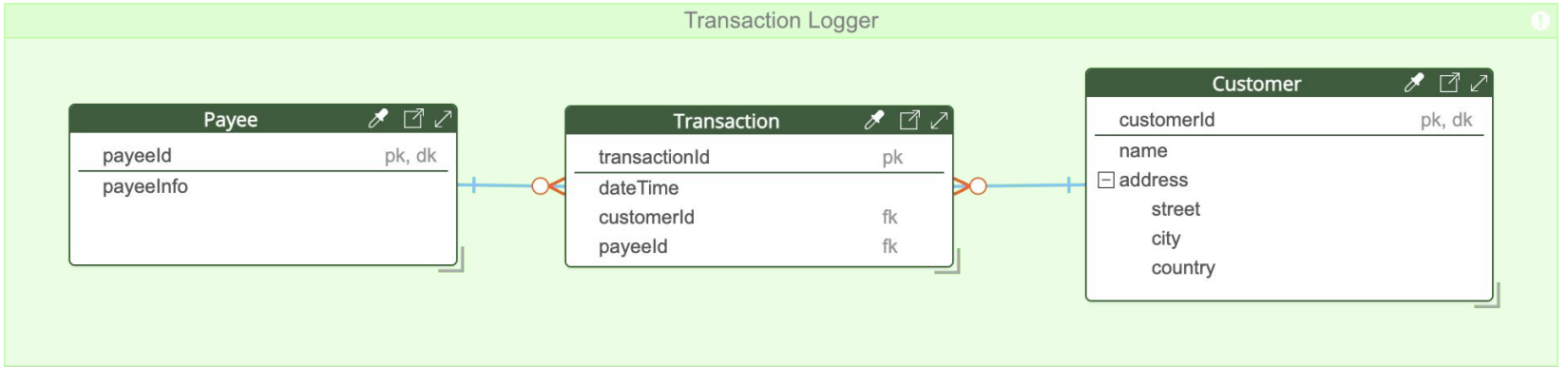
The Bank Secrecy Act (BSA), 31 USC 5311 et seq establishes program, recordkeeping and reporting requirements for national banks, federal savings associations, federal branches and agencies of foreign banks.

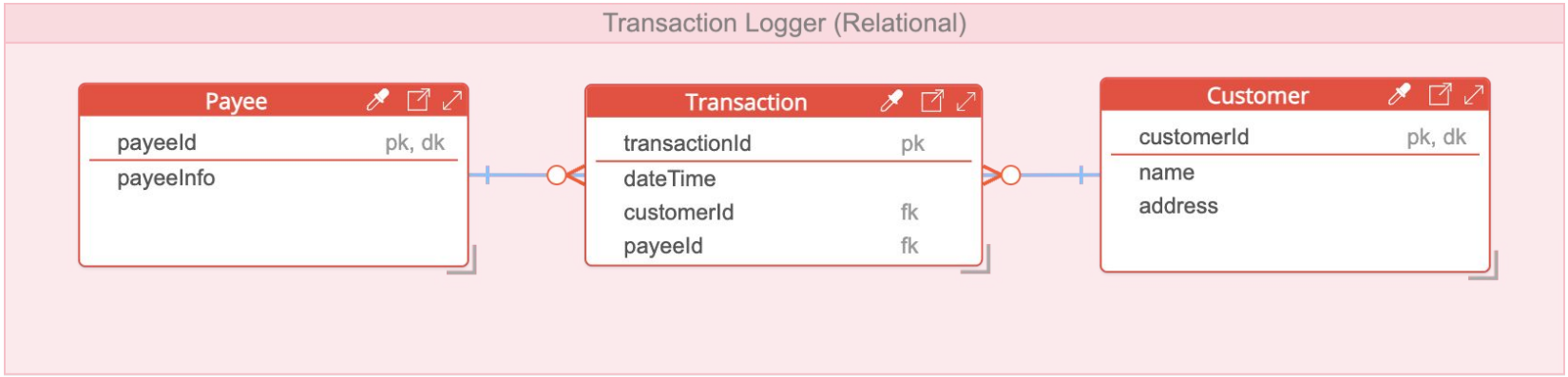
- This regulation requires every national bank to file a Suspicious Activity Report (SAR) when they **detect** certain known or **suspected violations** of federal law or **suspicious transactions** related to a money laundering activity or a violation of the BSA. A SAR filing is required for any potential crimes.

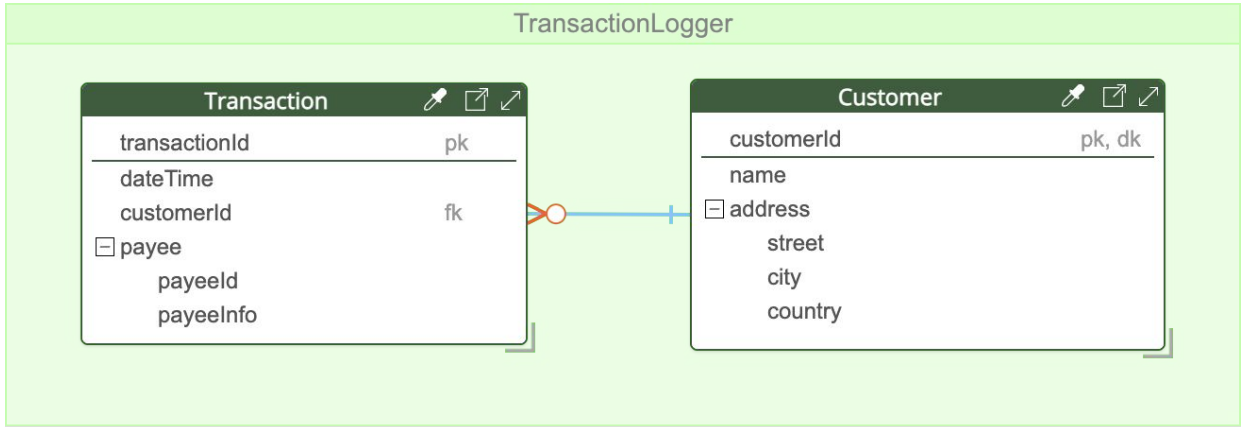
Financial institutions are required to assist U.S. government agencies in detecting and preventing money laundering.

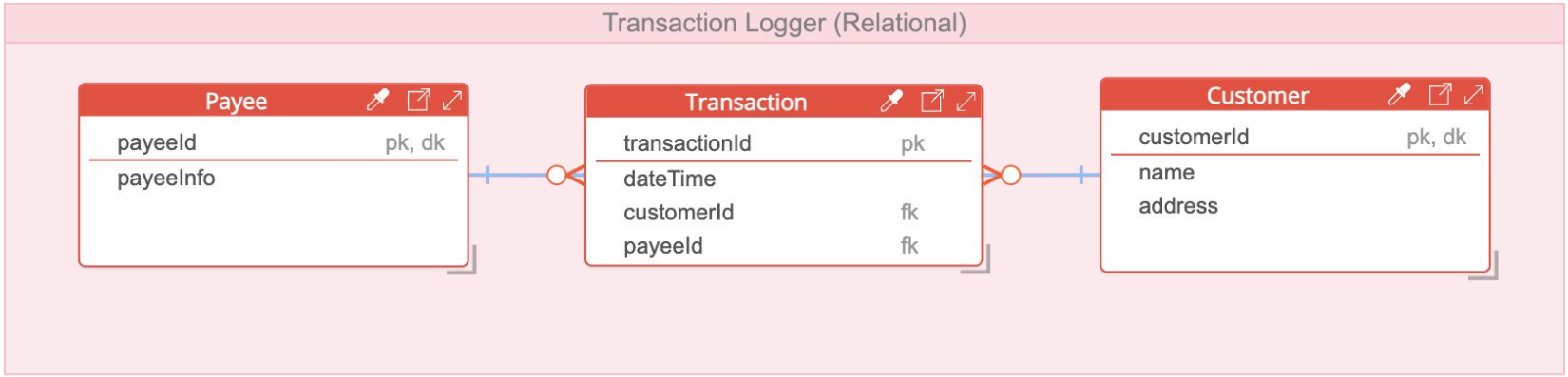

Transaction Logger (Relational)
Payee
payeeId pk, dk
payeeInfo
Transaction
transactionId pk
dateTime
customerId fk
payeeId fk
Customer
customerId pk, dk
name
address

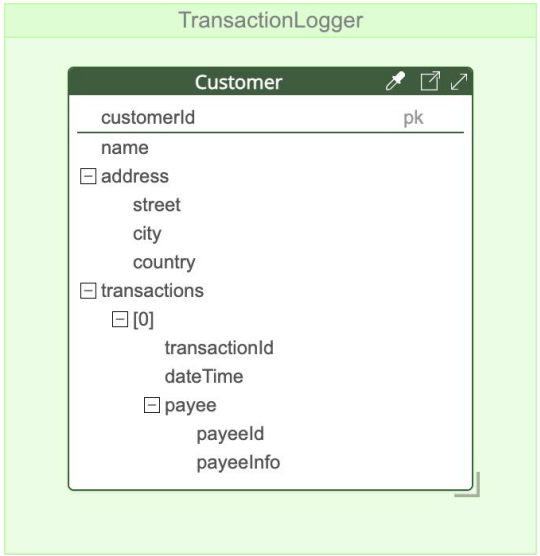

Transaction Logger (Relational)
Payee
payeeId pk, dk
payeeInfo
Transaction
transactionId pk
dateTime
customerId fk
payeeId fk
Customer
customerId pk, dk
name
address


Transaction Logger
Payee
payeeId pk, dk
payeeInfo
Transaction
transactionId pk
dateTime
customerId fk
payeeId fk
Customer
customerId pk, dk
name
address
street
city
country


Transaction Logger (Relational)
Payee
payeeId pk, dk
payeeInfo
Transaction
transactionId pk
dateTime
customerId fk
payeeId fk
Customer
customerId pk, dk
name
address


TransactionLogger
Transaction
transactionId pk
dateTime
customerId fk
payee
payeeId
payeeInfo
Customer
customerId pk, dk
name
address
street
city
country


Transaction Logger (Relational)
Payee
payeeId pk, dk
payeeInfo
Transaction
transactionId pk
dateTime
customerId fk
payeeId fk
Customer
customerId pk, dk
name
address


TransactionLogger
Customer
customerId pk
name
address
street
city
country
transactions
[0]
transactionId
dateTime
payee
payeeId
payeeInfo

Methodology to Model for MongoDB

Identifying the Type of Project

1

Lots of write operations

2

Lots of read operations

3

Reads with low latency

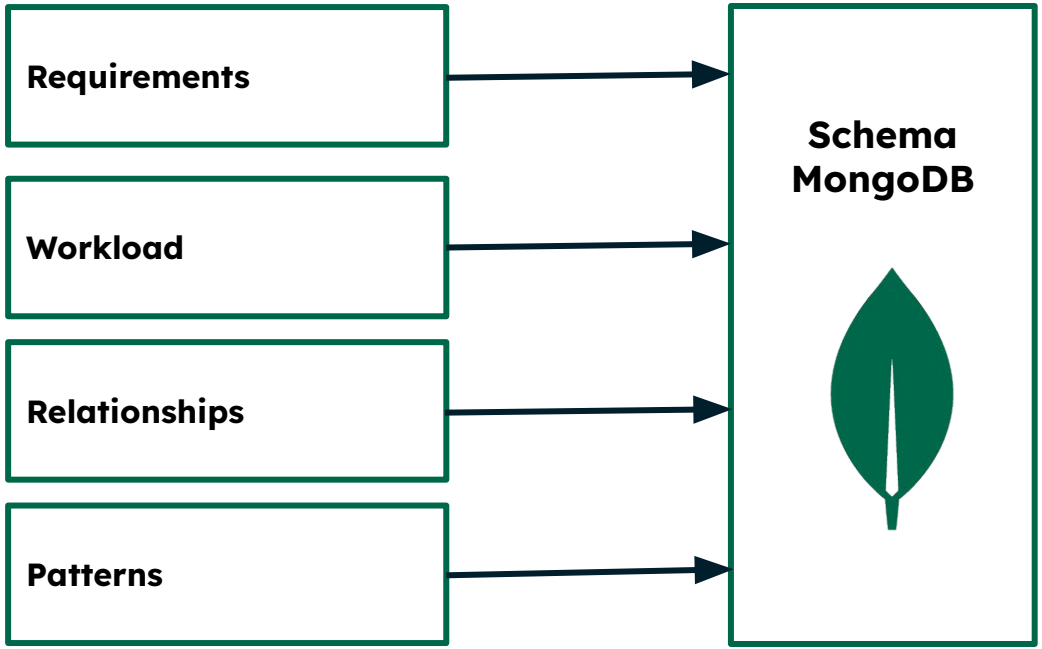
4

Tons of data

5

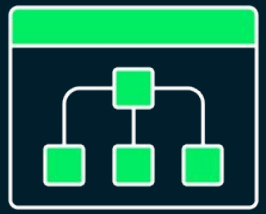
Simplicity

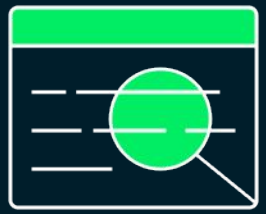
Data Modeling Methodology



Traditional
Relational DB

1. Data Model
2. Workload identification

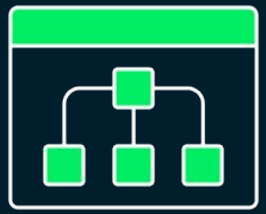


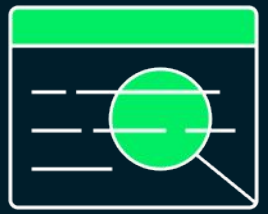


MongoDB

Traditional Relational DB

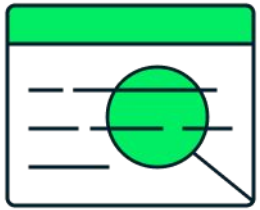
1. Data Model
2. Workload identification

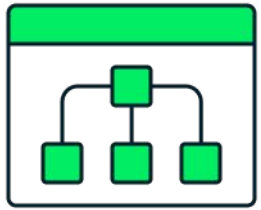




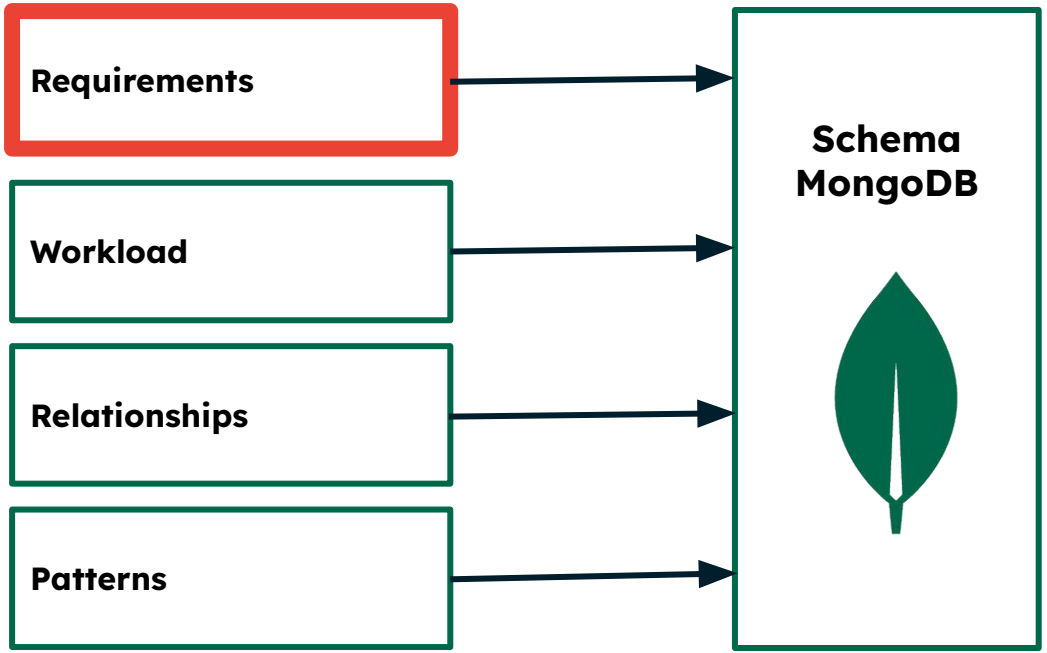
MongoDB

1. Workload identification
2. Data Model





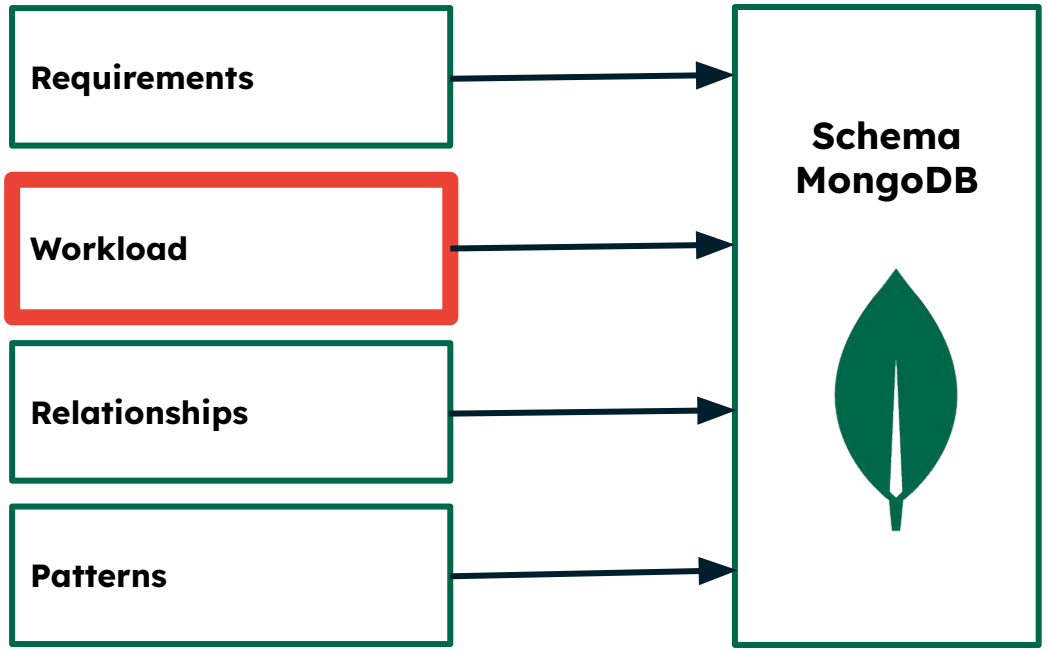
Data Modeling Methodology



Entities

- Transaction
- Customer
- Payee (customer or external party)

Data Modeling Methodology



Example: listing operations

Type	Operation	Information	Frequency	Criticality
write	new transactions	transaction, customer, payee	300 M per day	High
write (update)	transaction updates	customer and payee info	300 M per day (1/transaction)	High
read	processing flagged transactions	complete transaction	5 M per day (5 updates per transaction)	High
read	report generation on flagged	transaction info	10 reports per day, reading all daily transactions	Medium

Example: listing operations

Type	Operation	Information	Frequency	Criticality
write	new transactions	transaction, customer, payee	300 M per day	High
write (update)	transaction updates	customer and payee info	300 M per day (1/transaction)	High
read	processing flagged transactions	complete transaction	5 M per day (5 updates per transaction)	High
read	report generation on flagged	transaction info	10 reports per day, reading all daily transactions	Medium

Details for an Operation

Attribute	Details
Description	new transaction
Type	write
Frequency	300 M per day or **8.3K per second** over **10 hours** (100 M customers * 3 transactions per day)
Latency	**10 ms**
Size	1000 bytes (transaction) 300 GB per day, 27 TB per 90 days
Data lifespan	**90 days fast access** (hot data), 30 years archives (cold data)
Controls	Security/GDPR?

> We can best act on a new requirement if we know the original requirement.

Main Entity

- Transaction
- Customer
- Payee (customer or external party)

Identifying the Type of Project

1

Lots of write operations

2

Lots of read operations

3

Reads with low latency

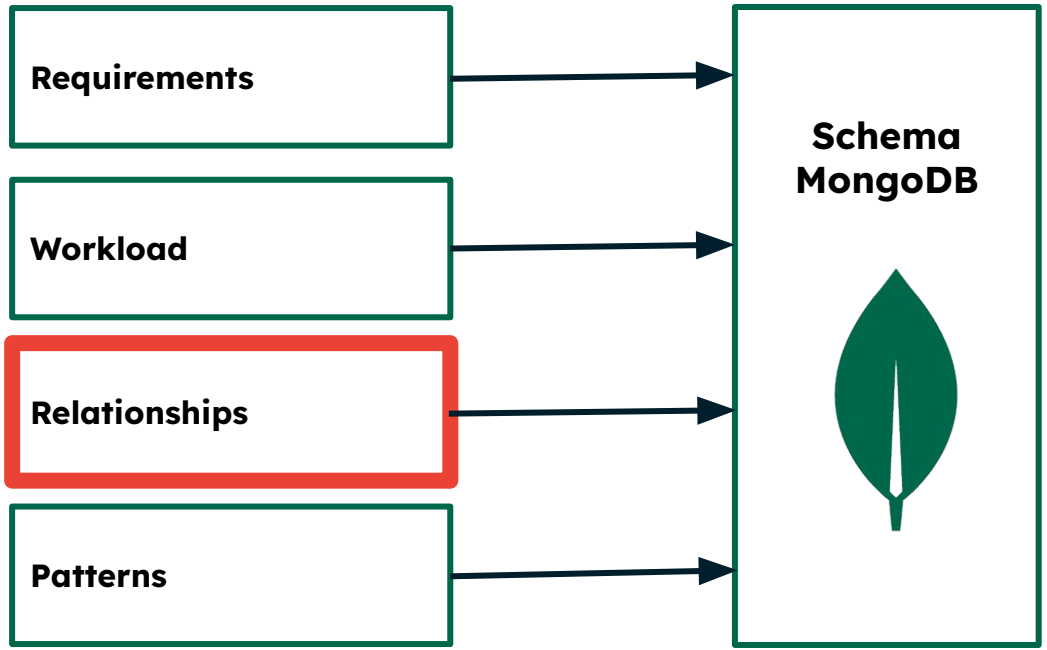
4

Tons of data

5

Simplicity

Data Modeling Methodology



“Choosing between embedding or referencing is one of the most important question to answer.”

Referencing

```
// Person
{
    "_id": "person001",
    "name": "Daniel Coupal",
    "driver_license": "B72398845"
}
```

```
// Driver's license
{
    "_id": "B72398845",
    "location": "California",
    "expiration": "10/2029"
}
```

Embedding

```
// Person
{
    "_id": "person001",
    "name": "Daniel Coupal",
    "driver_license": {
        "number": "B72398845",
        "location": "California",
        "expiration": "10/2029"
    }
}

call(person.driver_license)
```

“Embedding is like joining on write operations instead than at the time we read the data.”

Car in a Tabular/Relational Database (RDBMS)

Car in a Tabular/Relational Database (RDBMS)

Car in MongoDB

> What is used together in the application is stored together in the database.

Transaction - Payee Relationship

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	

Transaction - Payee Relationship

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	

Transaction - Payee Relationship

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	
Query Atomicity	Does the application query the pieces of information together?	Yes	

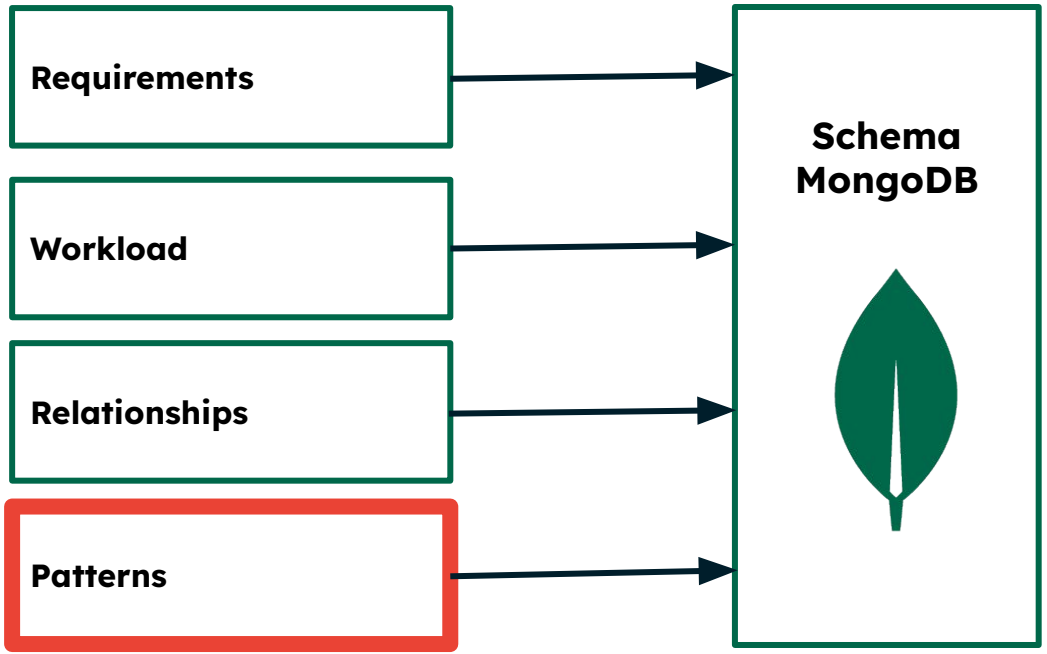
Transaction - Payee Relationship

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	
Query Atomicity	Does the application query the pieces of information together?	Yes	
Update Complexity	Are the pieces of information updated together?	Yes	
Archival	Should the pieces of information be archived at the same time?	Yes	
Cardinality	Is there a high cardinality (current or growing) in a "many" side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?		Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?		Yes

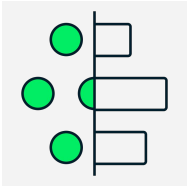
Transaction - Customer Relationship

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	
Query Atomicity	Does the application query the pieces of information together?	Yes	
Update Complexity	Are the pieces of information updated together?	Yes	
Archival	Should the pieces of information be archived at the same time?	Yes	
Cardinality	Is there a high cardinality (current or growing) in a "many" side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?		Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?		Yes

Data Modeling Methodology



Schema Design Patterns Families



Computation

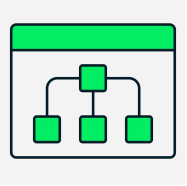
- Approximation
- Computed

Grouping

- Bucket
- Outlier
- Preallocated

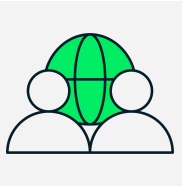
Life Cycle

- Archive
- Document Versioning
- Envelope
- Schema Versioning



Polymorphism

- Inheritance
- Single Collection

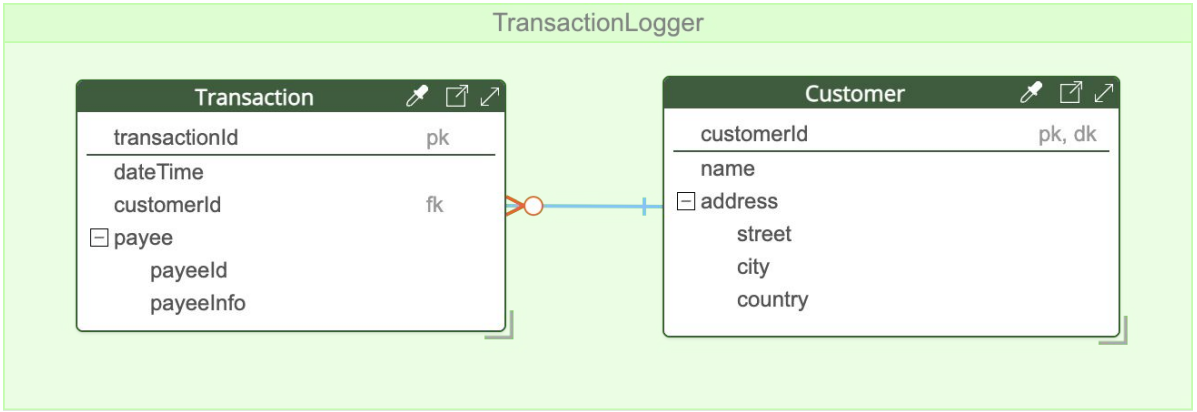


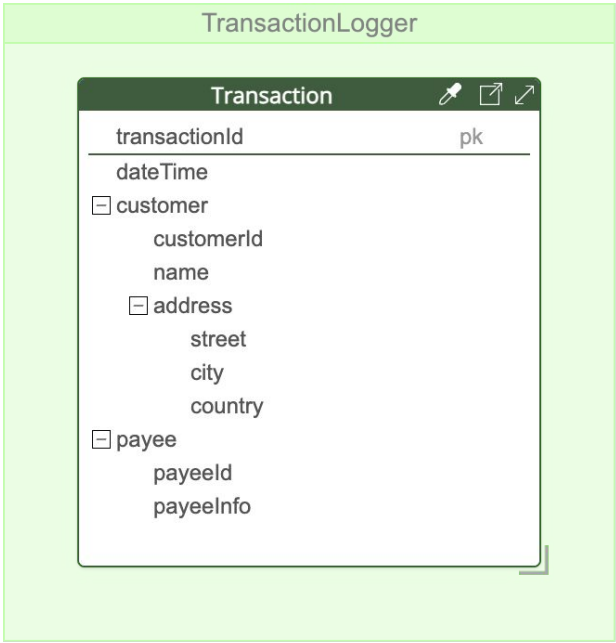
Relationships

- Attribute
- Extended Reference
- Graph
- Subset
- Tree

Only use Schema Design Patterns if needed.

Schemas don't always have to be complex.

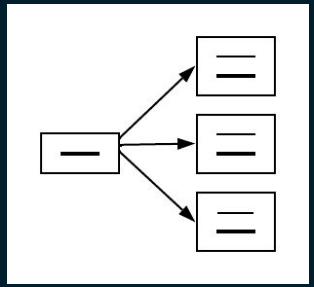

TransactionLogger
Transaction
transactionId pk
dateTime
customerId fk
payee
payeeId
payeeInfo
Customer
customerId pk, dk
name
address
street
city
country


TransactionLogger
Transaction
transactionId pk
dateTime
customer
customerId
name
address
street
city
country
payee
payeeId
payeeInfo

Additional Requirement 1: Bad Actors

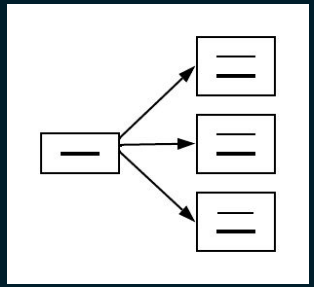
A. Track more information about bad actors.
B. Use information provided by the FBI and CIA databases.

Extended Reference Pattern



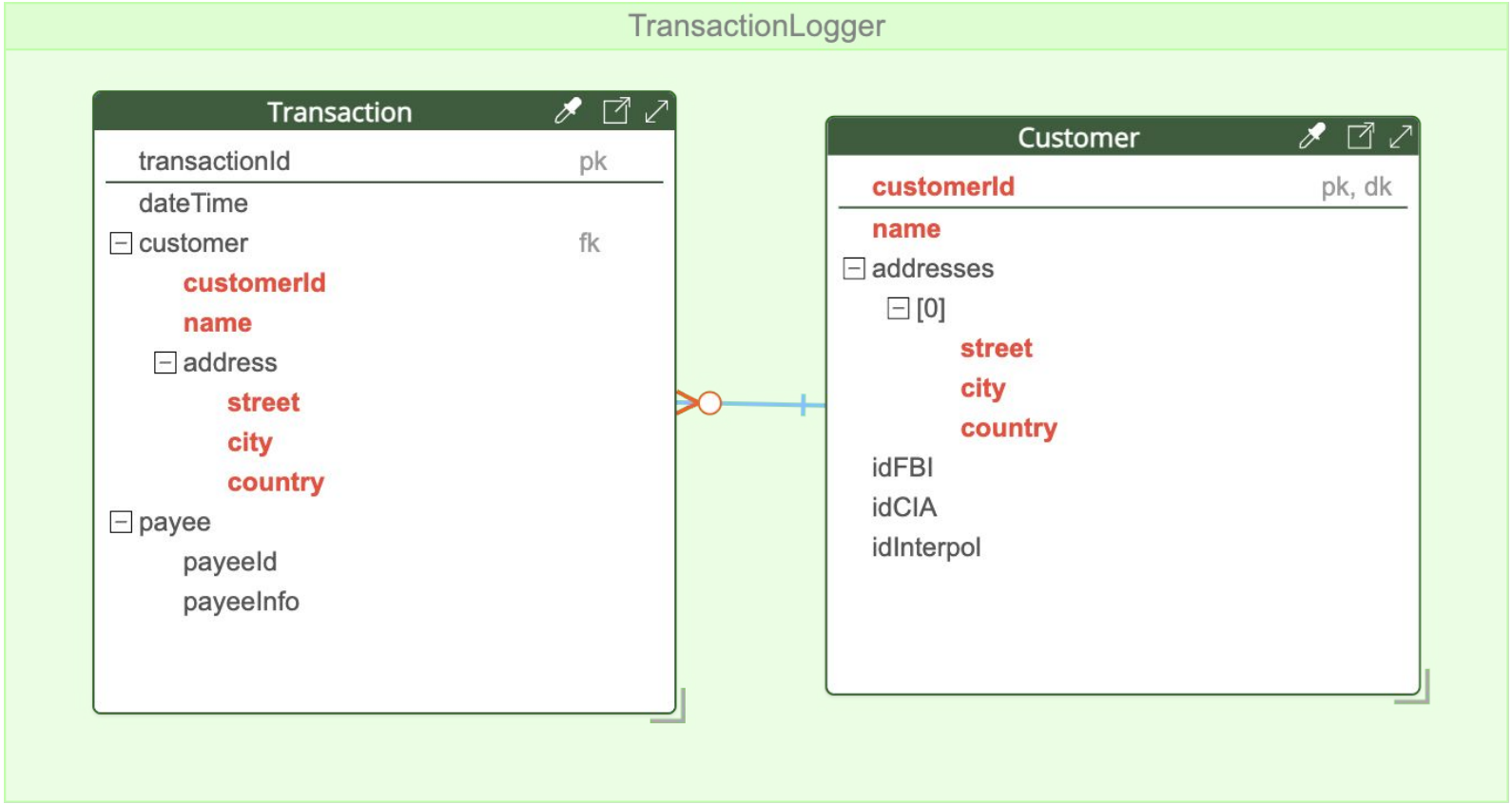
Problem	· Too many joins in read operations. · Embedding leads to documents that are too big or too much duplication.
Solution	· Identify fields using joins for the most common read operations. · Copy these fields as an embedded subdocument in the main document.
Use cases	· Catalog. · Mobile applications. · Real-time analytics.
Benefits	· Faster reads. · Lesser number of joins and lookups.
Trade-offs	· Potentially creates data duplication.

Extended Reference Pattern



Problem	· Too many joins in read operations. · Embedding leads to documents that are too big or too much duplication.
Solution	· Identify fields using joins for the most common read operations. · Copy these fields as an embedded subdocument in the main document.
Use cases	· Catalog. · Mobile applications. · Real-time analytics.
Benefits	· Faster reads. · Lesser number of joins and lookups.
Trade-offs	· Potentially creates data duplication.

Applying the Extended Reference Pattern

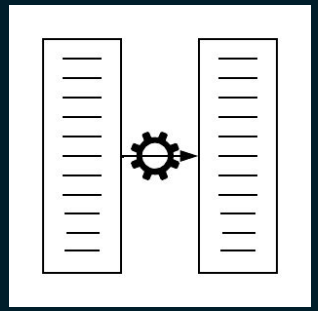


Additional Requirement 2: Analytics

Image generated using ChatGPT/Dall-E

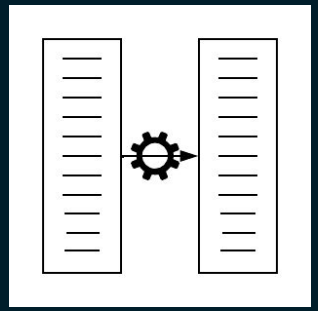
A. Compute different metrics on suspected transactions and bad actors.
B. Compute trends over time.

Computed Pattern



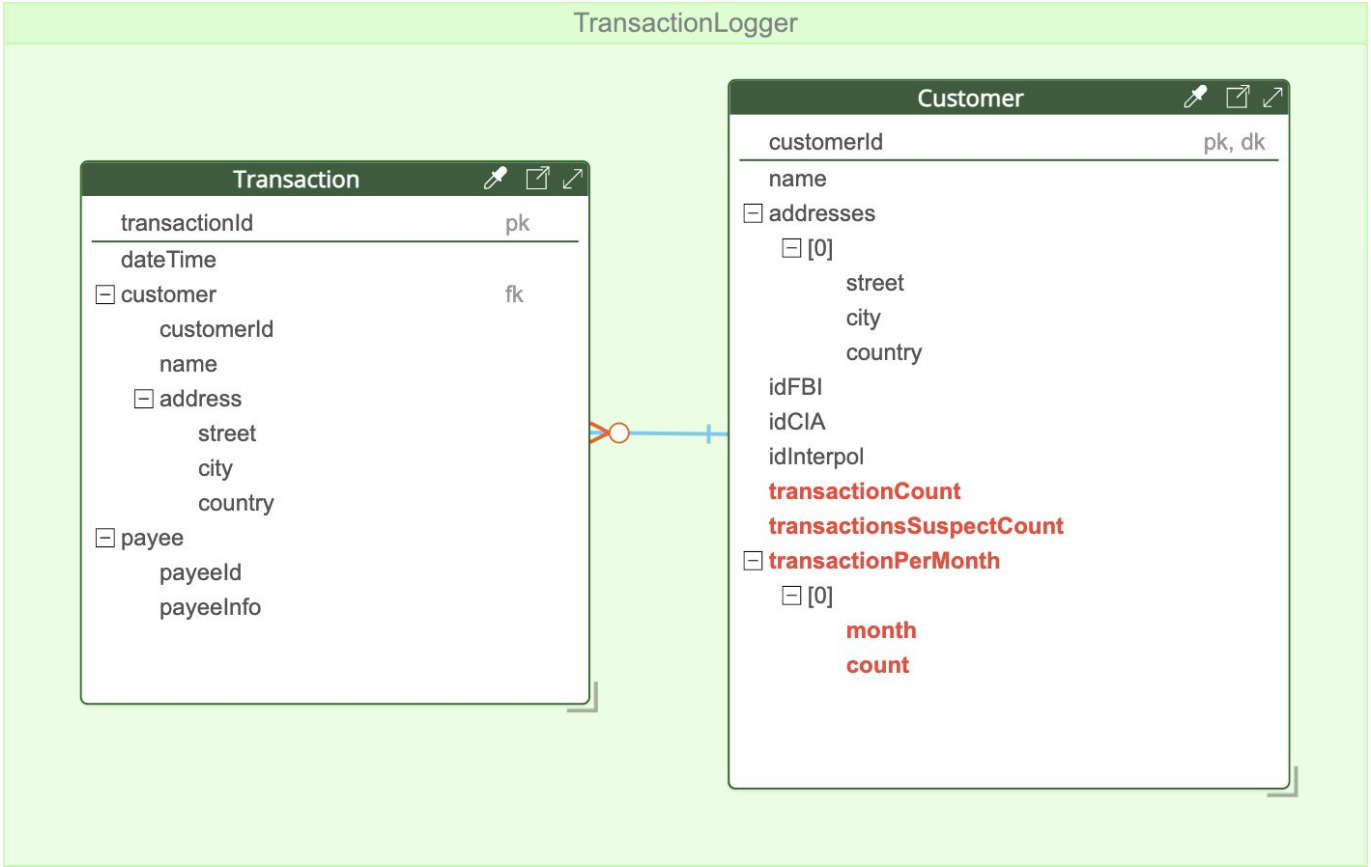
Problem	· Expensive computation or manipulation of data. High read-to-write ratio. · Computation executed frequently on the same data produces the same results.
Solution	· Execute the operation and store the result in the appropriate document either at write time or through a scheduler.
Use cases	· Internet of Things. · Event sourcing.
Benefits	· Read operations are faster. · Saving CPU and disk access resources.
Trade-offs	· Potentially creates staleness. · Potentially creates duplication of information.

Computed Pattern



Problem	· Expensive computation or manipulation of data. High read-to-write ratio. · Computation executed frequently on the same data produces the same results.
Solution	· Execute the operation and store the result in the appropriate document either at write time or through a scheduler.
Use cases	· Internet of Things. · Event sourcing.
Benefits	· Read operations are faster. · Saving CPU and disk access resources.
Trade-offs	· Potentially creates staleness. · Potentially creates duplication of information.

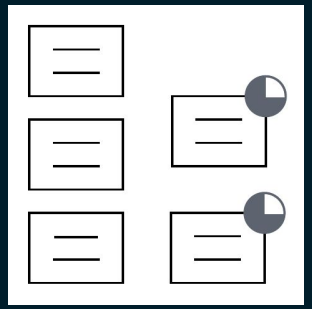
Applying the Computed Pattern



Additional Requirement 3: Long Data Life

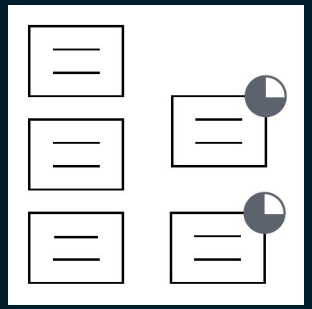
A. Must keep flagged Transactions for 30 years.
B. Must keep Transactions of a bad actor for 30 years.

Archive Pattern



Problem	· Some documents are used sparingly but must be kept for regulation purposes.
Solution	· Use different storage tiers. · Store older documents in a cheaper tier.
Use cases	· Financial applications. · Pharmaceutical applications.
Benefits	· Reduces the costs of managing older documents. · Satisfies the regulatory requirements to keep data for an extended time.
Trade-offs	· Slower access to older documents. · The database management system may not support federated queries to all tiers together. · Changing the schema of archived documents may not be possible.

Archive Pattern

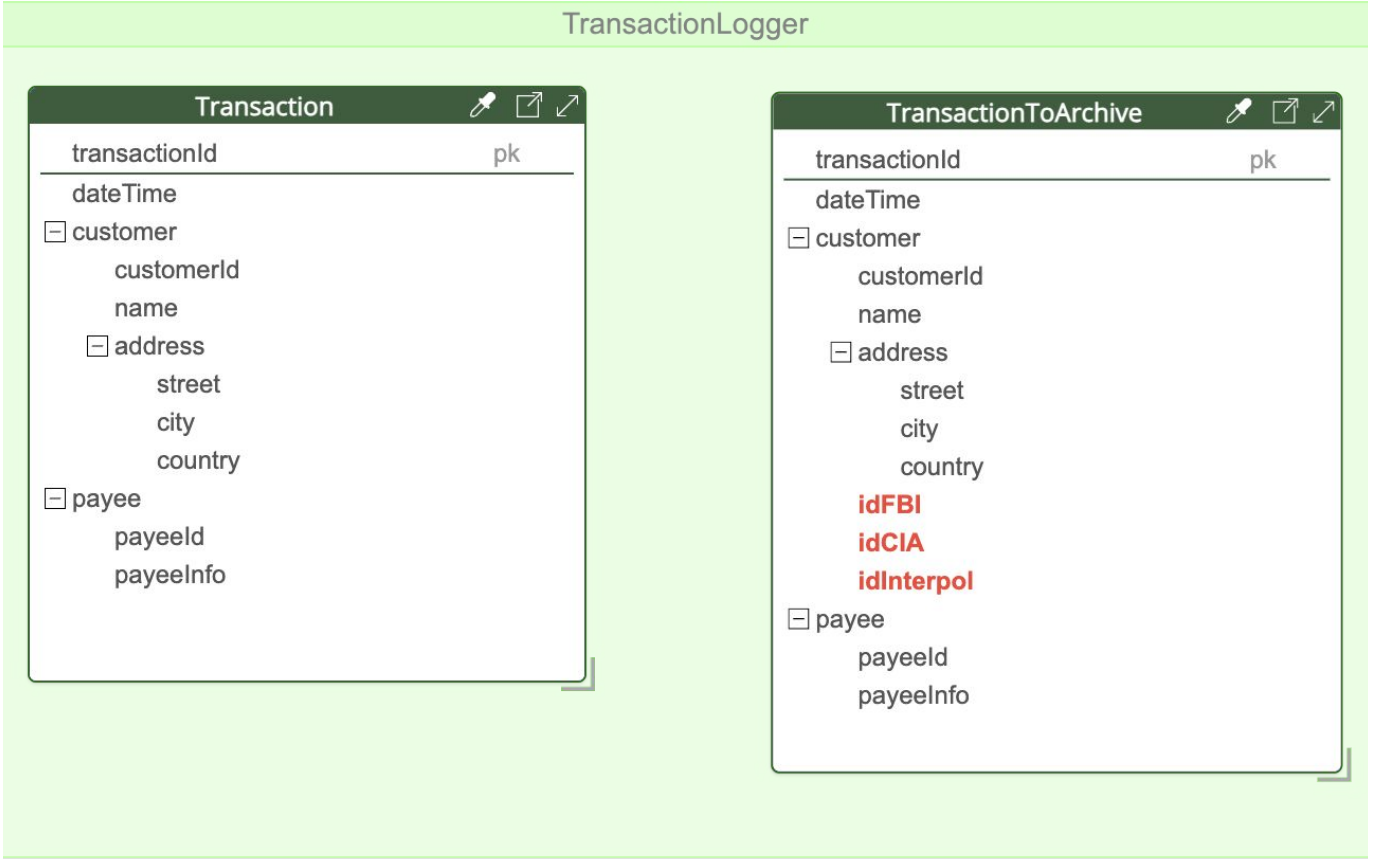


Problem	· Some documents are used sparingly but must be kept for regulatory purposes.
Solution	· Use different storage tiers. · Store older documents in a cheaper tier.
Use cases	· Financial applications. · Pharmaceutical applications.
Benefits	· Reduces the costs of managing older documents. · Satisfies the regulatory requirements to keep data for an extended time.
Trade-offs	· Slower access to older documents. · The database management system may not support federated queries to all tiers together. · Changing the schema of archived documents may not be possible.

Applying the Archive Pattern

Goal: standalone document

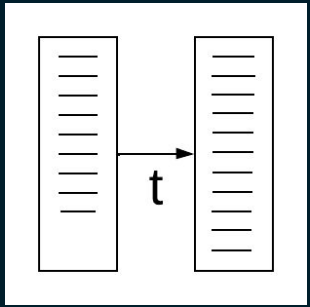
A. Archive as it is
B. Enrich before archiving



Additional Requirement 4: New Rules

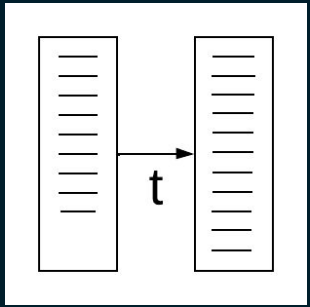
A. Every 2 years, new regulations may be issued.
B. These regulations may require tracking additional information.

Schema Versioning Pattern

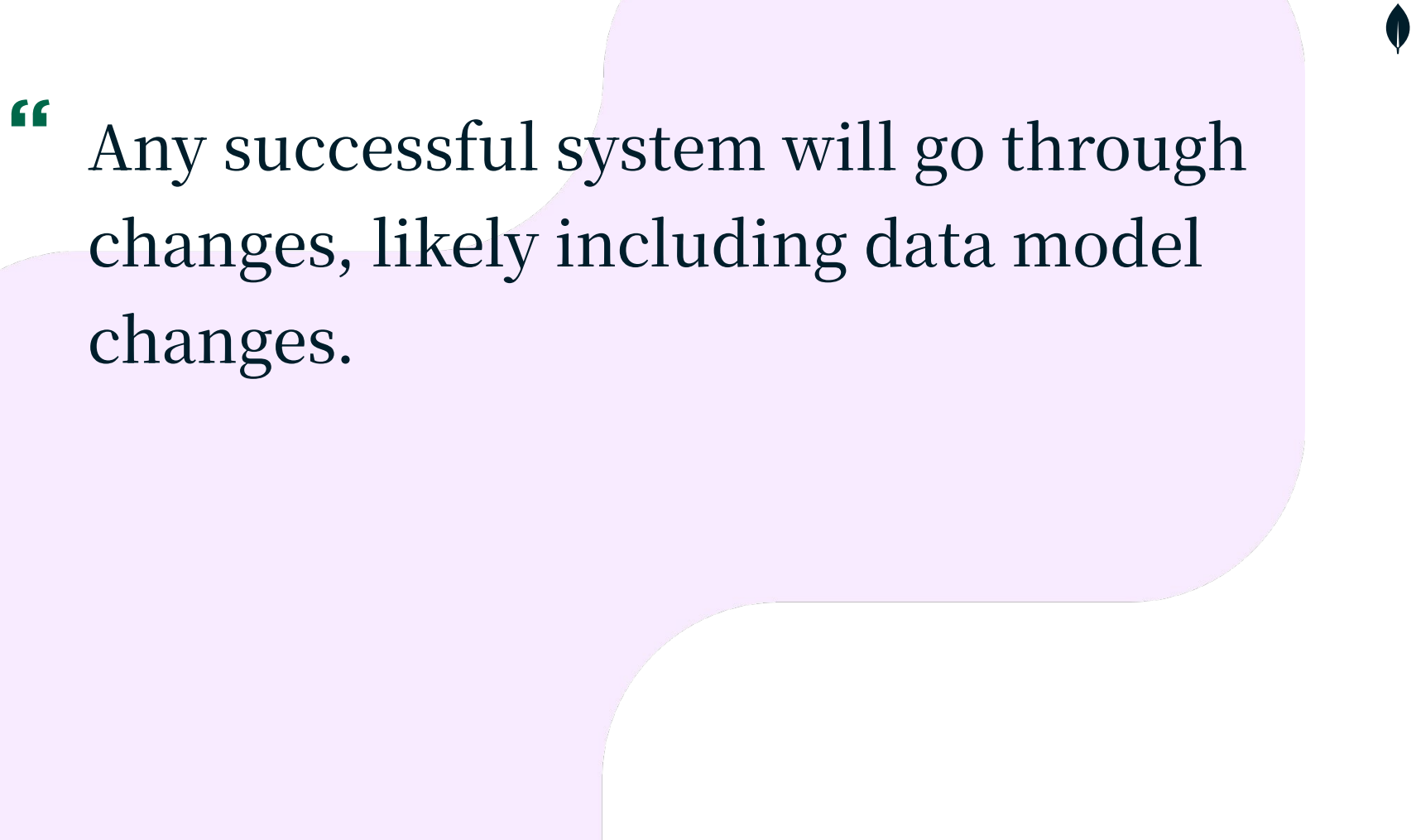


Problem	· Doing a schema migration without downtime.
Solution	· Add a schema version number to each document. · Modify the application code to handle each schema variant. · Progressively update each document.
Use cases	· Any application that can’t sustain any downtimes.
Benefits	· Allow for schema migration without downtime.
Trade-offs	· Adds temporary complexity to the code to handle different schema variants.

Schema Versioning Pattern



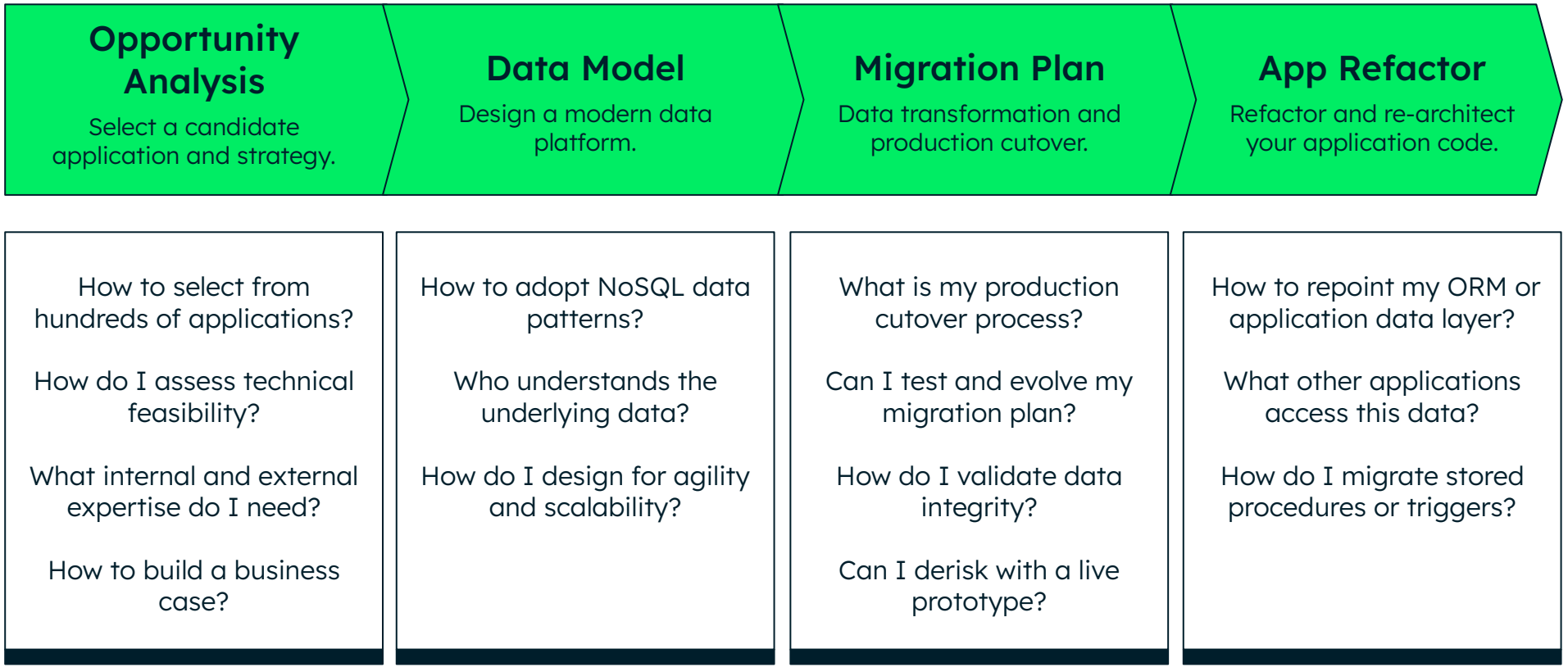
Problem	· Doing a schema migration without downtime.
Solution	· Add a schema version number to each document. · Modify the application code to handle each schema variant. · Progressively update each document.
Use cases	· Any application that can't sustain any downtimes.
Benefits	· Allow for schema migration without downtime.
Trade-offs	· Adds temporary complexity to the code to handle different schema variants.


“
Any successful system will go through changes, likely including data model changes.

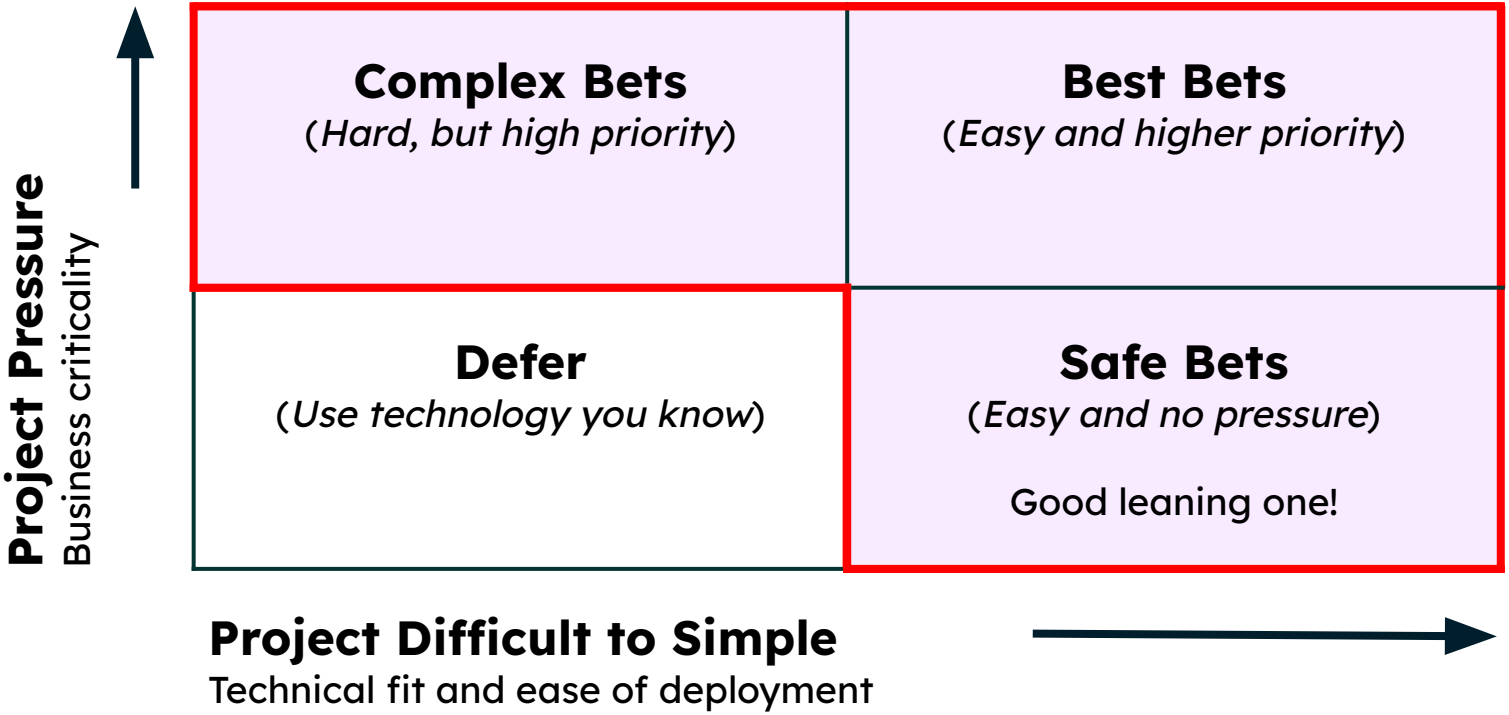
Executing the Plan: What to do next?

MongoDB New Projects and Migrations

Key App Migration Challenges

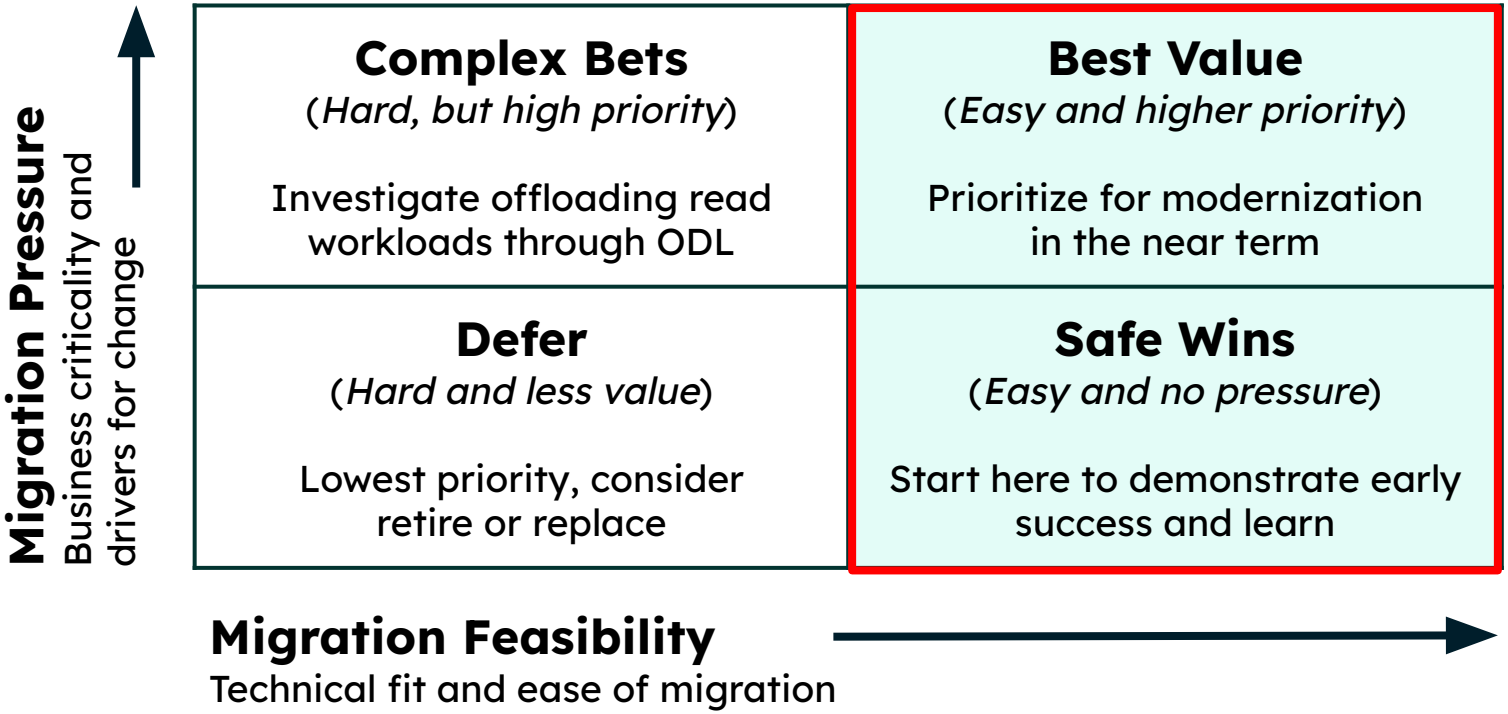


Candidates for First New Projects



Candidates for Modernizations

Assessing your Application Portfolio for Modernization



Relational Migrator can:

- Import and analyze your relational database schemas
    - From: Oracle, SQL Server, MySQL, PostgreSQL, SyBase
- Help you map your data to an appropriate MongoDB schema
- Transform and migrate your data and queries (!) into MongoDB

Relational Migrator can:

- Import and analyze your relational database schemas
  - From: Oracle, SQL Server, MySQL, PostgreSQL, SyBase
- Help you map your data to an appropriate MongoDB schema
- Transform and migrate your data and queries (!) into MongoDB

Relational Migrator is not:

- A silver bullet that will immediately modernize your application portfolio
- A substitute for experience and good advice on using MongoDB

MongoDB Design Reviews

MongoDB Design Reviews

A 1:1 session with a MongoDB data modeler for developer teams to explore how their own workloads might be supported by a NoSQL backend, and learn about how best to structure your data with the document model.

What is a Design Review?

- Get **NoSQL Data Modeling expert advice to prevent common mistakes** that can significantly impact cost and performance
- We partner with you to **enable your developer teams fine-tune your data models for specific projects of use cases**
- We will **review your logical model, relational schema or draft document model** and **suggest the most efficient MongoDB schema design**, exploring optimal design patterns and best practices

How to Book

Book your one-hour session through one of our MongoDB team members in-person

What to expect in a Design Review

Workload assessment
Determine your workload's data-related functional requirements.

Understanding the relationships
Identify and quantify the relationships of your data for your workload

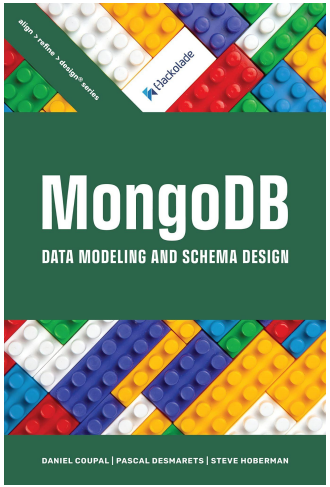
Recommend schema design patterns
Leave with a draft document model, and enough knowledge to continue building your application optimally

Learn MongoDB

Learn more

Book: MongoDB Data Modeling and Schema Design

Blogs: Patterns



Learn more

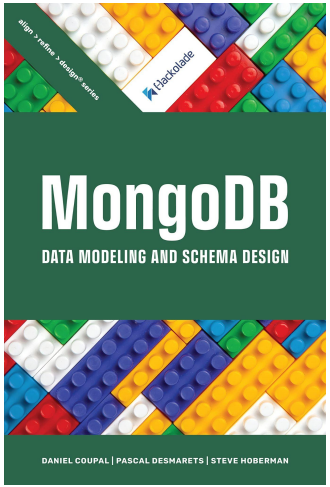
Book: MongoDB Data Modeling and Schema Design

Blogs: Patterns

MongoDB Days

MongoDB Dev Days

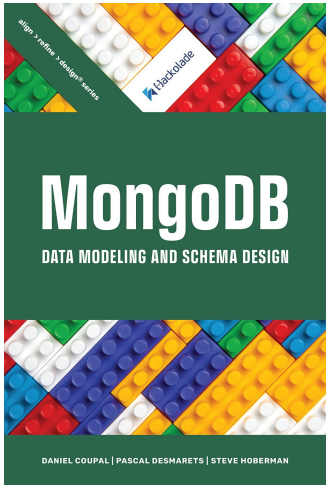
1:1 Design Reviews



Learn more

Book: MongoDB Data Modeling and Schema Design

Blogs: Patterns



MongoDB Days

MongoDB Dev Days

1:1 Design Reviews

MongoDB University

- Course:
  - MongoDB for SQL Professionals
- Path:
  - MongoDB Data Modeling
- Certification:
  - Data Modeling Certification

Conclusion

Conclusions

- Why do enterprise choose NoSQL?
    - Cost, Agility of Development, Performance.

Conclusions

- Why do enterprise choose NoSQL?
    - Cost, Agility of Development, Performance.
- How designing for NoSQL differs from RDBMS?
    - Similar; however think in objects and understand your workload at the beginning.

Conclusions

- Why do enterprise choose NoSQL?
    - Cost, Agility of Development, Performance.
- How designing for NoSQL differs from RDBMS?
    - Similar; however think in objects and understand your workload at the beginning.
- What to do next?
    - Get going on a new project or an easy project to migrate and learn!

Thank you for your time.