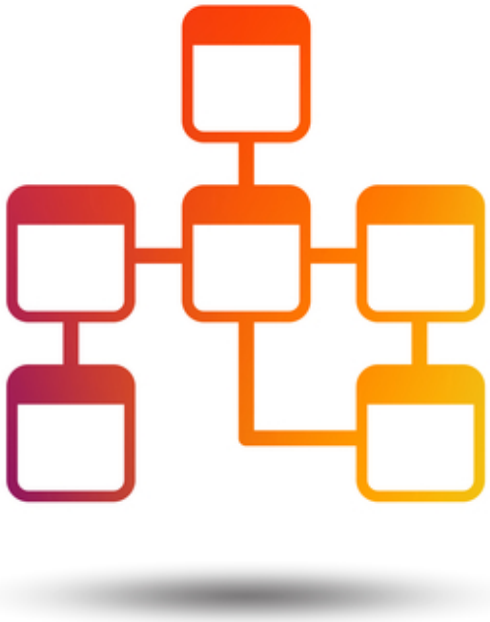
Full-Stack Web Development



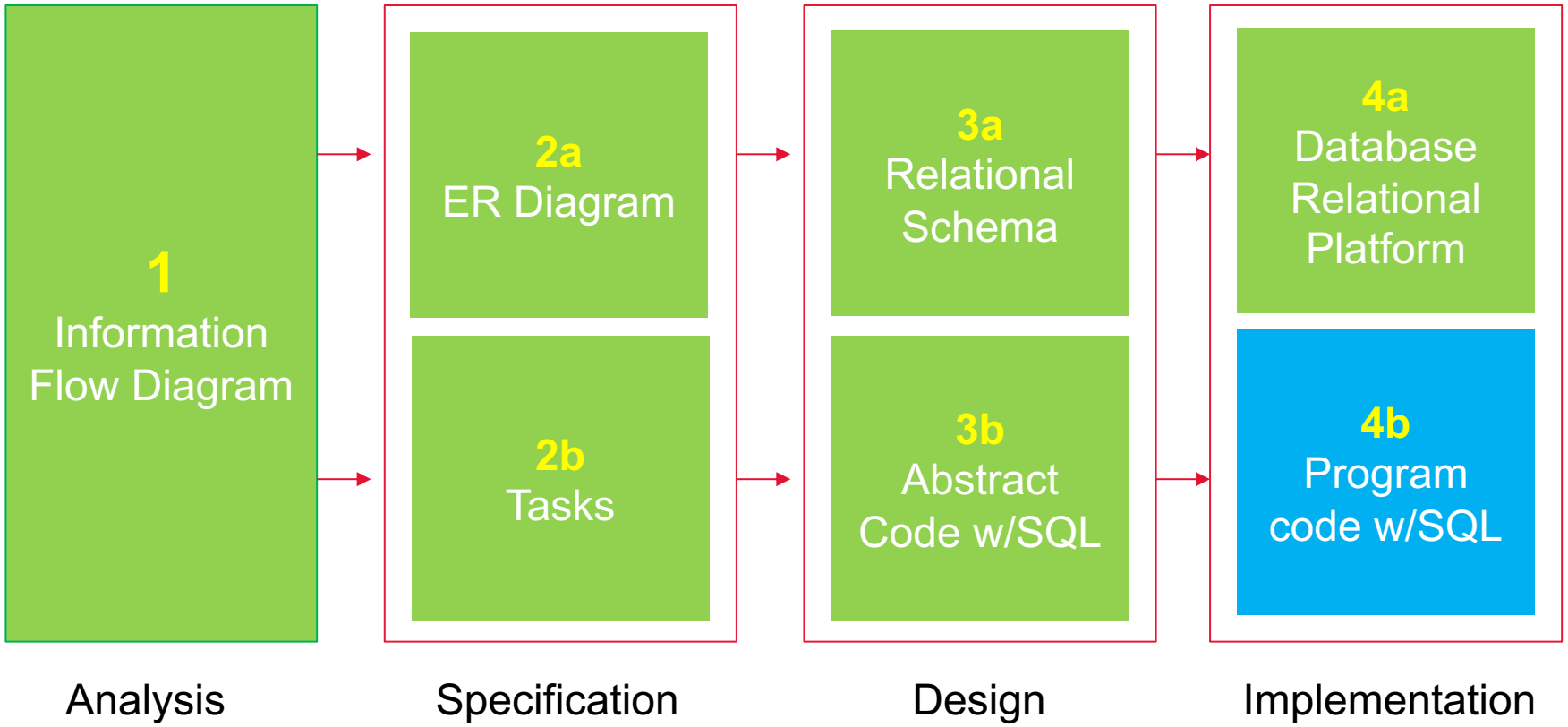
FS1030 – **Database Design and Principles**

school of continuing studies | YORK UNIVERSITÉ UNIVERSITY

What we learned so far

- Fundamentals of database
- Use cases around databases
- Why and when do we need various database types
- Data modeling and architecture
- Methodology
- Information flow diagrams
- Entity Relationship Diagrams
- Task decomposition
- Abstract Code
- Normalization Techniques
- Relational Algebra
- SQL Queries
- PHP/MYSQL
- NodeJS – Express – MySQL

What we are going to learn



Install MongoDB

MacOS

Install Homebrew if not installed

> /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

Install Mongodb

> brew tap mongodb/brew
> brew install mongodb-community@4.0

Open Mongodb

> brew services start mongodb

Install MongoDB

Windows

Download community edition

https://www.mongodb.com/download-center/community?jmp=docs

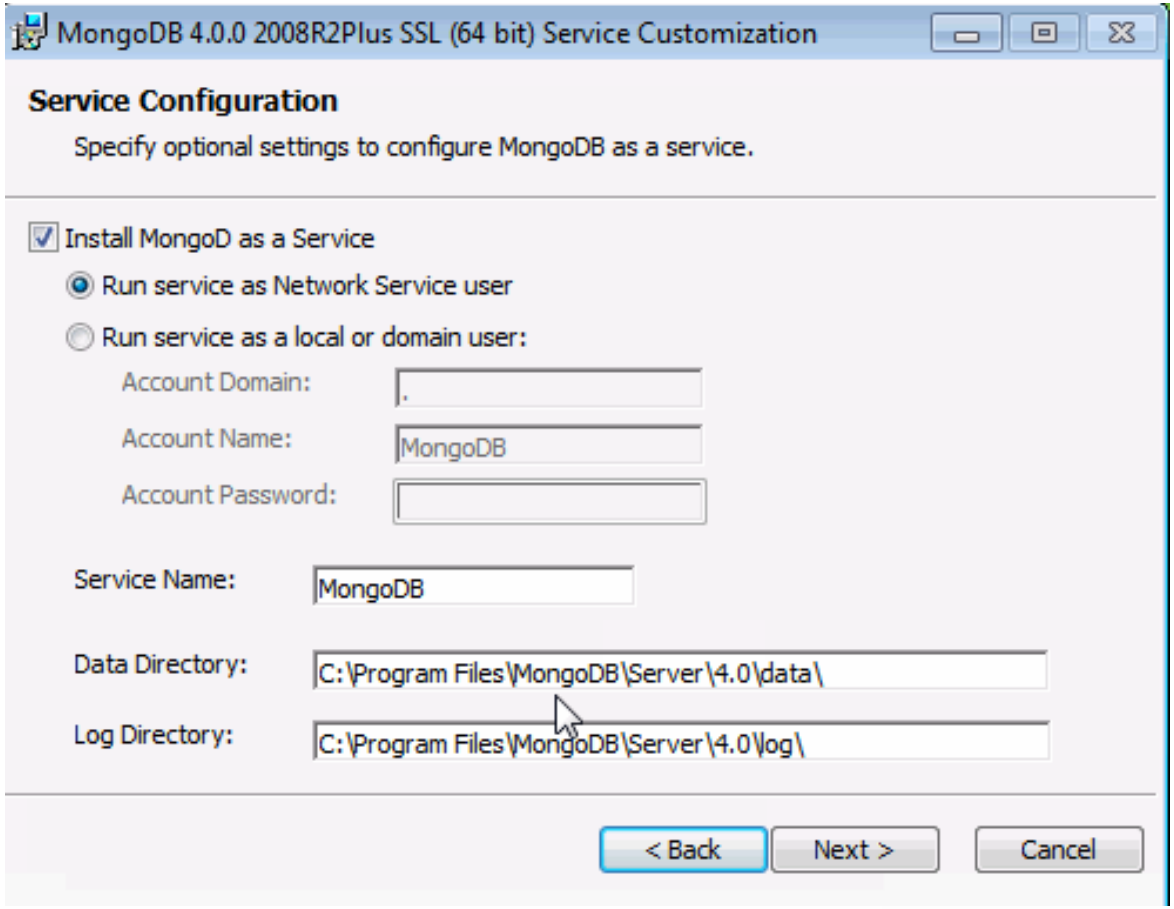
- The Download Center should display **MongoDB Community Server** download information. If not, select **Server**, then click the **MongoDB Community Server** tab.
- In the **Version** dropdown, select the version that corresponds to the latest MongoDB Server 4.0.
- In the **OS** dropdown, **Windows 64-bit X64** should be selected.
- In the **Package** drop down, **MSI** should be selected.
- Click **Download**.

Run the installer

- Go to the directory where you downloaded the MongoDB installer (.msi file). By default, this is your Downloads directory.
- Double-click the .msi file.

Install MongoDB

Windows



Source: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

Install MongoDB Compass

https://www.mongodb.com/products/compass

Source: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

MongoDB Basics

Database

Database is a container for collections.

Collections

Collection is a group of documents. Its is similar to table in RDBMS.

Document

Document is a set of key value pair

MongoDB Basics

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

MongoDB Basics

- NoSQL is schema less which means it is not structured
- It can contain documents of various different structure i.e. number of fields, content and size of document may differ with in the same collection.
- Easy to scale because of flat structure
- High availability and sharding support
- JSON style document storage

MongoDB Queries - Insert

db.collection.insertOne()	Inserts a single document into a collection.
db.collection.insertMany()	db.collection.insertMany() inserts *multiple* documents into a collection.
db.collection.insert()	db.collection.insert() inserts a single document or multiple documents into a collection.

```
> db.products.insertOne( { item: "card", qty: 15 } );
```

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Source: https://docs.mongodb.com

MongoDB Queries - Insert

```
> db.products.insertOne( { item: "card", qty: 15 } );

> db.products.insert(
   [
     { _id: 11, item: "pencil", qty: 50, type: "no.2" },
     { item: "pen", qty: 20 },
     { item: "eraser", qty: 25 }
   ]
)


> db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Source: https://docs.mongodb.com

MongoDB Queries – View/Select

RDBMS	MongoDB
Select * from inventory	db.inventory.find({})
SELECT * FROM inventory WHERE status = "D"	db.inventory.find({ status: "D" })
SELECT * FROM inventory WHERE status in ("A", "D")	db.inventory.find({ status: { $in: ["A", "D"] } })
SELECT * FROM inventory WHERE status = "A" AND qty < 30	db.inventory.find({ status: "A", qty: { $lt: 30 } })
SELECT * FROM inventory WHERE status = "A" OR qty < 30	db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })
SELECT * FROM inventory WHERE status = "A" AND (qty < 30 OR item LIKE "p%")	db.inventory.find({ status: "A", $or: [{ qty: { $lt: 30 } }, { item: /^p/ }] })

MongoDB supports regular expressions $regex queries to perform string pattern matches.

Nested search:
> db.inventory.find({ "size.h": { $lt: 15 }, "size.uom": "in", status: "D" })

Source: https://docs.mongodb.com

MongoDB Queries – Update

```
db.inventory.insertMany( [
   { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },
   { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },
   { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }
] );
```

The following example uses the db.collection.updateOne() method on the inventory collection to update the first document where item equals "paper":

```
> db.inventory.updateOne(
   { item: "paper" },
   {
     $set: { "size.uom": "cm", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

Source: https://docs.mongodb.com

MongoDB Queries – Update

The following example uses the db.collection.updateMany() method on the inventory collection to update all documents where qty is less than 50:

```
> db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

To replace the entire content of a document except for the _id field, pass an entirely new document as the second argument to db.collection.replaceOne().

```
> db.inventory.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty:
40 } ] }
)
```

Source: https://docs.mongodb.com

MongoDB Queries – Delete

```
> db.inventory.insertMany( [
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },
] );
```

The following example deletes *all* documents from the inventory collection:

```
> db.inventory.deleteMany({})
```

The following example removes all documents from the inventory collection where the status field equals "A":

```
> db.inventory.deleteMany({ status : "A" })
```

The following example deletes the first document where status is "D":

```
> db.inventory.deleteOne( { status: "D" } )
```

Source: https://docs.mongodb.com

NodeJS + MongoDB + REST API

Clone the repo

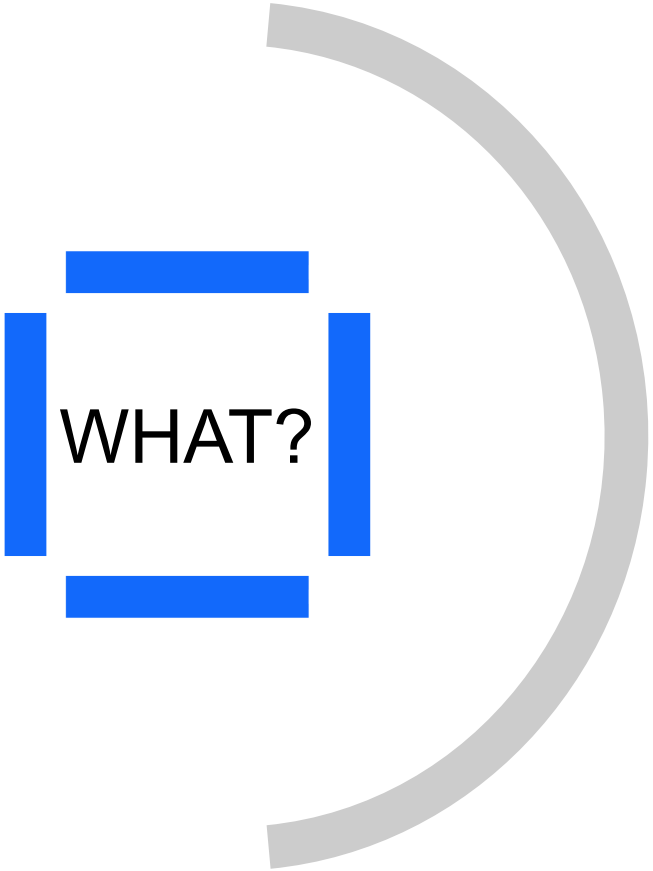
https://github.com/tarun27in/FS1030_NoSQL

Exercises

- Add one more value in the collection called “note_category” and “note_list_name”
- Create a route to add new records with note_category value as “Work Notes” and note_list_name as “Work List”
- Create a route to query all the records with note_list_name as “Work List”

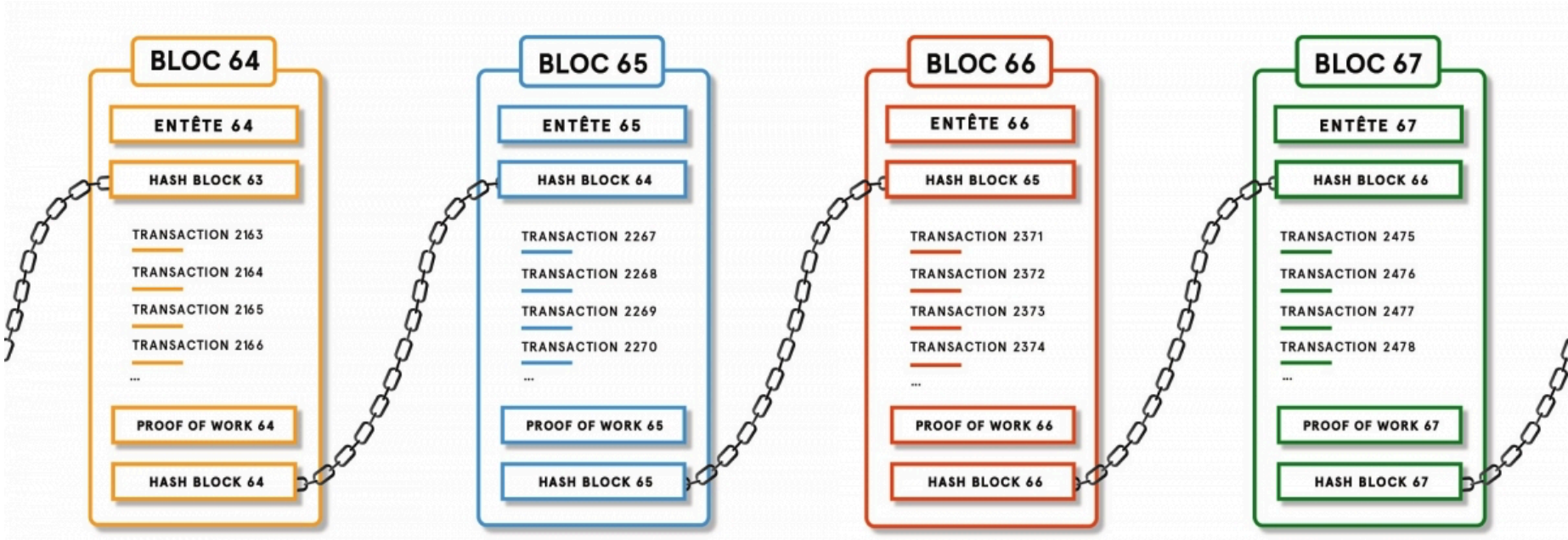
Next Section: Blockchain

What is Blockchain?



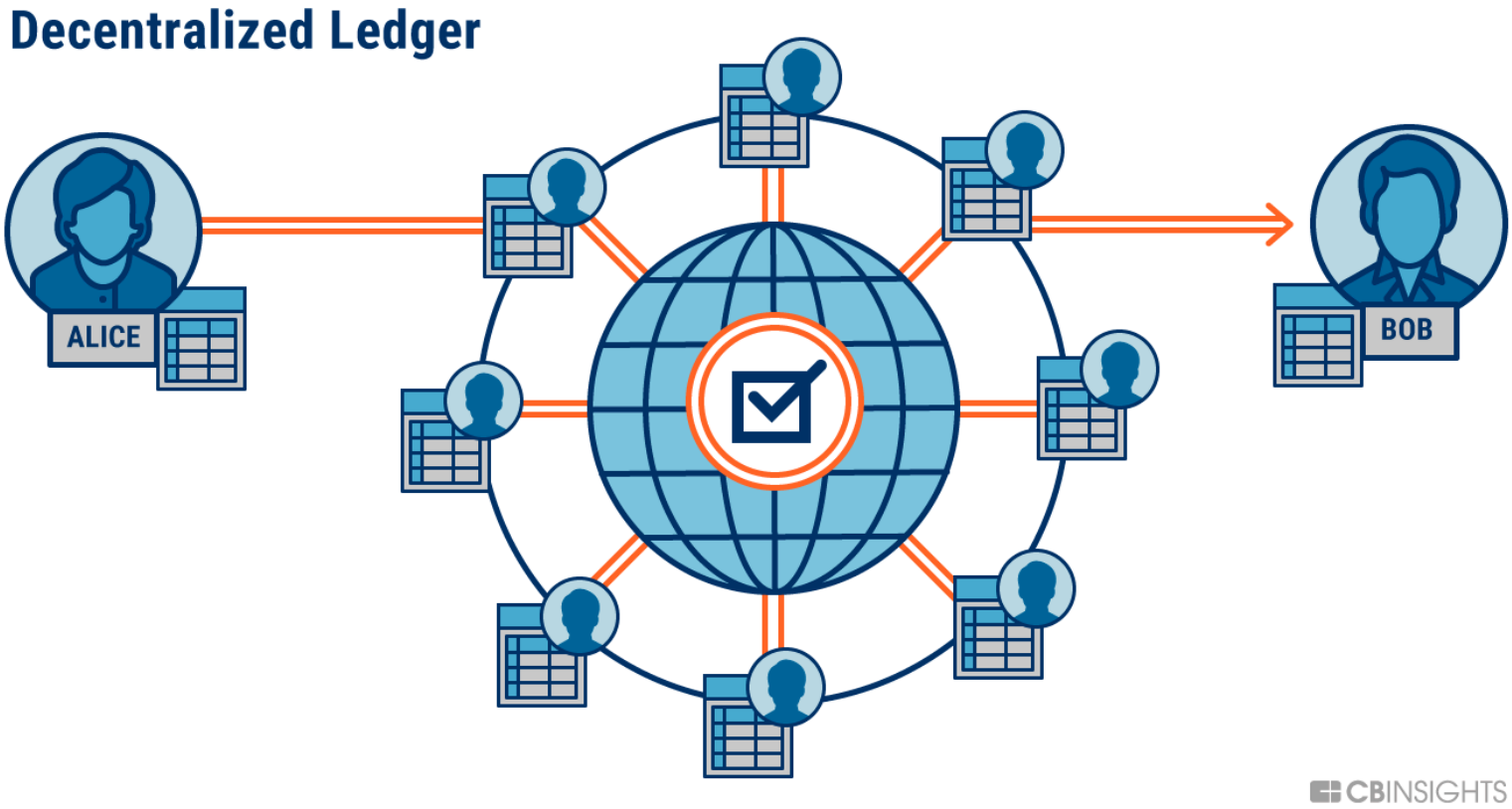
"The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value."

What is Blockchain?



Source: https://blog.theodo.com/2018/01/deploy-first-ethereum-smart-contract-blockchain/

Blockchain 101

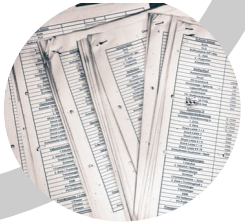


Blockchain 101

WHAT?

Business Networks
Group of entities that transact goods or services.

Assets
Anything (can be either digital or physical) capable of being owned or controlled to produce value.



Ledgers
System of records where transactions and contracts are digitally recorded.

Blockchain 101

WHY?

Business Networks are:

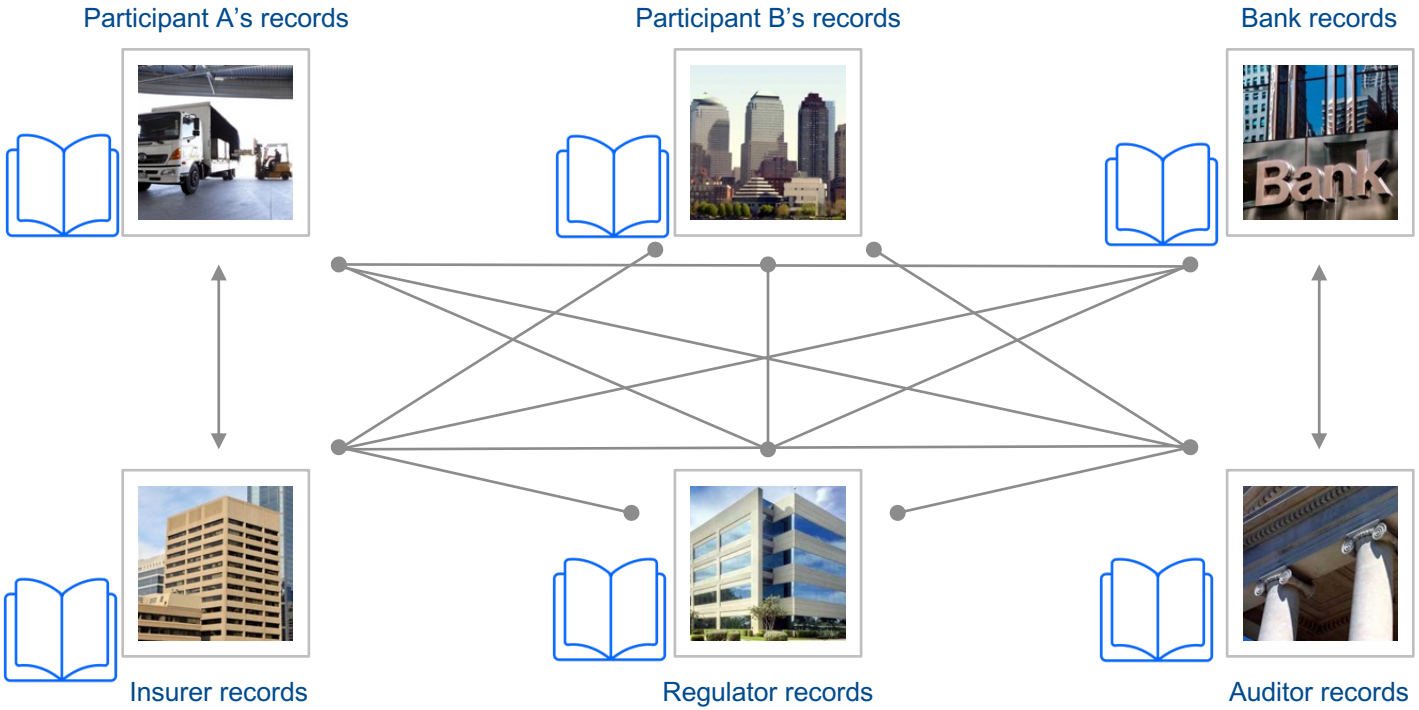
Expensive
Duplication of effort and intermediaries adding margin for services

Inefficient
Business conditions – the contract – is duplicated by every network participant

Vulnerable
a central system (e.g. Bank) is compromised due to an incidents this affects the whole business network. Incidents can include fraud, cyber attack or a simple mistake.

Blockchain 101

Problem with databases
inefficient, expensive, vulnerable

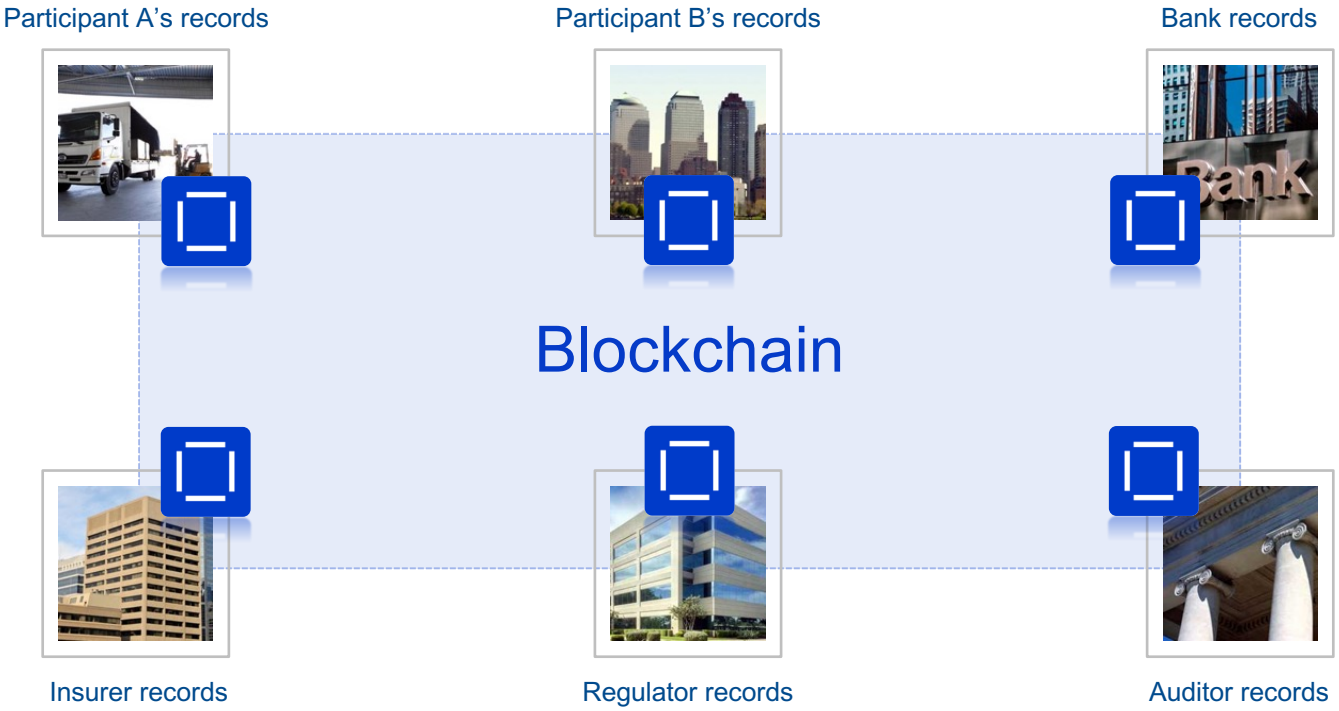


Blockchain 101

Solution

A shared, replicated, permissioned ledger…

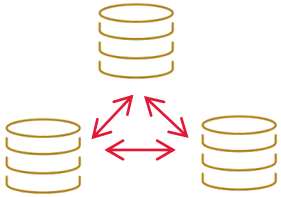
…with consensus, provenance, immutability and finality



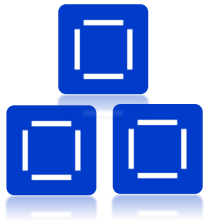
Blockchain 101

Traditional databases cannot be used in untrusted networks

- A traditional database is **centralized**
- Everyone needs to **trust** the administrator managing the database
- There's typically **no immutability or provenance**



- Databases shared across organizations do not alleviate the **trust** issue
- There are now **more copies** to worry about and **more administrators**



- **Blockchain** allows the concept of a distributed database to be deployed across an **untrusted network**
- Something a traditional database cannot handle

Blockchain 101

HOW?

CONSENSUS
All participants agree that a transaction is valid

PROVENANCE
Participants know where the asset came from and how it's ownership has changed over time

IMMUTABILITY
No participant can tamper with a transaction once it's agreed.

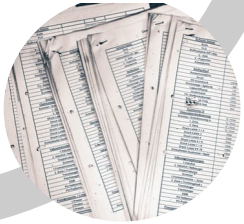
FINALITY
There is **ONE** place to determine the ownership of an asset or completion of a transaction

Blockchain 101

WHEN?

A multi-party problem
Does a business problem include more than two parties?

Reduced costs/Discoverability/Trust/Identity
Is the business problem revolves around efficiency, trust, discoverability, cost reduction, privacy?



Security
Is your centralized business system/process is vulnerable to tempering?

Blockchain Use Cases

Potential Blockchain Use Cases

Financial Institutions

- International payments
- Capital markets
- Trade finance
- Regulatory compliance & audit
- Anti-money laundering & know your customer
- Insurance
- Peer-to-peer transactions

Corporates

- Supply chain management
- Healthcare
- Real estate
- Media
- Energy

Governments

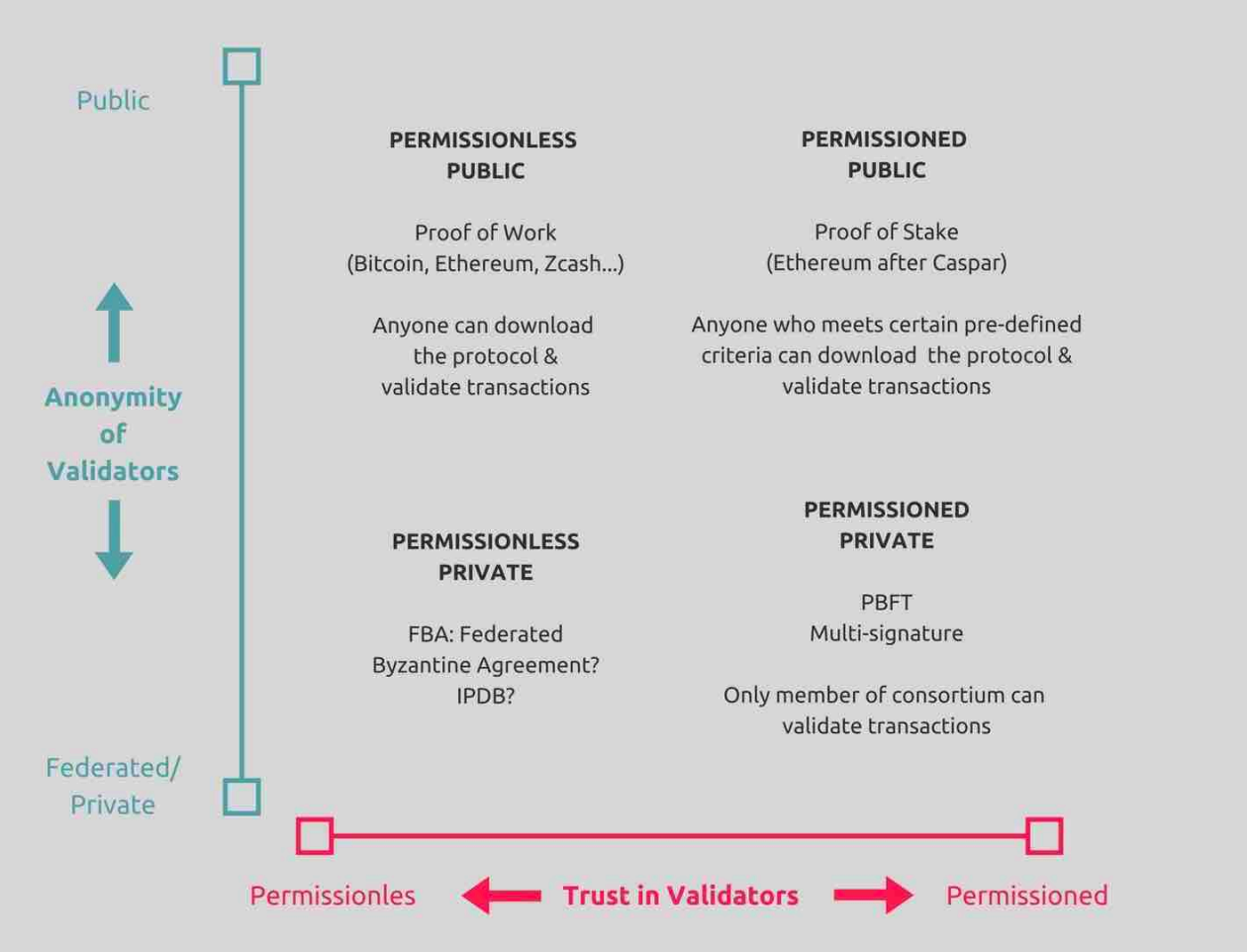
- Record management
- Identity management
- Voting
- Taxes
- Government & non-profit transparency
- Legislation, compliance & regulatory oversight

Cross-industry

- Financial management & accounting
- Shareholders' voting
- Record management
- Cybersecurity
- Big data
- Data storage
- Internet of Things

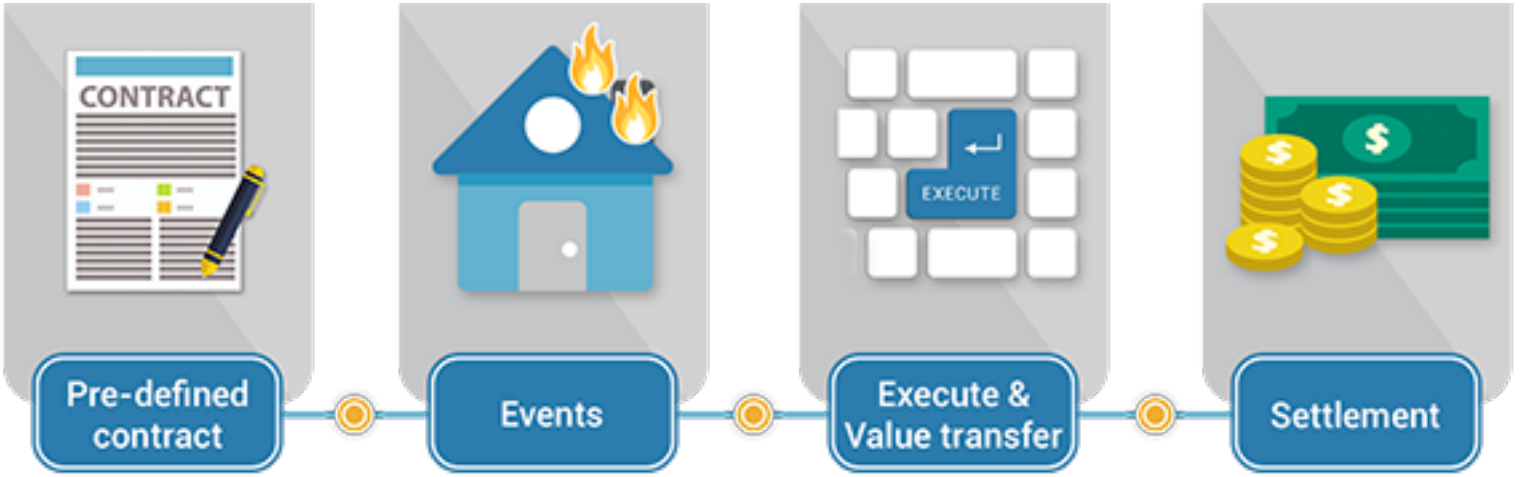
Source: https://www.n-ix.com/outsourcing-blockchain-development-how-to-make-it-work/

Blockchain Type



Source: https://medium.com/@pavelkravchenko/ok-i-need-a-blockchain-but-which-one-ca75c1e2100

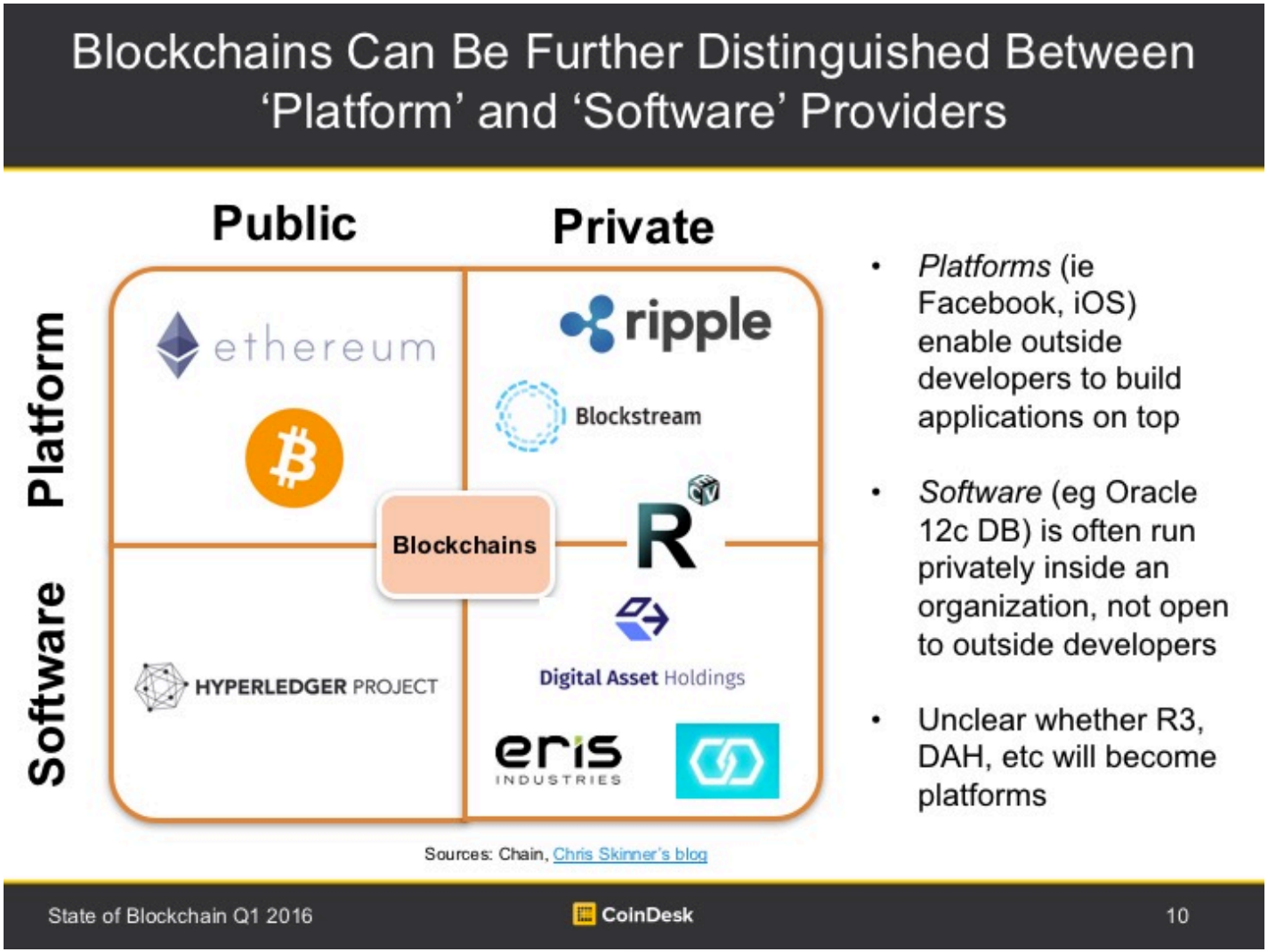
Blockchain 101 – Smart Contracts



- Terms of the policy are agreed by all counterparties
- These are hard coded into the smart contract and cannot be chan-ged without all parties knowing
- Event triggers insurance policy execution
- The smart contract policy is automatically executed based on the pre-agreed terms
- Payout / other settlement completed instantly and efficiently

Source: https://commons.wikimedia.org/wiki/File:Smart_contracts_in_insurance_policies.png

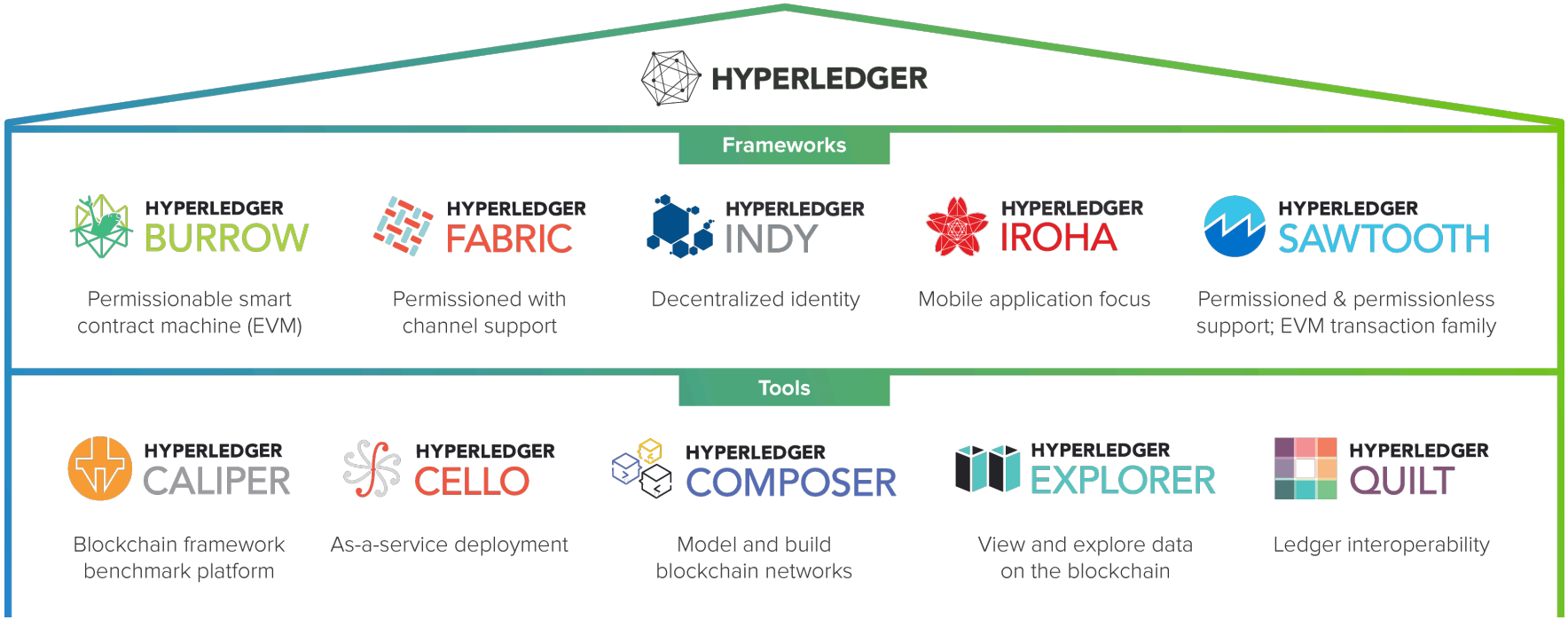
Blockchain 101 – Smart Contracts



- *Platforms* (ie Facebook, iOS) enable outside developers to build applications on top
- *Software* (eg Oracle 12c DB) is often run privately inside an organization, not open to outside developers
- Unclear whether R3, DAH, etc will become platforms

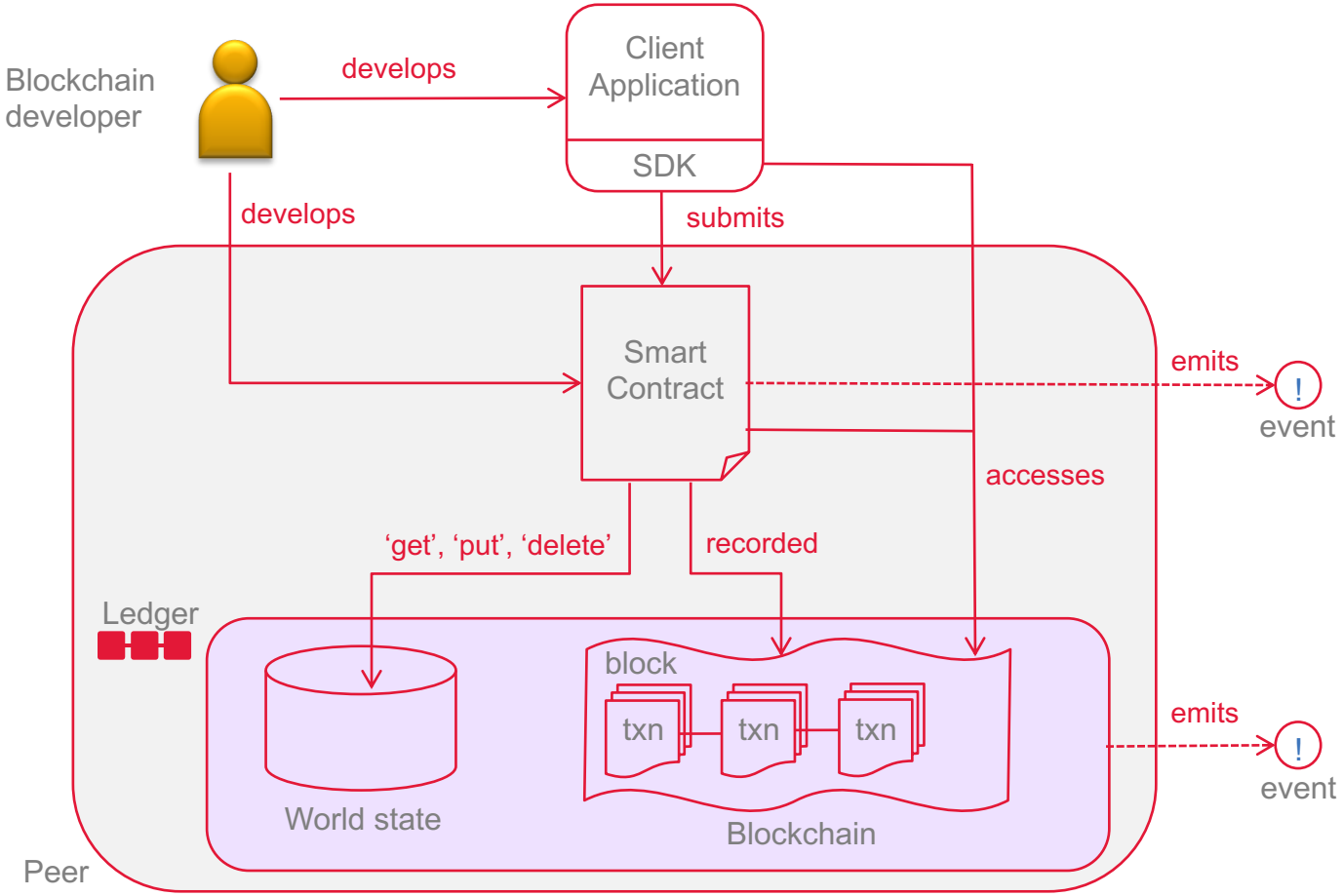
Source: https://www.coindesk.com/state-of-blockchain-q1-2016

Blockchain 101 – Hyperledger



Blockchain 101 – Application Development

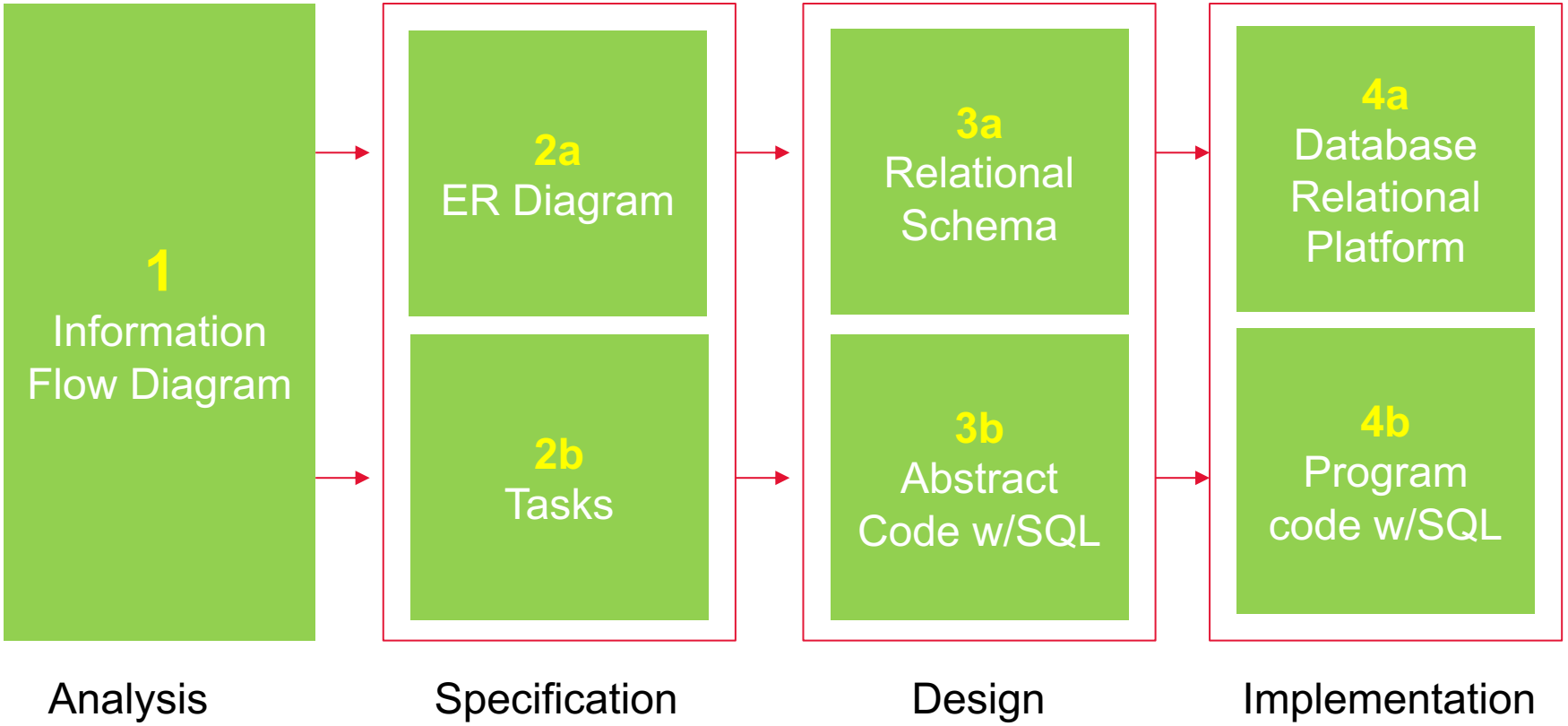
How applications interact with the ledger



Blockchain 101 – Example

Blockchain in Heathcare

What we learned today



Thank you!