Choosing the right database

Connor McDonald

Database Advocate



I know what you are thinking


EVERYTHING
ELSE

tackle the bigger issue


ALL THE
NEW TOYS
WHAT
YOU HAVE

Choosing the right database



Connor McDonald

~~Industry Visionary~~

Choosing the right database

Connor McDonald

~~Thought Leader~~

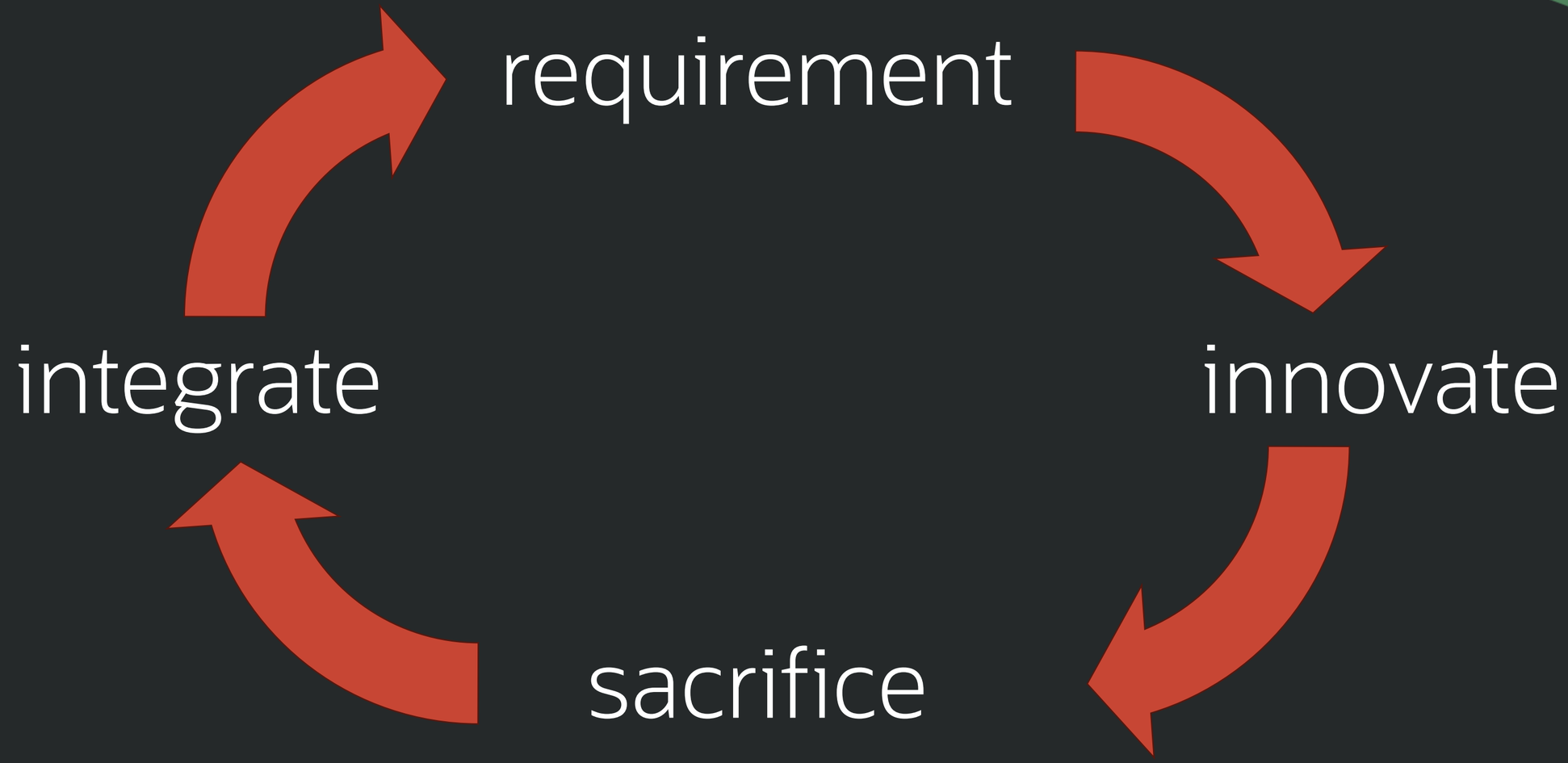
Choosing the right database

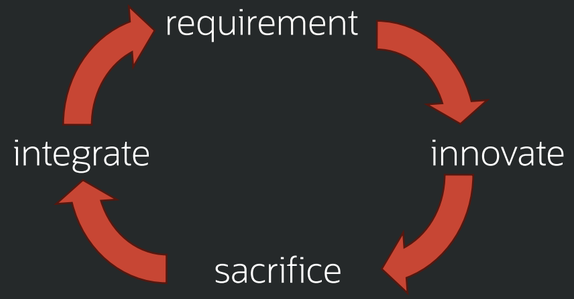
Connor McDonald

Old dude who's lived through much of this



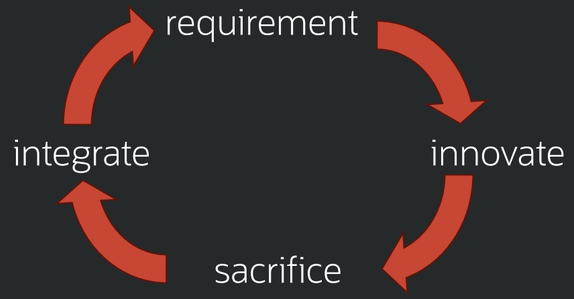
the cycle of database evolution


requirement
innovate
sacrifice
integrate



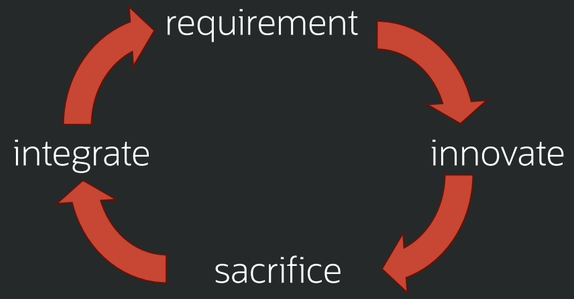
requirement

current solution does not provide something



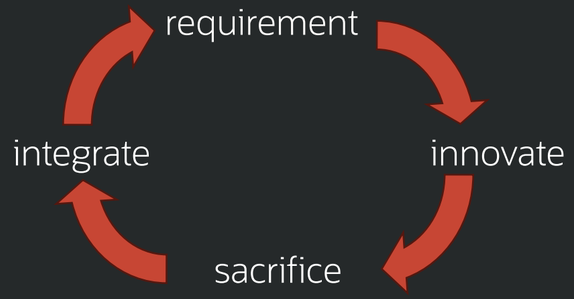
innovate

build something new that does provide it



sacrifice

did I have to give up something ?



integrate

solution has all the good parts of old and new

McDonald's Conjecture:



Unless you are a hyperscaler…

"The only database you should ever use is when it is at the INTEGRATE point"

a look at the history

1950's

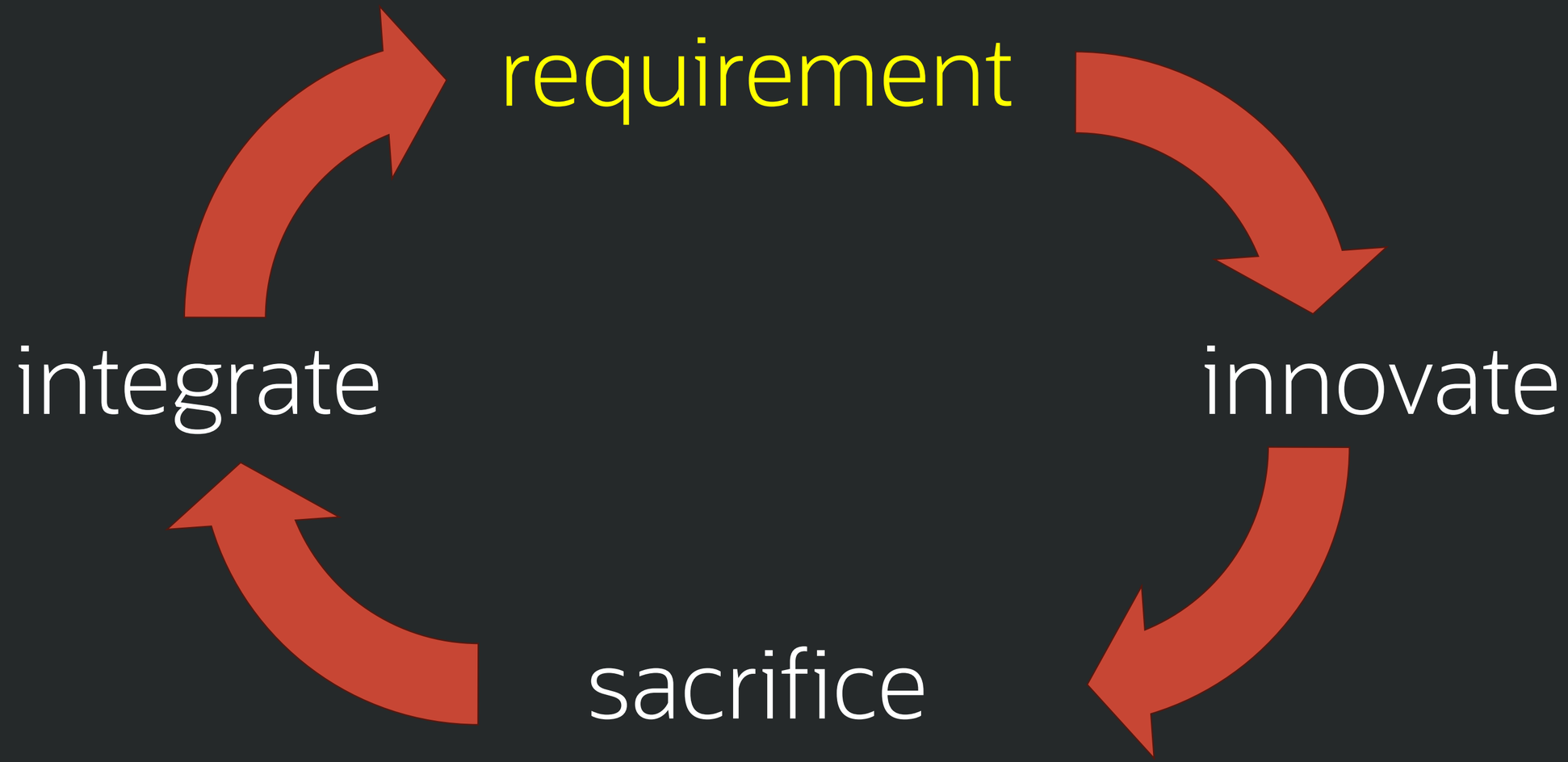


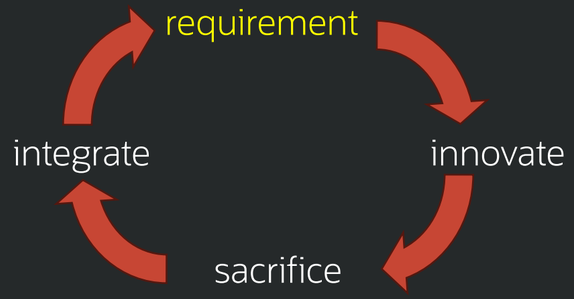
data access was sequential

the arrival of disk

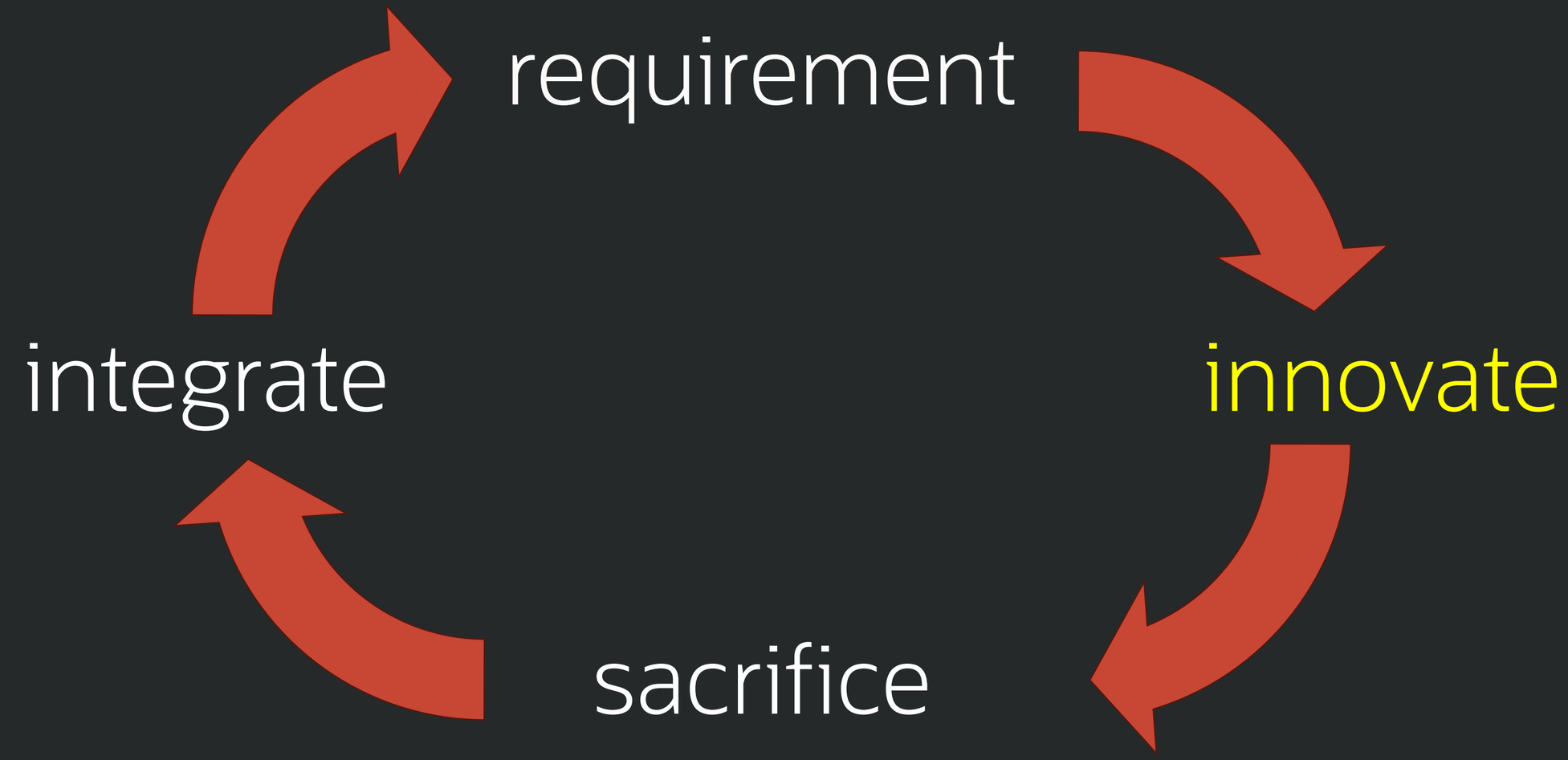


5MB

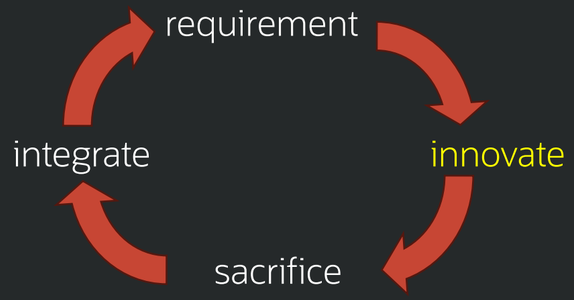

requirement
innovate
sacrifice
integrate



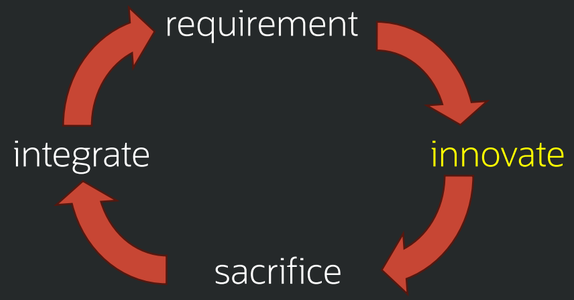
random data access …


requirement
innovate
sacrifice
integrate

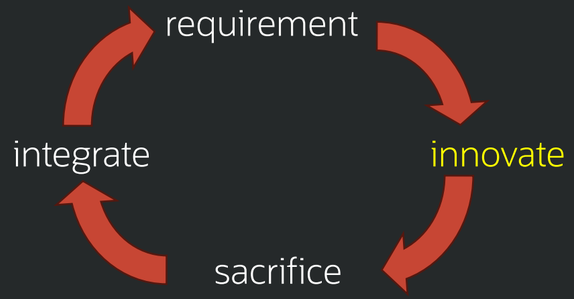
1960s



ISAM



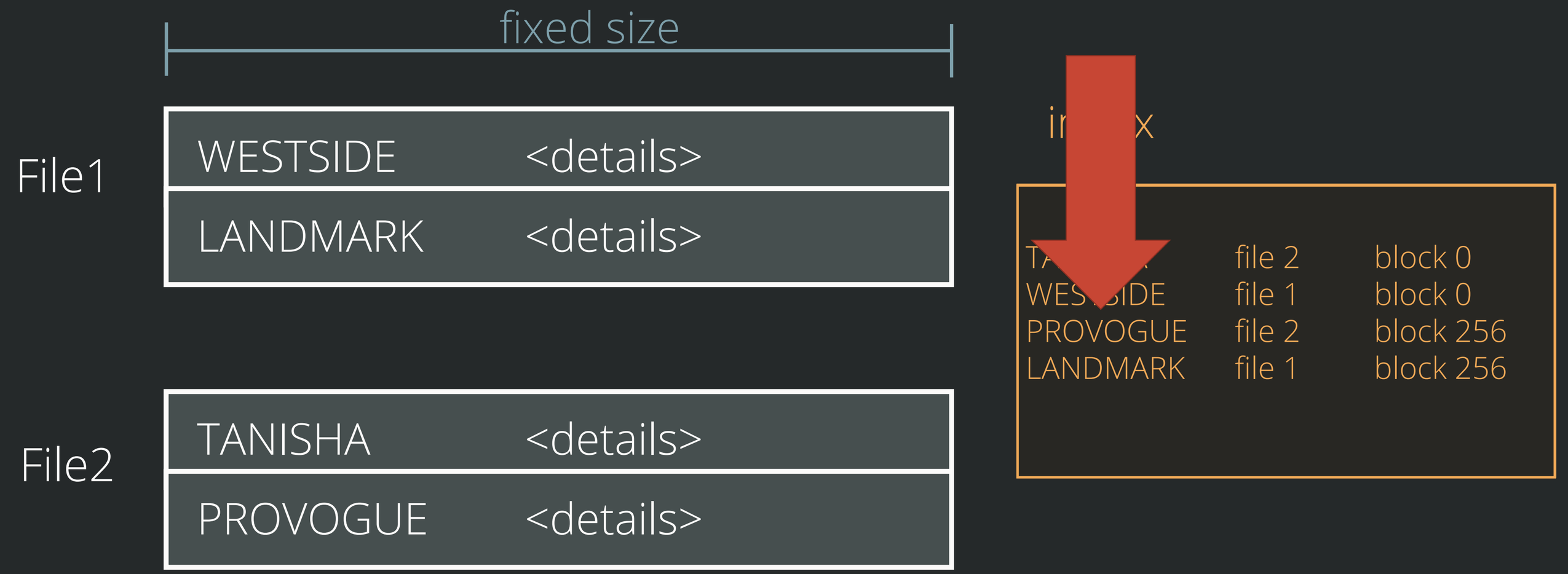
indexed sequential access method



first "databases"

let's go shopping :-)

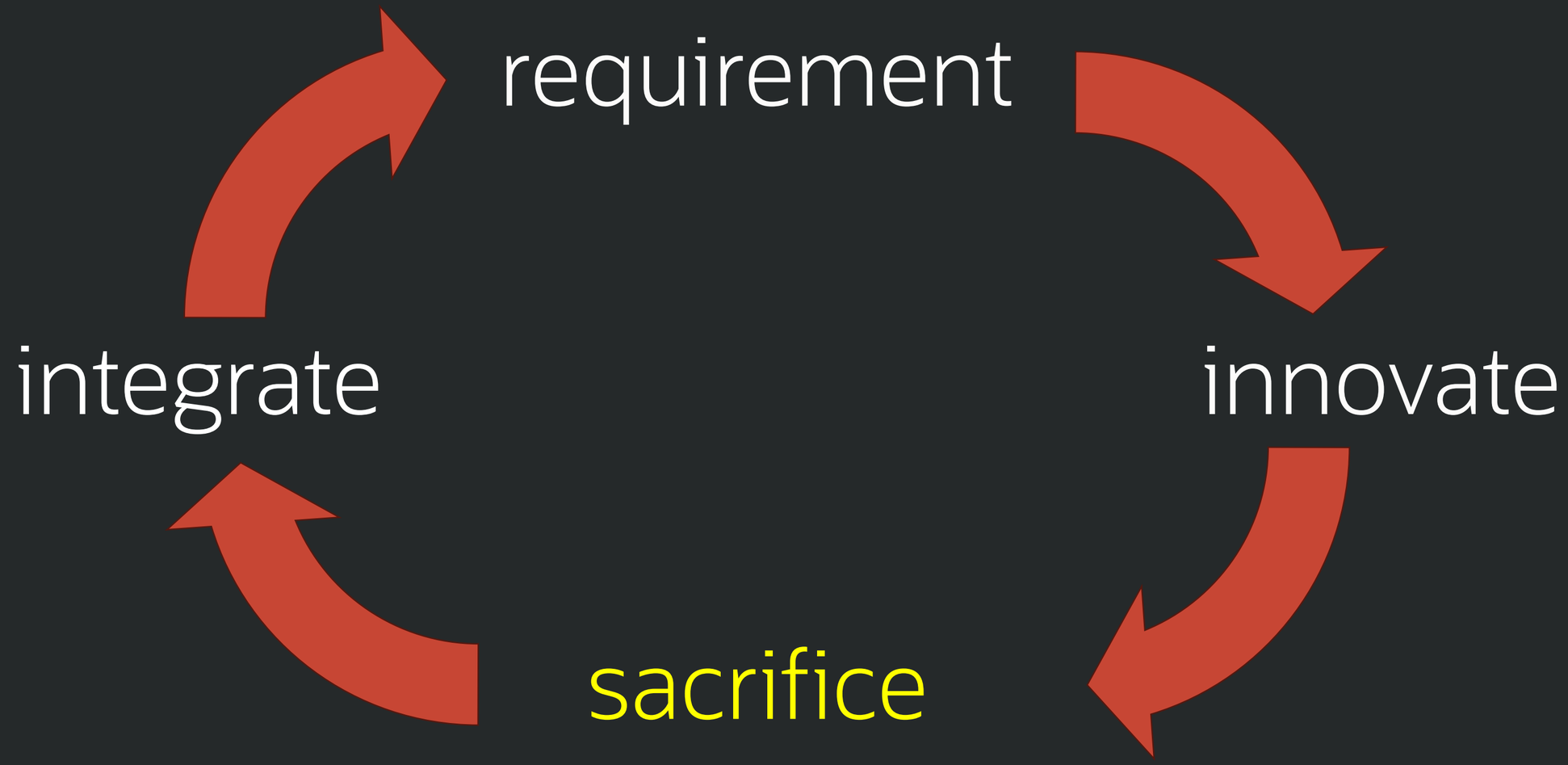
SHOP

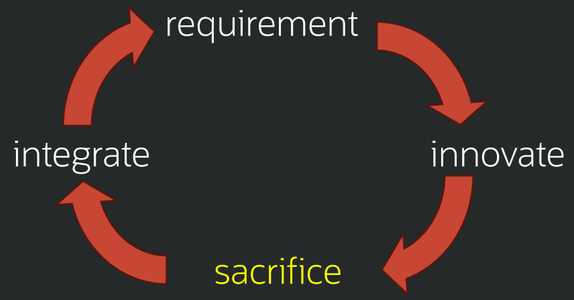

fixed size
File1
WESTSIDE <details>
LANDMARK <details>
File2
TANISHA <details>
PROVOGUE <details>
index
TA file 2 block 0
WES IDE file 1 block 0
PROVOGUE file 2 block 256
LANDMARK file 1 block 256

- open SHOP index
- *loop*
    read next entry
    if entry = 'WESTSIDE' exit;
- *end loop*
- *read record file 2, block 0*

- open PRODUCT index
- *loop*
    read next entry
    if entry = 'MILK' exit;
- *end loop*
- *read record file 23, block 64*

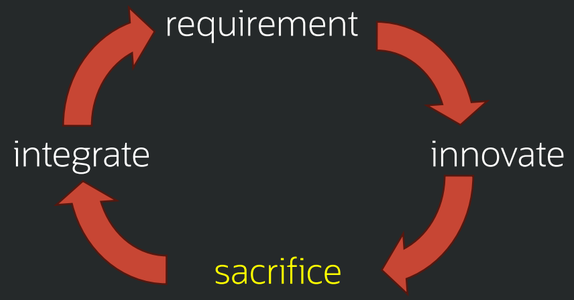
developer codes the access


requirement
innovate
sacrifice
integrate

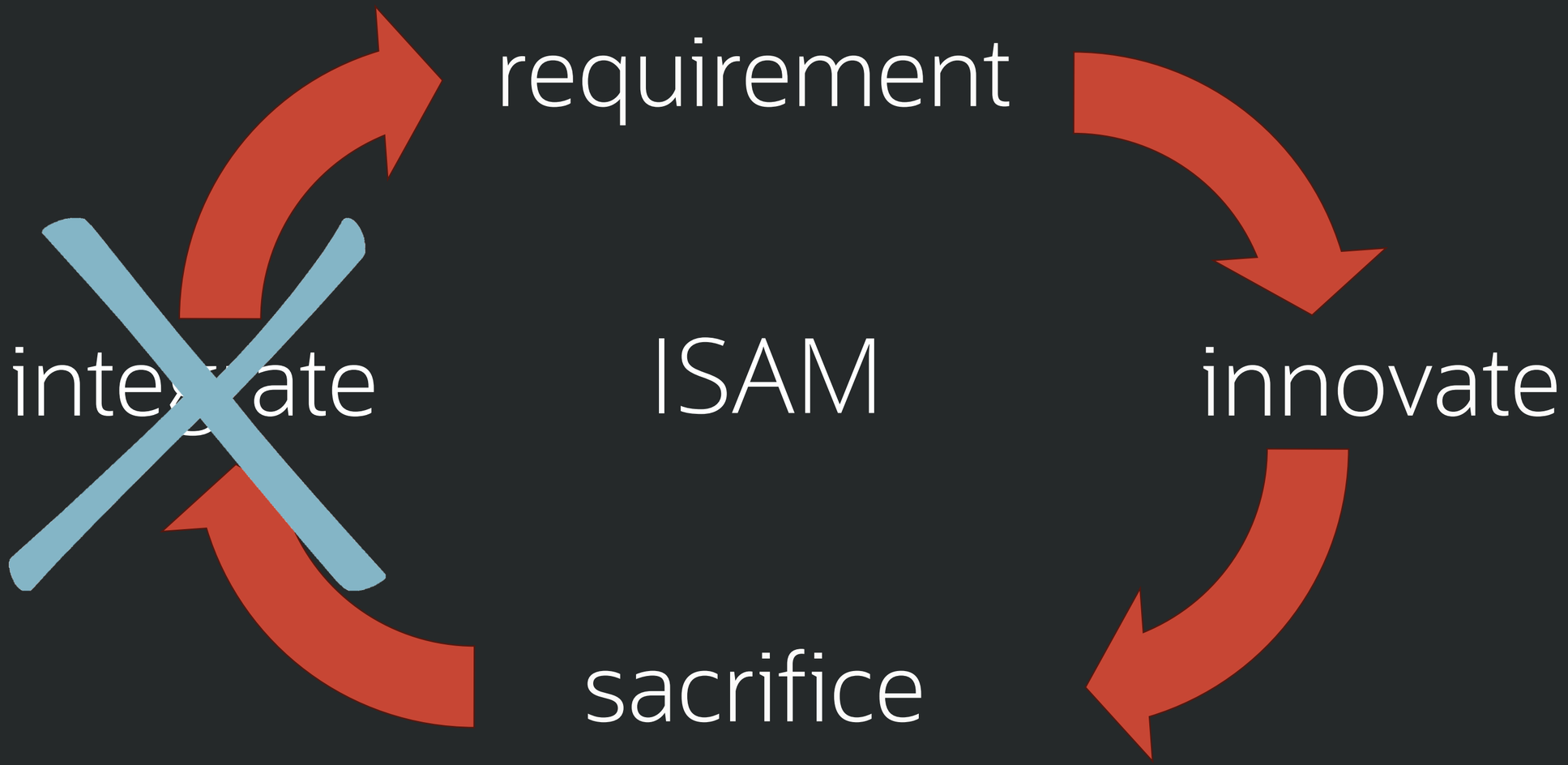


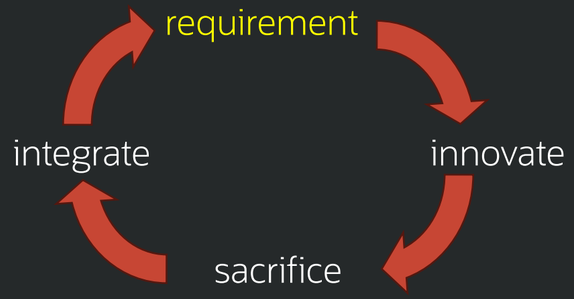
fixed size

WESTSIDE <details>

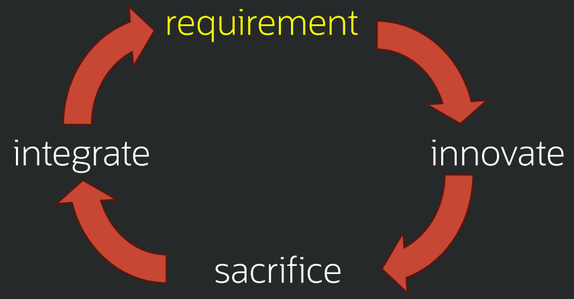


1	9	6	0

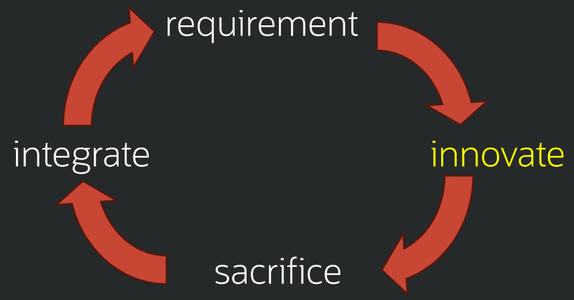

requirement
innovate
sacrifice
ISAM



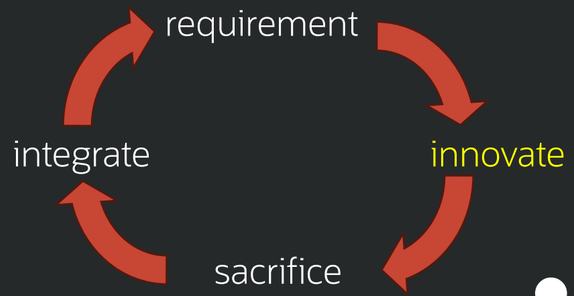
flexible record size



better direct record lookup

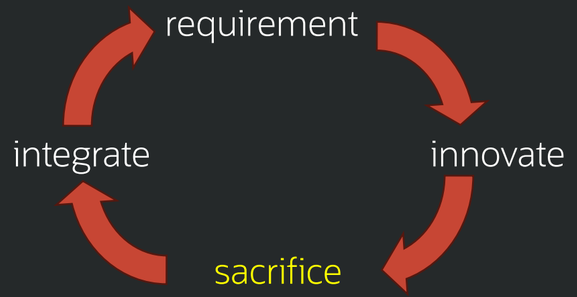


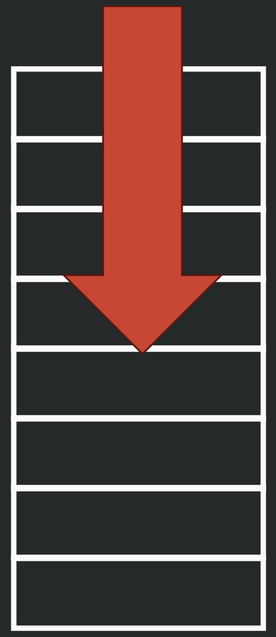
ISAM => VSAM



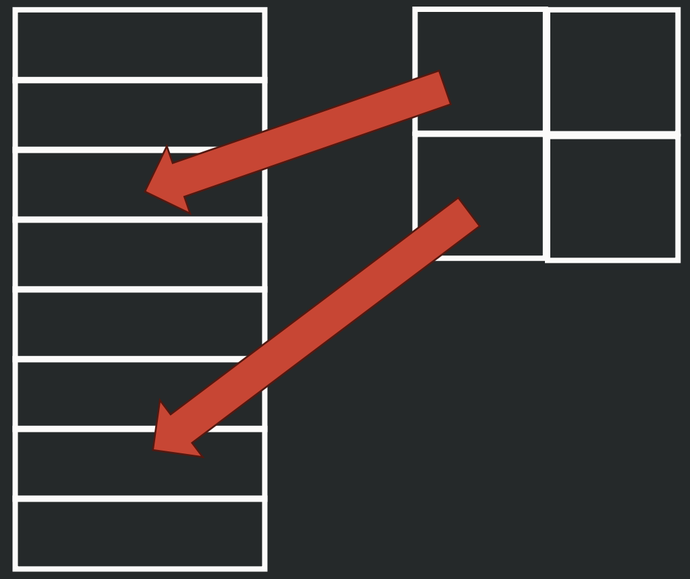
- flexible record size *plus*

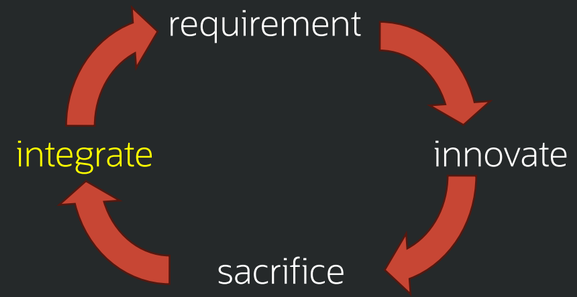
- larger file sizes
- record level locking
- error recovery, etc





performance

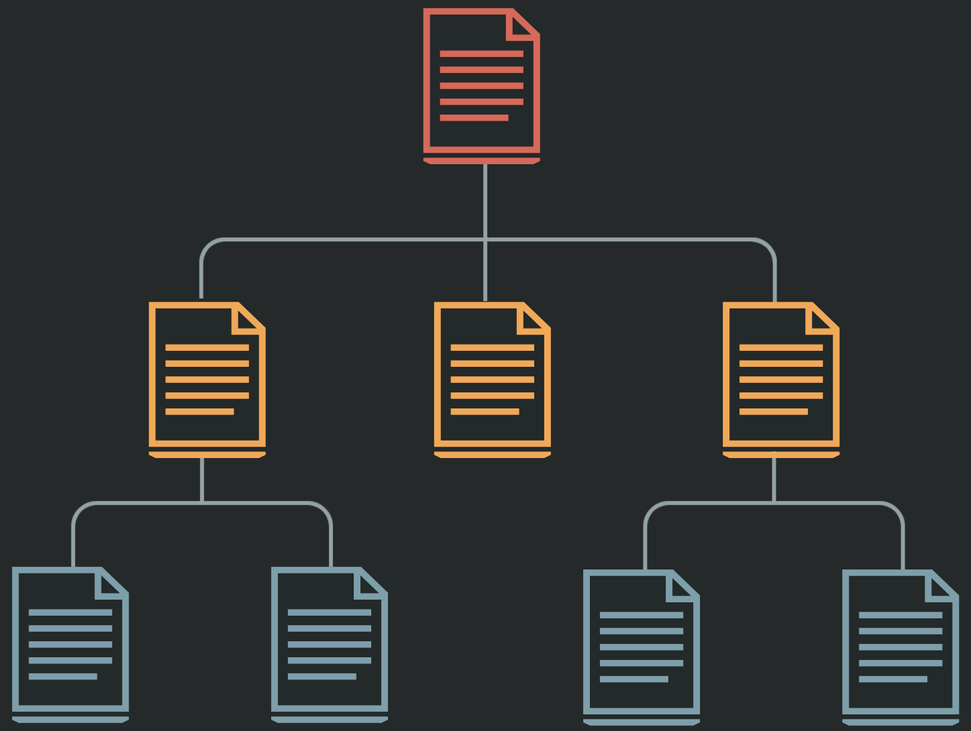




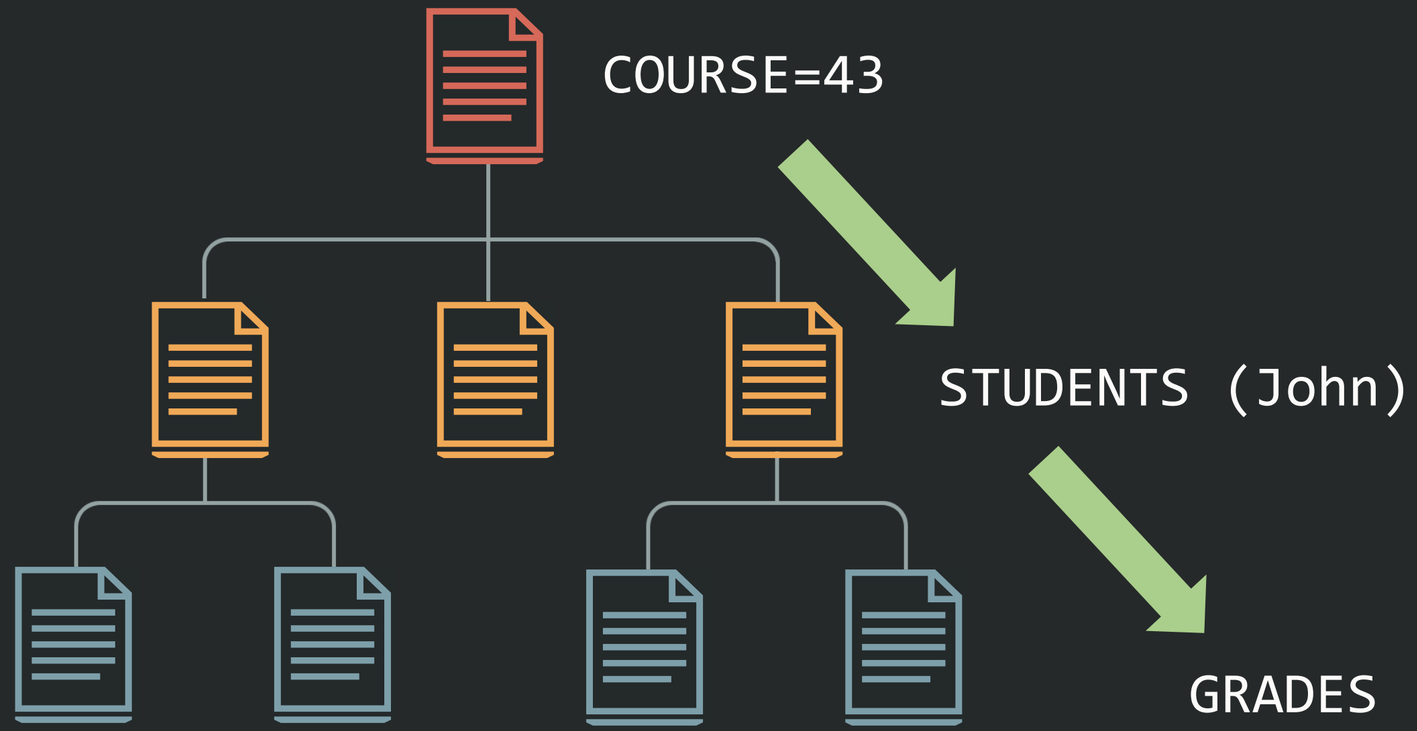
hierarchical databases

VSAM

Hierarchical Databases



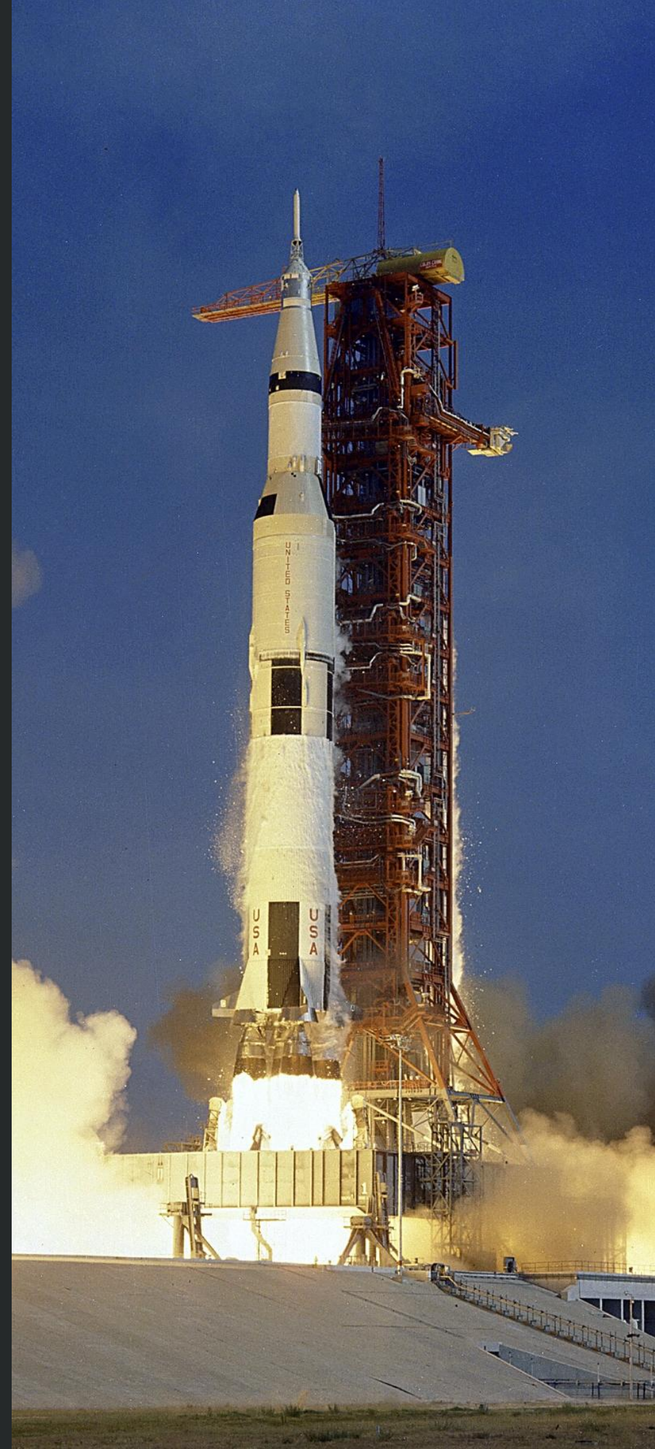
Hierarchical Databases



Information Management System

IMS
DLI / DL1

Data Language Interface / One





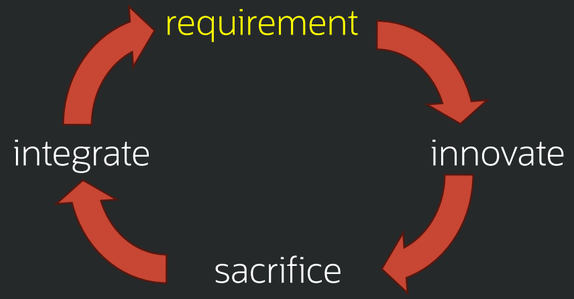


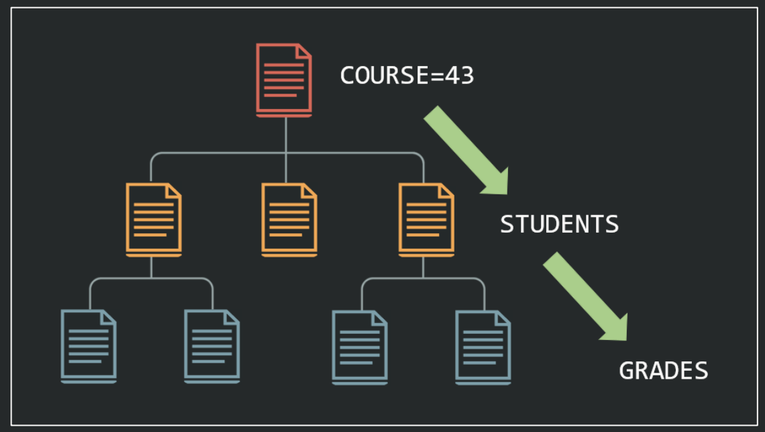
1967 - 1990 (and beyond)

COBOL / IMS / DL1

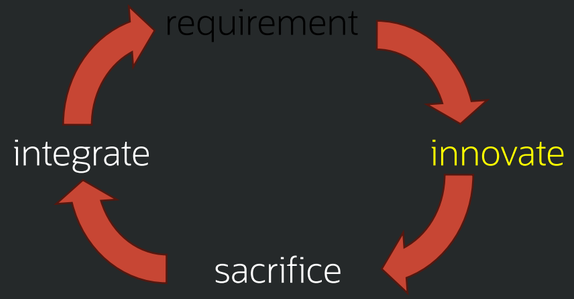
"The only database you should ever use is when it has past the INTEGRATE mark"

1970s



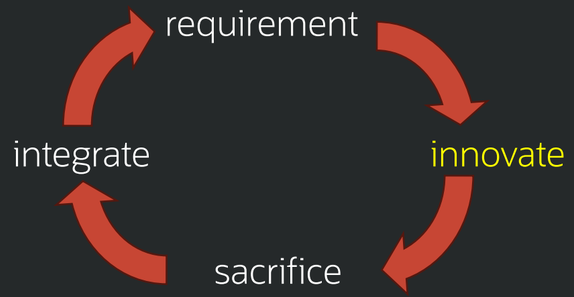


more flexible hierarchy navigation

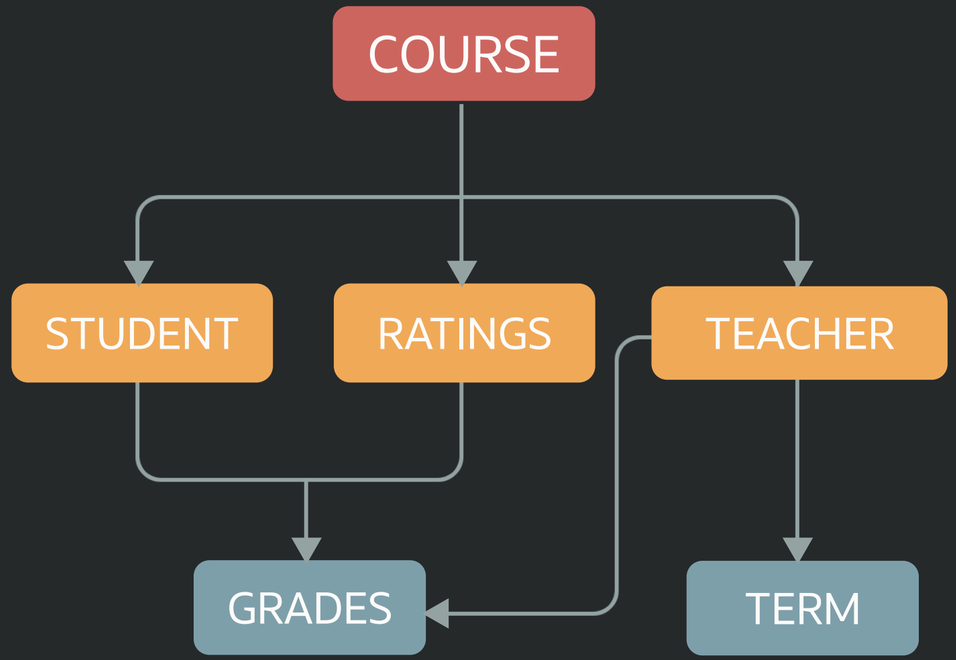


CODASYL

Committee on Data Systems Languages



Network Databases



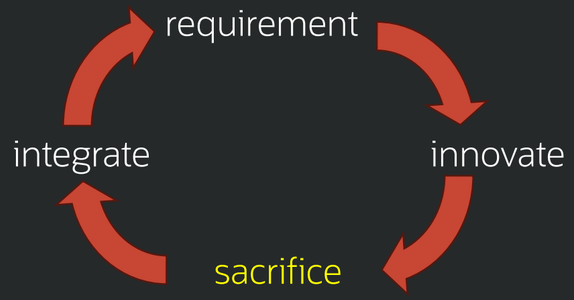


CA IDMS



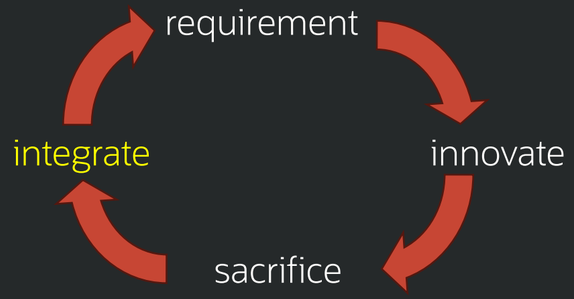
IDS

Charles Bachman

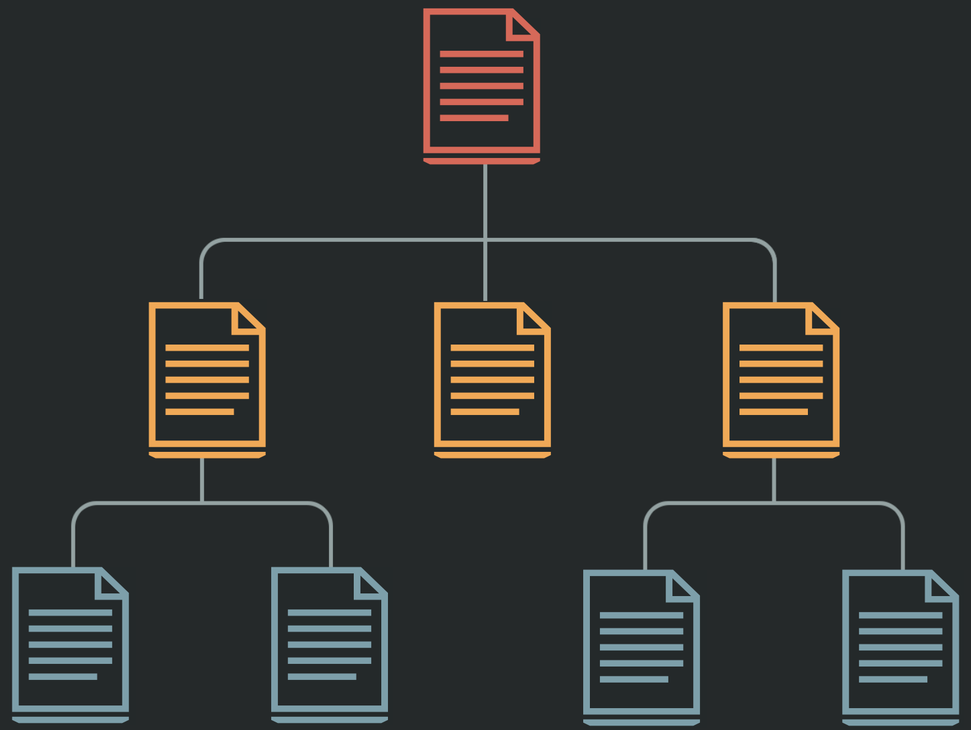


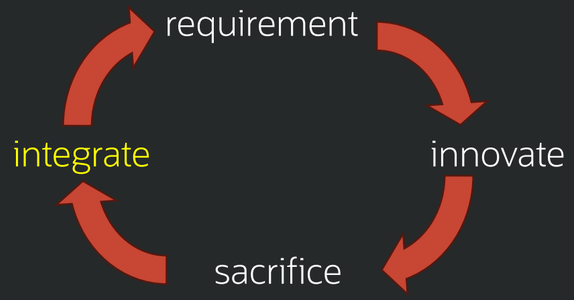
complexity

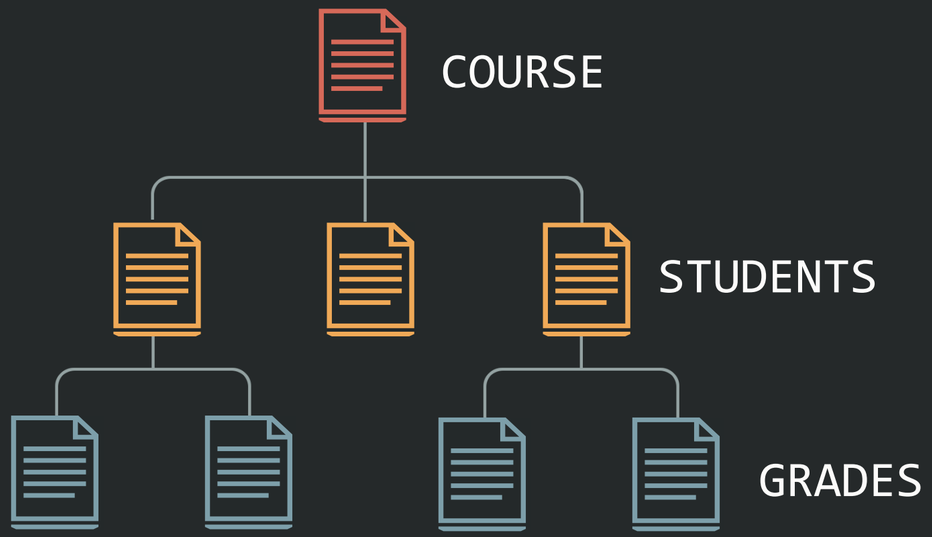
low level operations

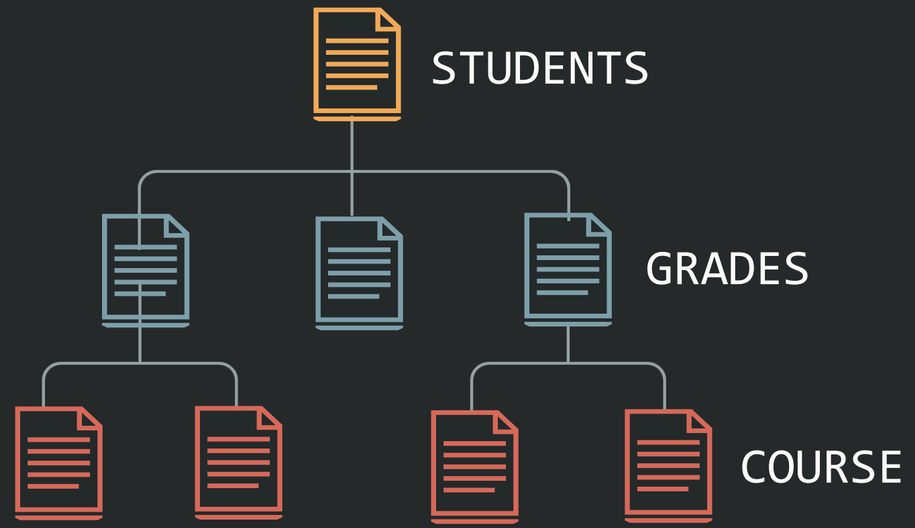


Hierarchical Databases

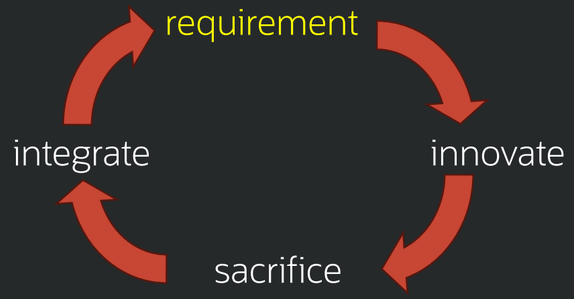



requirement
innovate
sacrifice
integrate


COURSE
STUDENTS
GRADES


STUDENTS
GRADES
COURSE

1980s

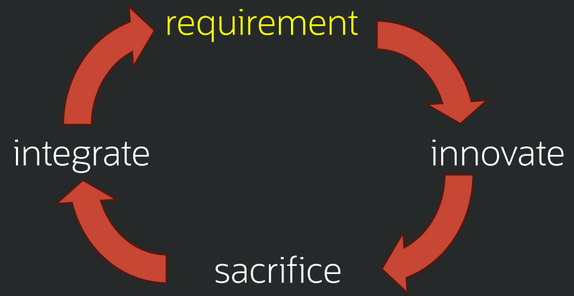


1) easier access to data

"Find Maria's details and derive a 10% commission"

```
PGM DEFINITION
  FILE1=DATASET.ISAM.7X4G.EMPLOYEE
  FILE2=DATASET.ISAM.A8G4.RESULTS

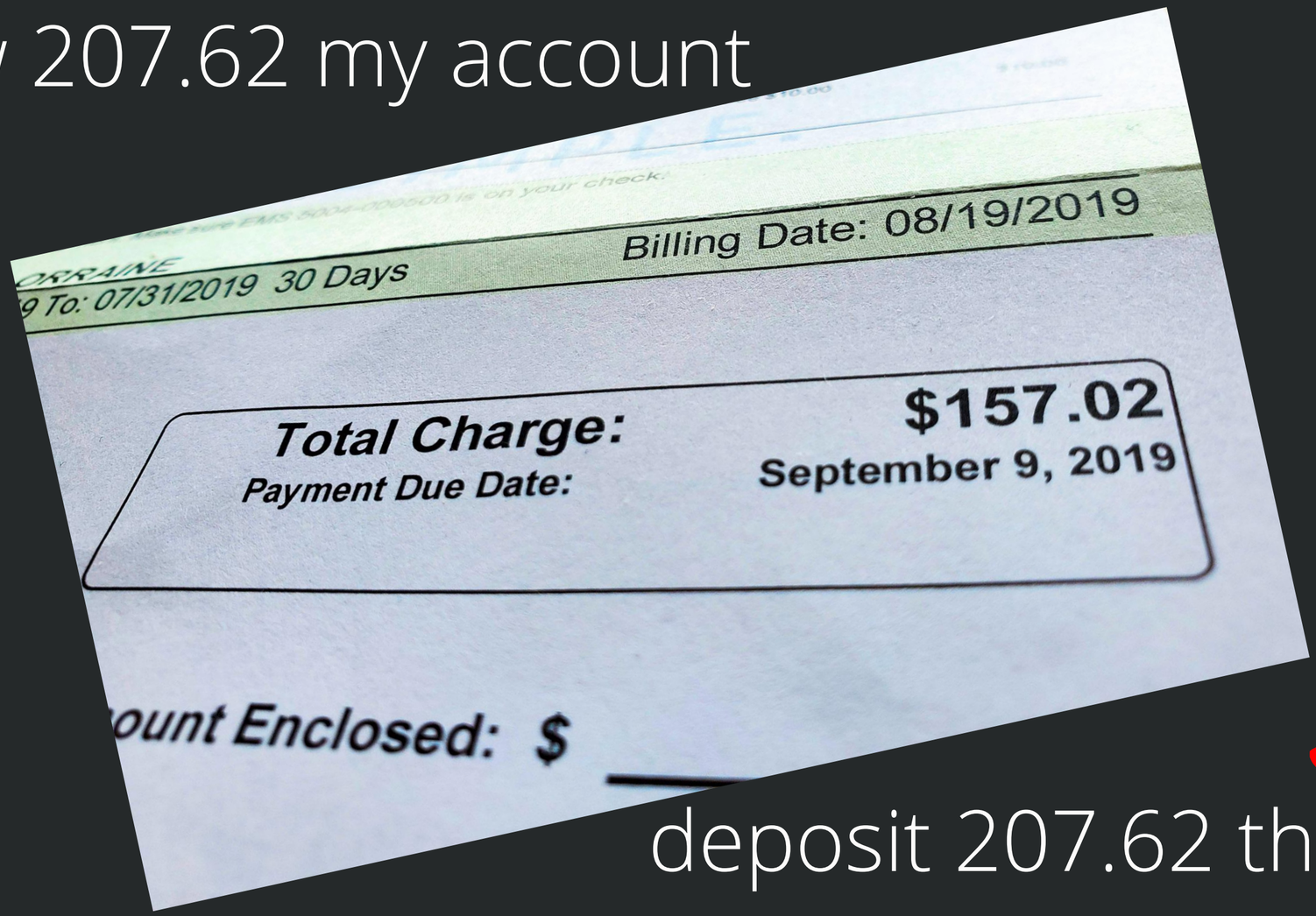
PGM EXECUTE
  OPEN FILE=FILE1,MODE=READ
  OPEN FILE=FILE2,MODE=WRITE
  MOVE "MARIA" TO KEY-ITEM
  GET UNIQUE, KEY=KEY-ITEM
  MOVE BYTES 1-20 TO NAME
  MOVE BYTES 21-30 TO DEPARTMENT
  MOVE BYTES 31-25 TO SALARY
  COMPUTE COMMISION=SALARY*0.10
  WRITE NAME,DEPARTMENT,SALARY,COMMISSION TO FILE2
```



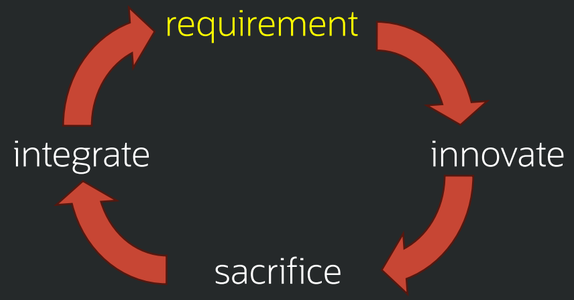
2) data accuracy

transactions

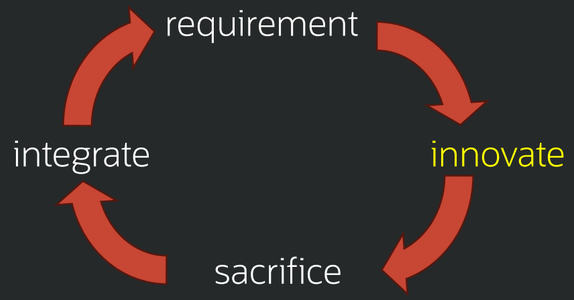
withdraw 207.62 my account

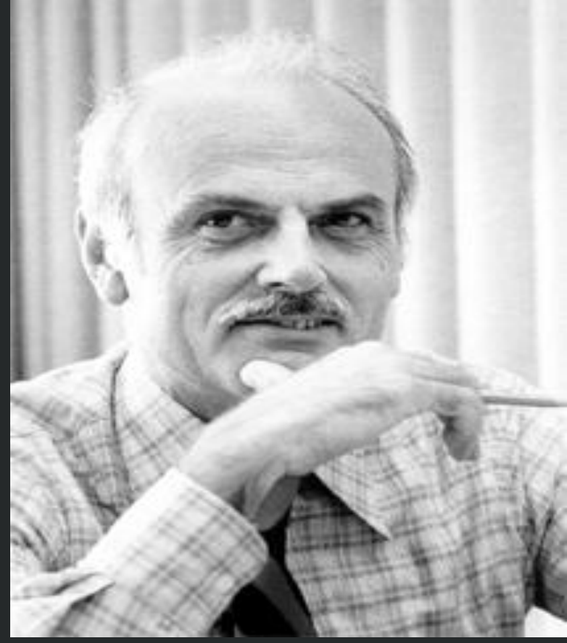


deposit 207.62 their account



3) data storage independence





Edgar (Ted) Codd

relational database theory

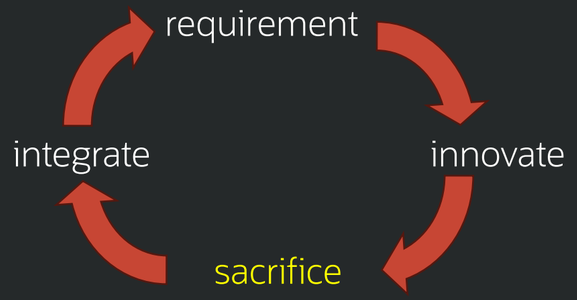
EMPID	NAME	DEPARTMENT
7369	SMITH	RESEARCH
7499	ALLEN	SALES
7521	WARD	SALES
7566	JONES	RESEARCH
7654	MARTIN	SALES
7698	BLAKE	SALES
7782	CLARK	ACCOUNTING
7788	SCOTT	RESEARCH

```
S =
{
  {7369,SMITH ,RESEARCH}
  {7499,ALLEN ,SALES}
  {7521,WARD  ,SALES}
  {7566,JONES ,RESEARCH}
  {7654,MARTIN,SALES}
  {7698,BLAKE ,SALES}
  {7782,CLARK ,ACCOUNTING}
  {7788,SCOTT ,RESEARCH}
}
```

1) easier access to data solved

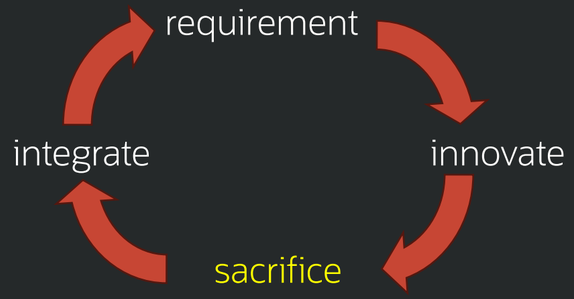
```
S1 =
{
  (EMPID,NAME,DEPARTMENT,COMMISSION) |
    (EMPID,NAME,DEPARTMENT,SAL*0.1) ∃ ORACLEORG ∧ NAME = 'Maria'
}
```

2) data accuracy solved


requirement
innovate
sacrifice
integrate

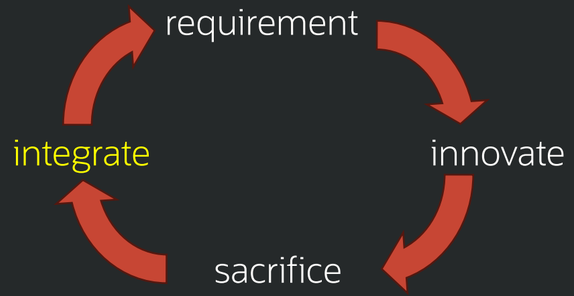
storage


DYSAN CE PACK 200 M/B



performance

but little else…



nothing from network ?

DDL

DML

Data Dictionary

Schema

CODASYL



Oracle



1979



1990 - 2005 (and beyond)

DB2 / ORACLE

"The only database you should ever use is when it has past the INTEGRATE mark"

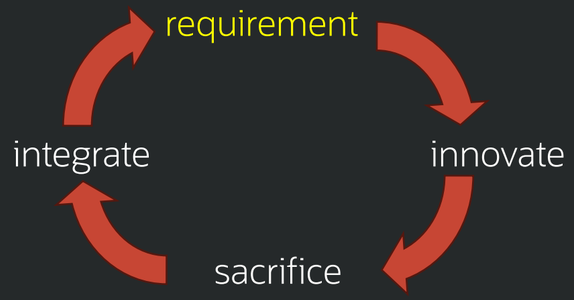
2007


iPhone
Apple rein the phone

everyone starts producing data



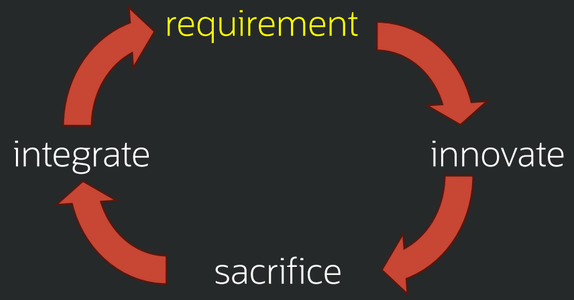




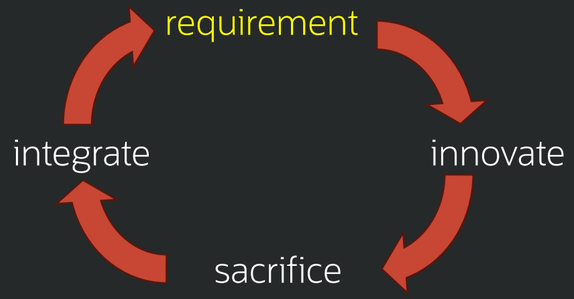
Google

massive data volume

YAHOO!

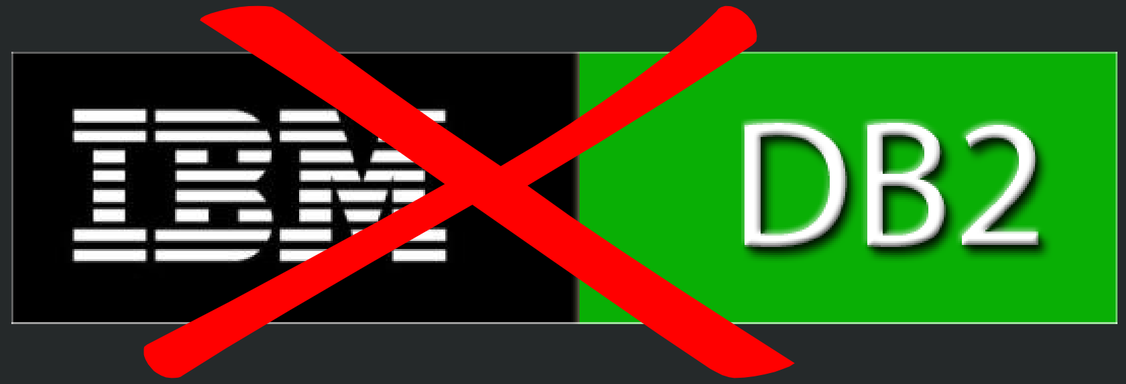


globally distributed data



unstructured data

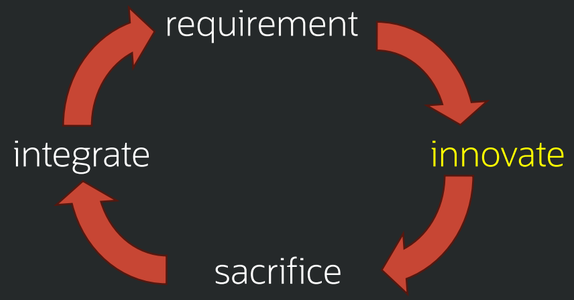




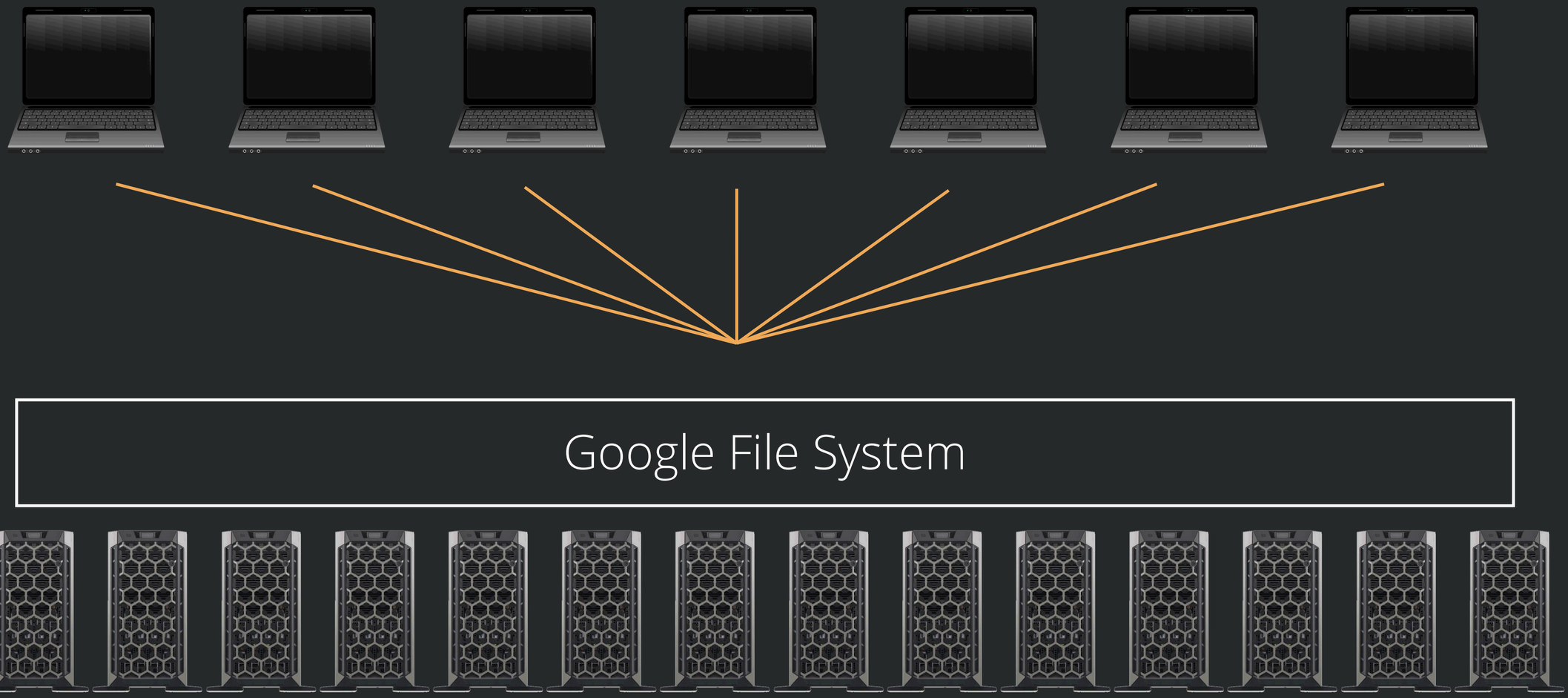
millions of users / petabytes







a new model for storing data

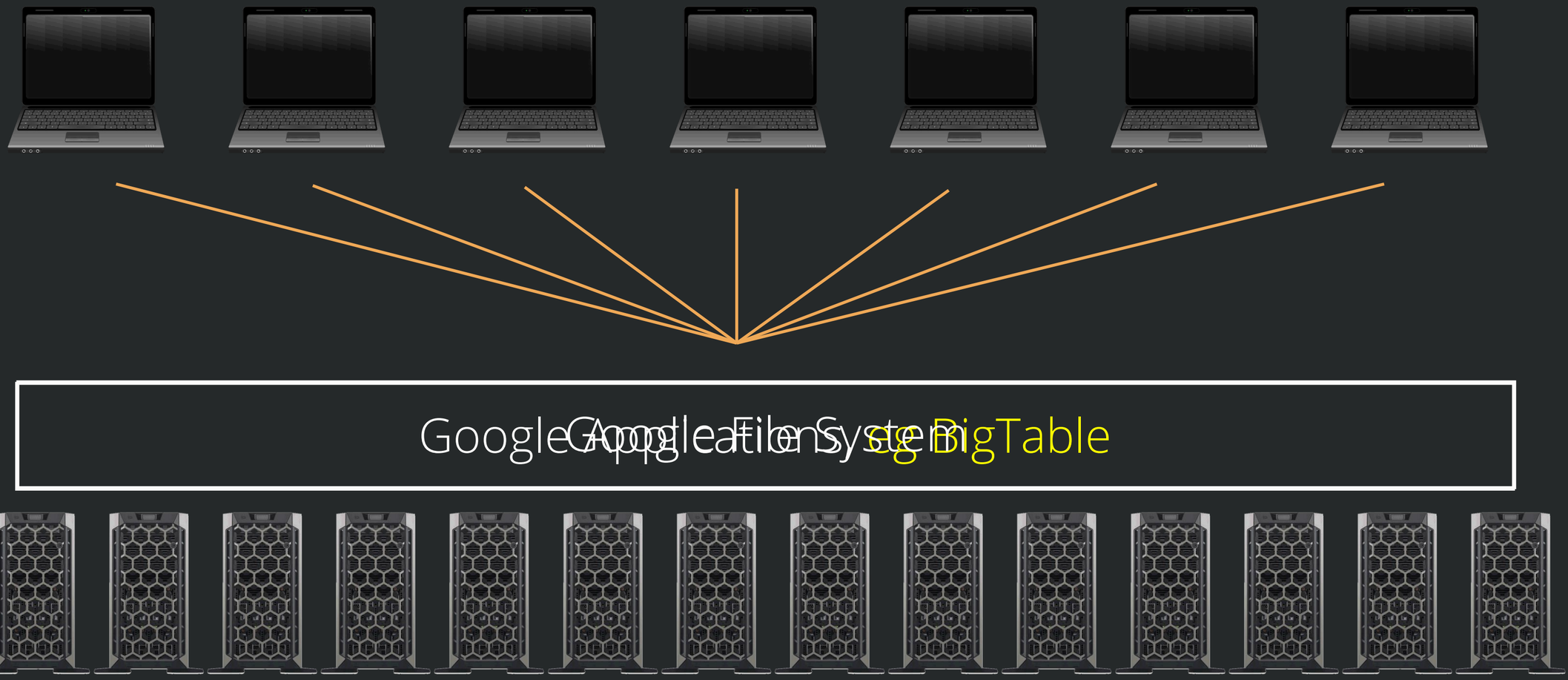

Google File System





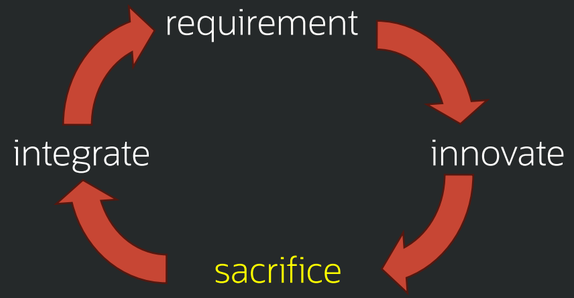
2006 300 servers
2009 4,000 servers
2012 42,000 servers




Google Application Stack
Google File System
Google BigTable

key/value store




requirement
innovate
sacrifice
integrate



1) no SQL

hence "NoSQL"

key lookup only



application responsible for structure



2) no ACID

atomicity
consistency
isolation
durability

data must be partitioned



Eric Brewer,

"Toward robust distributed systems", 2000

Consistency

all nodes show same data at the same time

Availability

nodes failures do not stop other nodes

Partition tolerance

arbitrary loss of messages do not cause failure
(aka, network failure)

"pick any two"


memCached


amazon
DynamoDB


ORACLE
NOSQL
DATABASE


redis





later players to the game

json, xml

successful but niche

(key/value) "...was a necessity because the relational databases could not answer the demand of scalability. If they had answered that demand, people would not have gone to key value stores."

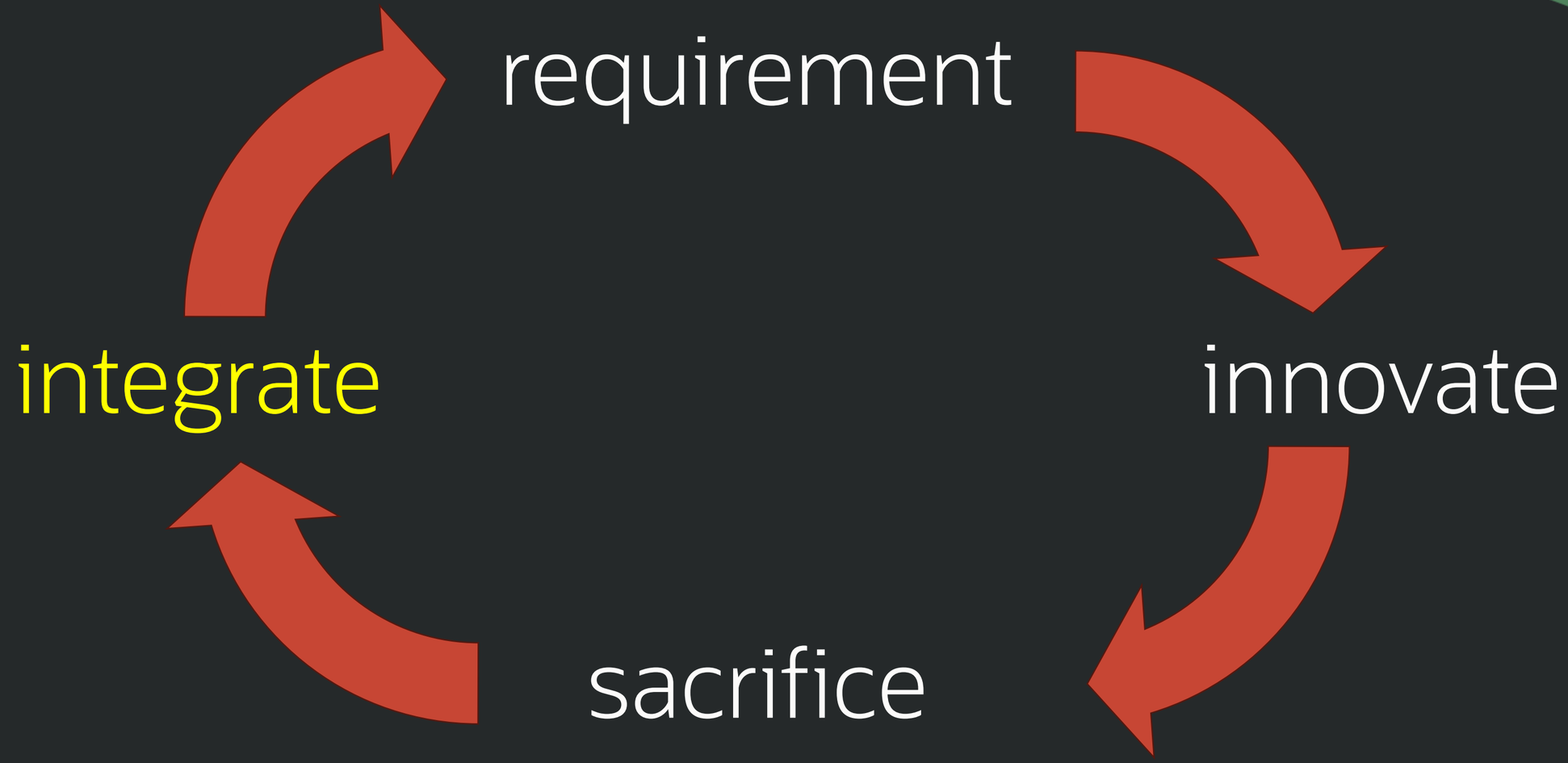
- Sugu Sougoumarane
YouTube

“[Facebook] started in the Hadoop world. We are now bringing in relational and SQL to enhance that. … [we] realized that using the wrong technology for certain kinds of problems can be difficult.”

- Ken Rudin, Facebook

"...the complexity of dealing with a non-ACID data store in every part of our business logic would be too great, and there was simply no way our business could function without SQL queries"

- Google,
A Distributed SQL Database That Scales

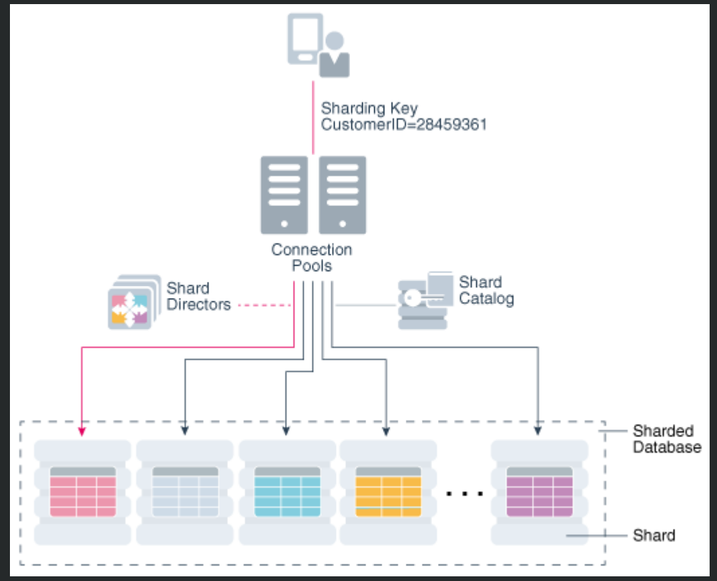

requirement
innovate
sacrifice
integrate

- infinite size
- infinite scale
- geo-location
- unstructured data
- performance over integrity

- nearly unlimited size
- nearly unlimited scale
- geo-location
- unstructured data
- performance over integrity

- geo-location
- unstructured data
- performance over integrity

sharding



- geo-location
- unstructured data
- performance over integrity

JSON / XML

```
SQL> create json collection table mydocs;

Table created.
```

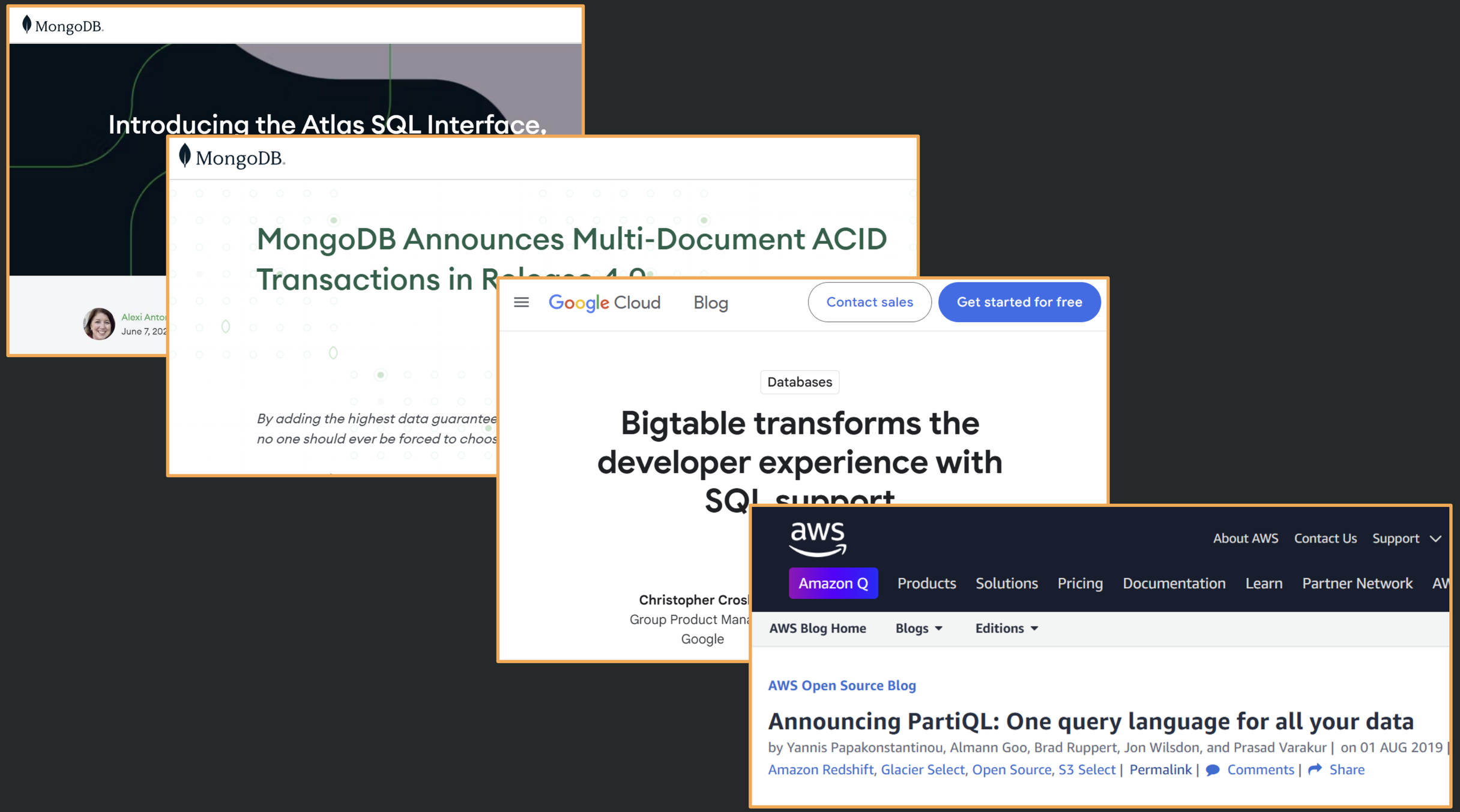
- geo-location
- unstructured data
- performance over integrity

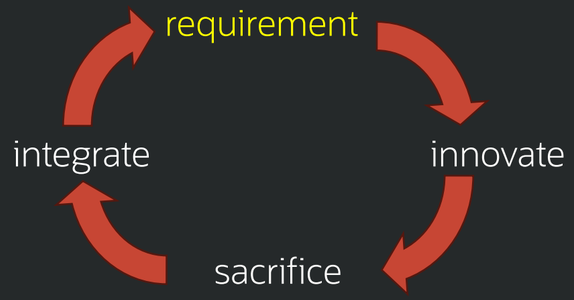
caching / batching

```
SQL> insert /*+ MEMOPTIMIZE_WRITE */
  2  into MYTABLE
  3  values ( ... )
```

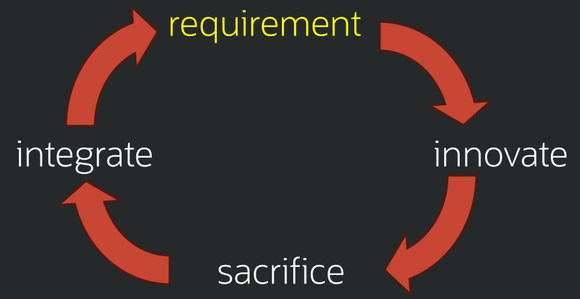
relational = relational plus document

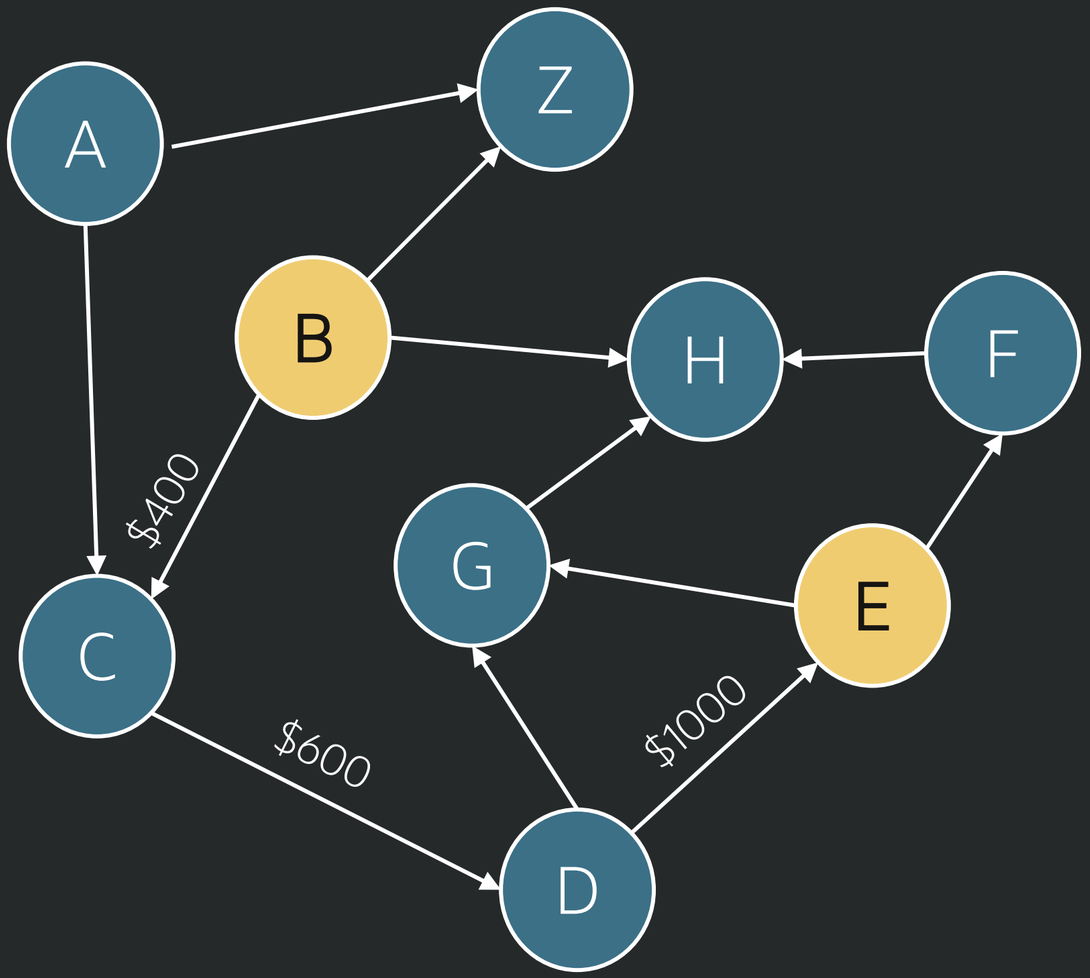
document = document minus relational


MongoDB.
Introducing the Atlas SQL Interface.
MongoDB.
MongoDB Announces Multi-Document ACID
Google Cloud
Blog
Contact sales
Get started for free
Databases
Bigtable transforms the
developer experience with
aws
About AWS
Contact Us
Support
Amazon Q
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Blog Home
Blogs
Editions
AWS Open Source Blog
Announcing PartiQL: One query language for all your data
by Yannis Papakonstantinou, Almann Goo, Brad Ruppert, Jon Wilsdon, and Prasad Varakur | on 01 AUG 2019 |
Amazon Redshift, Glacier Select, Open Source, S3 Select | Permalink | Comments | Share

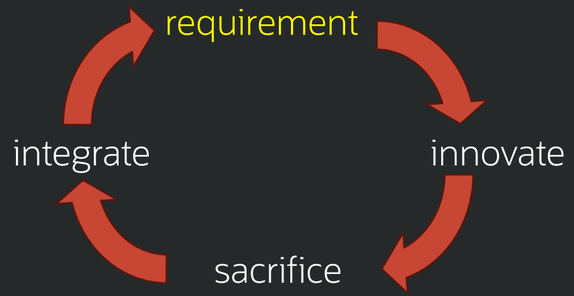


what abut relationships between data?



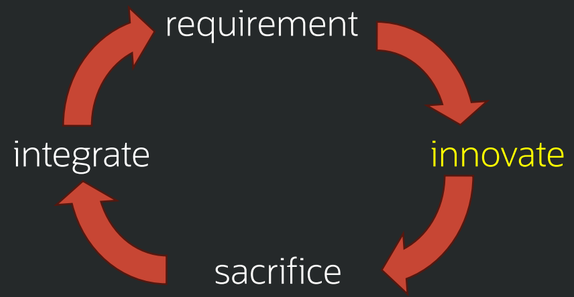


"Link between B and E?"




```
SELECT "v1.id" AS "THOM_ID", "v2.id" AS "LARRY_ID"
FROM (SELECT "V1"."ID" AS "v1.id", "V2"."ID" AS "v2.id"
FROM "GRAPHUSER"."USERS" "V1",
     "GRAPHUSER"."USERS" "V2",
     (WITH t1(src_table, src_key,dst_table, dst_key,exp, lvl)
      -- Anchor member.
      SELECT "src_table" AS src_table, "src_key" AS src_key,
      "src_table" AS dst_table, "src_key" AS dst_key,''AS exp,
      0 AS 1vl
      FROM
        (SELECT 'USERS' AS "src_table", v1.id AS "src_key"
         FROM "GRAPHUSER". "USERS" "V1"
         WHERE ("V1"."ID" = 1))
         UNION ALL
         -- Recursive member.
         SELECT t1.src_table, t1.src_key,t2."dst_table" AS dst_table,
                t2. "dst_key" AS dst_key,t1.exp || t2."exp" AS exp,t1.1vl + 1 AS lvl
         FROM
           (SELECT 'USERS' AS "src_table", "anonymous_1".follower_idAS "src_key",
                   'USERS' AS "dst_table", "anonymous_1".followed_id AS "dst_key",
                   '' AS "edge_table",'' AS "edge_key",
                   (('<EXPRESSIONS>' | | '') || '</EXPRESSIONS>') AS "exp"
            FROM "GRAPHUSER"."FOLLOWS" "anonymous_1"
            WHERE (NOT("anonymous_1"."FOLLOWER_ID" IS NULL)
            AND NOT("anonymous_1". "FOLLOWED_ID" IS NULL))) t2,
     t1AS (
     ...
     ...
```

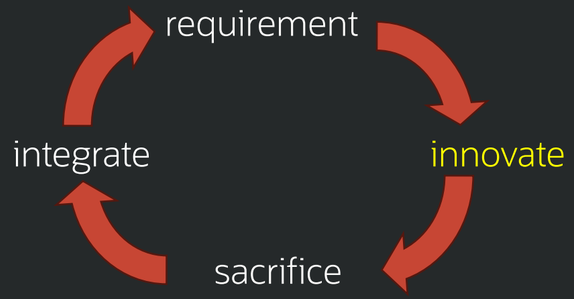
40 more lines



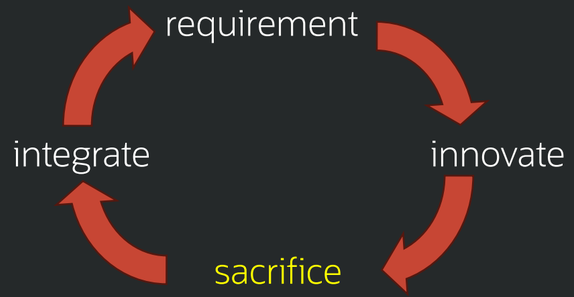


graph databases



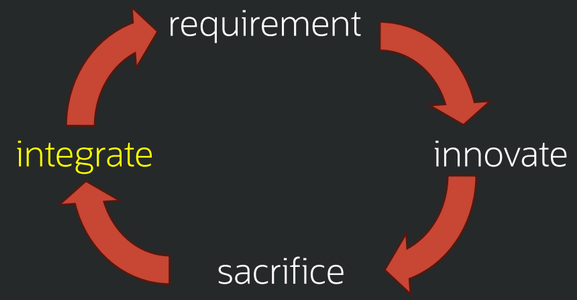



```
MATCH (source:Account{title: 'B'})-[:DEPOSIT]
      -> (target:Account{title: 'E'})
RETURN source.id
```

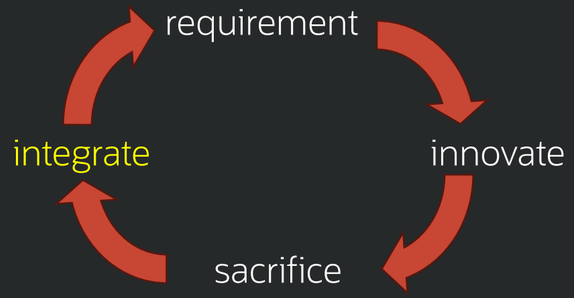


"Transactions, recovery and durability are features that you would take for granted when using a database, may not be working as expected - or worse, not be present at all"

- Johan Svensson
CTO Neo



property graph to relational



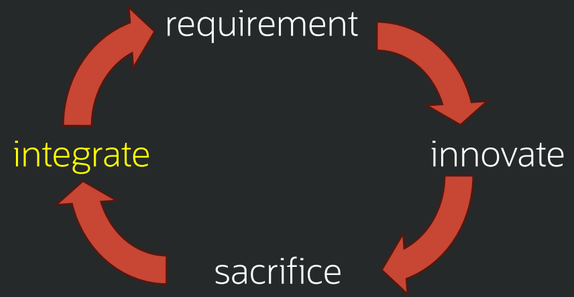

```
CREATE PROPERTY GRAPH bank_graph
       VERTEX TABLES (
         bank_accounts as accounts
         PROPERTIES (id, balance))
       EDGE TABLES  (
         bank_transfers
         SOURCE KEY      (from_acc) REFERENCES ACCOUNTS(ID)
         DESTINATION KEY (to_acc)   REFERENCES ACCOUNTS(ID)
         PROPERTIES (amount, to_acc));
```



BANK ACCOUNTS



MONEY TRANSFERS



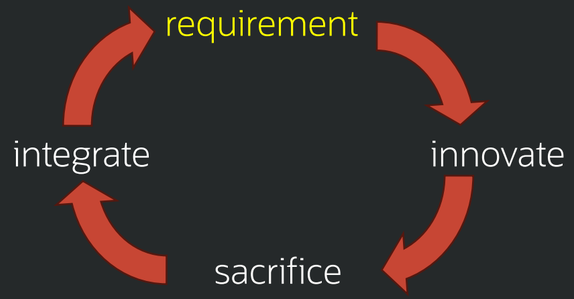

```
SELECT source_id, target_id
FROM graph_table(bank_graph
       MATCH (v1)-[IS deposit]->{1,5}(v2)
       WHERE v1.name = 'B'
       AND v2.name = 'E'
       COLUMNS (v1.id AS source_id, v2.id AS target_id)
);
```

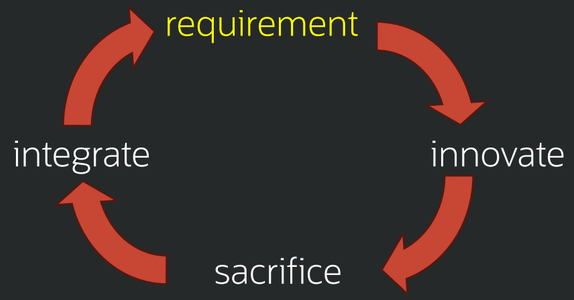
which brings us (almost) up to date

ChatGPT

what is a vector database

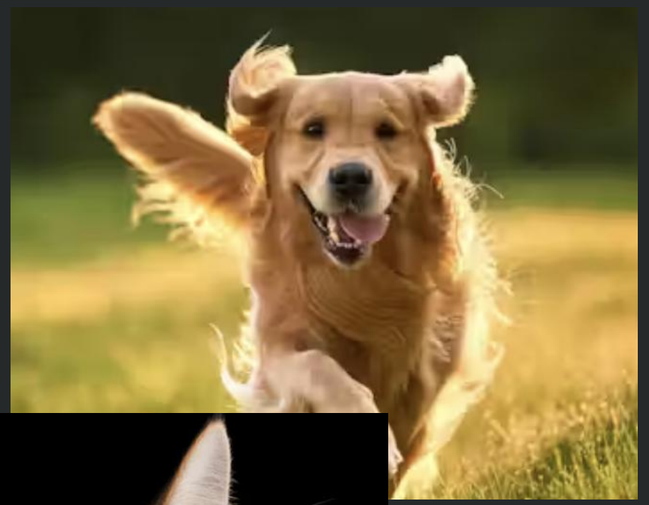
A **vector database** is a type of database designed to store, index, and search **high-dimensional vector embeddings** efficiently. Instead of storing traditional structured data like rows and columns, it manages numerical vectors, which represent data points in multi-dimensional space.


requirement
innovate
sacrifice
integrate



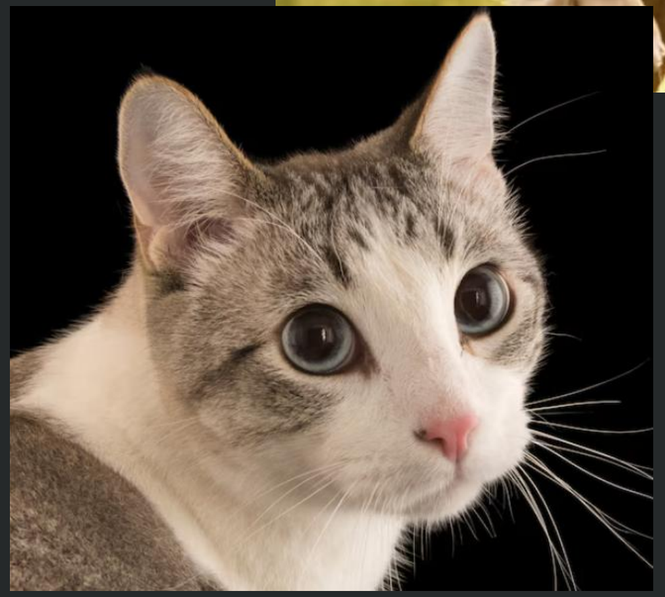
bring human meaning to data







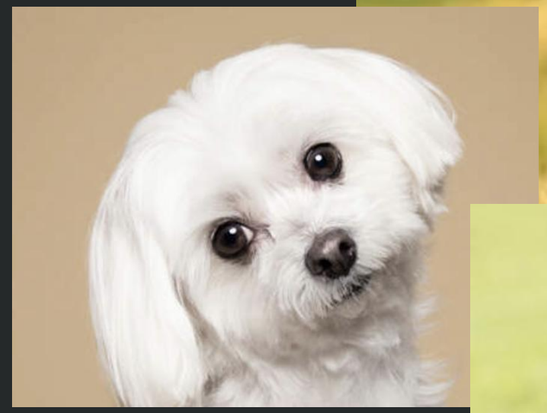








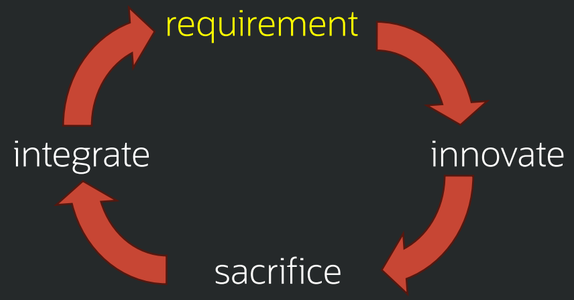




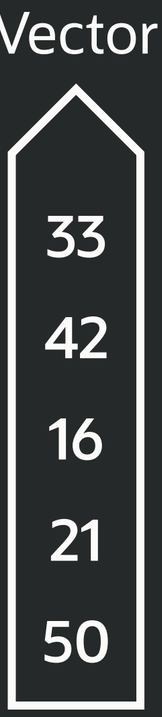






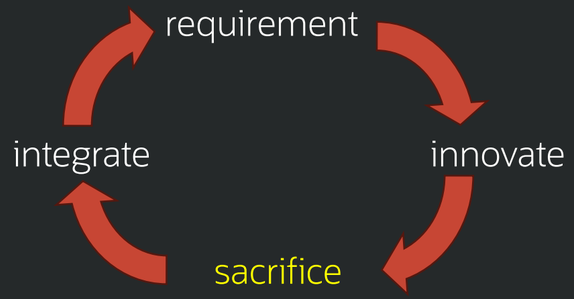


handling of vectors



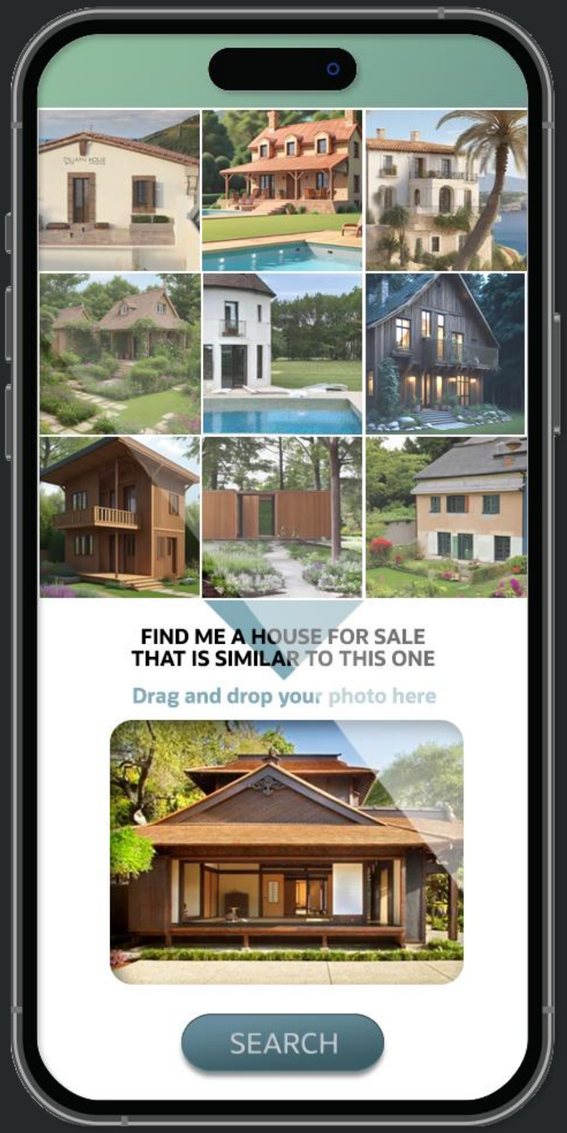

Pinecone

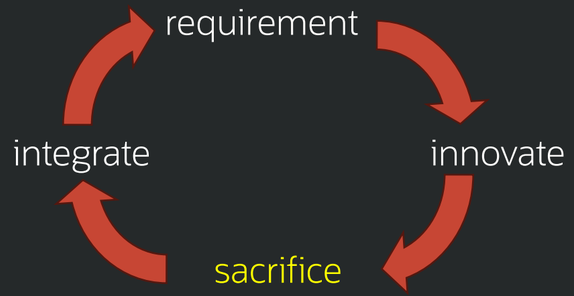

Weaviate



buying a house

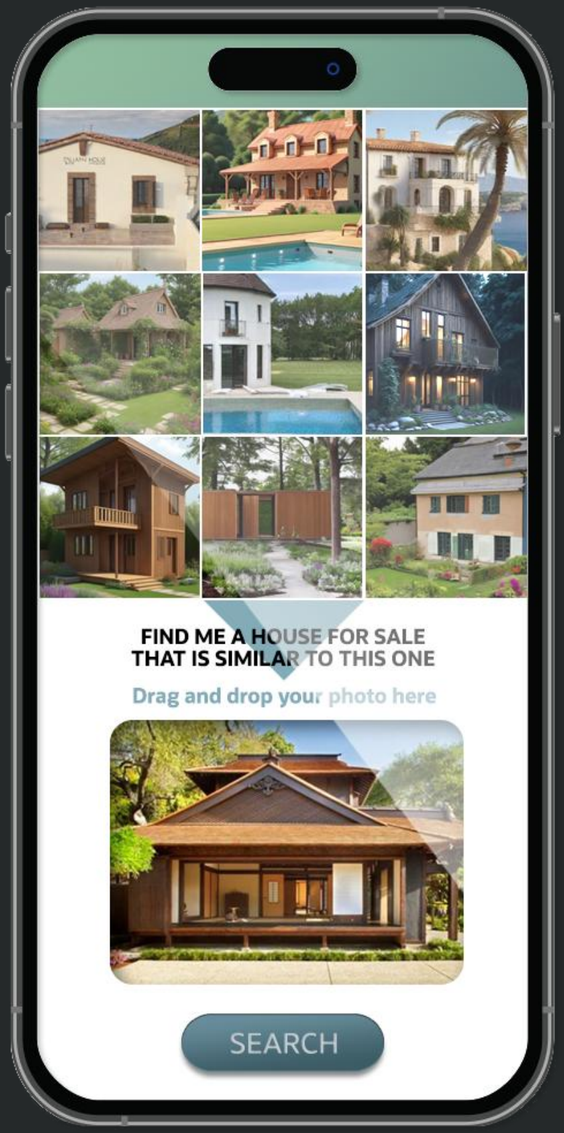
1) image similarity ✓

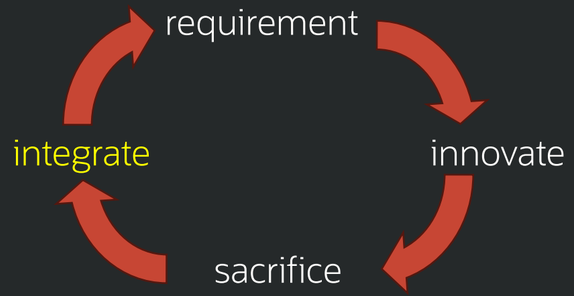


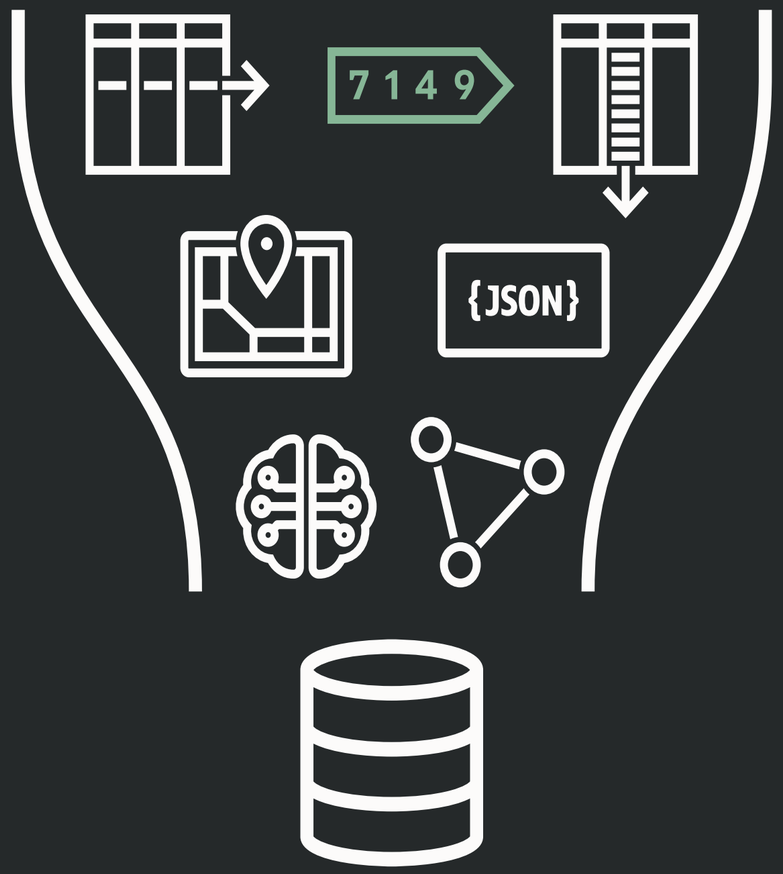


buying a house

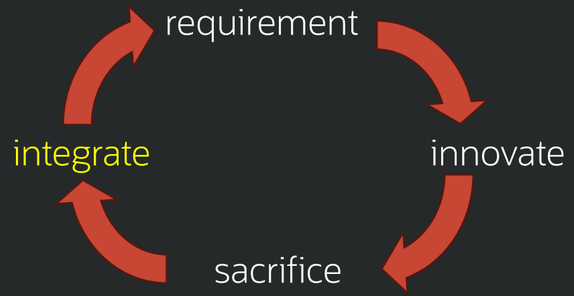
2) budget
crime stats
locality
schools
public transport







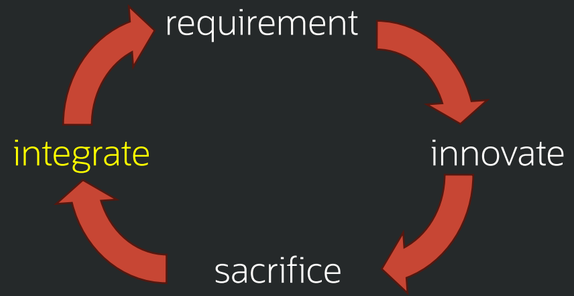
Converged Database



new datatype



```
CREATE TABLE house_for_sale
    (house_id      number,
     price         number,
     city          varchar2(400),
     house_photo   blob,
     house_vector VECTOR
);
```





convergence

```
SELECT …
FROM   house_for_sale,
       customer c
WHERE  price <= c.budget
AND    city  = c.city
ORDER BY vector_distance(house_vector,:myvector);
```

wrap up

are you convinced ?

McDonald's Conjecture:



Unless you are a hyperscaler…

"The only database you should ever use is when it has past the INTEGRATE mark"

in simpler terms…

don't buy a "feature product" ...

… they become a product feature

- AverageDB
- LintDB
- StormiDB
- BroDB
- Redict
- Tablespace
- ChromoDB
- SableDB
- TidesDB
- Dare-DB
- ScopeDB
- Tonbo
- HaystackDB
- SereneDB
- Valkey
- JameSQL
- SlateDB
- VelarixDB

history has shown …

a declarative language

SQL

logical vs physical

storage independence

data integrity / transactions

it's probably relational

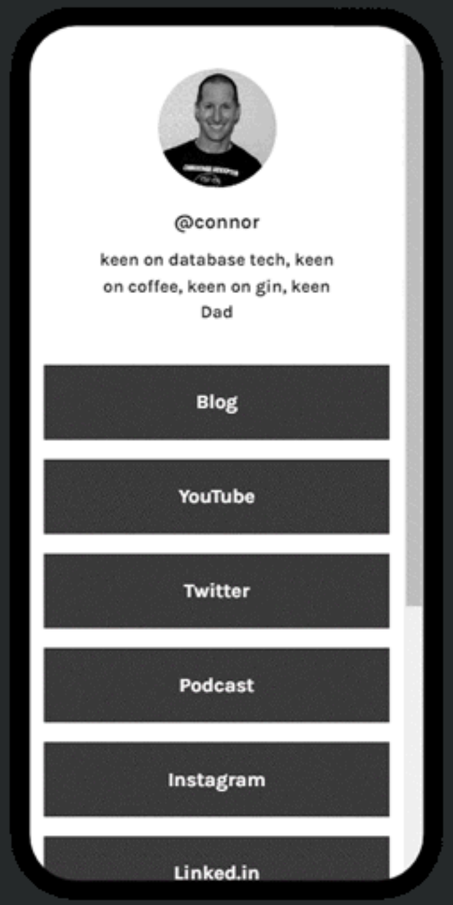
still not perfect

- configuration
- text search
- multi language
- any data format

still lots of decisions for you

exciting time to be database person

Stay in touch



@connor_mc_d

https://linktr.ee/connor