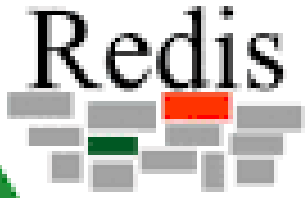
DATABASE INTRO

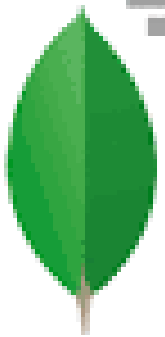
JOHN JERNIGAN 8/22/2023

So many databases...
Why?

PostgreSQL



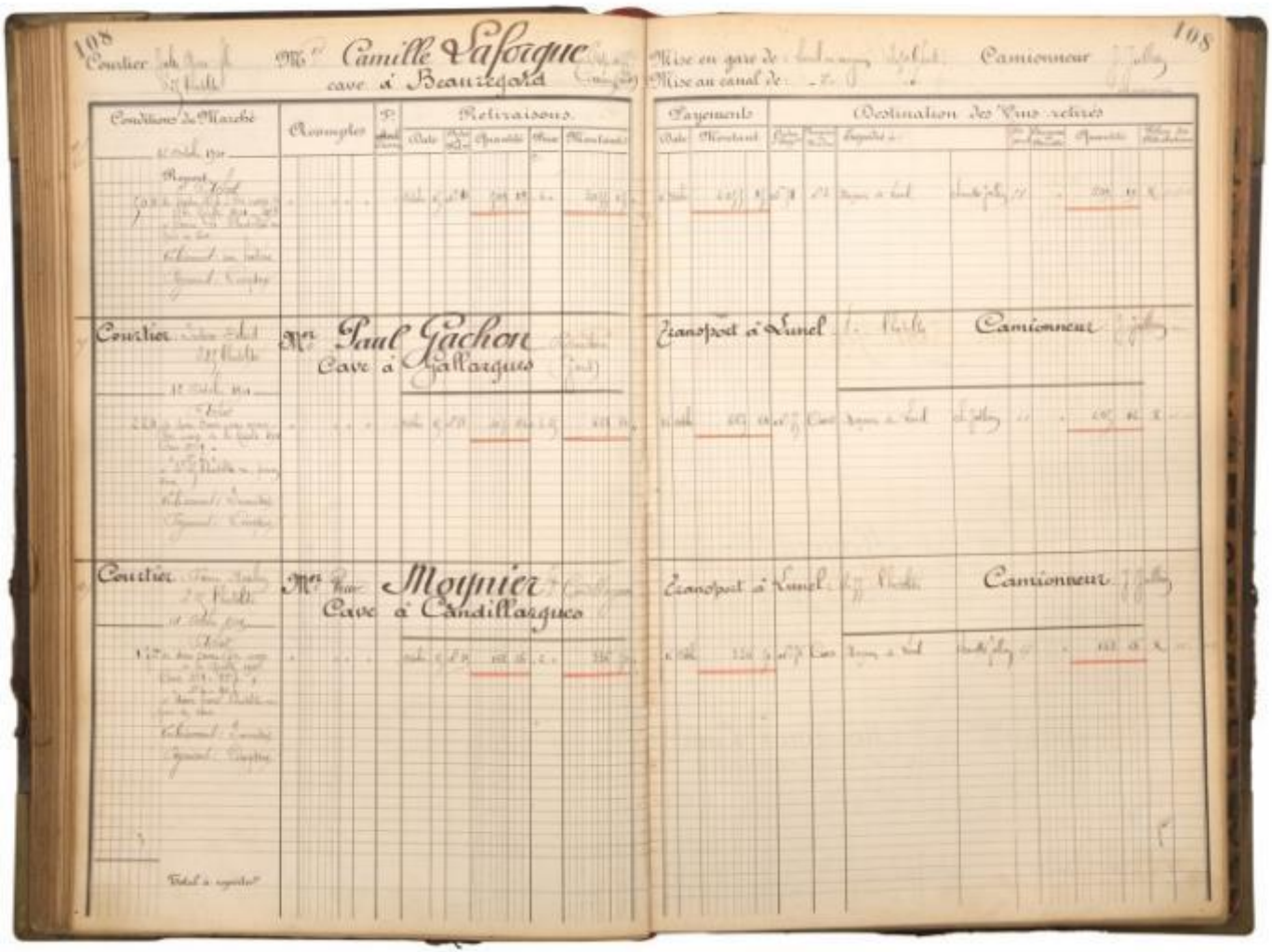
Cassandra



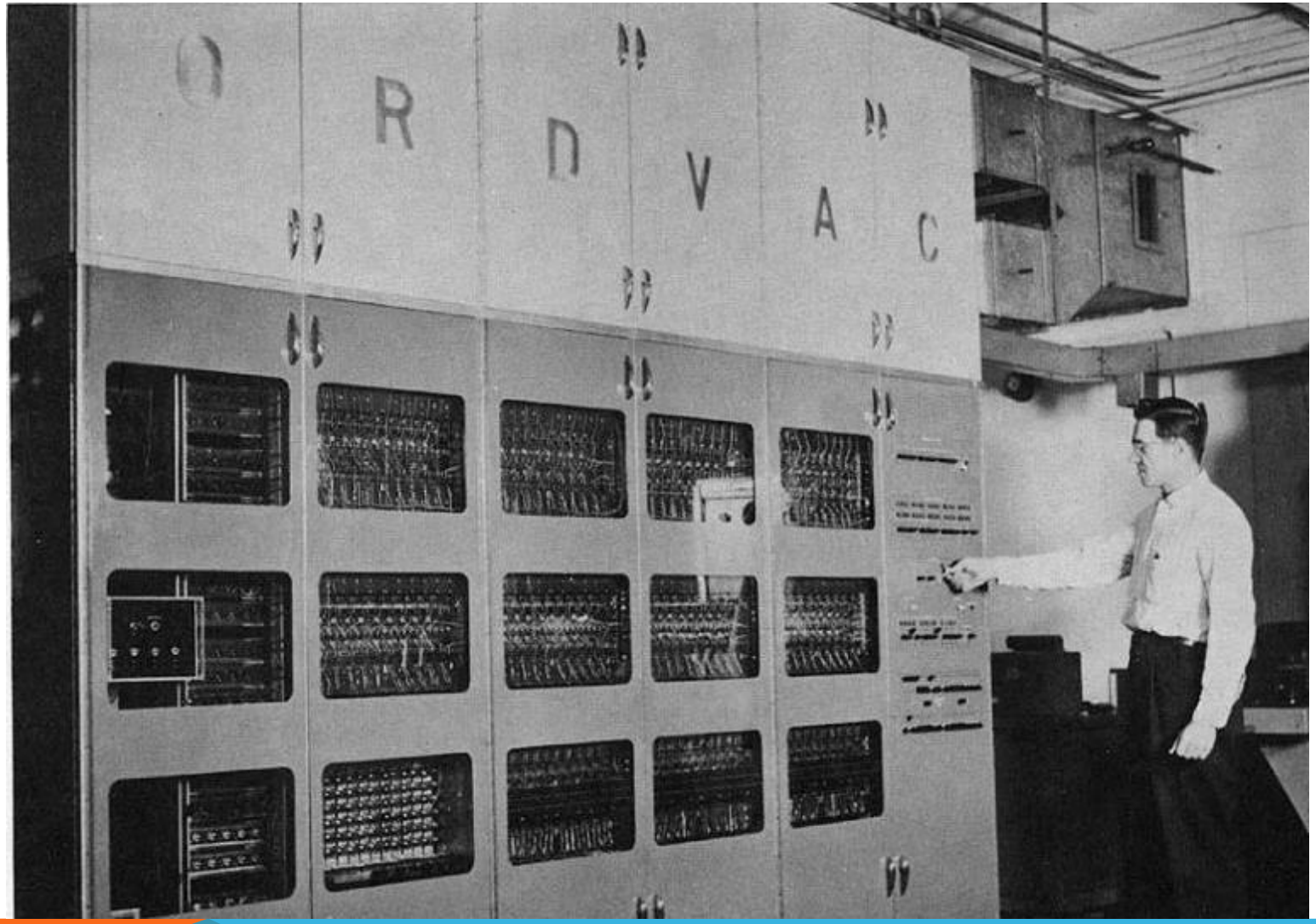
mongoDB

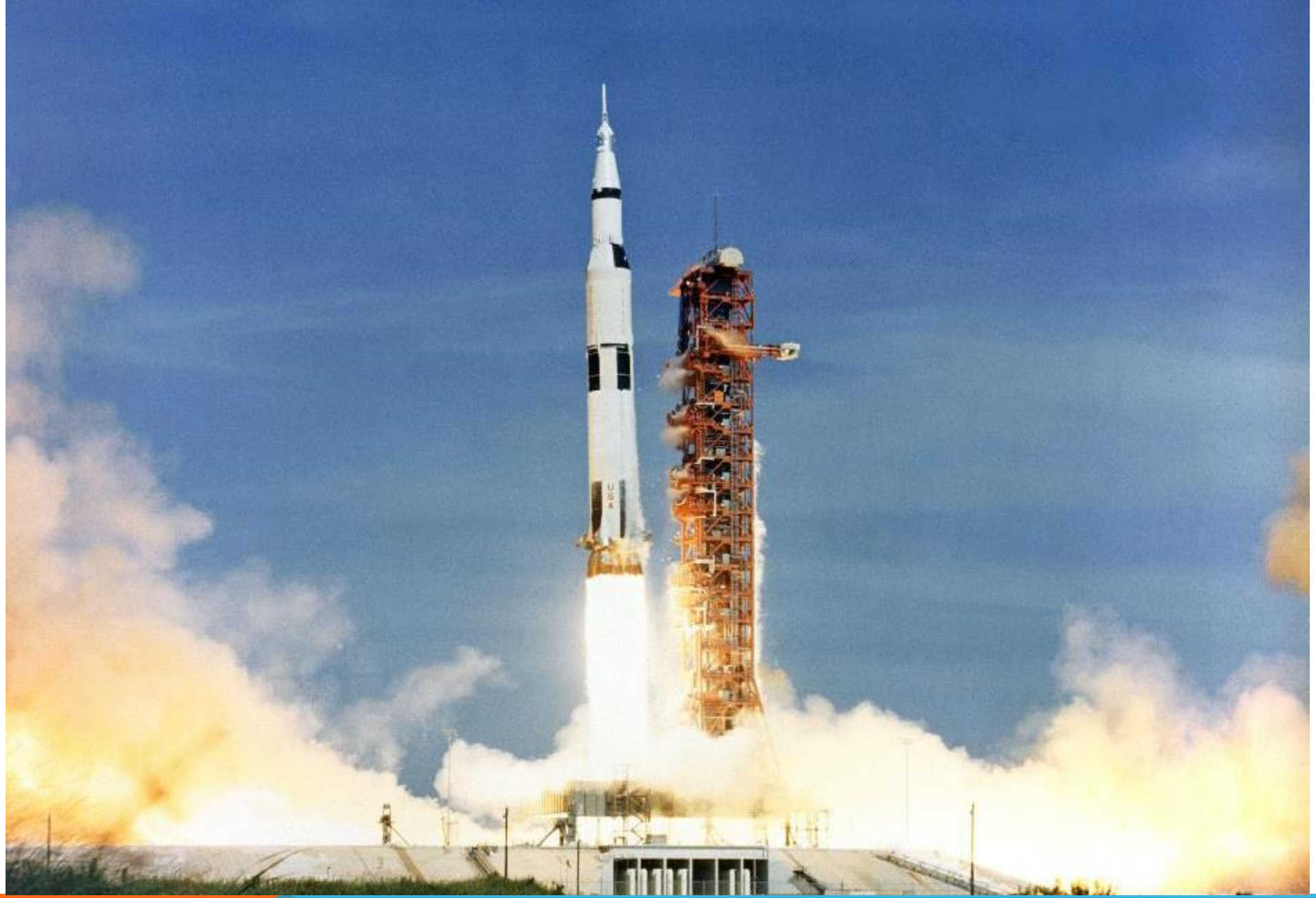
membase

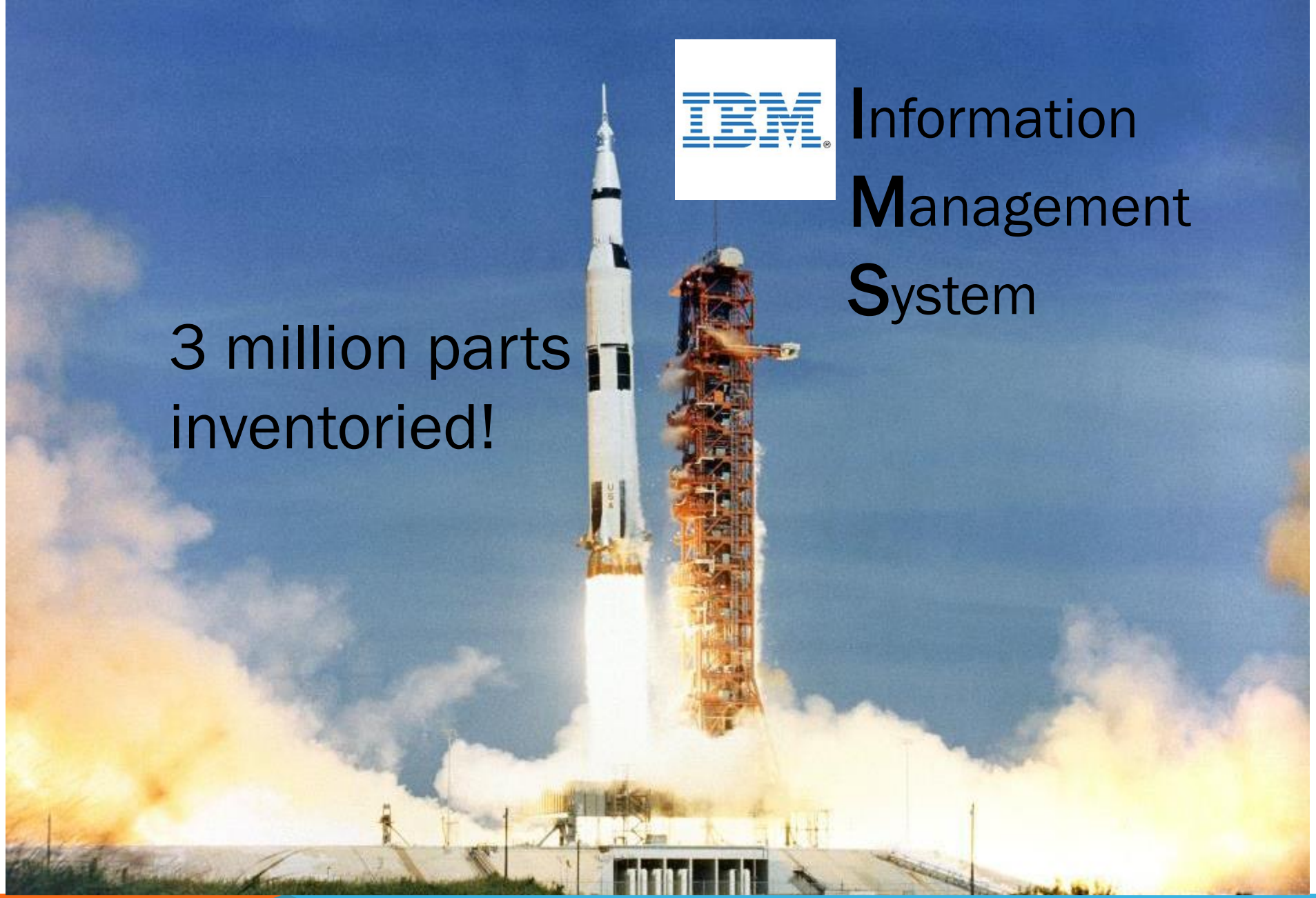
The spreadsheet has been a workhorse during the Industrial Revolution…



Computers offered data storage and computation…
Could the spreadsheet be electrified?

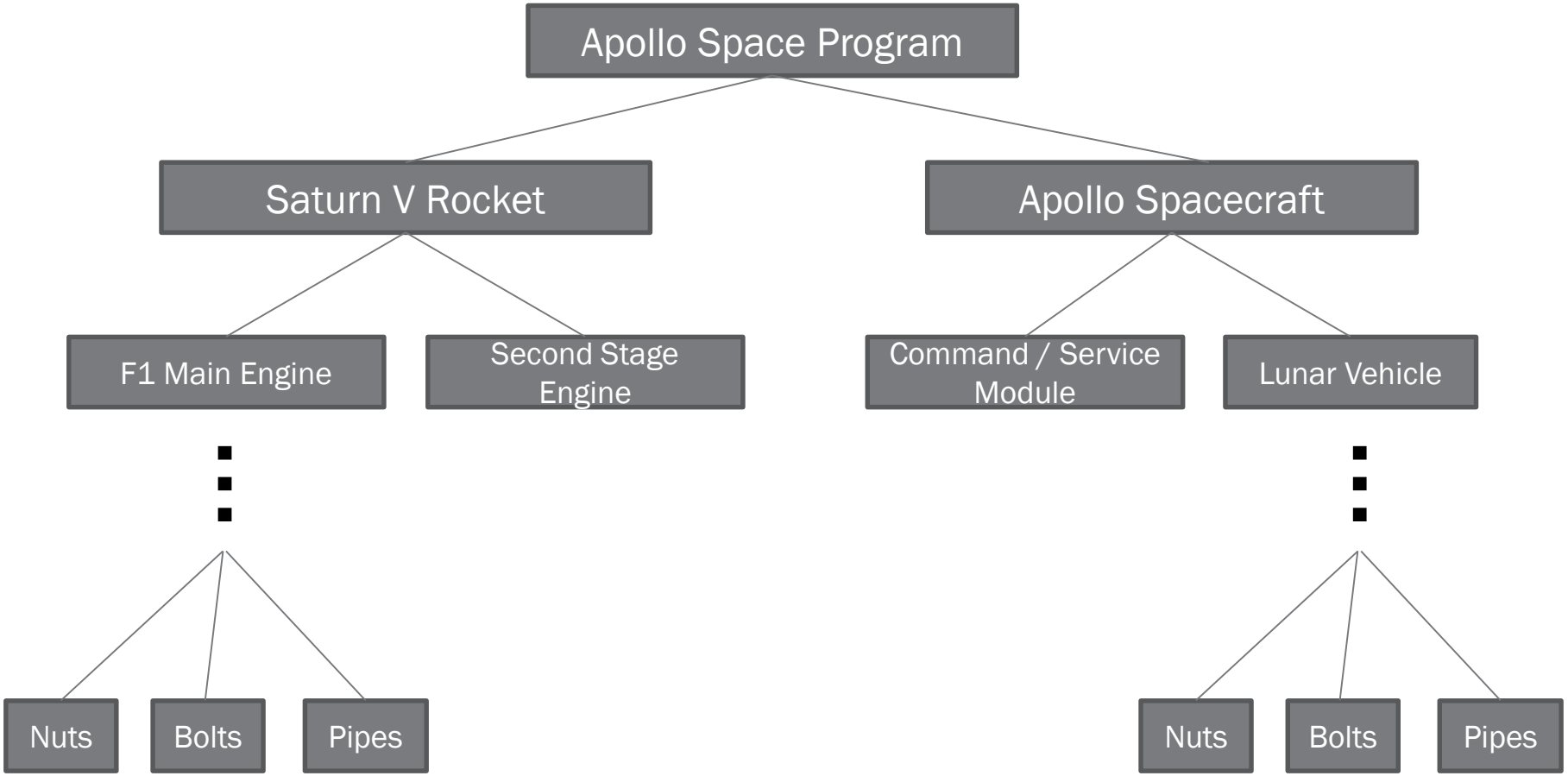






1966: First Database Management System (DBMS)

IMS had a hierarchical “tree” structure, internally



(This is not really how it was organized)

Tree structures are fast and efficient!

Hierarchical Database Disadvantages:

- Developers MUST understand internal database structure
- Which means you really don't want to fundamentally alter the database
- Because that will break your existing applications that use it

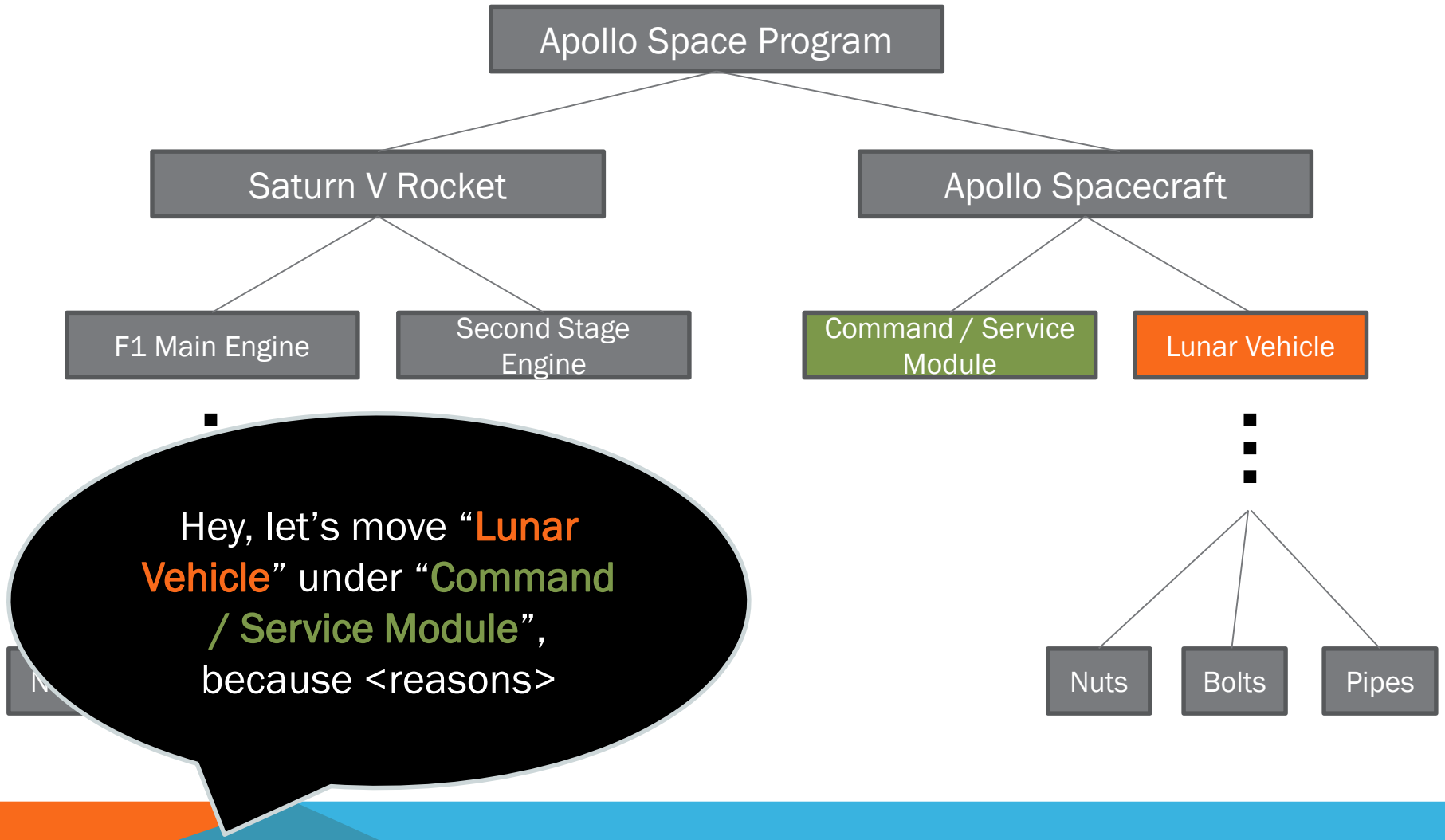
Hierarchical Database Disadvantages:

- Developers MUST understand internal database structure
- Which means you really don't want to fundamentally alter the database
- Because that will break your existing applications that use it

Hierarchical Database Disadvantages:

- Developers MUST understand internal database structure
- Which means you really don't want to fundamentally alter the database
- Because that will break your existing applications that already use it

You Really Don't Want to Fundamentally Alter a Hierarchical Database



But *what if* there were a database model…

But *what if* there were a database model…

…where expensive database applications *continued to work…*

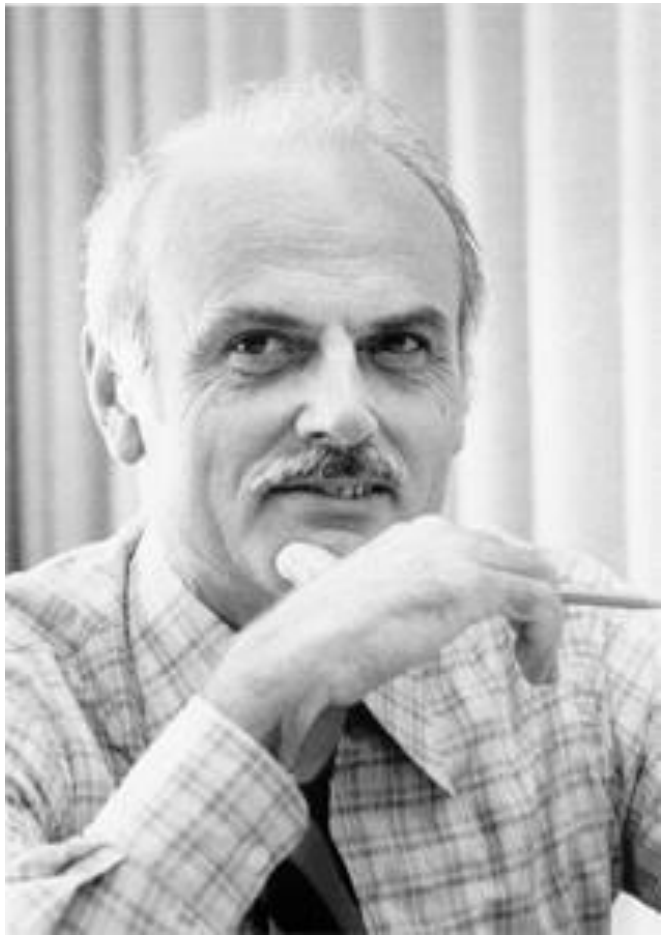
But *what if* there were a database model…

…where expensive database applications *continued to work…*

…even if the internal database structure were altered?

In 1970, Dr. Edgar Codd invents:

"*Relational* model for database management"



Information Retrieval

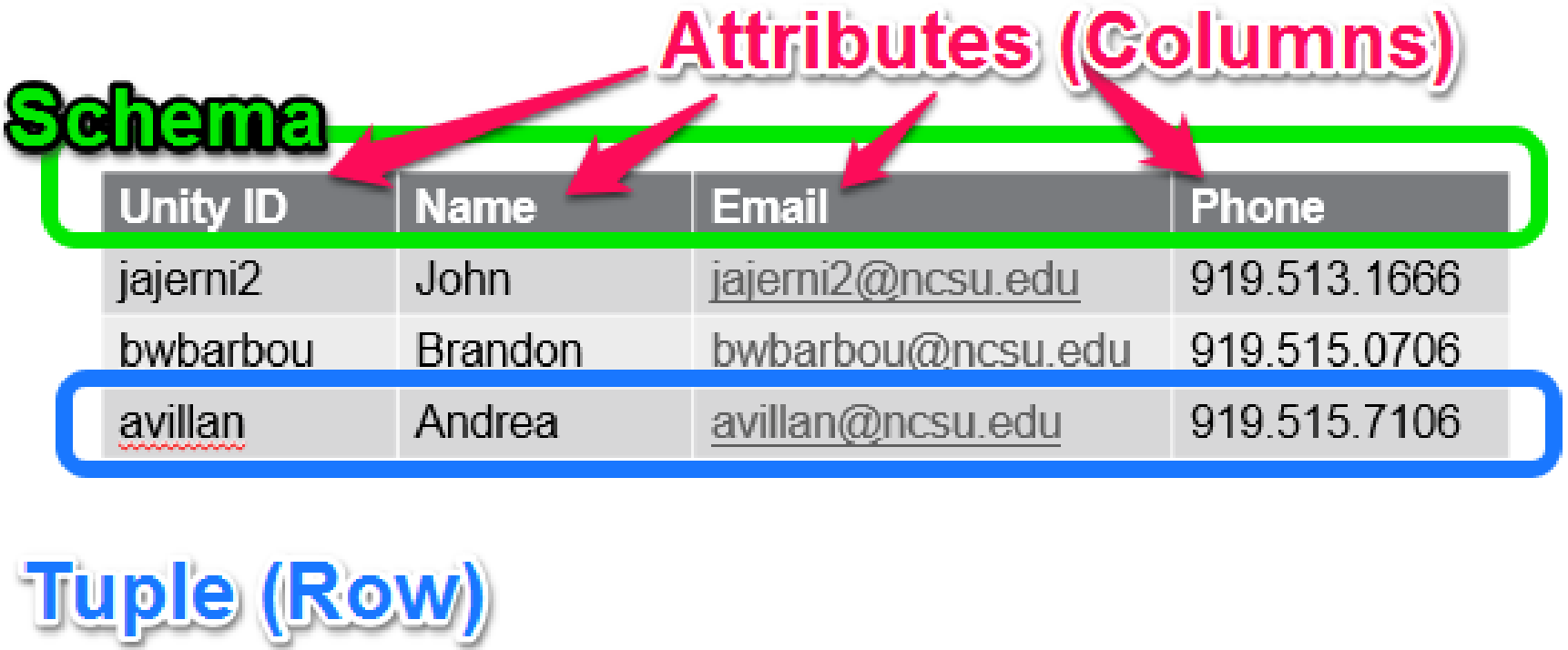
A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation

Communications of the ACM
Volume 13 Issue 6, June 1970

Relational Databases Use Table Structures, Not Trees



Unity ID	Name	Email	Phone
jajerni2	John	jajerni2@ncsu.edu	919.513.1666
bwbarbou	Brandon	bwbarbou@ncsu.edu	919.515.0706
avillan	Andrea	avillan@ncsu.edu	919.515.7106

Use Structured Query Language (SQL) to Interface with Database

Unity ID	Name	Email	Phone
jajerni2	John	jajerni2@ncsu.edu	919.513.1666
bwbarbou	Brandon	bwbarbou@ncsu.edu	919.515.0706

mysql> SELECT Name,Email FROM staff WHERE ‘Unity ID’=‘jajerni2’;

```
+---------+--------------------+
| Name    | Email              |
+---------+--------------------+
| John    | jajerni2@ncsu.edu  |
+---------+--------------------+
```

Creating the First Relational Databases…

Creating the First Relational Databases…

Ed Oates Bob Miner Larry Ellison

Creating the First Relational Databases…

Creating the First Relational Databases…

Some Top Relational Database Management Systems

- Oracle
- Microsoft SQL Server
- MySQL (free, open-source; see: MariaDB)
- Postgres (free, open-source)

Some Top Relational Database Management Systems

- Oracle
- Microsoft SQL Server
- MySQL (free, open-source; see: MariaDB)
- Postgres (free, open-source)

Some Top Relational Database Management Systems

- Oracle
- Microsoft SQL Server
- MySQL (free, open-source; see: MariaDB)
- Postgres (free, open-source)

Some Top Relational Database Management Systems

- Oracle
- Microsoft SQL Server
- MySQL (free, open-source; see: MariaDB)
- PostgreSQL (free, open-source)

Special Mention of Popular Database:

- Stripped-down
- Bare-essentials
- Still powerful (also: free)

What's Wrong with Relational Databases?

What's Wrong with Relational Databases?

- Not much

What's Wrong with Relational Databases?

- Not much
- Until you start dealing with Big Data

What's Wrong with a Honda Fit?

- Not much
- Until you start dealing with Big Cargo

What's Wrong with a Honda Fit?

- Not much
- Until you start dealing with Big Cargo

CLASSIC "BIG DATA" DEFINITION

What Can Go Wrong With Relational Databases:

Big Data

Database Too Slow?

Scale Resources Vertically

What Can Go Wrong With Relational Databases:

Poor Performance

Database Too Slow?

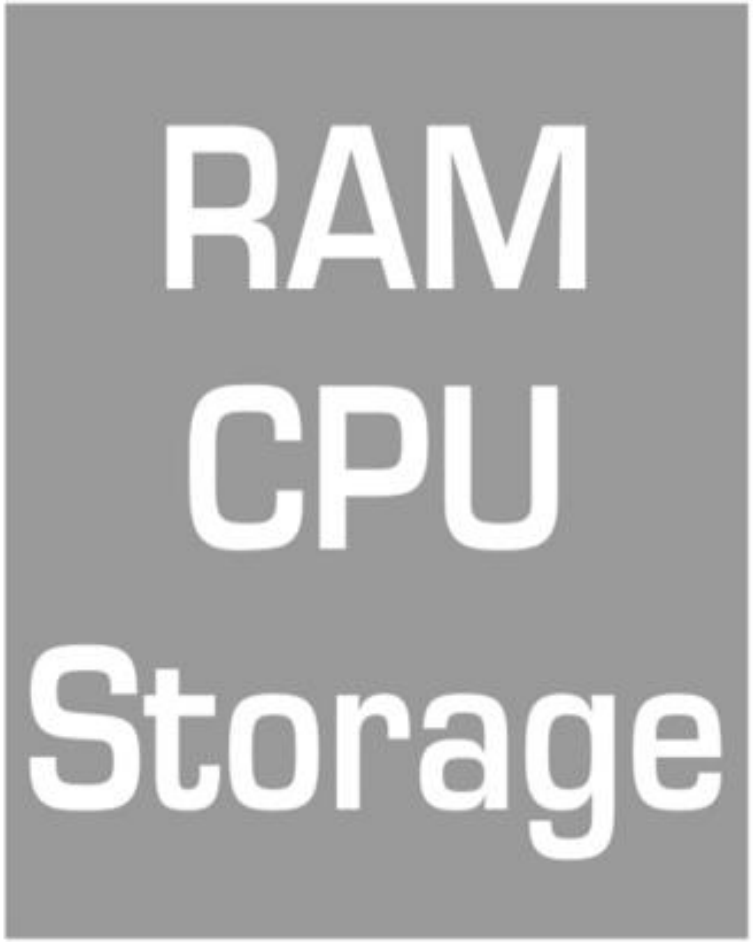
Scale Resources Vertically

What Can Go Wrong With Relational Databases:

Poor Performance

Database Too Slow?

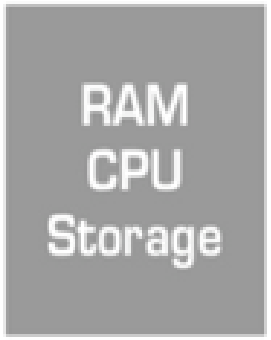
Scale Resources Vertically



What Can Go Wrong With Relational Databases:

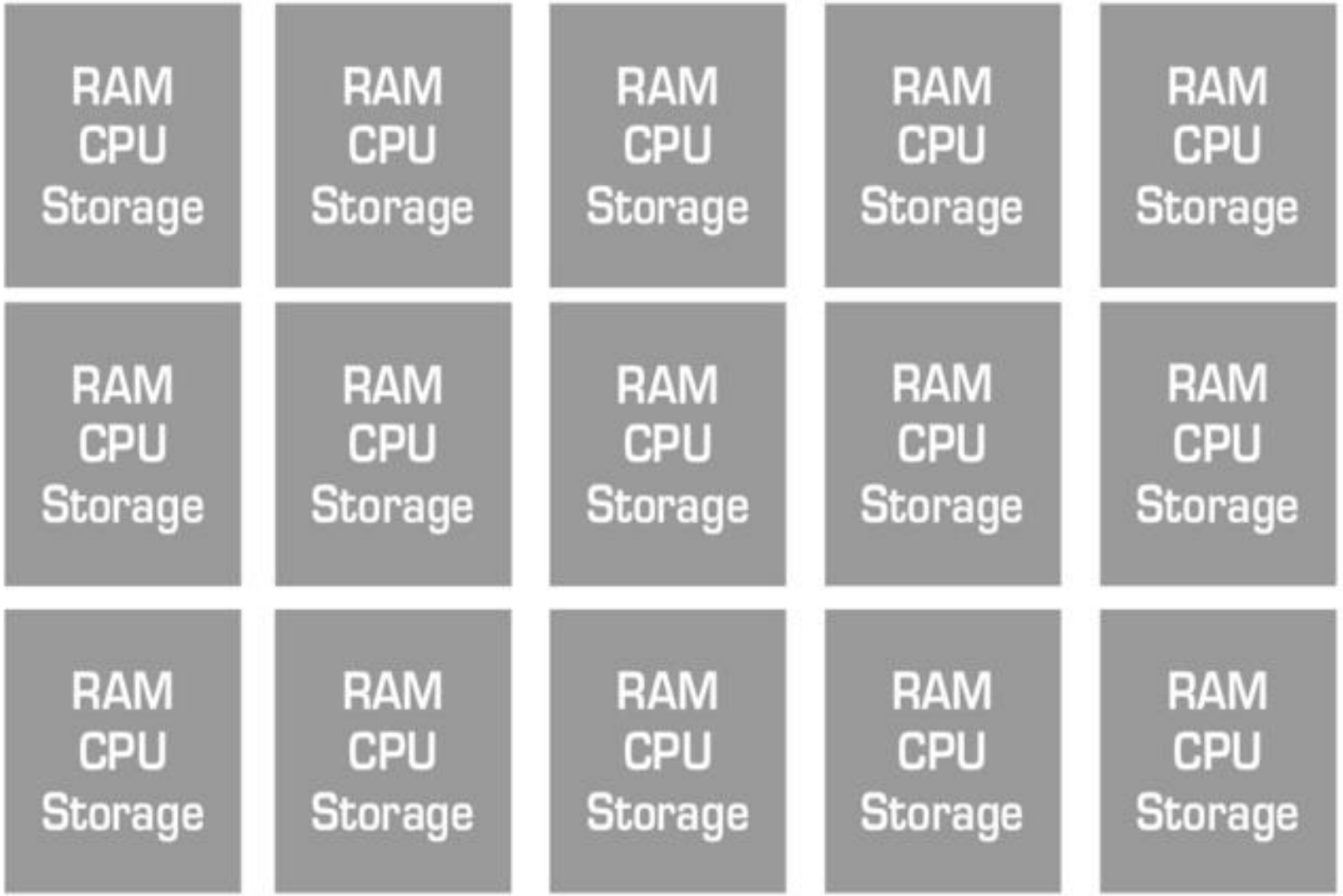
Poor Performance

Or … Scale Resources Horizontally



Or … Scale Resources Horizontally

Or … Scale Resources Horizontally



What Can Go Wrong With Relational Databases:

Poor Performance

Scaling Vertically: Limited by physics, and state-of-the-art

Scaling Horizontally: Works! But … presents new problems

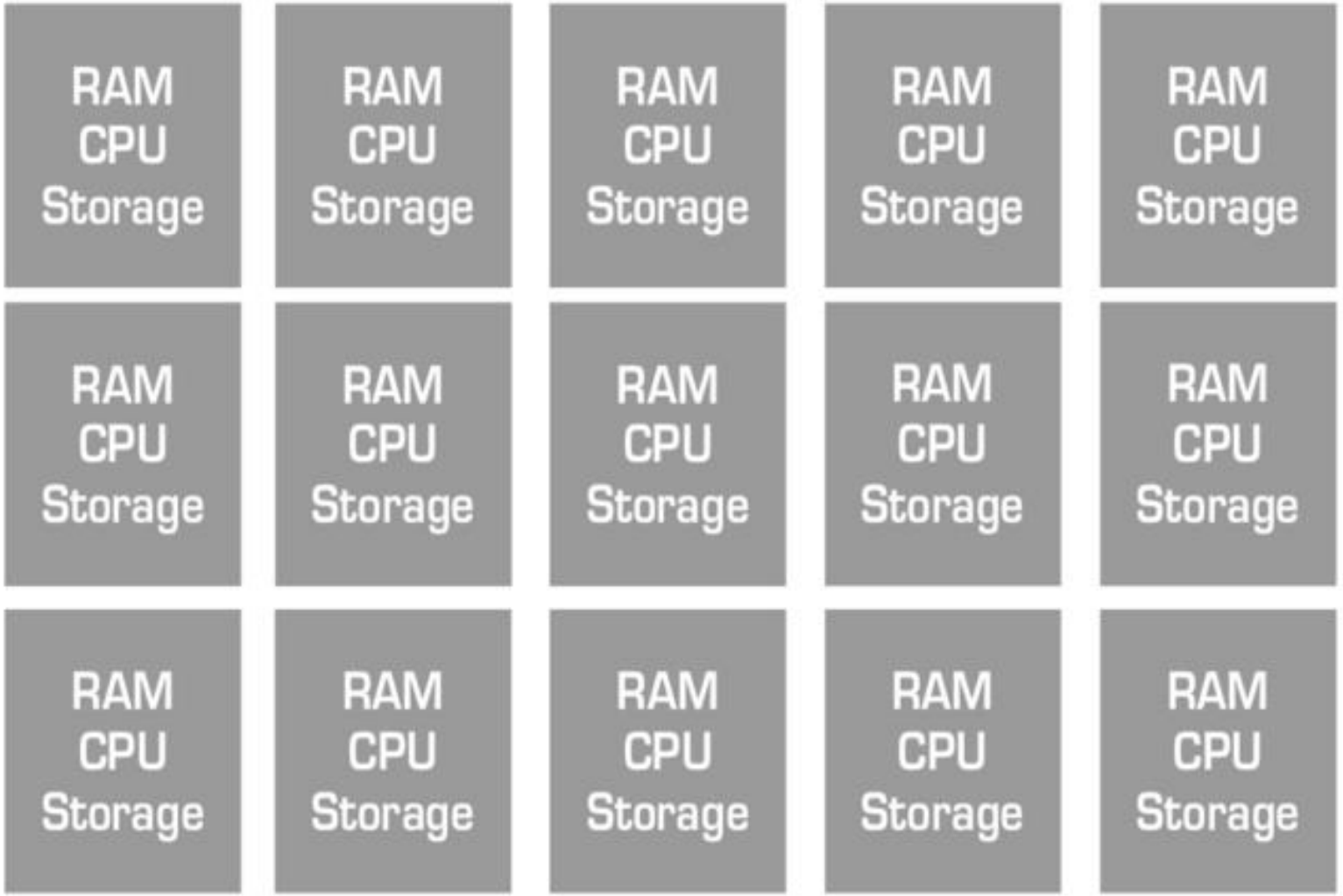
What Can Go Wrong With Relational Databases:

Poor Performance

Scaling Vertically: Limited by physics, and state-of-the-art

Scaling Horizontally: Works! But … presents new problems

Consider the synchronization issues when updating the database simultaneously on multiple nodes



What Can Go Wrong With Relational Databases:
Poor Performance

Engineers attempted to build **better performing** databases.

They were called…

NoSQL - A misleading term

Have you heard the term NoSQL?

NoSQL - A misleading term

NoSQL?

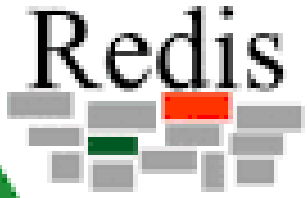
It *seems* to mean
“not a relational database”

NoSQL - A misleading term

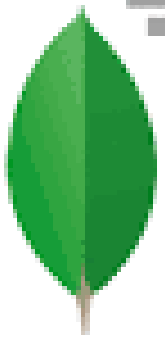
A better way of interpreting it:

So many databases...
Why?

PostgreSQL



Cassandra

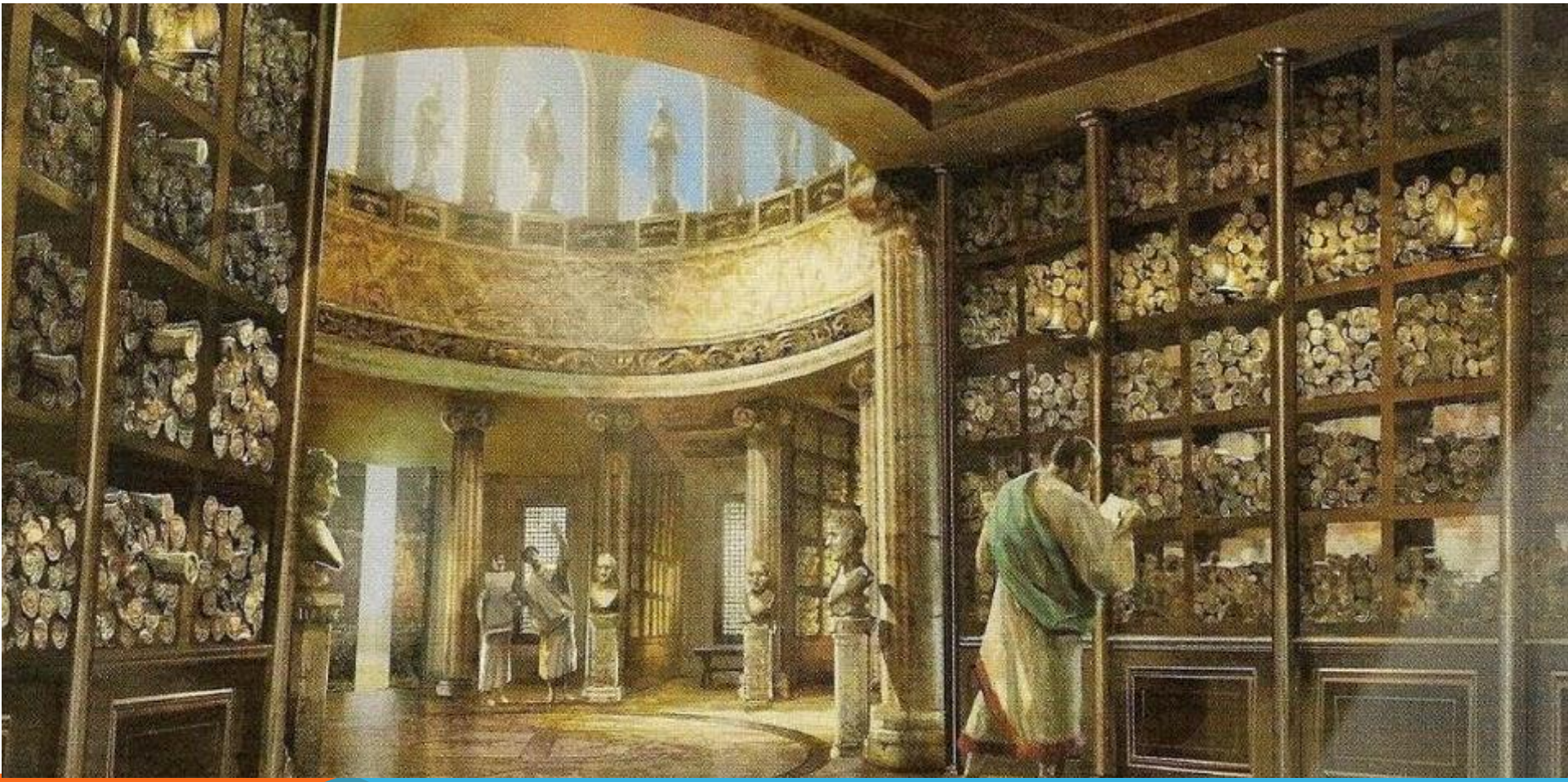


mongoDB

membase

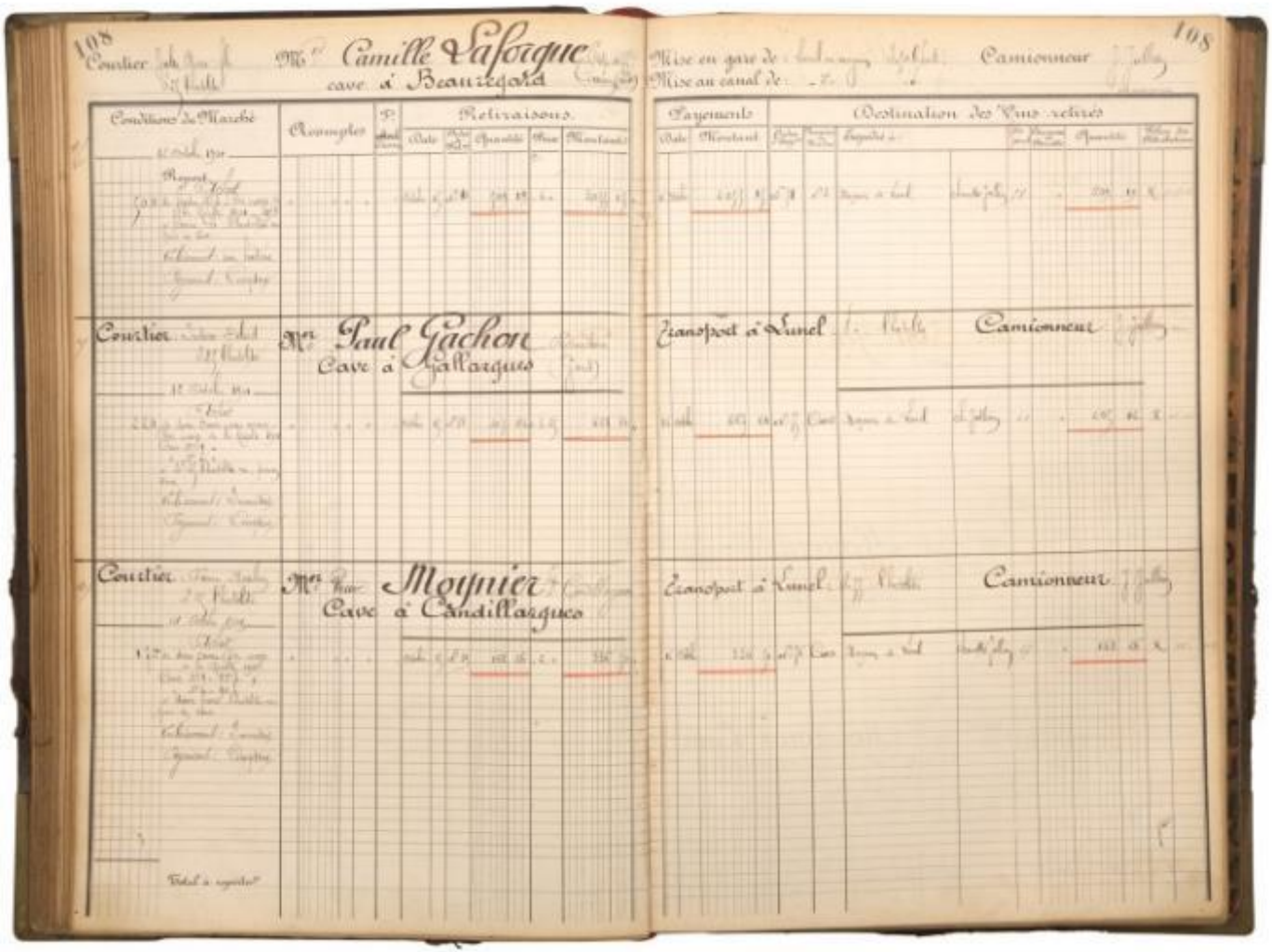
Library of Alexandria

First use of alphabetical ordering?

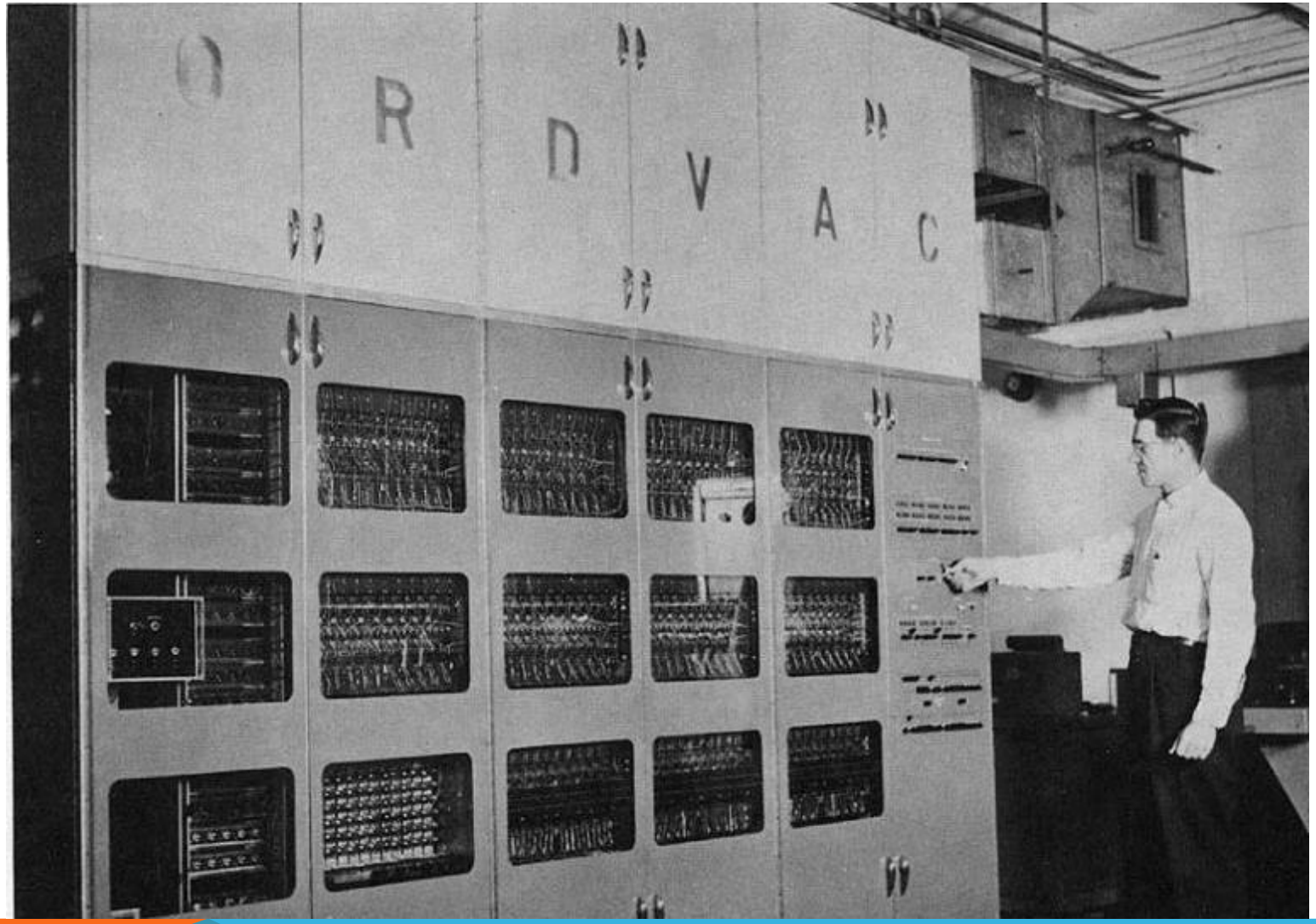


Casson, L. (2001). Libraries in the ancient world. ProQuest Ebook Central (pg 37)

The spreadsheet has been a workhorse during the Industrial Revolution…



Computers offered data storage and computation…
Could the spreadsheet be electrified?



First Relational Database Management System (ca. 1979)

What's Wrong with Relational Databases?

- Not much
- Until you start dealing with Big Data

NoSQL - A misleading term

A better way of interpreting it:

NoSQL - A misleading term

Think: Solving Big Data database challenges with *application-specific solutions*.

NoSQL - A misleading term

Think: Solving Big Data database challenges with *application-specific solutions*.

Working with big JSON data?
Try:

CouchDB
relax

NoSQL - A misleading term

Think: Solving Big Data database challenges with *application-specific solutions*.

Working with big key-value pairs?
Try:

redis

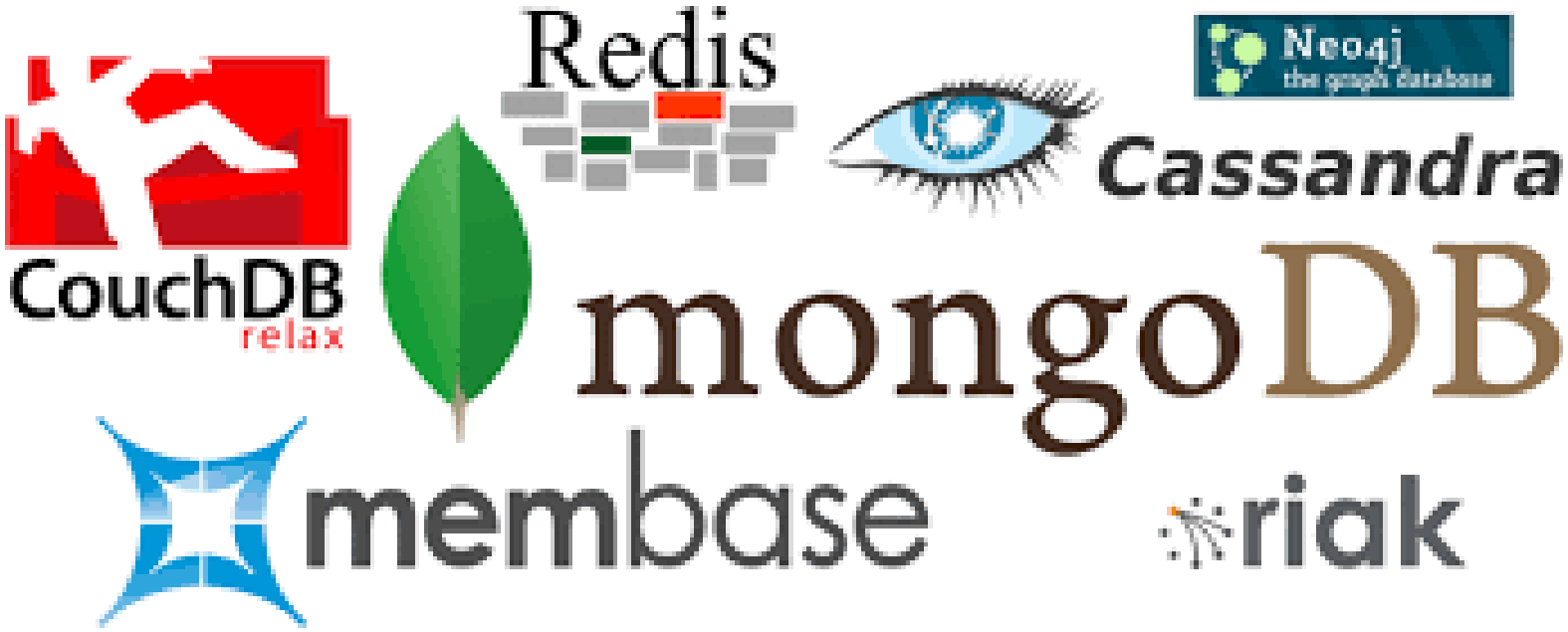
NoSQL - A misleading term

Think: Solving Big Data database challenges with *application-specific solutions*.

Working with big graph data?
Try:

NoSQL - A misleading term

Think: Solving Big Data database challenges with *application-specific solutions*.



Recap: The Big Picture of Relational Databases and NoSQL

Recap: The Big Picture of Relational Databases and NoSQL

IBM IMS → **Relational** Database Management Systems (RDBMS) → **NoSQL** Database Management Systems

Recap: The Big Picture of Relational Databases and NoSQL

Gartner "Strategic Planning Assumption":

By 2017, all leading operational DBMSs will offer multiple data models, relational and NoSQL, in a single DBMS platform.

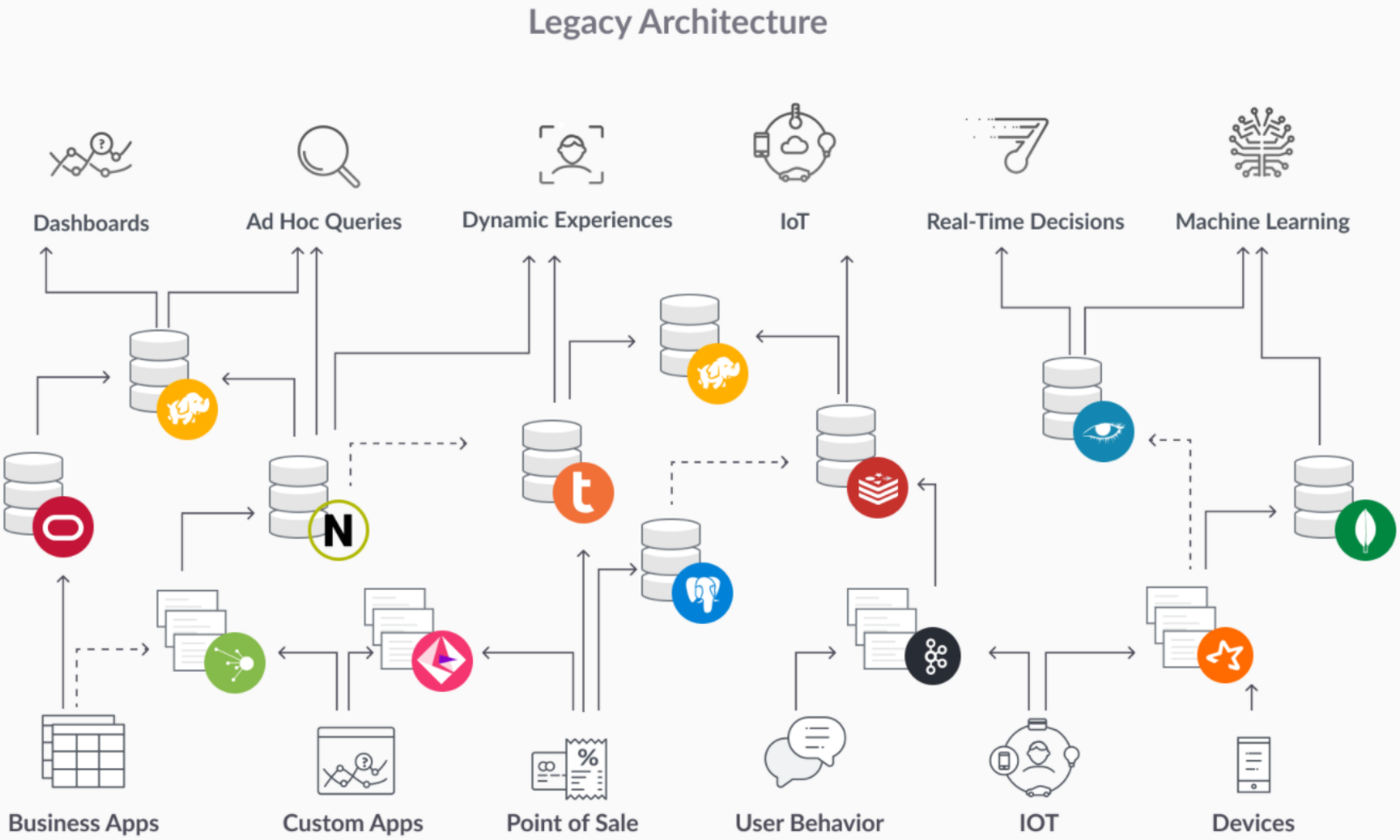
https://www.gartner.com/doc/reprints?id=1-2PMFPEN&ct=151013&st=sb

RDBMS

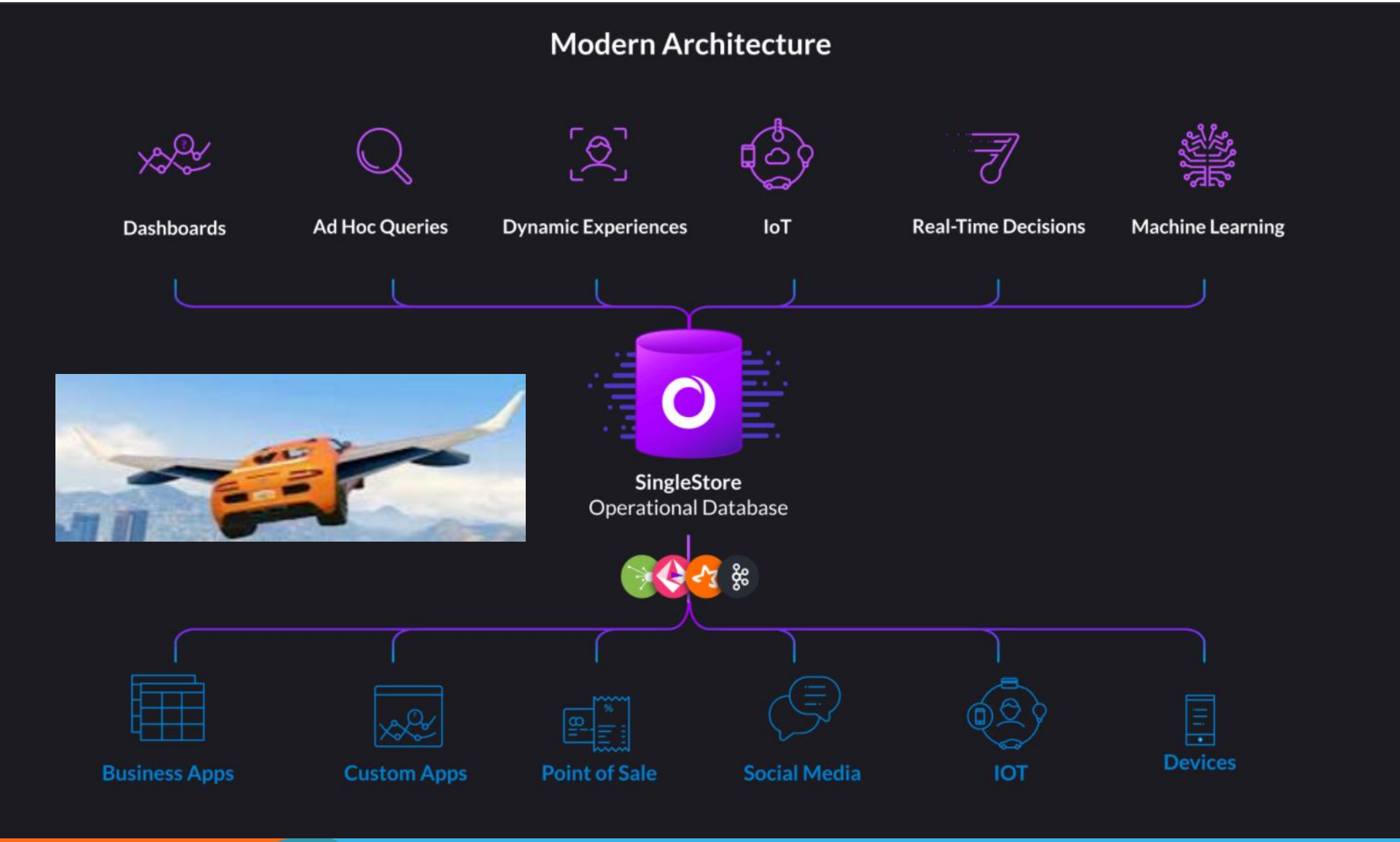
NoSQL

RDBMS + NoSQL =
Flying Car

Trend is moving away from all these different database solutions…



Instead, seeing data storage solutions try to “do it all”



PRACTICUM TIP: USE INDEXING

- Indexes are created *per-column (attribute) and you can index multiple (or all) columns in a table*.
- Indexes can hugely speed up your queries!
- Index updates are *performance overhead when data is added to database* which is a tradeoff to consider. Data scientists are rarely adding data, however, so USE INDEXES OFTEN, ON MANY COLUMNS.

Questions

PostgresQL Demo

Never had any hands-on experience writing SQL code against a simple database?

Try this browser-based Postgres database with some sample data loaded:

- https://www.crunchydata.com/developers/playground?sql=https://gist.githubusercontent.com/jaidetree/11cce77331a82bcab52563d1f63a9c46/raw/a0f0650c2655bc9923fd1a98fc2b888f6df2beb3/create-births-table.sql
- List the tables available:
  - \dt (think: display tables)

```
postgres=# \dt
                List of relations
 Schema |          Name          | Type  |  Owner
--------+------------------------+-------+----------
 public | us_births_20002014_ssa | table | postgres
(1 row)
```

This table contains US number of births for each day of the year between 2000 – 2014, provided by the Social Security Administration.

PostgresQL Demo

- Describe the table schema (i.e. tell me what columns it has)
  - \d us_births_20002014_ssa (think: describe us_births_20002014_ssa)

```
postgres=# \d us_births_20002014_ssa
                              Table "public.us_births_20002014_ssa"
    Column     |  Type   | Collation | Nullable |                       Default
---------------+---------+-----------+----------+-------------------------------------------------
 id            | integer |           | not null | nextval('us_births_20002014_ssa_id_seq'::regclass)
 year          | integer |           |          |
 month         | integer |           |          |
 date_of_month | integer |           |          |
 day_of_week   | integer |           |          |
 births        | integer |           |          |
Indexes:
    "us_births_20002014_ssa_pkey" PRIMARY KEY, btree (id)
```

Hint: use the Tab key to auto-complete the table name instead of typing it in every command. Type "\d us" and then hit Tab.

PostgresQL Demo

- **Display the first 10 rows of data**
  - SELECT * FROM us_births_20002014_ssa LIMIT 10;

```
postgres=# SELECT * FROM us_births_20002014_ssa LIMIT 10;
 id | year | month | date_of_month | day_of_week | births
----+------+-------+---------------+-------------+--------
  1 | 2000 |     1 |             1 |           6 |   9083
  2 | 2000 |     1 |             2 |           7 |   8006
  3 | 2000 |     1 |             3 |           1 |  11363
  4 | 2000 |     1 |             4 |           2 |  13032
  5 | 2000 |     1 |             5 |           3 |  12558
  6 | 2000 |     1 |             6 |           4 |  12466
  7 | 2000 |     1 |             7 |           5 |  12516
  8 | 2000 |     1 |             8 |           6 |   8934
  9 | 2000 |     1 |             9 |           7 |   7949
 10 | 2000 |     1 |            10 |           1 |  11668
(10 rows)
```

Looks like fewer births on Saturday and Sunday (day_of_week 6 and 7)…

Hint: use the Tab key to auto-complete the table name instead of typing it in every command. Type “us” and then hit Tab.

PostgresQL Demo

- Some other ideas…
  - Count how many rows of data there are:
    - SELECT count(births) FROM us_births_20002014_ssa;
  - Find out what year is the latest year in the data:
    - SELECT MAX(year) FROM us_births_20002014_ssa;
  - Find the average number of births on Sunday across all data:
    - SELECT AVG(births) FROM us_births_20002014_ssa WHERE day_of_week='7';
  - Find the average number of births on Monday across all data:
    - SELECT AVG(births) FROM us_births_20002014_ssa WHERE day_of_week='1';

Interesting…fewer births on the weekend consistently over 14 years…

PostgresQL Demo

- This example uses one single table of data, but in the real world, your data is probably spread over many tables. Learning how to link the tables together with unique identifiers and join the partial results into a large resultant table takes skill and practice.

A Short Review of This Material:

https://www.oreilly.com/library/view/sql-pocket-guide/9781492090397/ch01.html

Chapter 1: