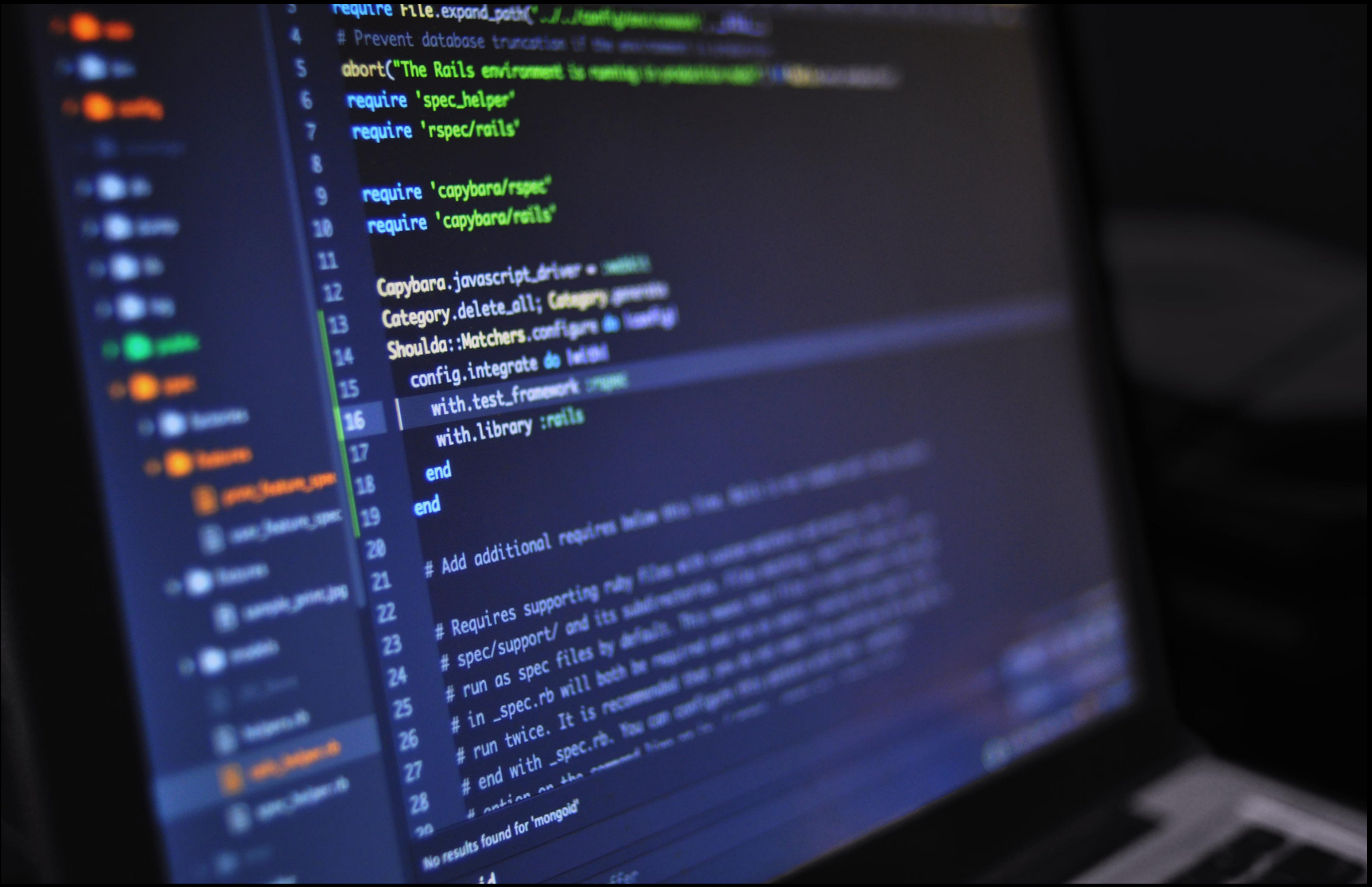
Role of DBMS in Advancing AI

: A New Era of Innovation

By

Ms. Dyuti Lal

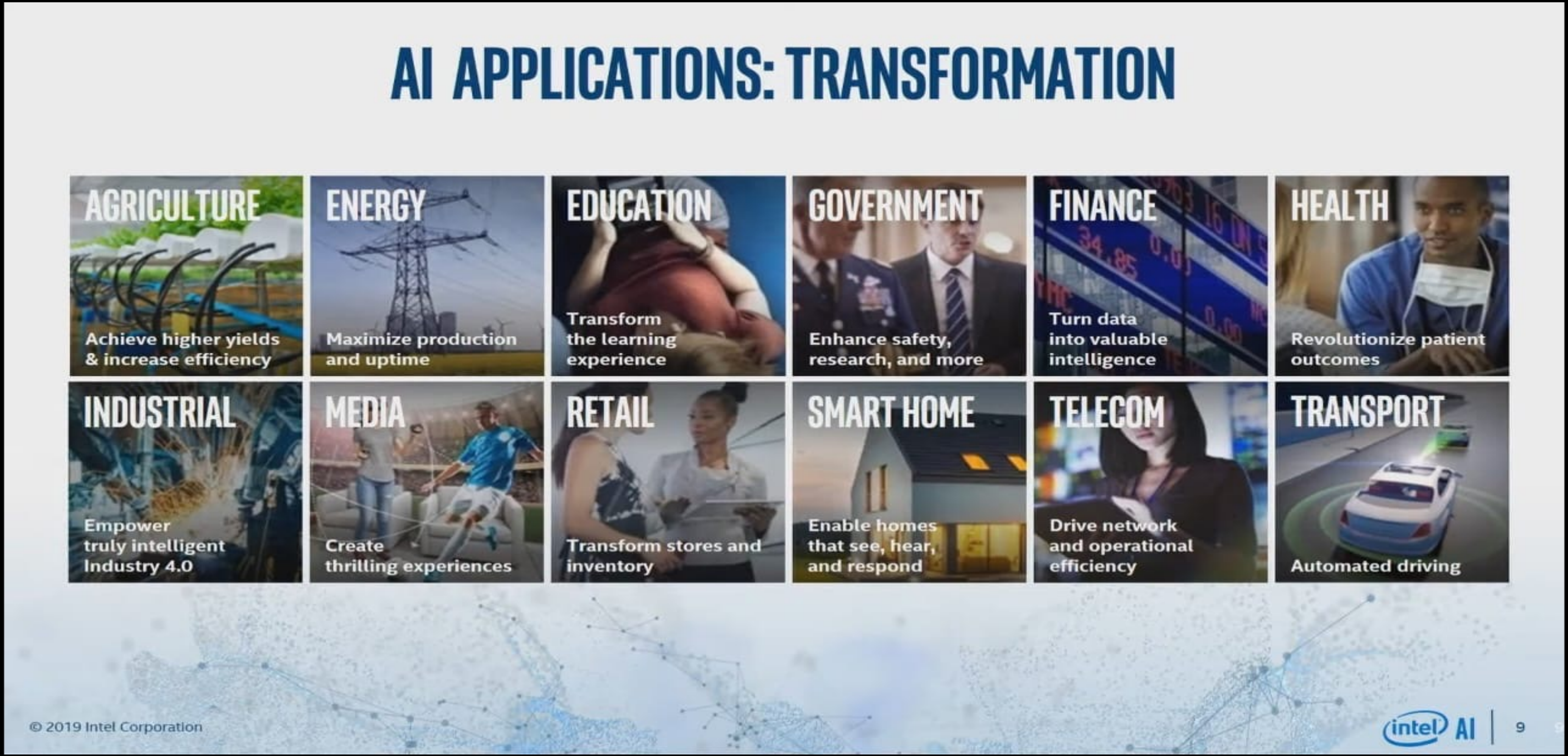
Introduction



```
# Prevent database
abort("The Rails environment
require 'spec_helper'
require 'rspec/rails'

require 'capybara/rspec'
require 'capybara/rails'

Capybara.javascript_driver
Category.delete_all;
Shoulda::Matchers.
  config.integrate
    with.test_framework
    with.library :rails
  end
end

# Add additional requires
```

Open-source databases are becoming increasingly popular as a means to power AI. This trend is creating a new era of innovation and is transforming the way we think about data management. In this presentation, we will explore the benefits of using open-source databases to power AI and discuss some of the most exciting developments in this field.

Domains in which AI is applied



AI APPLICATIONS: TRANSFORMATION

AGRICULTURE
Achieve higher yields & increase efficiency

ENERGY
Maximize production and uptime

EDUCATION
Transform the learning experience

GOVERNMENT
Enhance safety, research, and more

FINANCE
Turn data into valuable intelligence

HEALTH
Revolutionize patient outcomes

INDUSTRIAL
Empower truly intelligent Industry 4.0

MEDIA
Create thrilling experiences

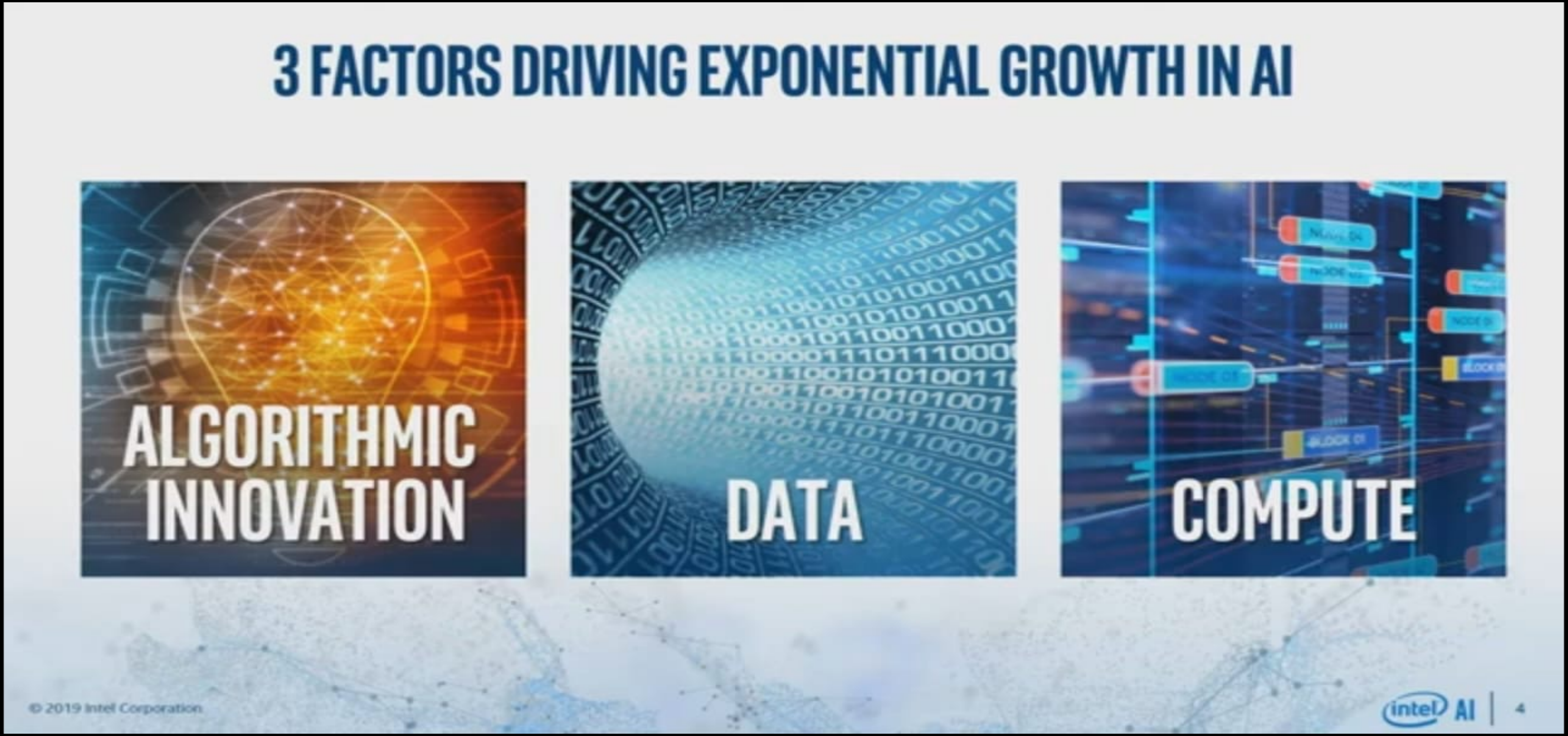
RETAIL
Transform stores and inventory

SMART HOME
Enable homes that see, hear, and respond

TELECOM
Drive network and operational efficiency

TRANSPORT
Automated driving

3 Major Factors



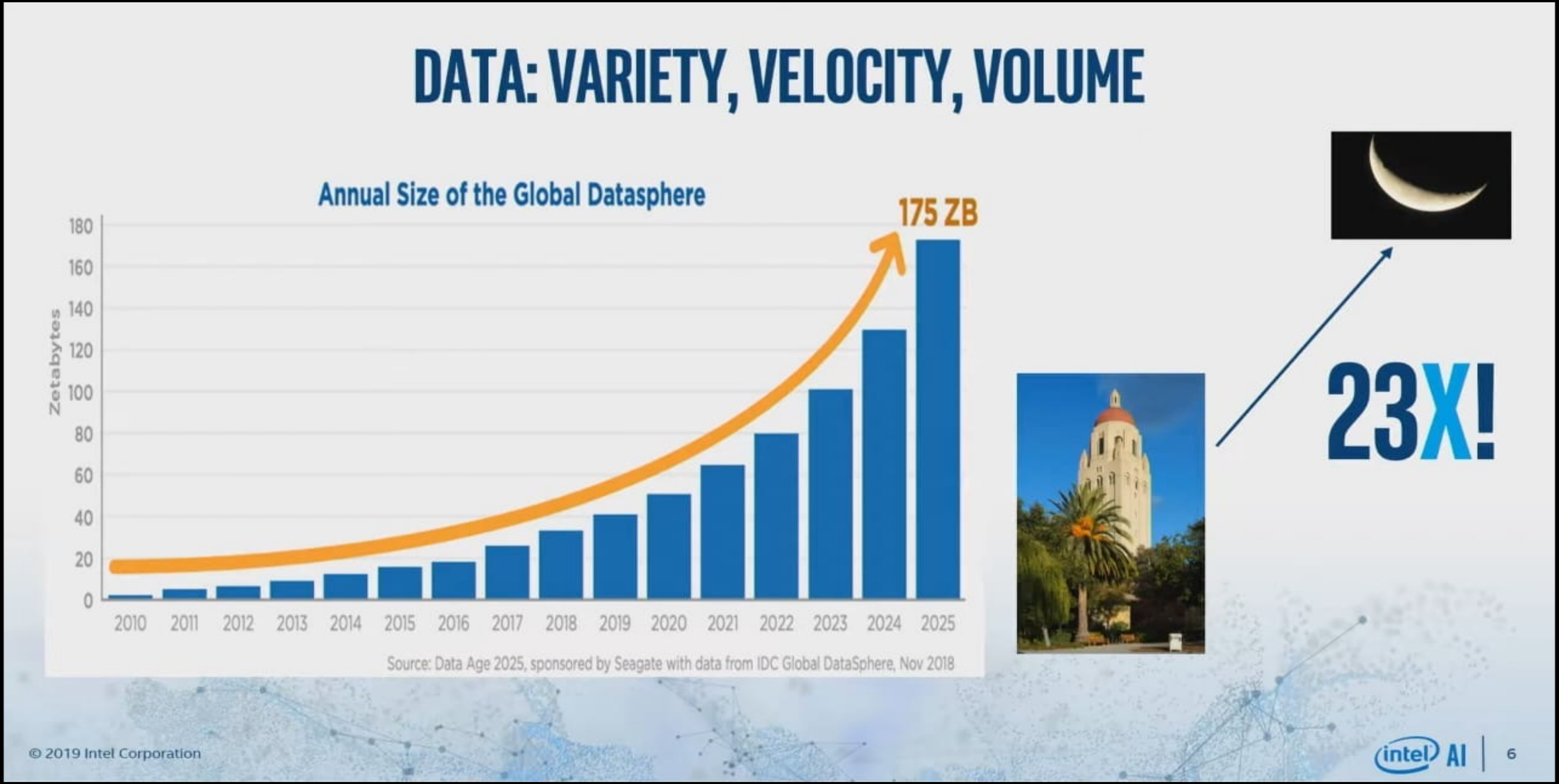
3 FACTORS DRIVING EXPONENTIAL GROWTH IN AI

ALGORITHMIC INNOVATION

DATA

COMPUTE

Data is the new Oil



DATA: VARIETY, VELOCITY, VOLUME

Annual Size of the Global Datasphere

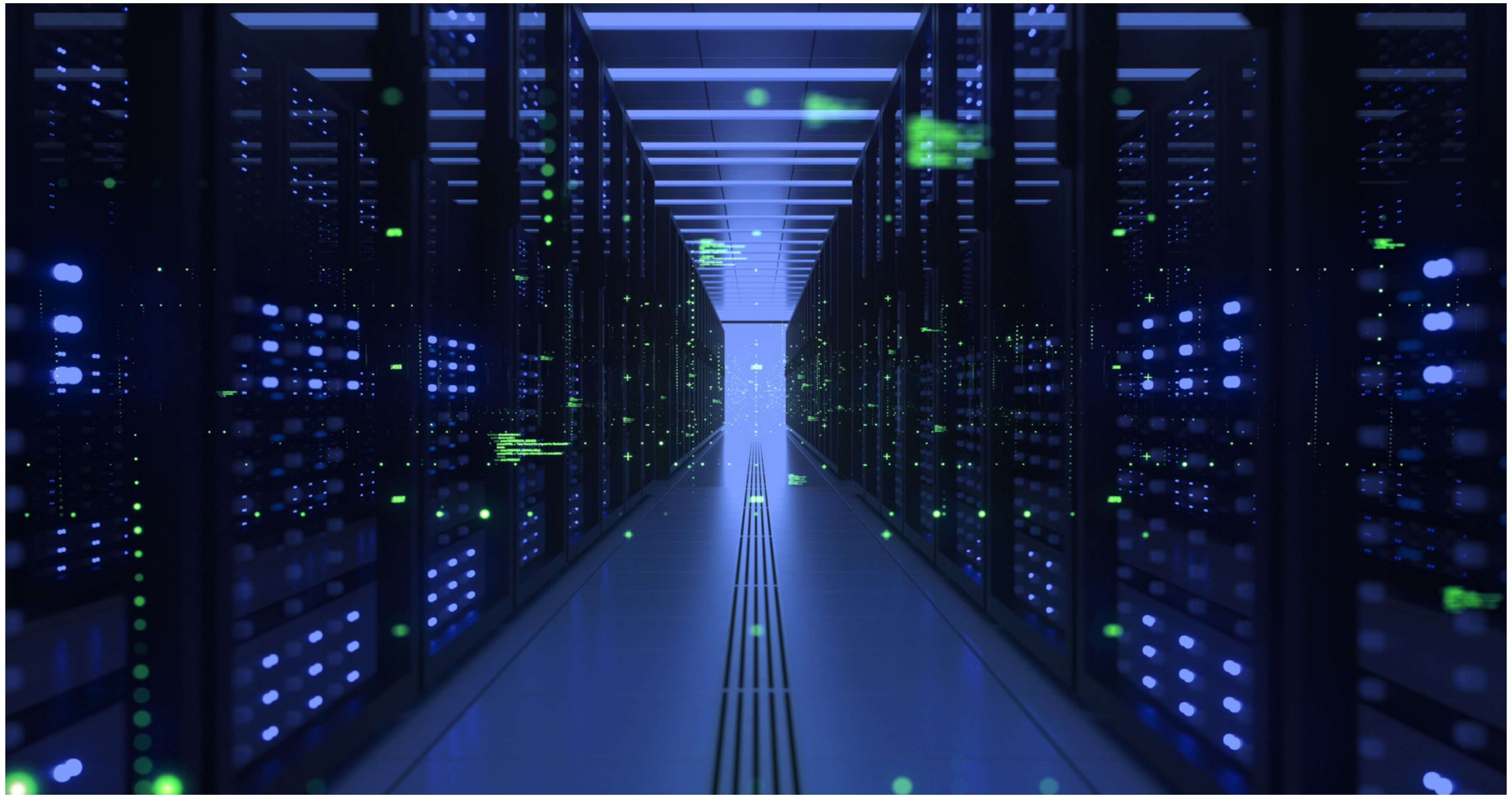
Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018

23X!

Database Management System

DBMS

DBMS stands for "Database Management System." It is software that facilitates the creation, organization, management, and manipulation of databases. A database is a structured collection of data that is organized and stored for efficient retrieval and analysis.



Key Components and Functions of a DBMS

Data Storage and Organization

Data Retrieval and Querying

Data Manipulation

Data Security

Concurrency Control

Key Components and Functions of a DBMS

Data Integrity

Backup and Recovery

Data Independence

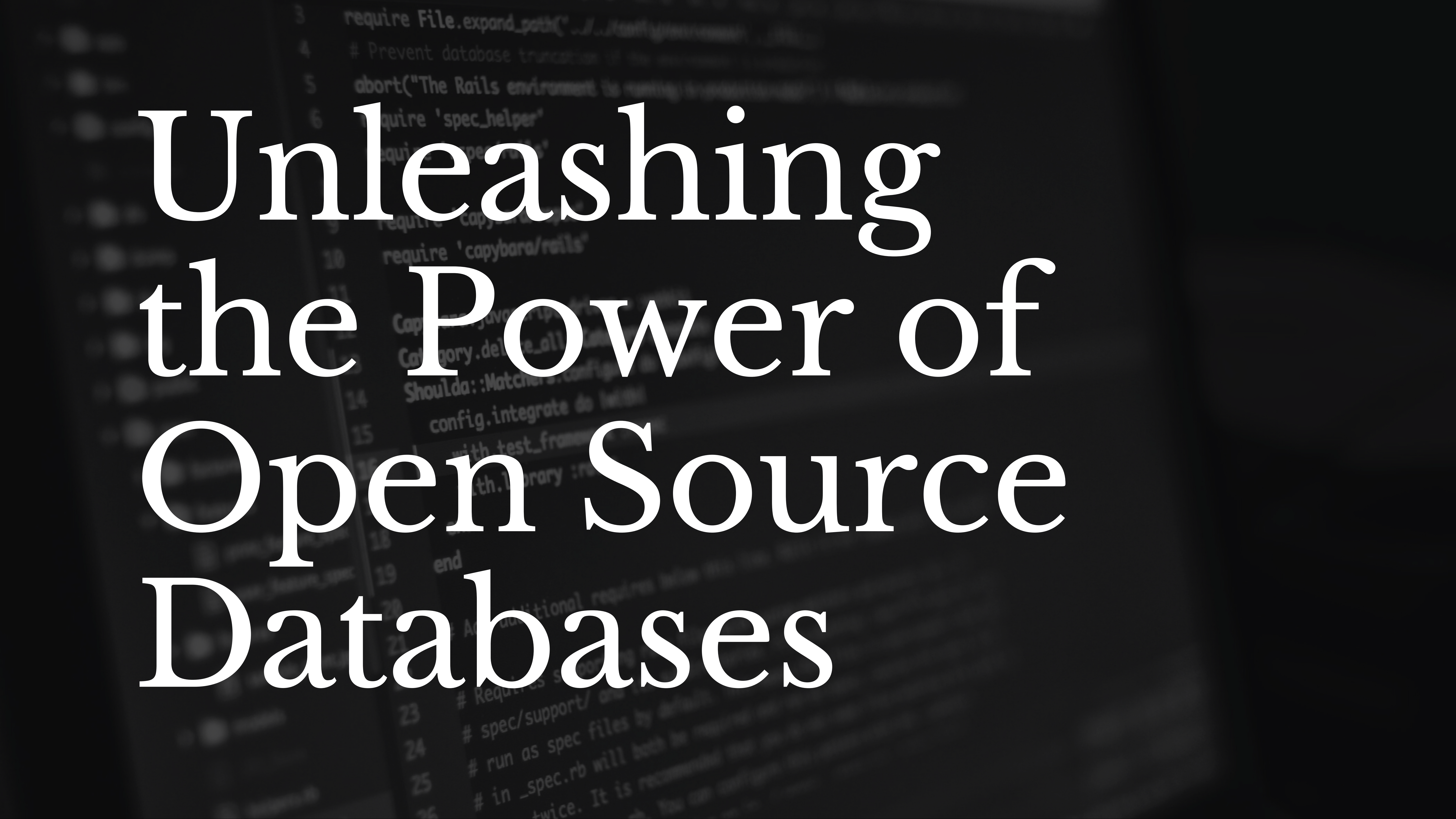
Data Modeling

Transaction Management

Key Components and Functions of a DBMS

Scalability

Data Dictionary



Unleashing the Power of Open Source Databases

Open-Source Database

Open-source databases are database management systems (DBMS) that are developed, distributed, and maintained under open-source licenses. An open-source license allows the source code of the software to be freely available to the public, enabling anyone to view, modify, and distribute the code.

```
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
        if (A>B) A
        else B=B-A
    }
    return A;
}

integer euclidAlgori
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
        if (A>B) A
        else B=B-A
    }
    return A;
}

integer euclidAlgori
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
```

Several key Characteristics of Open-Source Databases

Licensing

Transparency

Community Collaboration

Customizability

Flexibility

Several key Characteristics of Open-Source Databases

- Cost-effectiveness
- Innovation
- Security Auditing
- Education
- Interoperability

Several key Characteristics of Open-Source Databases

- Community Support
- Reduced Vendor Lock-in
- Adaptability
- Global Accessibility
- Ethical Considerations

Few examples of popular Open-Source Databases

MySQL

MySQL is used in various applications, from small websites to large-scale enterprise systems.

PostgreSQL

It emphasizes compliance with SQL standards and supports advanced features such as JSON storage, full-text search, and spatial data.

MongoDB

MongoDB is a popular open-source NoSQL database that stores data in flexible, JSON-like documents.

Few examples of popular Open-Source Databases

SQLite

SQLite is a self-contained, serverless, and zero-configuration open-source SQL database engine.

Cassandra

Apache Cassandra is an open-source distributed NoSQL database designed for managing large amounts of structured and unstructured data across multiple commodity servers.

Redis

Redis is an open-source, in-memory data structure store often used as a cache, message broker, and real-time analytics tool.

Few examples of popular Open-Source Databases

Elasticsearch

While primarily known as a search engine, Elasticsearch is also used as an open-source distributed document-oriented NoSQL database.

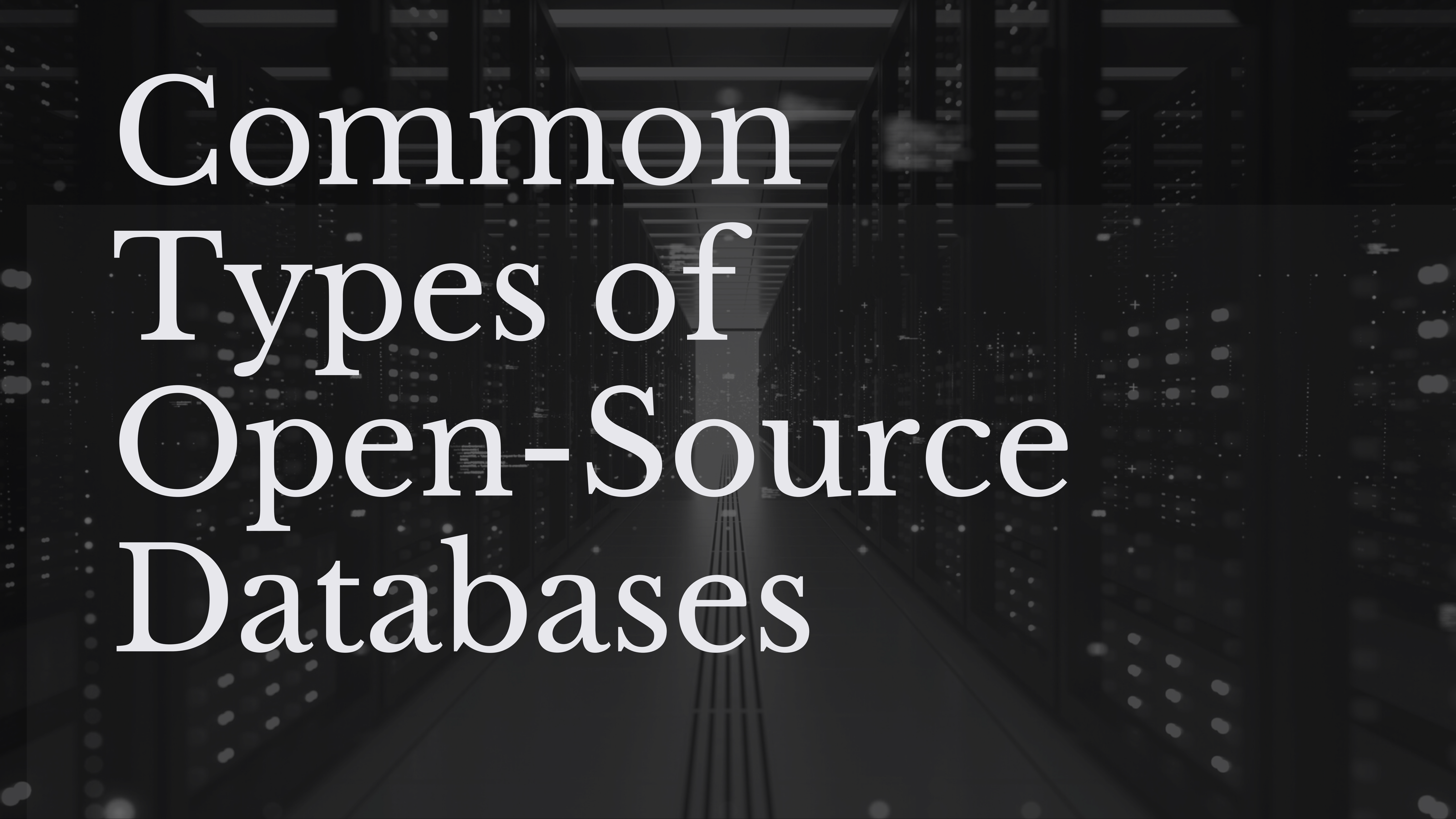
MariaDB

MariaDB is a community-developed open-source fork of MySQL.

InfluxDB

InfluxDB is an open-source time-series database designed for handling time-stamped data.

Common Types of Open-Source Databases



Relational Databases

- MySQL
- PostgreSQL
- SQLite

NoSQL Databases

- MongoDB
- Cassandra
- Redis
- Neo4j

Time-Series Databases

- InfluxDB
- OpenTSDB

Document Stores

- CouchDB
- RethinkDB

Columnar Databases

- Apache HBase

Search Engines and Full-Text Databases

- Elasticsearch
- Apache Solr

NewSQL Databases

- CockroachDB

- TiDB

The Rise of Open-Source Databases

Open-source databases are gaining popularity due to their flexibility, cost-effectiveness, and scalability. They allow developers to build and customize their own solutions, and offer a wide range of features and functionalities. This makes them ideal for powering AI, which requires large amounts of data and complex algorithms.

Open-Source Databases contribution to the advancement of AI

- Data Availability
- Training Data
- Benchmarking and Research
- Innovation and Collaboration
- Model Development and Testing

Open-Source Databases contribution to the advancement of AI

Transfer Learning

Data Labeling and Annotation

Ethical AI

Real-world Applications

Open-Source Databases contribution to the advancement of AI

- Educational Resources
- Reducing Barriers to Entry

Challenges of Using Open-Source Databases for AI

- Complexity and Learning Curve
- Lack of Professional Support
- Limited Features in Some Cases
- Security Concerns
- Performance Tuning

Challenges of Using Open-Source Databases for AI

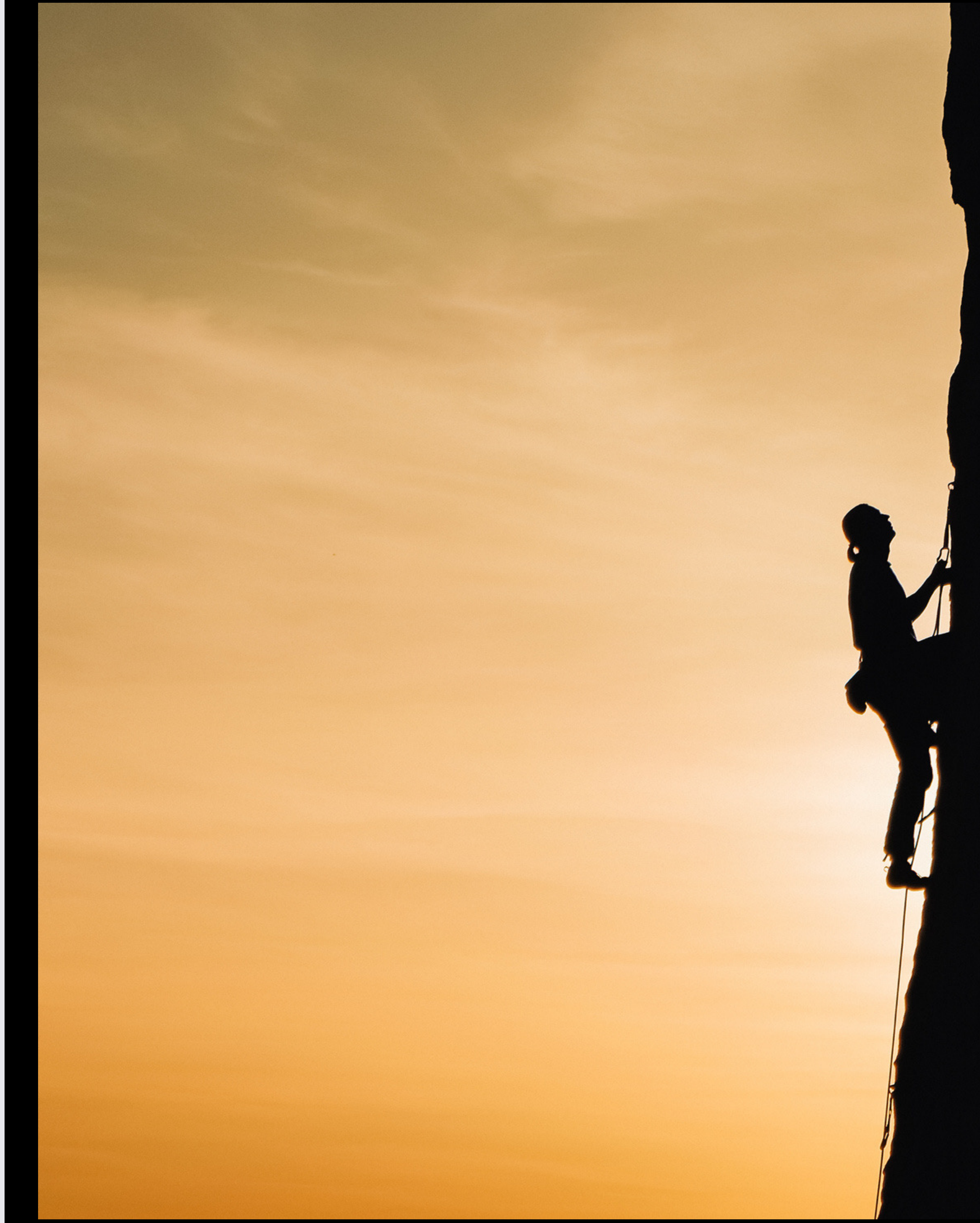
- Scalability Challenges
- Integration with AI Frameworks
- Data Consistency and Replication
- Lack of Vendor Lock-in Solutions
- Documentation and Resources

Challenges of Using Open-Source Databases for AI

- Migration and Compatibility

- Lack of Industry Compliance

- Long-Term Maintenance



Real-World Examples of Open-Source Databases for AI

Image Recognition and Classification

Natural Language Processing (NLP)

Recommendation Systems

Real-World Examples of Open-Source Databases for AI

IoT Data Processing

Healthcare Analytics

Autonomous Vehicles

Real-World Examples of Open-Source Databases for AI

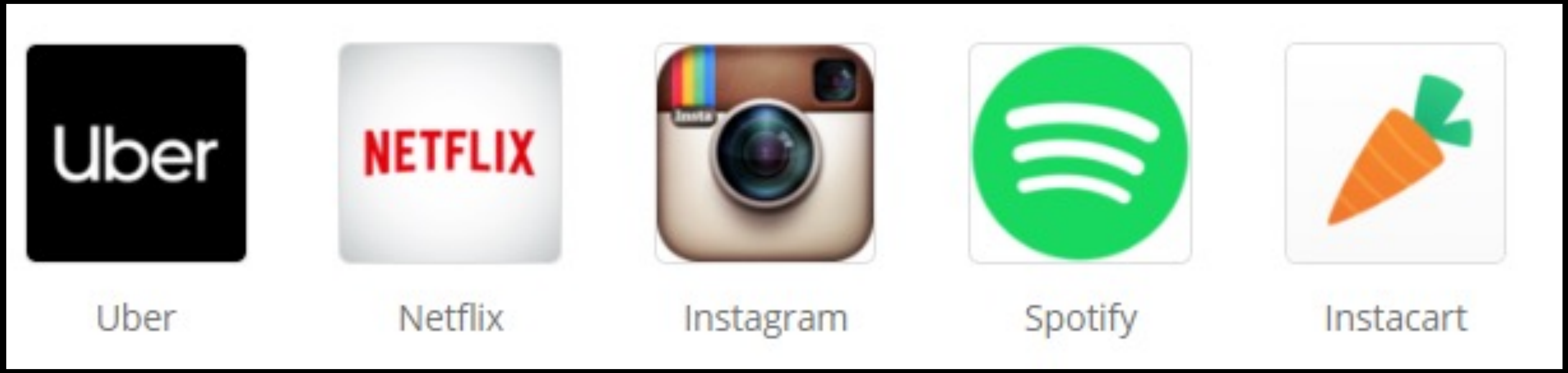
Fraud Detection

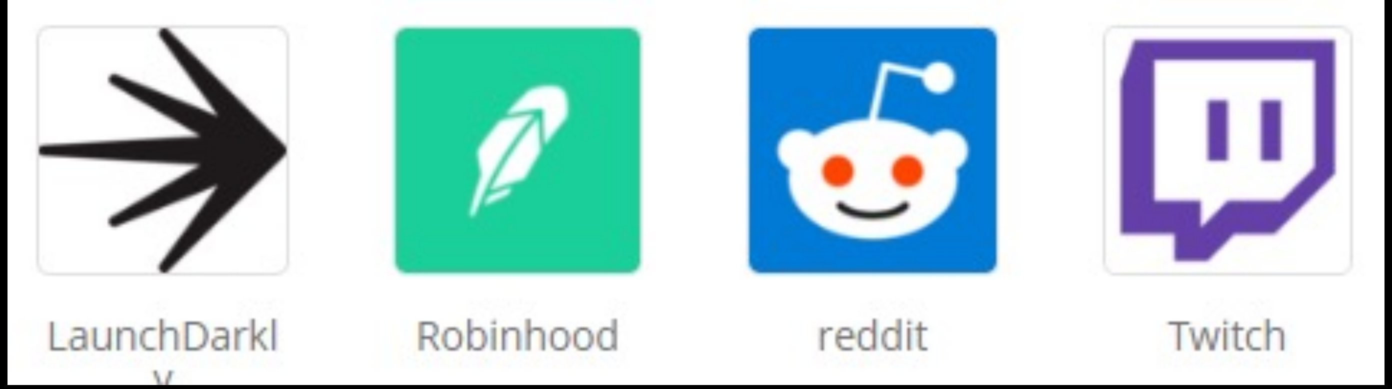
Financial Analysis

Social Media Analysis

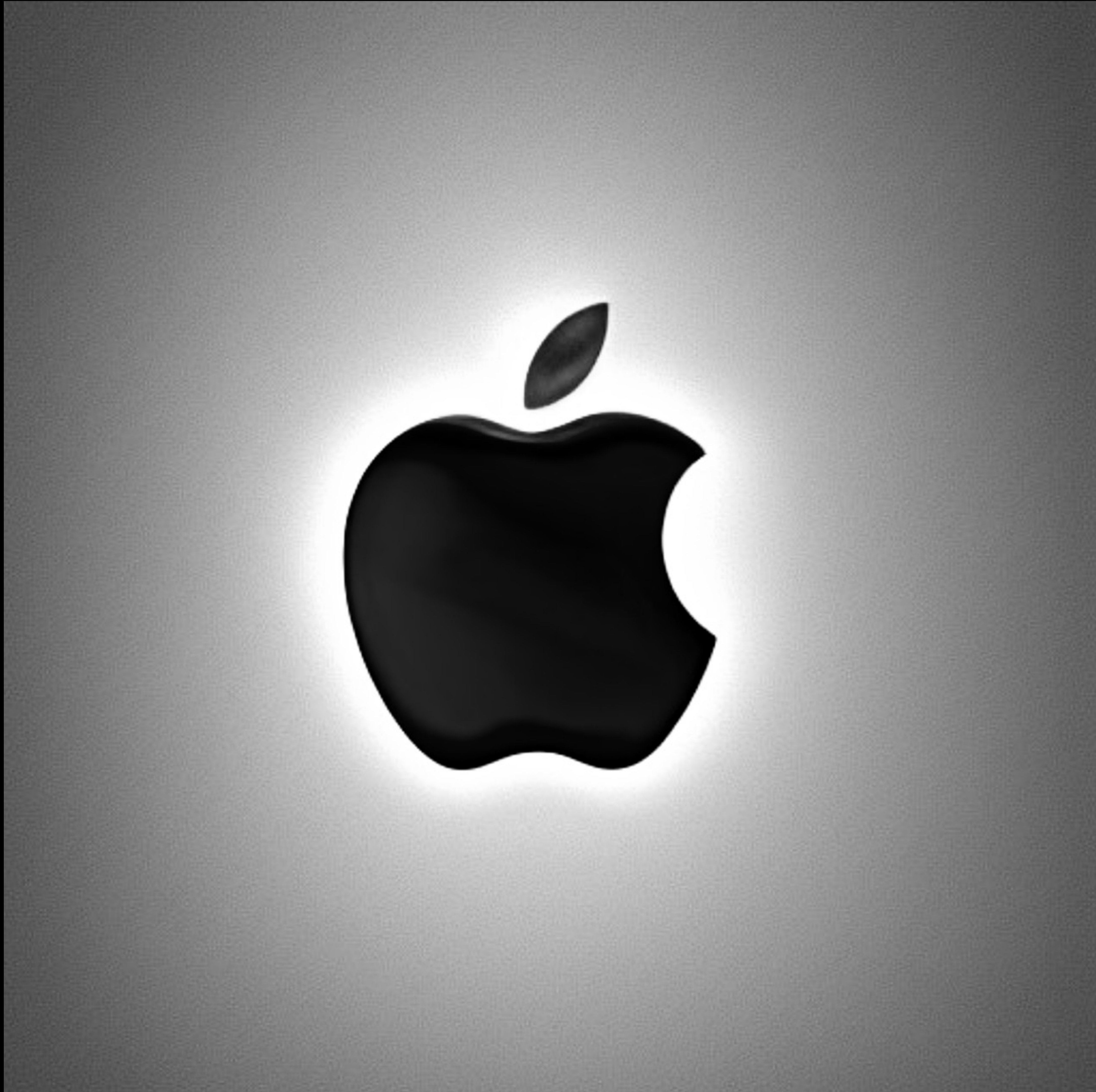
Energy Management

6306 companies reportedly use PostgreSQL in their tech stacks, including Uber, Netflix, and Instagram.





Apple systems support PostgreSQL.



A lot of data of IMDB is processed in PostgreSQL.

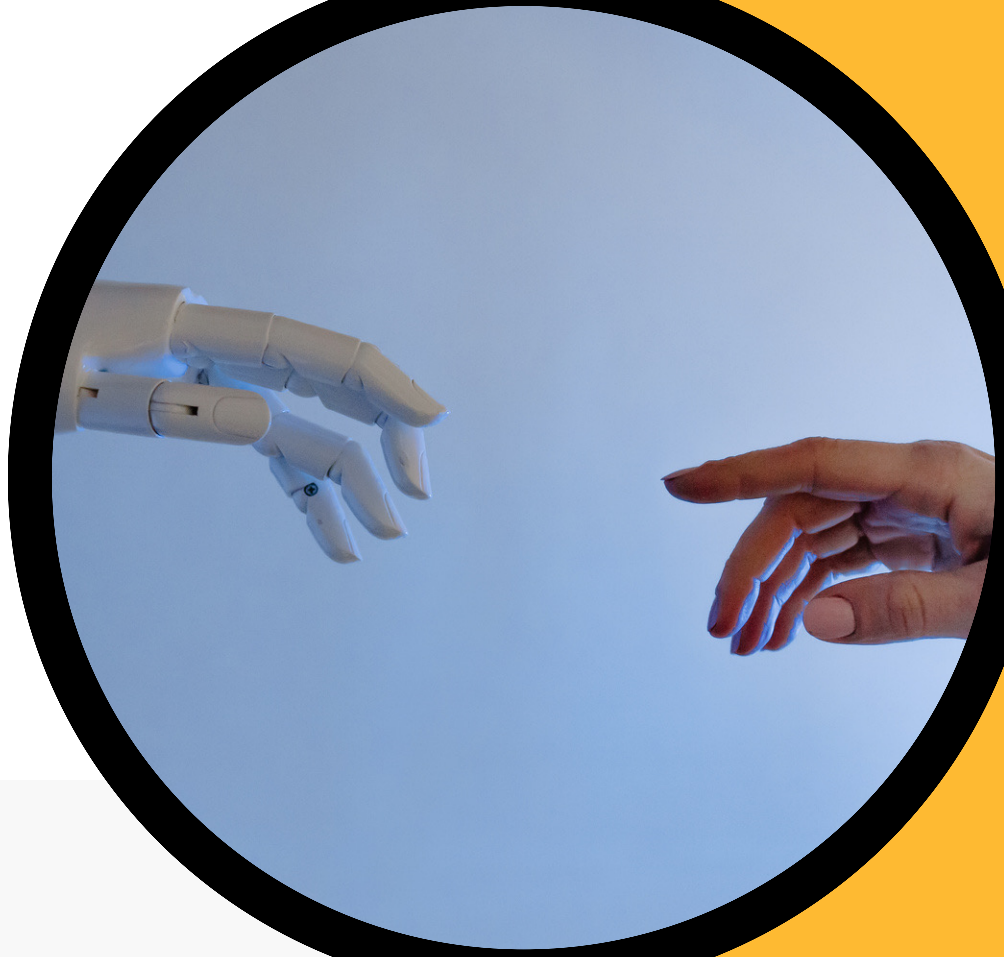
Instagram uses many RDBMSs, but PostgreSQL and Cassandra were chosen for the main tasks.

Uber uses Apache Cassandra to manage its ride and driver data.

Netflix uses Apache Druid to power its real-time analytics platform.

CONCLUSION

Open-source databases are a powerful tool for powering AI and are transforming the way we think about data management. While there are challenges to using open-source databases, the benefits are clear. As more companies adopt open-source databases for AI, we can expect to see even more innovation and growth in this exciting field.



Thank you

Any questions?

How Postgres is shaping AI trajectory

Vibhor Kumar
Global Vice President, Performance Engineering & Architecture

EDB™

About - PostgreSQL

The World's Most Advanced Open Source Database

- Relational, SQL based database.
- Fully enterprise ready; increasingly replacing Oracle, SQL Server, DB2 and more.
- Used in pretty much every sector: government, law enforcement, financial, healthcare...
- Possibly the most SQL Standard compliant database there is.
- Highly extensible:
  - Plugin extension modules
  - Plugin procedural languages (e.g. Python, Perl, R, Java, V8)
  - Low level code hooks

EDB IN SUMMARY

EDB is the world's largest software, support, and services company focused exclusively on PostgreSQL. With over 5,000 customers, we are proud to serve some of the world's leading financial services, government, media & communications, and information technology organisations. Our 16 offices worldwide enable us to deploy our global expertise in all your business locations.

POSTGRESQL **COMMUNITY** LEADERSHIP

- **30%** of Postgres code contributed
- **>300** Dedicated Postgres engineers
- **3 of 7** Postgres Core Team Members

EDB **PLATFORM** (SOFTWARE & TOOLS)

- Databases: PostgreSQL, EPAS
- Tools: Variety of supported open source and proprietary tools for High availability, backup, monitoring and migration

EDB **SUPPORT**

- 24/7 world-class support
- Experienced support engineers, with the world's leading Postgres contributors
- Cloud/Remote DBA Service, Technical Account Management, CTO Office

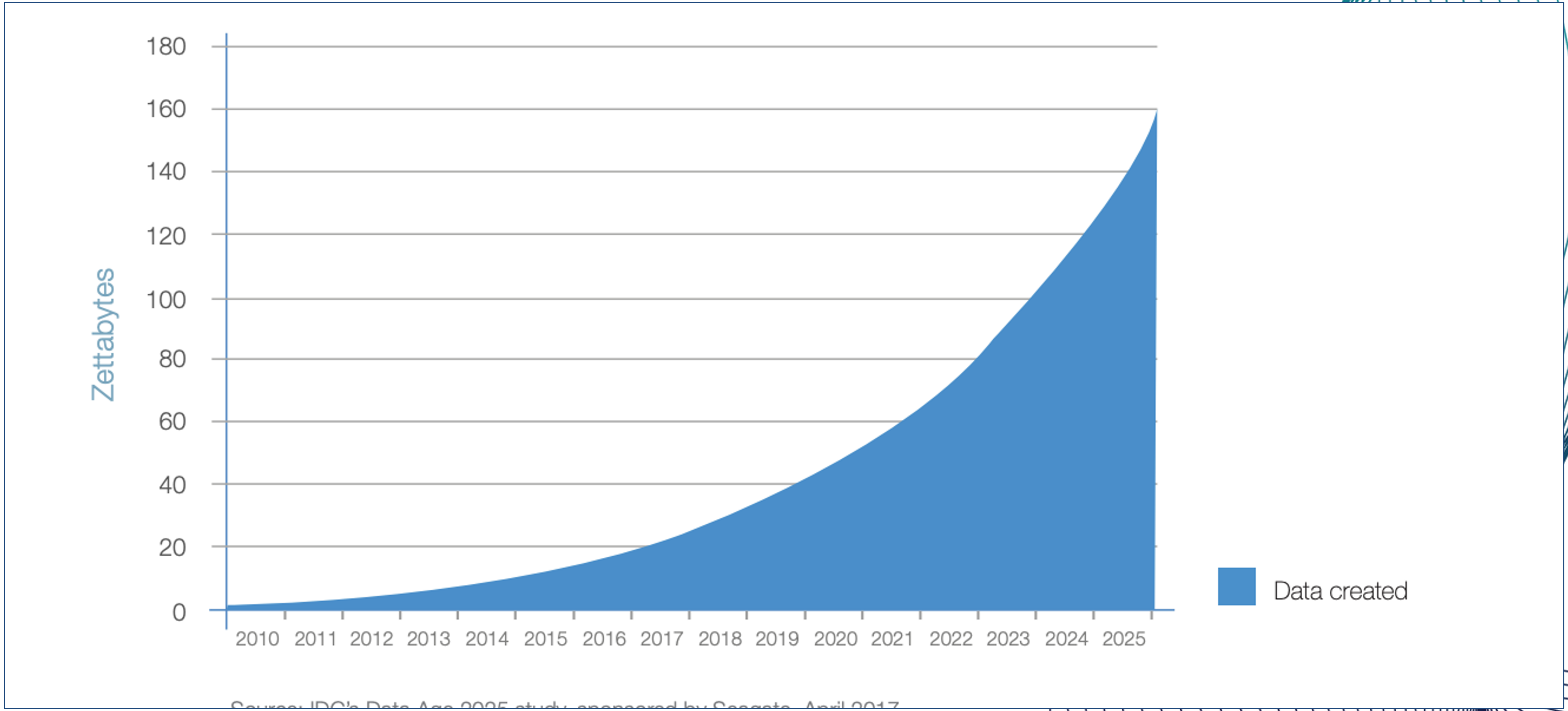
EDB **SERVICES**

- Services offerings and packages:
  - PostgreSQL deployment, design, migration
  - Postgres Optimization: Best practices
  - Enterprise Strategy: Use-case driven PostgreSQL architectures
  - Embedded PostgreSQL experts

EDB™

The future of AI rests on a foundation of solid data management.

AI Growing Data Need



Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017

- Key Takeaway: From gigabytes to petabytes, AI demands more data than ever before.
- Reference: IDC's Data Age 2025 study.

Postgres’ key feature for AI

- Rich data types
  - NUMBER
    - SMALLINT, BIGINT, …
    - DECIMAL, DOUBLE, NUMERIC
    - SMALLSERIAL, BIGSERIAL, SERIAL
  - BYTEA
  - TIMESTAMP, TIME, DATE, INTERVAL
  - MONEY
  - BOOLEAN
  - **GEOMETRY (LINE, POINTS, LINE SEGMENTS(LSEG), PATH, POLYGON, CIRCLE)**
  - NETWORK ADDRESS TYPE (INET, CIDR, MACADDR, MACADDR8)
  - BIT STRING TYPE (BIT(n), BIT VARYING
  - **VARACHAR(n)/CHAR(n)/TEXT**

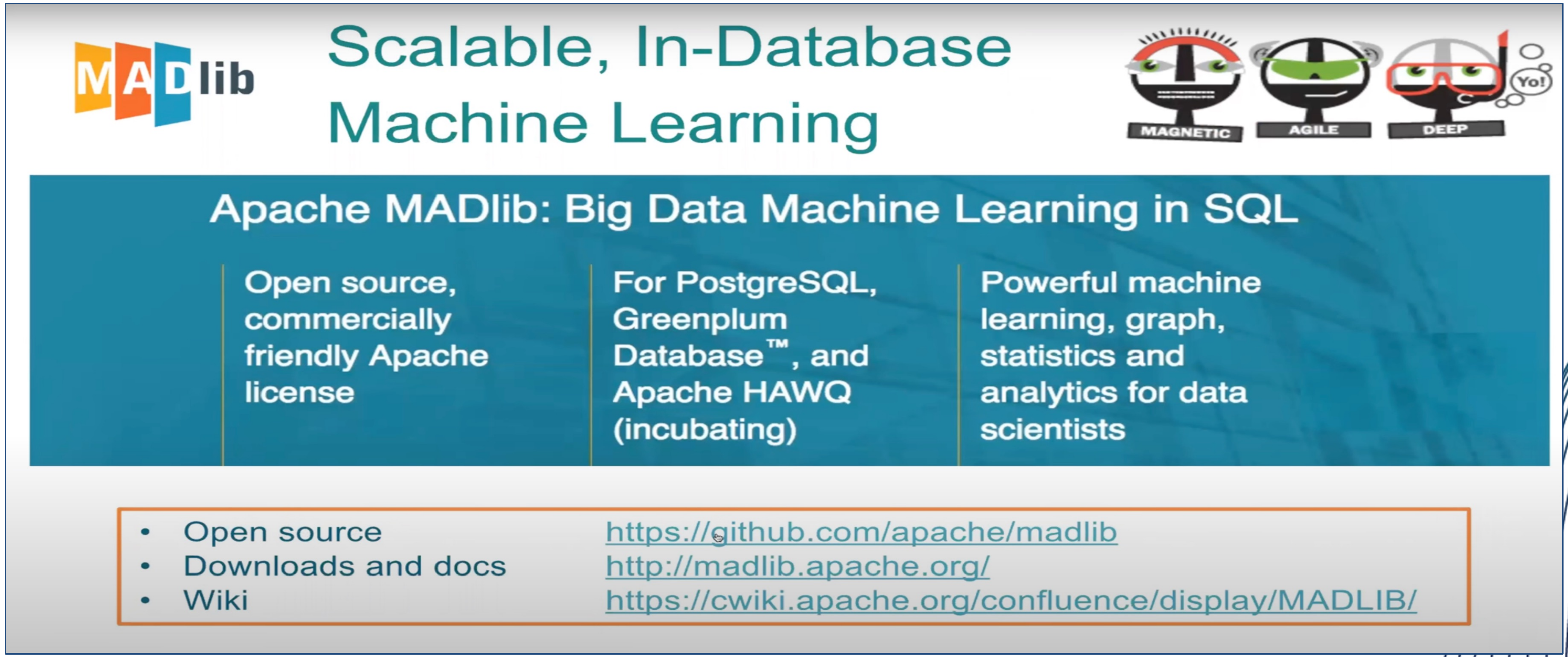
- Rich data types
  - UUID
  - **XML**
  - **JSON/JSONB**
  - **ARRAYS**
  - COMPOSITE TYPE
  - RANGE TYPES (INT4RANGE, INT8RANGE, NUMRANGE, TSRANGE, TSTZRANGE, DATERANGE, …)
  - DOMAIN TYPE
  - **TEXT SEARCH TYPES (FULL TEXT SEARCH)**

- **Geospatial**
  - **PostGIS**

Richness of data types allows for versatile data modeling

EDB™

Machine Learning Integration



Scalable, In-Database Machine Learning

Apache MADlib: Big Data Machine Learning in SQL

Open source, commercially friendly Apache license

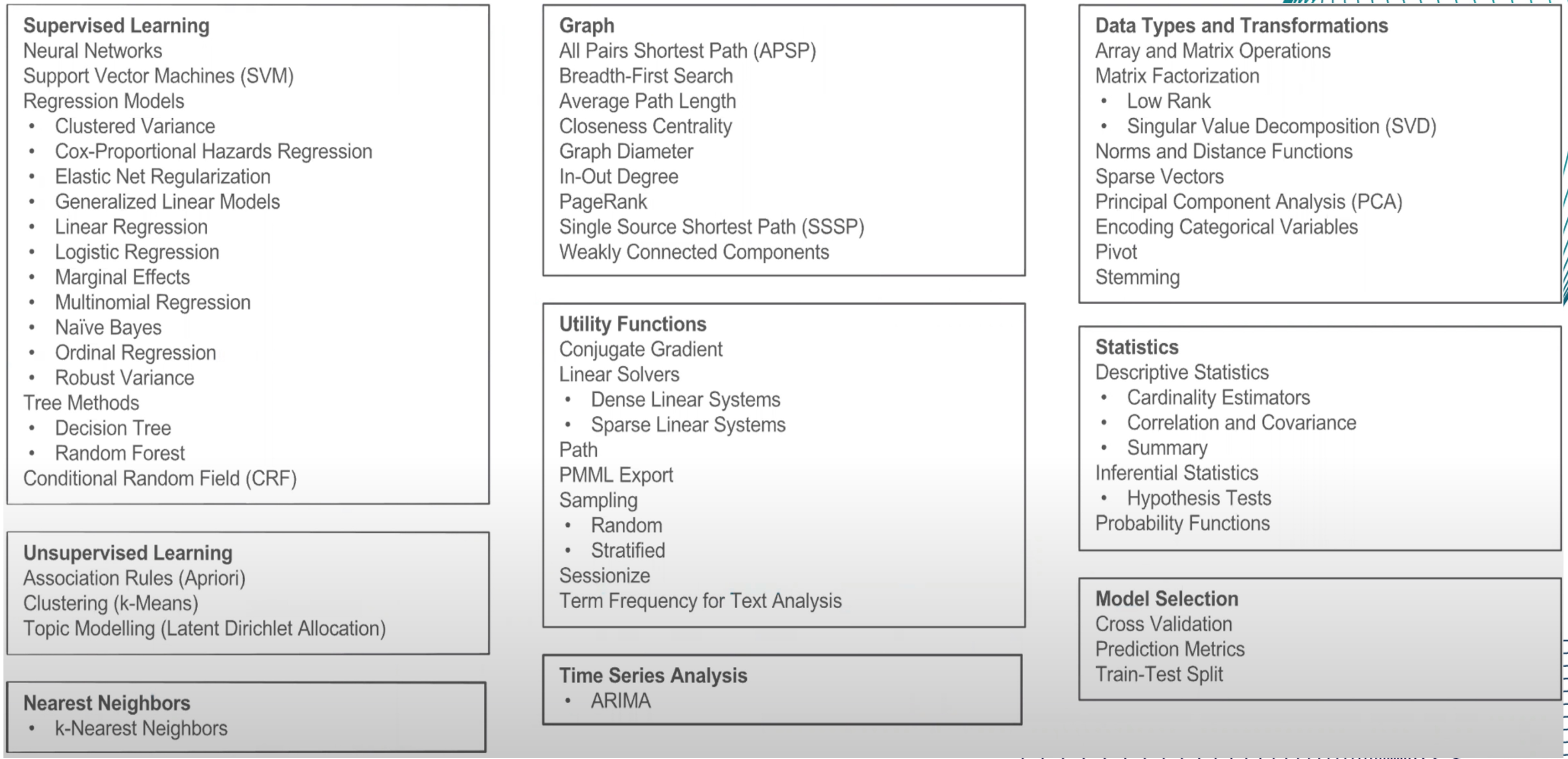
For PostgreSQL, Greenplum Database™, and Apache HAWQ (incubating)

Powerful machine learning, graph, statistics and analytics for data scientists

- Open source https://github.com/apache/madlib
- Downloads and docs http://madlib.apache.org/
- Wiki https://cwiki.apache.org/confluence/display/MADLIB/

In-database analytics reduces the need for data movement.

Machine Learning Integration - Apache MADLib



Supervised Learning
Neural Networks
Support Vector Machines (SVM)
Regression Models
- Clustered Variance
- Cox-Proportional Hazards Regression
- Elastic Net Regularization
- Generalized Linear Models
- Linear Regression
- Logistic Regression
- Marginal Effects
- Multinomial Regression
- Naïve Bayes
- Ordinal Regression
- Robust Variance

Tree Methods
- Decision Tree
- Random Forest

Conditional Random Field (CRF)

Unsupervised Learning
Association Rules (Apriori)
Clustering (k-Means)
Topic Modelling (Latent Dirichlet Allocation)

Nearest Neighbors
- k-Nearest Neighbors

Graph
All Pairs Shortest Path (APSP)
Breadth-First Search
Average Path Length
Closeness Centrality
Graph Diameter
In-Out Degree
PageRank
Single Source Shortest Path (SSSP)
Weakly Connected Components

Utility Functions
Conjugate Gradient
Linear Solvers
- Dense Linear Systems
- Sparse Linear Systems

Path
PMML Export
Sampling
- Random
- Stratified

Sessionize
Term Frequency for Text Analysis

Time Series Analysis
- ARIMA

Data Types and Transformations
Array and Matrix Operations
Matrix Factorization
- Low Rank
- Singular Value Decomposition (SVD)

Norms and Distance Functions
Sparse Vectors
Principal Component Analysis (PCA)
Encoding Categorical Variables
Pivot
Stemming

Statistics
Descriptive Statistics
- Cardinality Estimators
- Correlation and Covariance
- Summary

Inferential Statistics
- Hypothesis Tests

Probability Functions

Model Selection
Cross Validation
Prediction Metrics
Train-Test Split

JSON/JSONB - Flexible Storage

JSON and NoSQL Support

- Creating a table with a JSONB field

```
CREATE TABLE json_data (data JSONB);
```

- Simple JSON data element:

```
{"name": "Apple Phone", "type": "phone", "brand": "ACME", "price": 200,
"available": true, "warranty_years": 1}
```

- Inserting this data element into the table json_data

```
INSERT INTO json_data (data)  VALUES
     (’ { "name": "Apple Phone",
          "type": "phone",
          "brand": "ACME",
          "price": 200,
          "available": true,
          "warranty_years": 1
     } ')
```

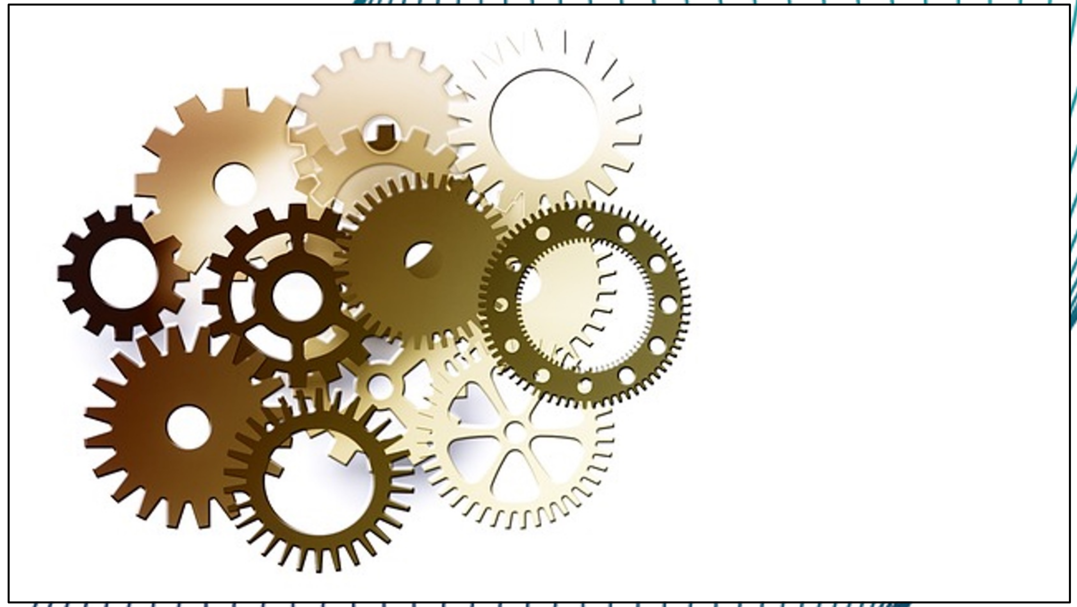
A QUERY THAT RETURN JSON DATA

```
SELECT data FROM json_data;
data
------------------------------------------
 {"name": "Apple Phone", "type": "phone", "brand": "ACME", "price": 200,
"available": true, "warranty_years": 1}
```

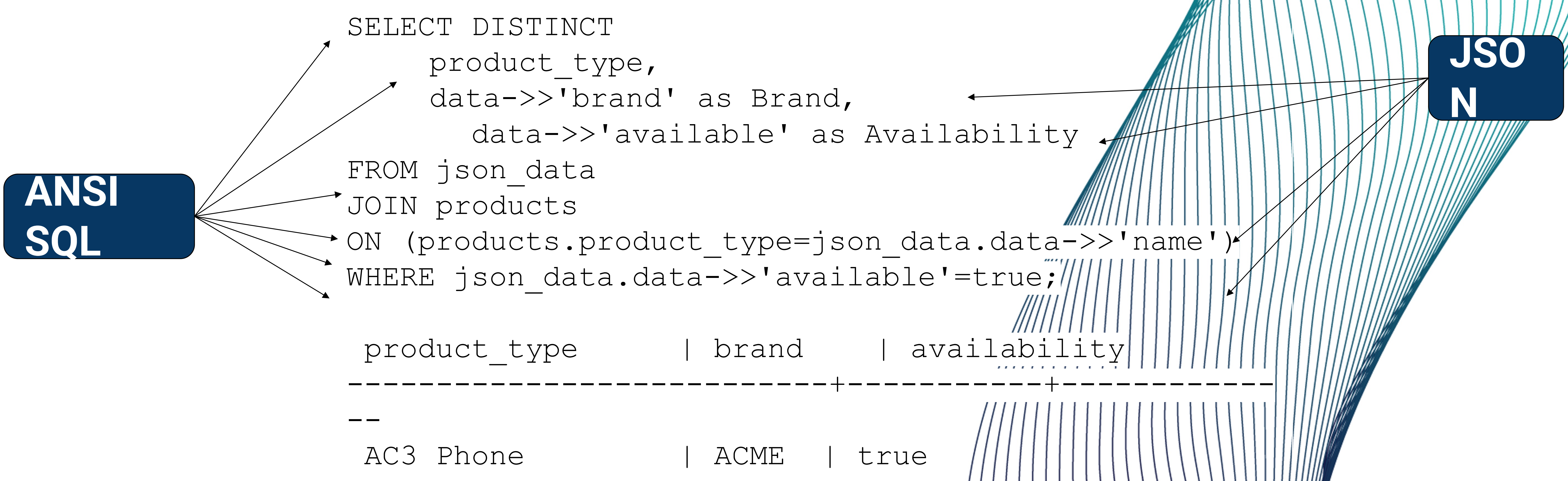
JSON(B) AND ANSI SQL IN POSTGRES – A NATURAL FIT

- JSON is naturally integrated with ANSI SQL in Postgres
- JSON and SQL queries use

  the same language, the same planner, and the same ACID compliant transaction framework
- JSON is an elegant and easy to use extensions of the underlying object-relational model



JSON AND ANSI SQL – EXAMPLE



ANSI SQL

JSON

```
SELECT DISTINCT
     product_type,
     data->>'brand' as Brand,
       data->>'available' as Availability
FROM json_data
JOIN products
ON (products.product_type=json_data.data->>'name')
WHERE json_data.data->>'available'=true;
```

```
 product_type      | brand    | availability
---------------------------+-----------+-------------
--
 AC3 Phone         | ACME  | true
```

No need for programmatic logic to combine SQL and NoSQL in the application – Postgres does it all

Bridging Between SQL And JSON

Simple ANSI SQL Table Definition

```
CREATE TABLE products (id integer, product_name text );
```

Select query returning standard data set

```
SELECT * FROM products;

 id | product_name
----+---------------
  1 | iPhone
  2 | Samsung
  3 | Nokia
```

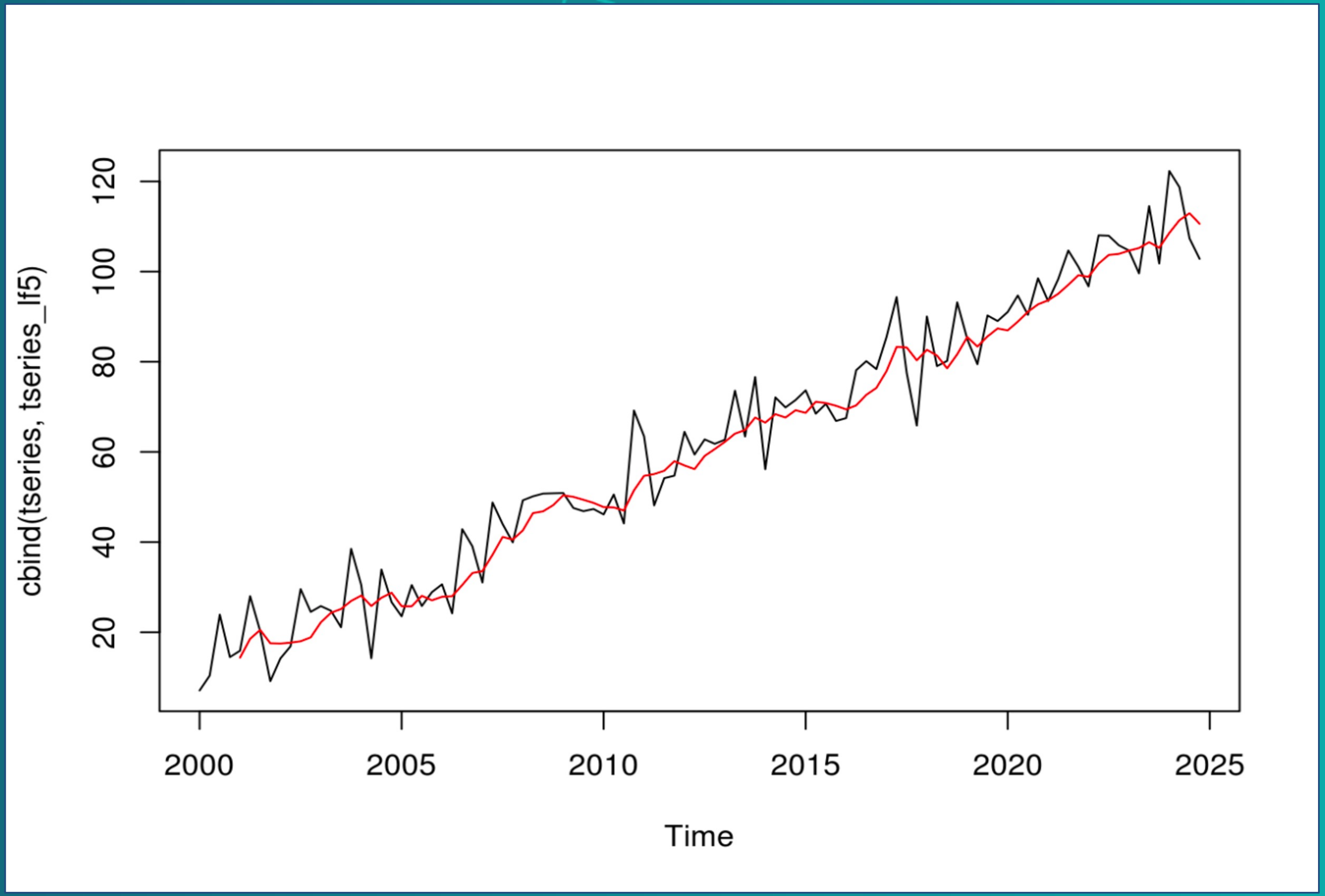
Select query returning the same result as a JSON data set

```
SELECT ROW_TO_JSON(products) FROM products;

 {"id":1,"product_name":"iPhone"}
 {"id":2,"product_name":"Samsung"}
 {"id":3,"product_name":"Nokia”}
```

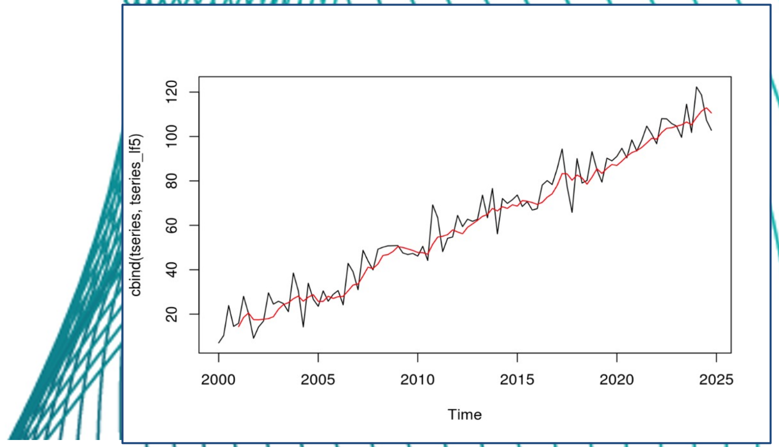
JSON DATA TYPES EXAMPLE

```
{
  "firstName": "John",           -- String
Type
  "lastName": "Smith",           -- String
Type
  "isAlive": true,               -- Boolean
Type
  "age": 25,                     -- Number
Type
  "height_cm": 167.6,            -- Number
Type
  "address": {                   -- Object
Type
    "streetAddress": "21 2nd Street”,
    "city": "New York”,
    "state": "NY”,
    "postalCode": "10021-3100”
  },
  "phoneNumbers": [        // Object Array
    {                     // Object
      "type": "home”,
      "number": "212 555-1234”
    },
    {
      "type": "office”,
      "number": "646 555-4567”
    }
  ],
  "children": [],
  "spouse": null         // Null
}
```

TIME SERIES DATA



Time-Series Data Management

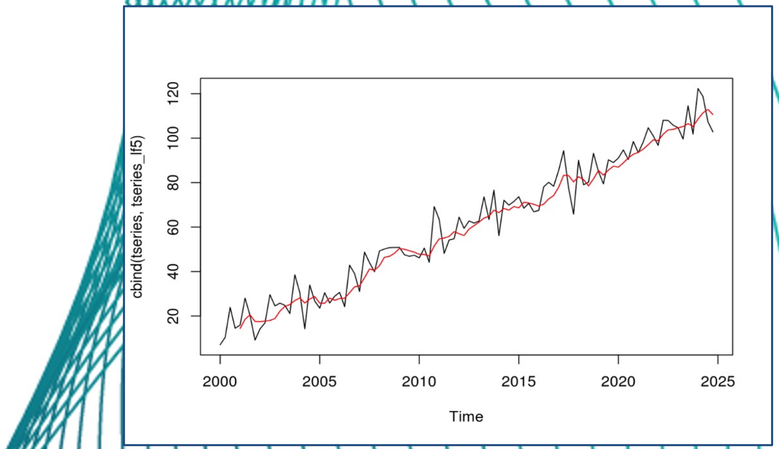


- Support for partitioning: Store time-series data in a way that optimizes for queries that need to access data from a specific time range.

```
-- Create a table to store the time-series
data
CREATE TABLE temperature (
  id INT,
  time TIMESTAMP,
  temperature FLOAT
);
```

```
-- Create a partition for the data from 2023-01-01 to
2023-03-01
CREATE TABLE temperature_partitioned (
  id INT,
  time TIMESTAMP,
  temperature FLOAT
)
PARTITION BY RANGE (time)
(
  PARTITION p1 VALUES LESS THAN ('2023-03-01
00:00:00'),
  PARTITION p2 VALUES LESS THAN ('2023-06-01
00:00:00'),
  PARTITION p3 VALUES LESS THAN ('2023-09-01
00:00:00')
);
```

Time-Series Data Management



- Interval partition in EDB Postgres Advanced Server
- Compression using custom functions like gzip
- Tablespace for storing data on filesystems with higher compression
- Indexes -
  - Local indexes
  - B-Tree
  - GiST indexes
  - BRIN Indexes
  - Custom Indexes
- Functions
  - AVG
  - SUM/MAX/MIN
  - DATE functions etc…
  - Mathematical functions - SIN/COS/TAN etc…

```
CREATE TABLE sales (
   sale_date              DATE,
   units                  INTEGER
) PARTITION BY RANGE (sale_date) INTERVAL
(NUMTOYMINTERVAL(-1,'MONTH'))
(
  PARTITION part_01 values LESS THAN
(TO_DATE('01-FEB-2020','DD-MON-YYYY'))
);
```

Procedure Languages (PL/perl, PL/python, PL/C, …)

PL/Python3u - Python programing language

- Can be used with an common Python machine learning and related libraries:
    - Tensorflow
    - pvTorch
    - Numpy
    - Pandas

- Gives you complete control to write the functionality you need.

- Install Python modules in the Python environment used by the PostgreSQL server.

Advanced Analytics with Postgres

Window functions

```
SELECT depname, empno, salary, rank() OVER (PARTITION BY depname ORDER BY salary DESC)
FROM empsalary;

 depname  | empno | salary |          avg
-----------+-------+--------+-----------------------
 develop   |    11 |   5200 | 5020.0000000000000000
 develop   |     7 |   4200 | 5020.0000000000000000
 develop   |     8 |   6000 | 5020.0000000000000000
 develop   |    10 |   5200 | 5020.0000000000000000
 personnel |     5 |   3500 | 3700.0000000000000000
 personnel |     2 |   3900 | 3700.0000000000000000
 sales     |     3 |   4800 | 4866.6666666666666667
 sales     |     1 |   5000 | 4866.6666666666666667
 sales     |     4 |   4800 | 4866.666666666666667
(9 rows)
```

Advanced CTE feature

Delete a given order,all the items associated with order and place order in a historical table.

```
WITH source (order_id) AS (
     DELETE FROM orders WHERE name = ’my order’ RETURNING order_id
), source2 AS (
     DELETE FROM items USING source WHERE source.order_id = items.order_id )
INSERT INTO old_orders SELECT order_id FROM source;
```

Less code to maintain than on any other database
Fewer round trips with the server than on any other database

```
 GROUPING SETS, CUBE, and ROLLUP - more ways processing
```

Window functions

Function **Description**
row_number () → bigint Returns the number of the current row within its partition, counting from 1.
rank () → bigint Returns the rank of the current row, with gaps; that is, the row_number of the first row in its peer group.
dense_rank () → bigint Returns the rank of the current row, without gaps; this function effectively counts peer groups.
percent_rank () → double precision Returns the relative rank of the current row, that is (rank - 1) / (total partition rows - 1). The value thus ranges from 0 to 1 inclusive.
cume_dist () → double precision Returns the cumulative distribution, that is (number of partition rows preceding or peers with current row) / (total partition rows). The value thus ranges from 1/*N* to 1.
• And more…

Specialized Indexes

Specialized indexes for all data types and access patterns

Index Type	**Optimized For**
B-Tree	Range queries with low selectivity and largely unique values. The traditional database index.
BRIN	Time series data, multi-terabyte tables
HASH	Equality lookups on large datasets (key / value store) use cases.
GiST	Unstructured Data i.e. Geo Spatial Types
GIN	JSON Data, Full Text Search, JSONB Data
SP-GiST	SP-GIST is ideal for indexes whose keys have many duplicate prefixes

Specialized Indexes

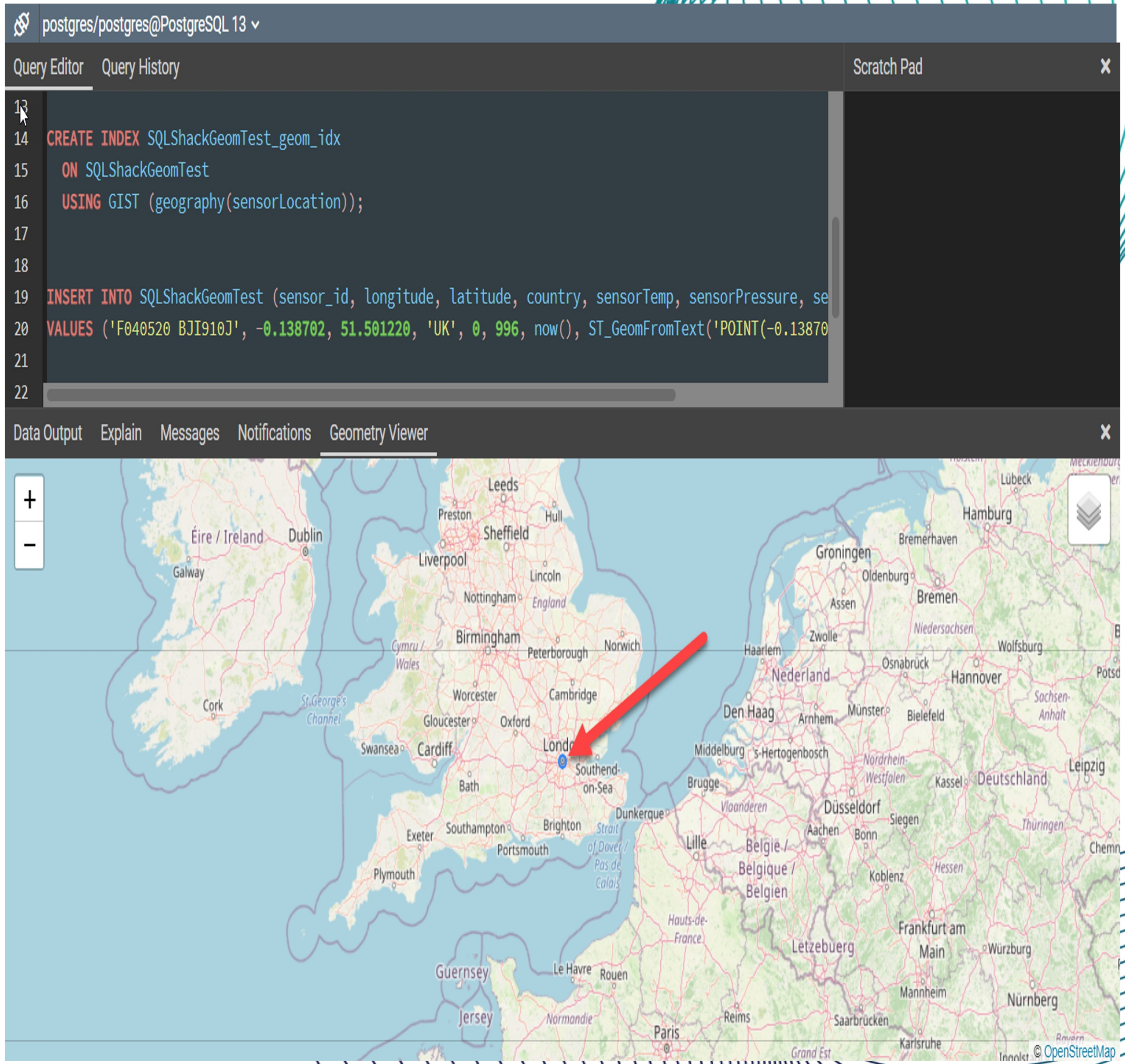
Specialized indexes for non relational data

Index Type	Optimized For
PARTIAL	When only a specific set of values will be looked up
COVERING	For access patterns to unindex values navigated to by an index.
EXPRESSION	Allow for variances in keys

PostGIS - Spatial Data

Spatial Data and Geolocation

- Extension - PostGIS
- PostGIS Functions
    - ST_Accum - Aggregate. Constructs an array of geometries
    - ST_Collect - Return a specified ST_Geometry value from a collection of other geometries.
    - etc…
- Indexes
    - GiST - most commonly used for PostGIS
    - R-Tree - Break up data into rectangles, and sub-rectangles
    - Quad Tree - (images/spatial)

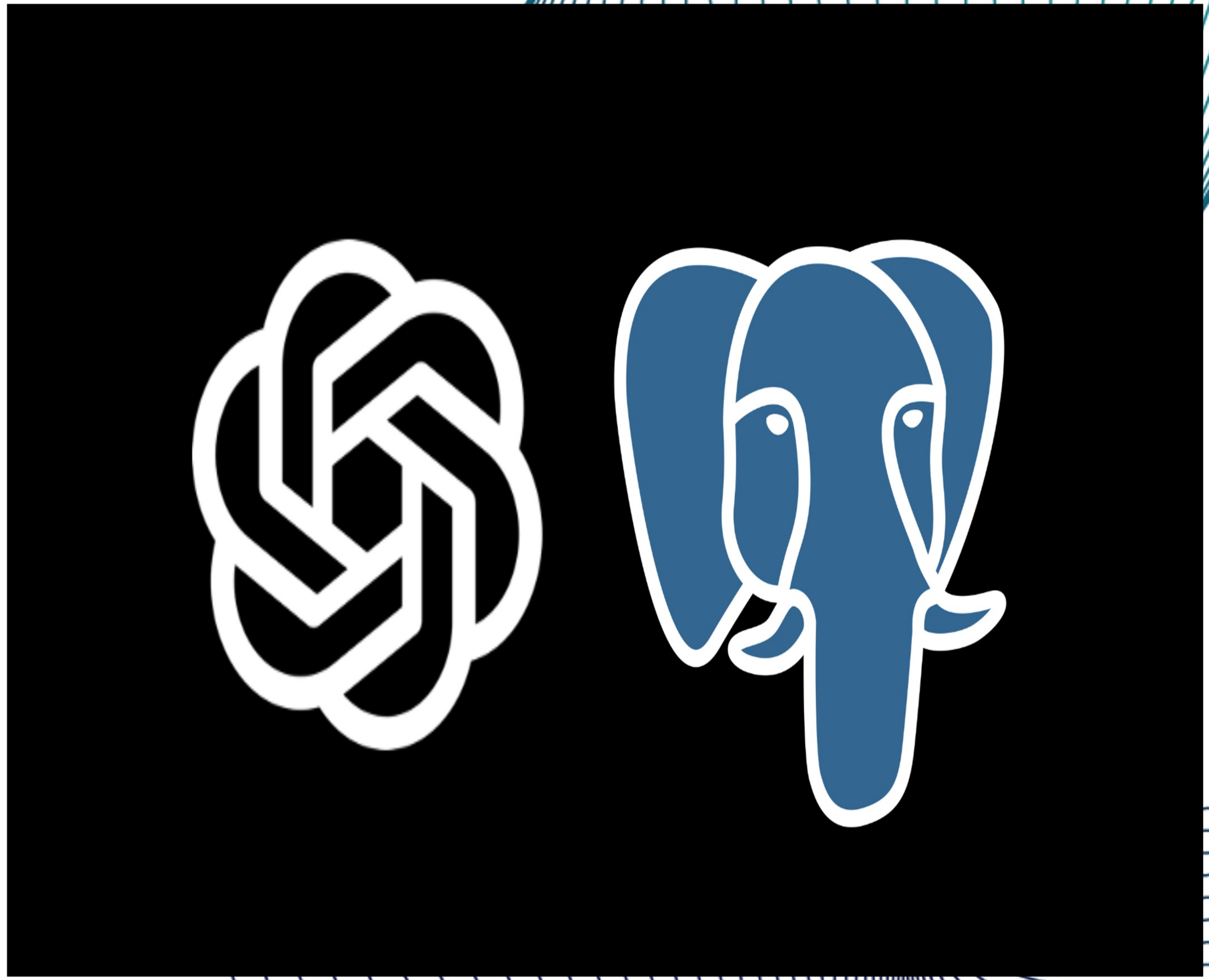


```
CREATE INDEX SQLShackGeomTest_geom_idx
  ON SQLShackGeomTest
  USING GIST (geography(sensorLocation));


INSERT INTO SQLShackGeomTest (sensor_id, longitude, latitude, country, sensorTemp, sensorPressure, se
VALUES ('F040520 BJI910J', -0.138702, 51.501220, 'UK', 0, 996, now(), ST_GeomFromText('POINT(-0.13870
```

pgvector

pgvector - An Extension For Similarity Search

- Vector similarity search is a type of search that allows you to find similar vectors.
- Vectors data type represents
  - points in a multidimensional space.
- Can be used for a variety of applications, such as:
  - Recommendation systems
  - Image search
  - Natural language processing



More Extensions For AI

- PostgresML: An open-source extension provides support for training and deploying machine learning models in PostgreSQL.

- pgRouting: An open-source extension for routing algorithms.
    - FInd the shortest path between two points in a road network.

Open Source Advantage

A community-driven approach ensures continuous updates and innovations.

Conclusion

With every challenge comes an opportunity for innovation.

THANK YOU

(We can’t wait to see what you create)