

# Data Horizons with Postgres

BRUCE MOMJIAN



This presentation surveys the horizon for data management and Postgres.

<https://momjian.us/presentations>



*Creative Commons Attribution License*

*Last updated: March 2024*

# Outline

1. What are the data needs of the future?
2. How can Postgres meet them?
3. What if it can't?

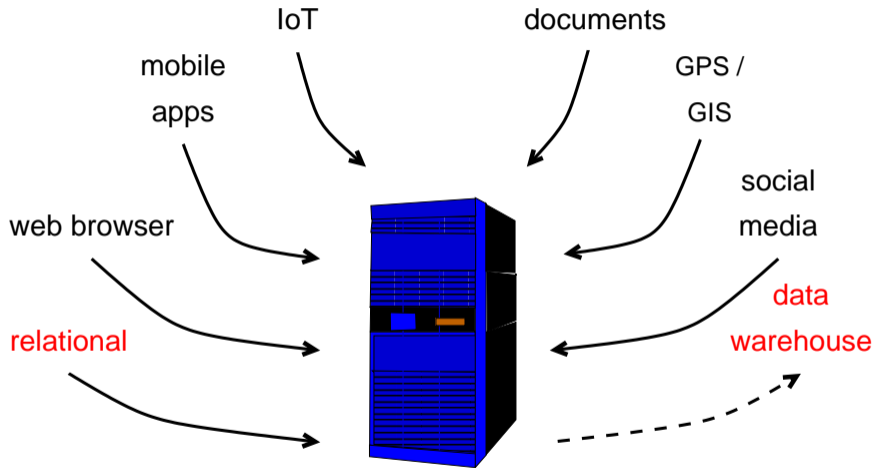
# Data Ingestion Methods Have Changed

- 1960's: electronically-readable paper (punch cards)
- 1970's: teletypes
- 1980's: dumb terminals
- 1990's: fat & thin clients
- 2000's: web browsers
- 2010's: web applications, mobile apps, documents, text communication, geolocation data, Internet of Things (IoT), sensor data



<https://www.flickr.com/photos/retroweb/>

# Today's Data Sources

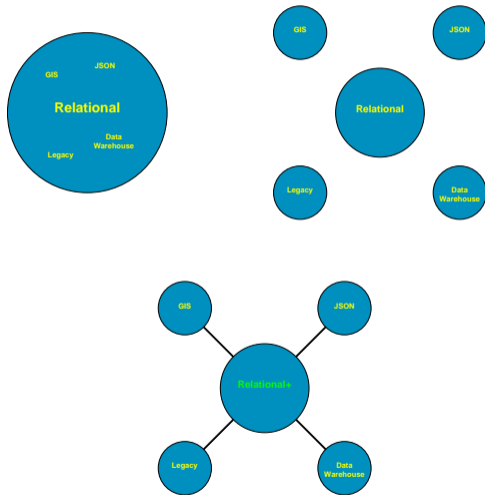


No longer constrained by data ingestion and storage constraints.

# How To Ingest New Data Sources?

- Store in a relational structure
  - must be converted to relational
  - loses original structure
  - limited indexing ability
- Use data stores specialized for each data type
  - e.g., JSON , GIS, and full-text search data stores
  - must use microservices, no monolithic data store
  - data integration and governance become difficult

# Relational+

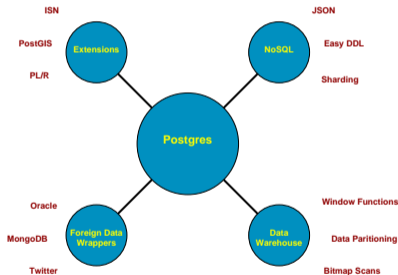


# Specialized Data Stores or Relational+

Easy For	Agile Teams	Data Integration	Data Governance
Microservices with specialized data stores	✓		
Microservices with relational+	✓	✓	✓
Monolithic architecture with relational+		✓	✓

# Postgres == Relational+

Postgres is really the only database with full relational+ features that can function in microservice and monolithic architectures. Specialized data stores still fill a need, but only when they are required.



# Why Is Postgres Relational+?

- Designed for extendability in 1986
- Ignored in 1996
- Praised for its extendability today

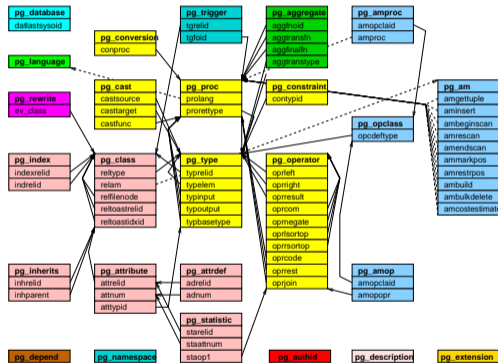


Michael Stonebraker

# How Is Postgres Relational+?

Extendability built in:

- Data types
- Indexing methods, not just btree
- Functions
- Operators
- Server-side languages



<https://momjian.us/main/presentations/extended.html#central>

# What Makes Postgres Relational+?

- Built in
  - full text search
  - JSON support
  - data warehouse capabilities
- Geographical Information System (GIS) support installed via PostGIS extension

# Mixing Relational and Non-Relational

A row can mix:

- Relational columns using btree and hash indexes
- Non-relational data like GIS using specialized index types like GIST
- Unstructured data like JSONB using specialized index types like GIN

## Data Row

INTEGER	VARCHAR	Full text search	GIS	JSONB
---------	---------	------------------	-----	-------

Consistent visibility, durability, and storage

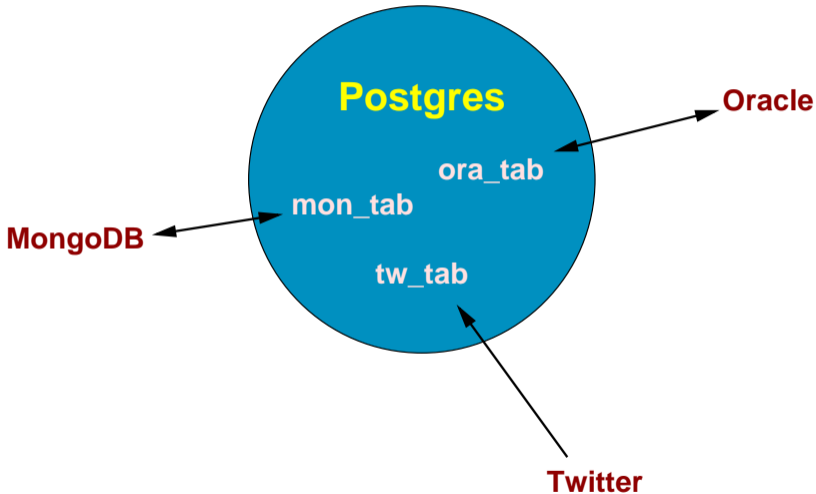
# What If Postgres Is Insufficient?

What if your workload needs:

- Data warehouse optimizations unavailable in Postgres
- Complex full text search options unavailable in Postgres and can't use Rum extension
- Columnar storage for unstructured and denormalized data with much duplication and can't use Citus extension
- Data sharding for high write volume and can't use manual sharding or pg\_shardman extension

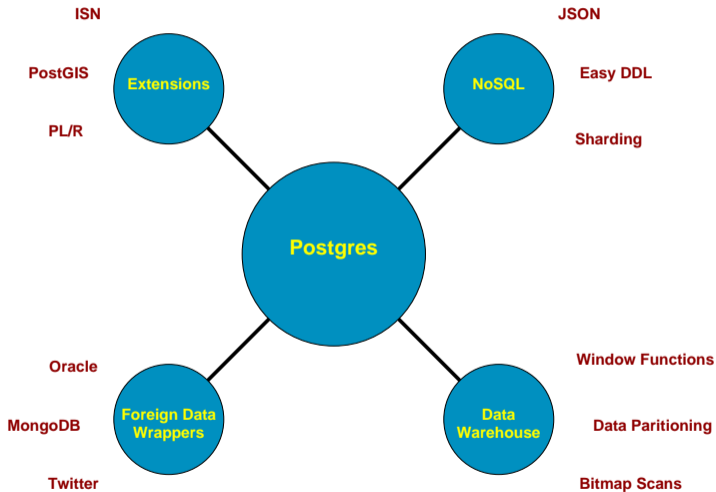
or you don't want to move all your data to Postgres, or can't yet.

# Install Foreign Data Wrappers, Query Postgres



[https://wiki.postgresql.org/wiki/Foreign\\_data\\_wrappers](https://wiki.postgresql.org/wiki/Foreign_data_wrappers)

# Postgres == Relational+



# Conclusion



<https://momjian.us/presentations>

<https://www.flickr.com/photos/marfis75/>