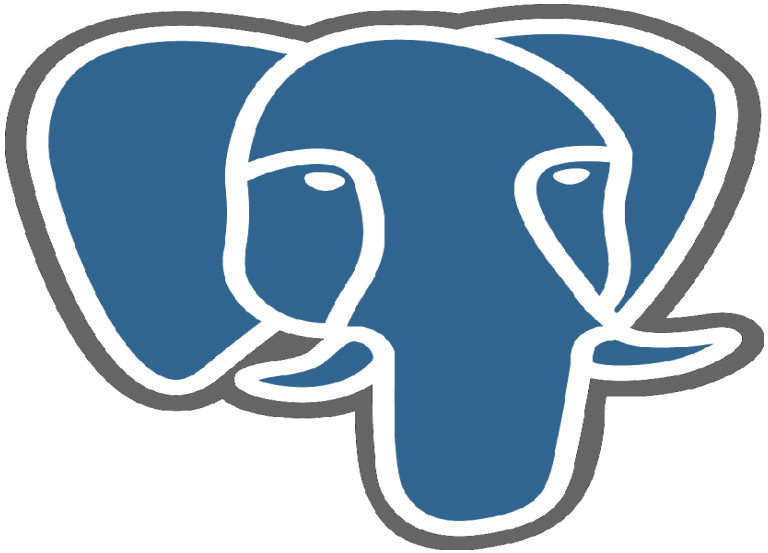
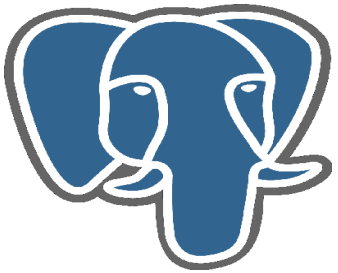
2ndQuadrant

Professional PostgreSQL

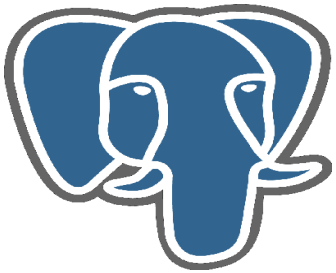
PostgreSQL and NoSQL



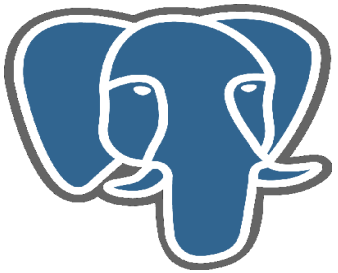


Simon Riggs

- 30 years development experience
  - Binary toggle, Pilot, Assembler, Basic, Fortran, Prolog, Pascal, COBOL, C, perl, bash, DCL, JCL, REXX, PL/SQL, blah blah blah
- 25 years professional database experience
  - Developer on pre-SQL relational analysis tool
- 20 years as a Consulting Database Architect
  - Architect of first British Airways Data Warehouse
- Database Architect at Top UK Bank/Mortgager
- Certifiable in 4 separate RDBMS

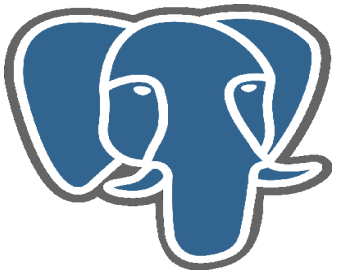


Old **** !



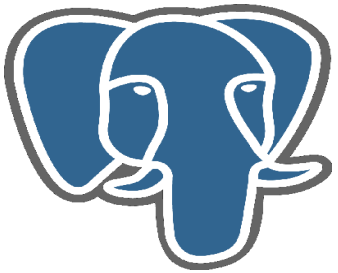
Historical Perspective

- No data storage
- Logical Key stores
- Codasyl
- Relational
- SQL
- Object Relational
- Nested Relational
- Parallel DBMS
- ROLAP
- Column stores
- Streaming data
- EAI, ETL, ELT
- XML & Documents
- GIS systems



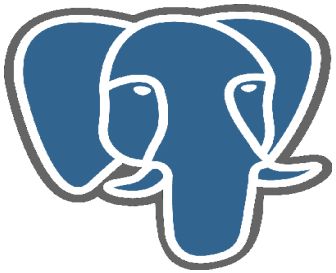
Acquisition and Assimilation

- TCP/IP
- Filesystems
- Tools
- Drivers
- SQL
- Gateways
- Database languages
- GIS

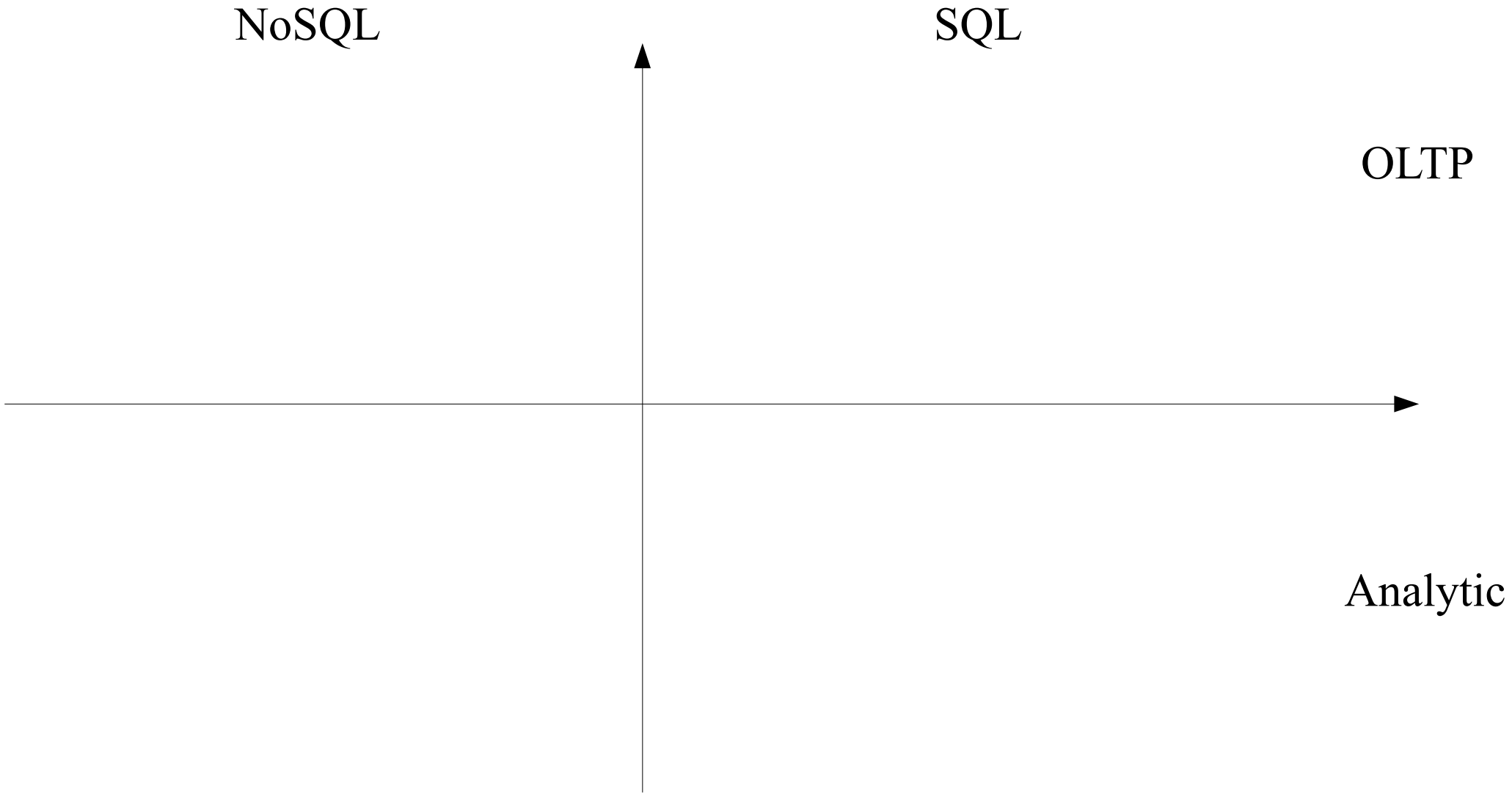


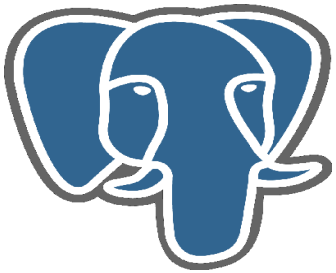
Natural and Inevitable

- Databases become **Data Operating Systems**
- Teradata now includes
  - Hybrid row/column store
- Oracle now includes
  - Replication technology from Golden Gate
  - InMemory technology from TimesTen
  - MOLAP technology from Express
  - MVCC technology from PostgreSQL

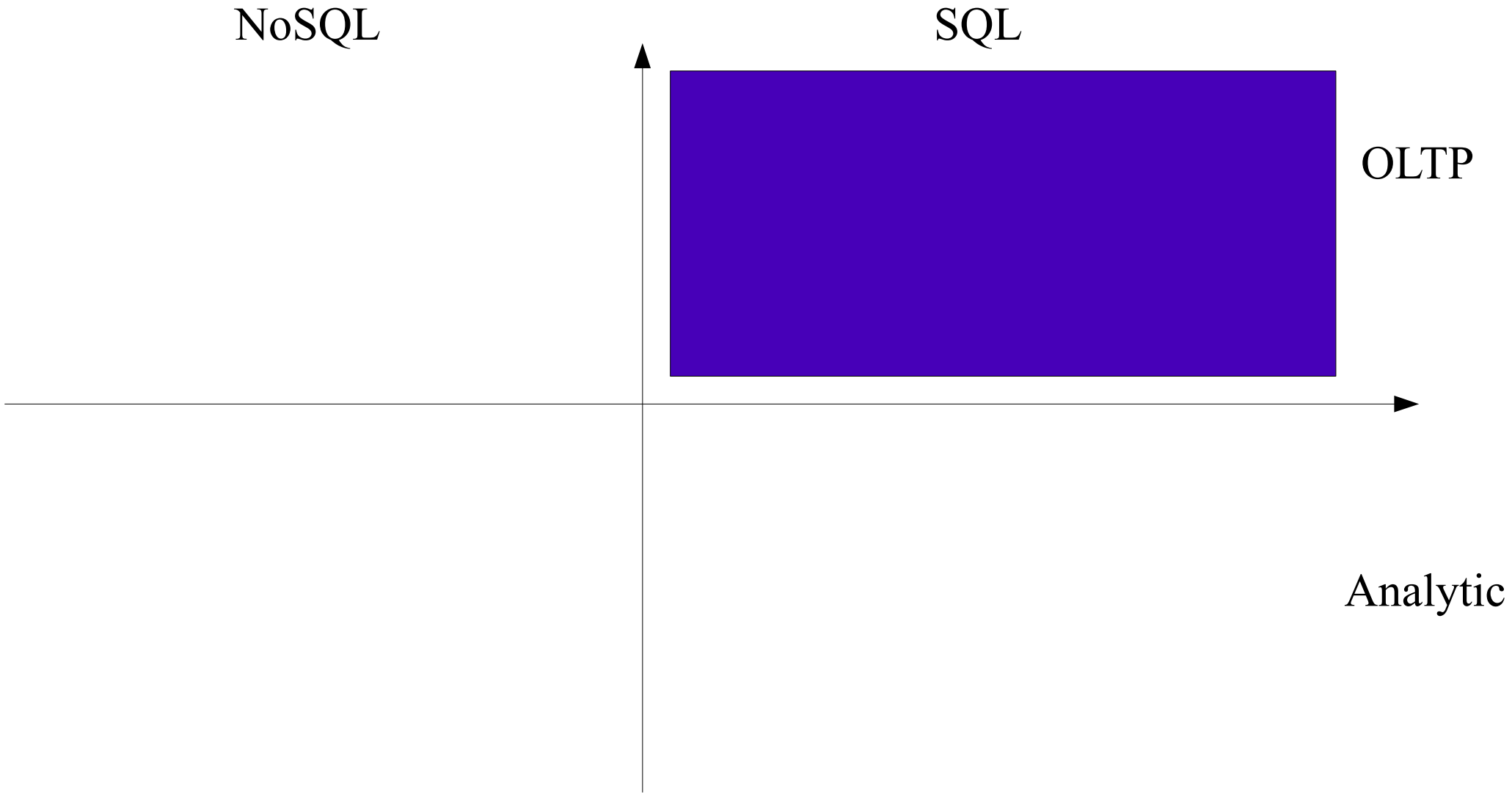


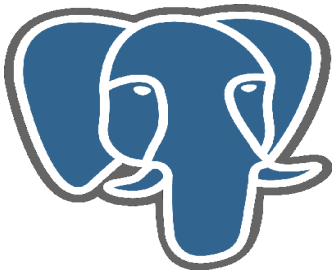
Positioning



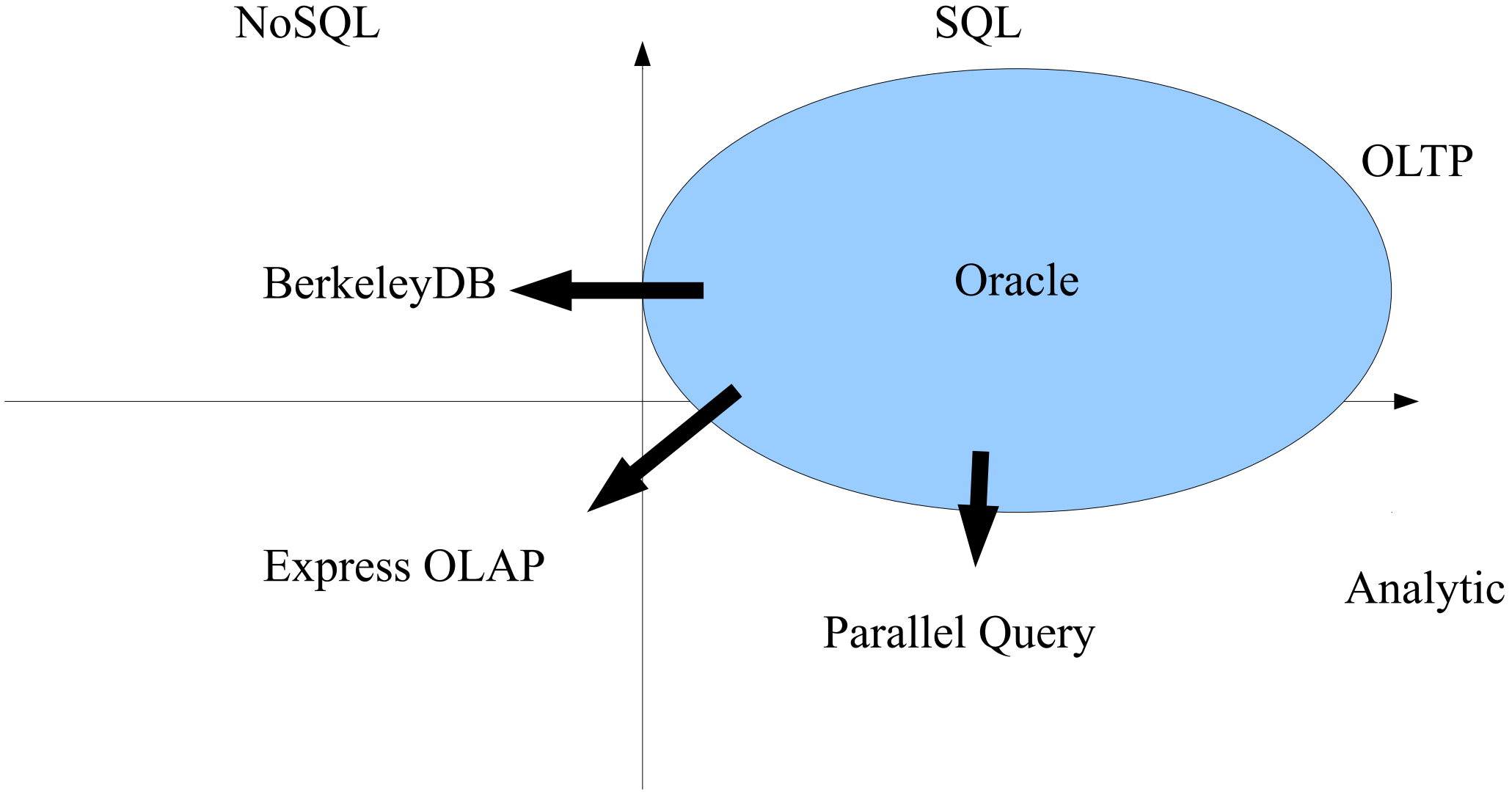


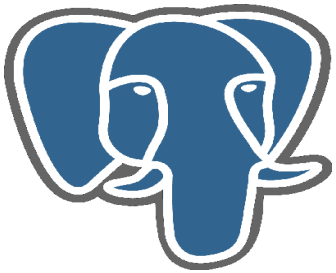
Positioning



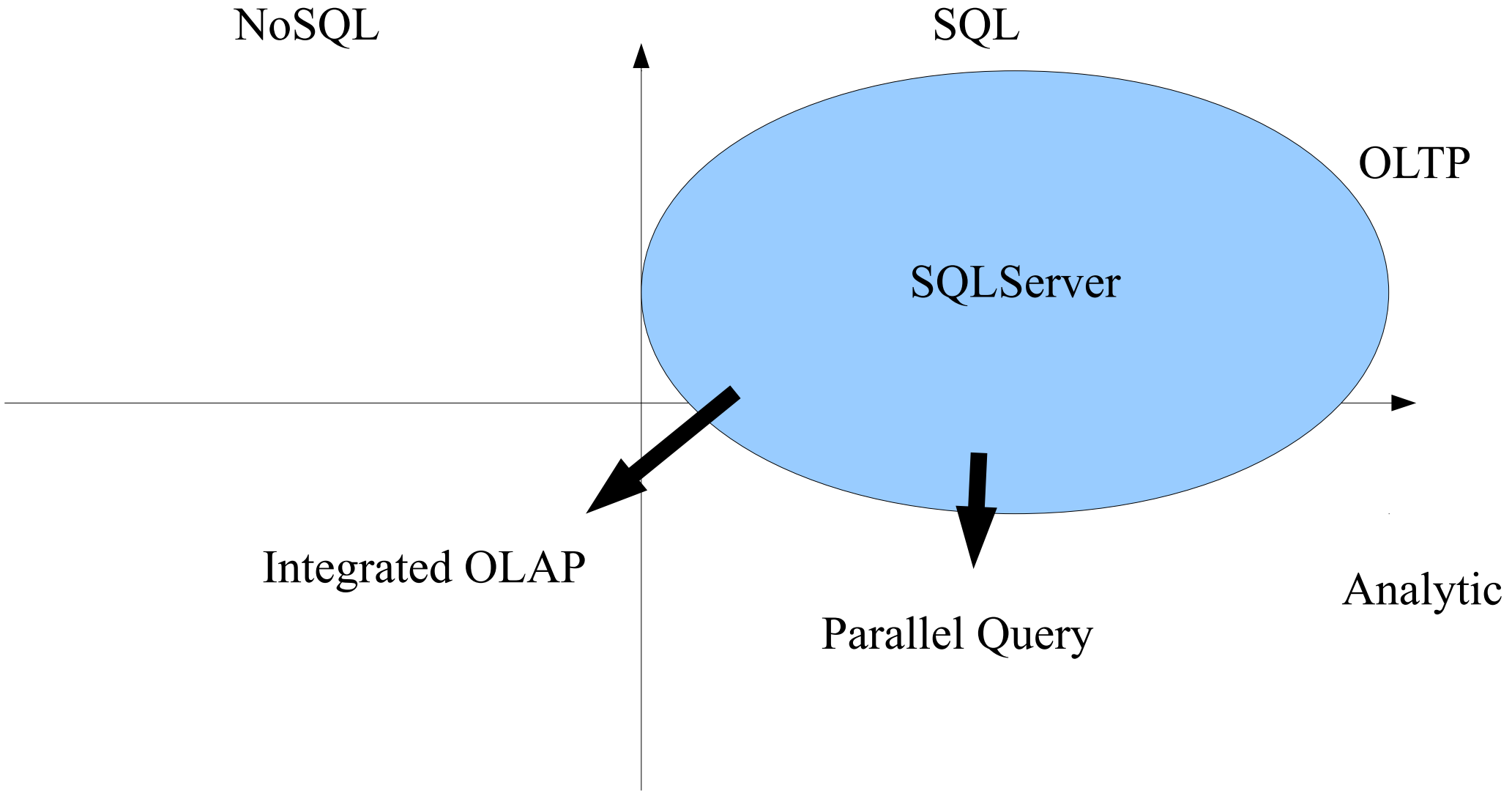


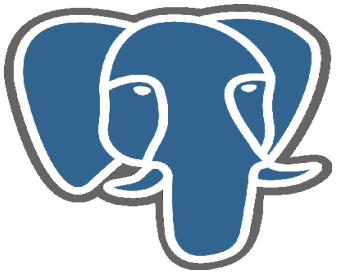
Positioning





Positioning

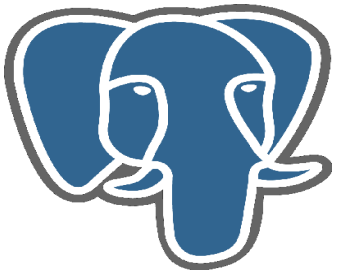




PostgreSQL includes

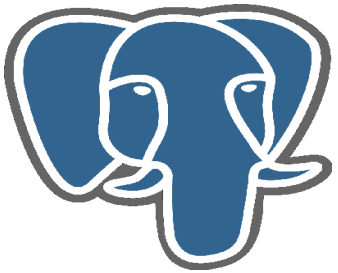
- Relational and SQL
- Object relational
- XML technology
- Extensible document datatypes (hstore)

- What isn't included? Need to look elsewhere to see what is available



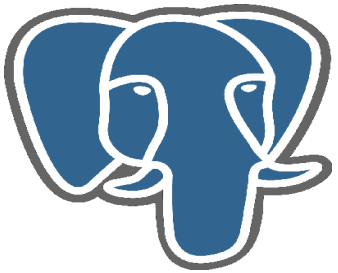
Early Non SQL Datastores

- Berkeley-DB 1986-'06
- C-ISAM 1980s
- FAME – time series database
- BLAST - Fast biodata – specialist formats
- Express 1970-1995+
- Ab Initio 2000+
- SAS/S-Plus 1966+



Later Non SQL Datastores

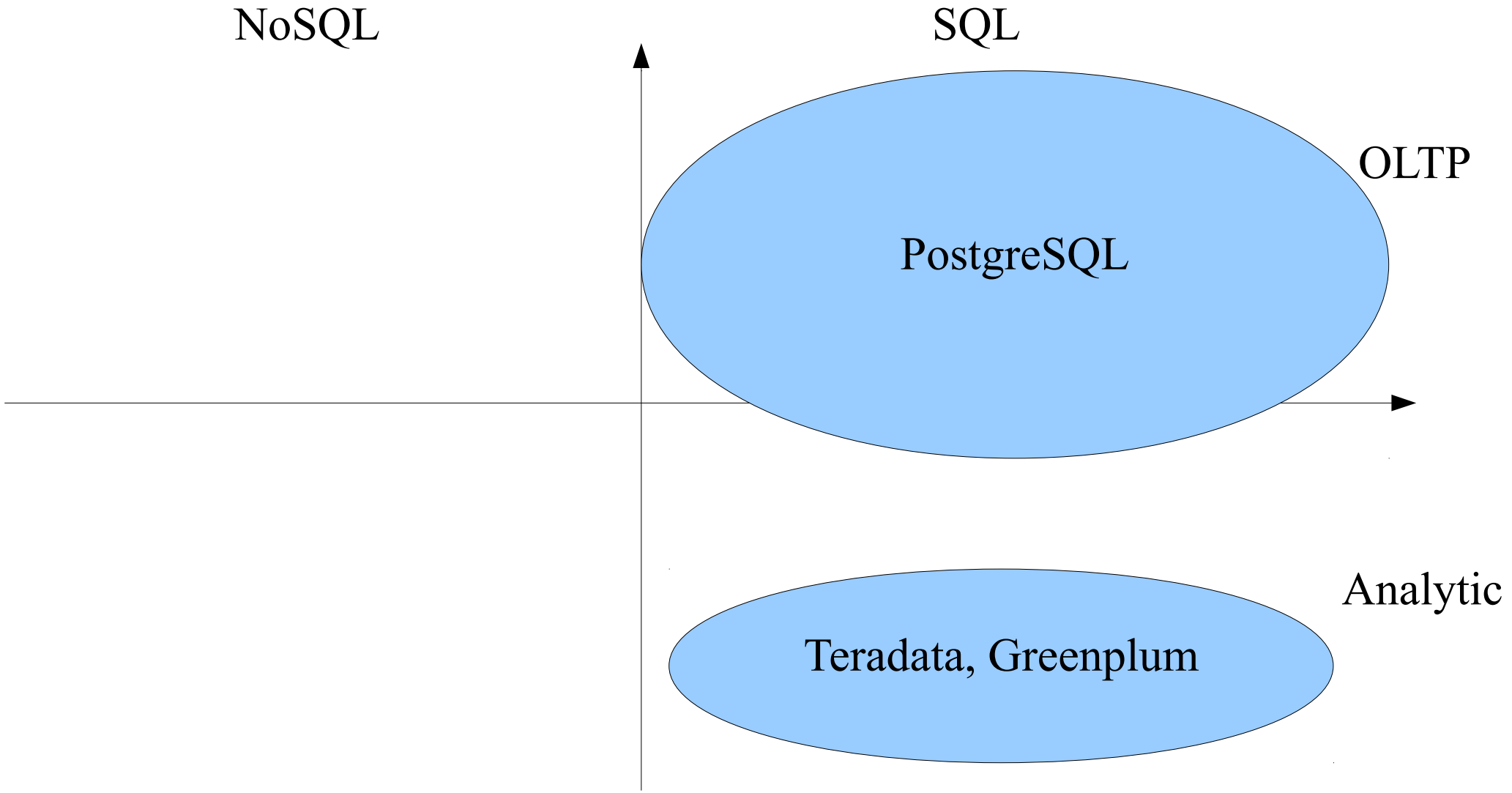
- Key access: Mongo, Cassandra, BigTable, SDS
- Specialist data formats
- MOLAP – SQLServer
- Parallel file scans – Map/Reduce - Hadoop

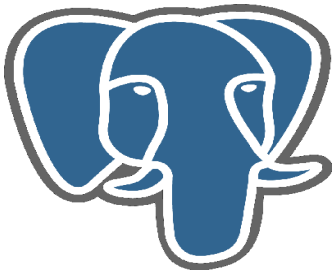


Other database products

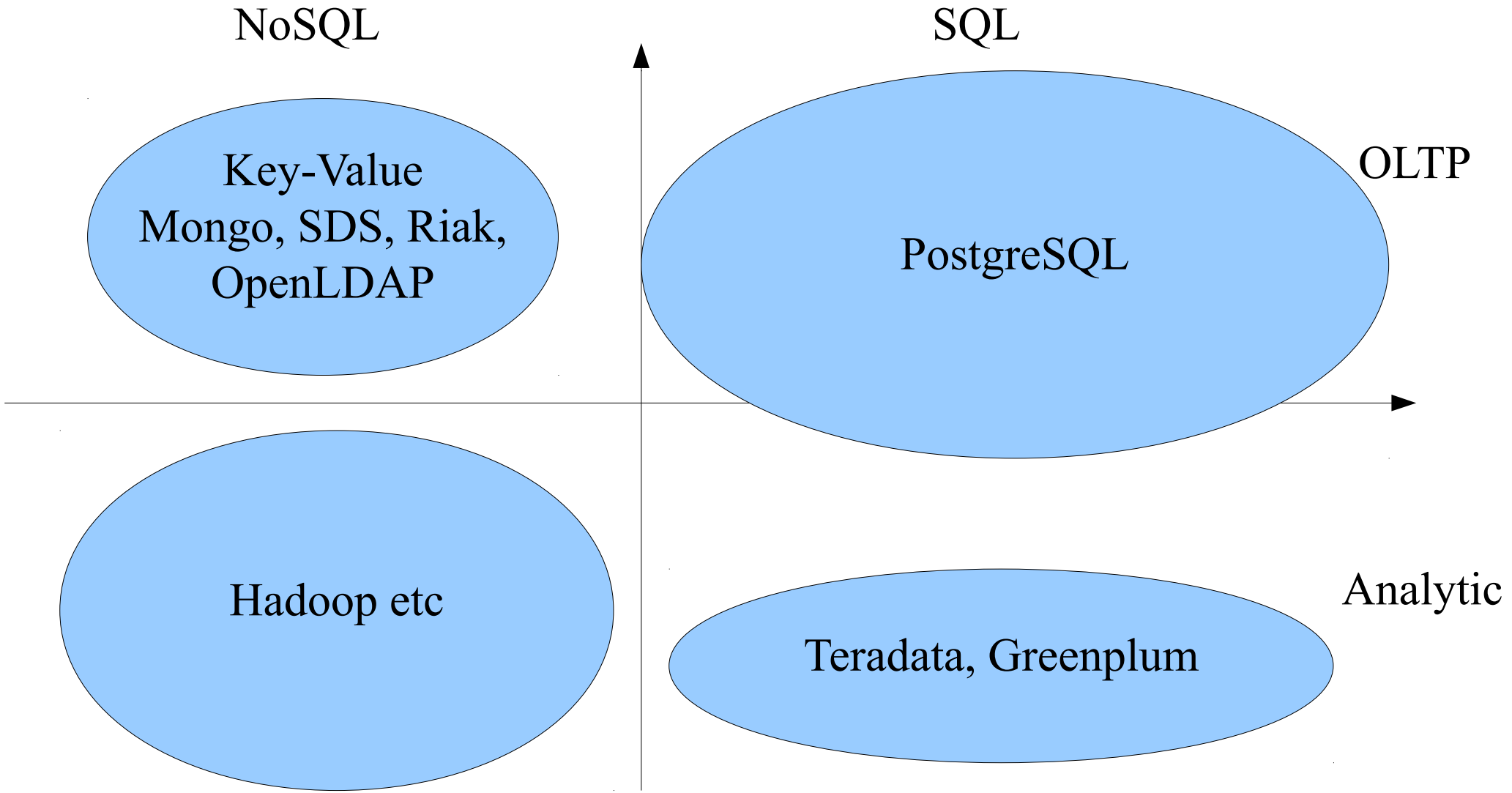
- VoltDB
- SciDB
- MayBMS
- CopperEye
- MarkLogic XML server
- JackRabbit
- Distributed hash table
  - CHORD etc

Positioning

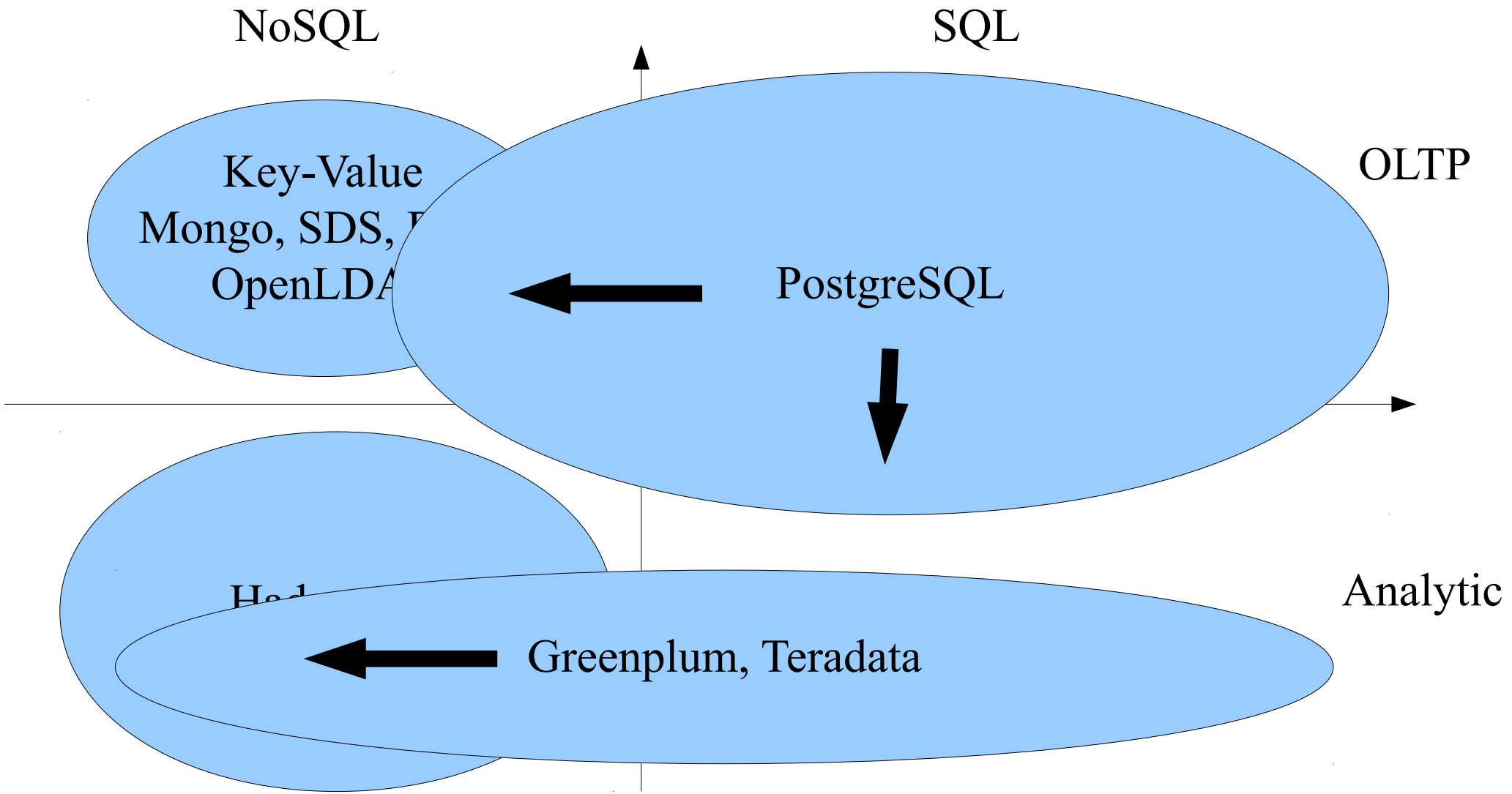


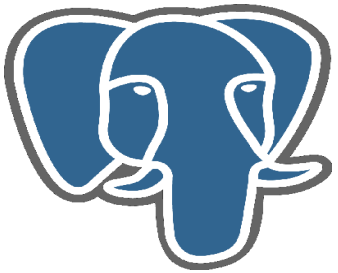


Positioning



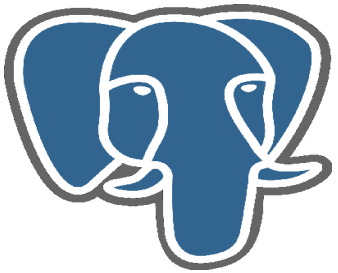
Positioning





Differentiating factors

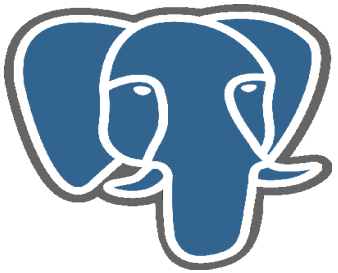
- Performance
- User API
- Scalability
- Data Model



Performance Architecture

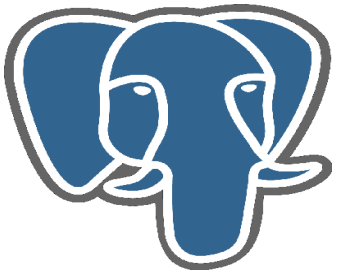
- SQL Parsing overhead
- Planning overhead
- Locking overhead 16%
- Execution overhead 14%
- Transaction Log overhead 11%
- In Memory database 34%

- “OLTP Through the Looking Glass, and What We Found There”
  Harizopolous et al, SIGMOD 2008



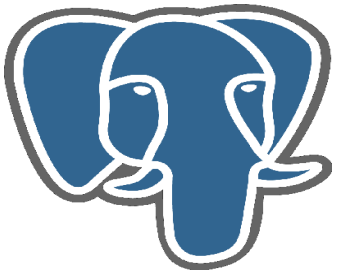
User API

- SQL API
  - API that delivers SQL language calls
  - Execute works, but Prepare/Execute is preferred
  - Many functions, many options
  - Too difficult for most developers
  - Hence many SQL API wrappers and Object Relational Mappers
- Memcache
  - Simple API with very few calls



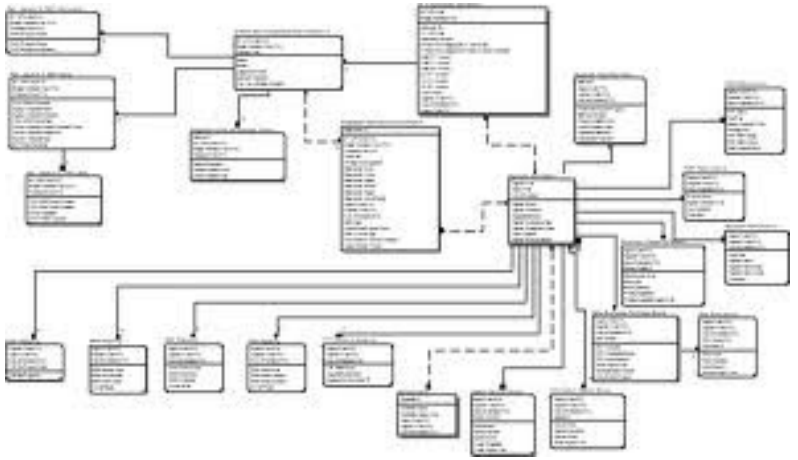
Scalability

- If all access to data is by single records only, then we can spread **data** and **data requests** across multiple nodes
- The true benefit of a simple API is it keeps requests simple, so the workload scales
- This only works for equality conditions (=)
- As soon as you do an ordered index scan it becomes an all-node operation and that is ***not scalable***

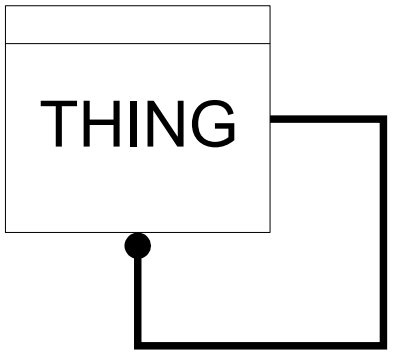


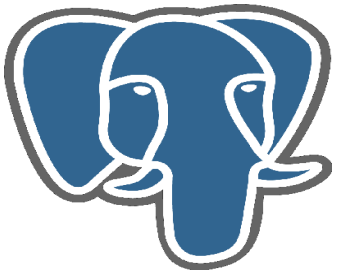
Data Model

- 3NF
  - Static Types



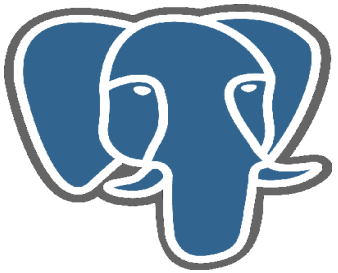
- NoSQL – 2NF
  - Just one table
  - Dynamic Types
  - Static Types





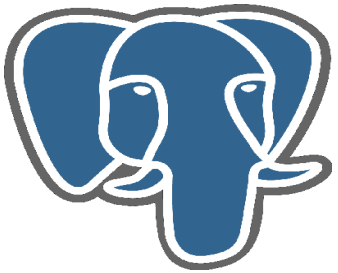
Scalability Restrictions

- You ***must*** access all your data by the Key only
  - Like having a table with only one index
- Any other indexing needs to be built by you
  - Can't just do a CREATE INDEX
- If you perform all node operations they are non-scalable and so get more expensive as you add nodes to the cluster
  - Quickly use up all the resources on the cluster
- One table, one index...
- Just like putting your application on an LDAP server



Scalability Restrictions

- If you are willing to accept those restrictions
- Why not use PL/Proxy?
- Same benefits, same restrictions as NoSQL
- But underneath, its PostgreSQL, so you have all the benefits of durability and functionality



Conclusions

- Many NoSQL concepts already in PostgreSQL
- We have much to learn from other data store servers
- Further advances arriving soon
- Scaling databases causes some hard trade-offs
- NoSQL approaches do fit some applications
  - PL/Proxy fits for all of those applications also
- General Purpose database applications **don't need** and **can't use** NoSQL
- PostgreSQL will continue to be an assimilation point for the best open source database tech