The Design of Postgres

Ruijia Mao

The Design of Postgres

- History
- Process Structure
- Design Goals & Implementation
- Comments

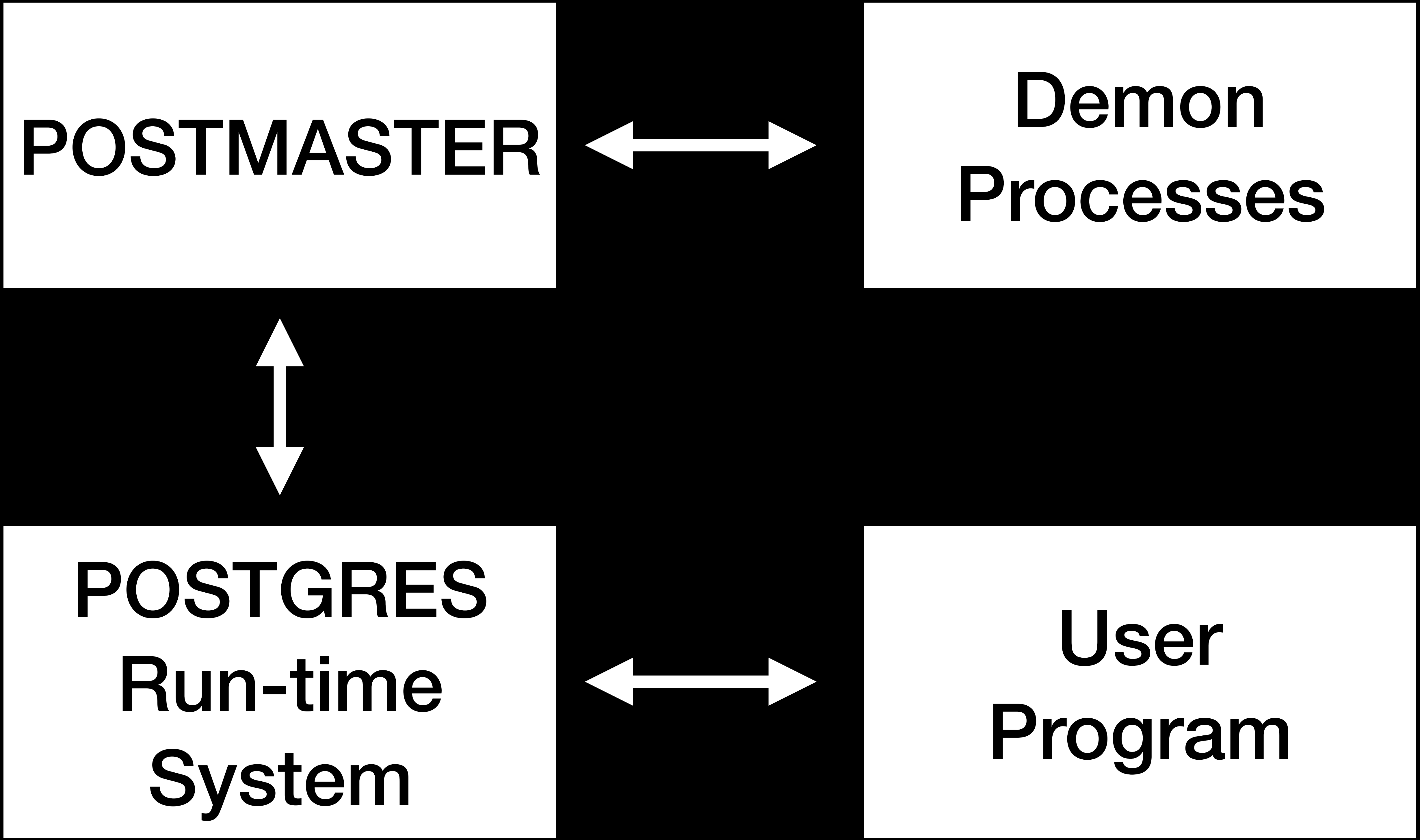
History

History

- INGRES
  - Implemented during 1975 - 1977
- POSTGRES - POST inGRES
  - POSTGRES uses POSTQUEL as its query language
- PostgreSQL

Process Structure

Process Structure



POSTMASTER

Demon Processes

POSTGRES Run-time System

User Program

Design Goals & Implementation

Design Goals

1. Provide better support for complex objects
2. Provide user extendibility for data types, operators and access methods
3. Provide facilities for active database (i.e. alerts and triggers) and inferencing including forward- and backward-chaining

Design Goals

4. Simplify the DBMS code for crash recovery
5. Produce a design that can take advantage of optical disks
6. Make as few changes as possible to the relational model

Complex Objects

Complex objects

- Can be represented by a field of type POSTQUEL

Complex objects: Example

- Polygon (id, other_fields)
- Line (id, other_fields)
- create OBJECT(name=char[10], obj=postquel)

Complex objects: Example

Name	Obj
Apple	retrieve (POLYGON all) where POLYGON.id = 10
Orange	retrieve (LINE all) where LINE.id = 17 retrieve (POLYGON all) where POLYGON.id = 11

Complex objects:
Support for Procedural Data

- Precomputation

Complex objects: Precomputation

- Compiling an access plan for POSTQUEL commands
- Executing the access path to produce an answer

Complex objects: Compilation and Fast-path

- A demon process will compile queries in idle time
- The time to parse and optimize the query is avoided
- The fast-path can accept binary form arguments and run even faster

Complex objects: Invalidate

- Use I-lock to support invalidation of plans and answers

Complex objects: Invalidate

	R	W	I
R	Ok	No	Ok
W	No	No	*
I	Ok	No	Ok

User-defined Types

User-defined Types

- Existing access methods must be usable for user-defined data types
- New access method must be definable

User-defined Types: example

- B-tree
- { <, =, >, >=, <=}

Alerts, Triggers and Inference

Alerts, Triggers and Inference

- T-lock is used to support alerts and triggers
- D-lock is used to support inference

Alerts and Triggers: example

- create EMP(name=char[20], mgr=char[20])
- retrieve **always** (EMP all) where EMP.name=“Bill”

Alerts and Triggers: compatibility matrix

	R	W	I	T
R	Ok	No	Ok	Ok
W	No	No	*	#
I	Ok	No	Ok	Ok
T	Ok	No	Ok	Ok

Inference

Inference: example

- The employees need to work 8 hours a day.
- The salary for new employees are $14 per hour.

Alerts and Triggers: compatibility matrix

	R	W	I	T	D
R	Ok	No	Ok	Ok	&
W	No	No	*	#	No
I	Ok	No	Ok	Ok	Ok
T	Ok	No	Ok	Ok	Ok
D	Ok	No	*	#	Ok

Crash Recovery

Crash Recovery

- Force: When a transaction commits, it is pushed to the disk
- Steal
- When a crash is observed, abort all active transactions

Comments

Comments

- This paper adds supports for user-defined types, alerting and triggers, and other things.
- The POSTGRES database evolves to one of the leading open-source relational databases
- The process structure is relatively simple. Modern PostgreSQL uses client-server model.

Q&A

Thanks