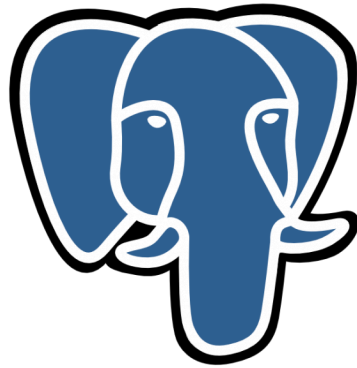


# Design of Postgres



PostgreSQL

---

PRESENTED BY: TARANJEET GILL

SIMON FRASER UNIVERSITY

# Design of Postgres

---

## **Outline**

What is Postgres?

Why Postgres?

Design Goals of Postgres

POSTQUEL

Programming Language Interface

# What is Postgres?

---

- **post-Ingres** is successor to INGRES implemented in 1970s by Michael Stonebraker
- Open Source object relational database system
- Highly reputed for data integrity, correctness
- ACID compliant
- Supports many data types- INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE
- Native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby

# Why Postgres?

---

## Existing relational database system limitations

- Able to solve business data processing problems
- Lacks support for engineering applications e.g. CAD systems, geographic data and graphics
- The inclusion of substantial new function is extremely difficult.

# Three main points

---

- Design goals of Postgres
- POSTQUEL (Query language supported by Postgres)
- Programming Language Interface

# Design goals of Postgres

---

First goal is to support complex objects such as POLYGON,CIRCLE,LINE

This info can be stored in a single relation. Traditional command

```
foreach OBJ m {POLYGON, CIRCLE, LINE} do range of 0 1sOBJ  
range of D 1sDISPLAY  
retrieve (D all, 0 all)
```

```
where D obje&id = 0 id and D obj-type = OBJ
```

- Not fast enough to paint screen in real time
- To show these objects on the screen information such as color,position, scaling factor is needed.
- POSTGRES will only need one display field to show these objects

# Design goals of Postgres

---

Second goal is to support new

- Data types
- Operators
- User Defined Access methods

# Design goals of Postgres

---

The third goal is to provide active database facility and rules.

- Alerts
- Triggers



# Design goals of Postgres

---

Fourth goal is to shorten the length of code written for crash recovery by treating the log information as normal data managed by DBMS.

Existing crash recovery code limitations-

- Tricky
- Full of special cases
- Difficult to test and debug

# Design goals of Postgres

---

Next goal is utilize advanced technology

- Optical disk – price performance, storage hierarchy and reliability.
- Workstation sized processors with several CPUs
- Custom designed VLSI chips

# Design goals of Postgres

---

Last goal is to make minimal changes to the existing framework.

## **Why?**

- Business community is familiar with it
- Model should be small, simple and extendible

# POSTQUEL

---

## **Commands included-**

Create Relation, Destroy Relation, Append, Delete, Replace, Retrieve, Define View

## **Commands excluded-**

Modify- specifies storage structure of relation

# POSTQUEL

---

## Data Definition

- Integers, Floating Point, Fixed length character strings, varying length of arrays
- POSTQUEL
- Procedure

EMP(name, age, salary, hobbies, dept)

execute(EMP hobbies)

where EMP name="Smith"

# POSTQUEL

---

## Complex objects

Name	Object
apple	retrieve (POLYGON all) where POLYGON id = 10 retrieve (CIRCLE all) where CIRCLE id=40
orange	retrieve (LINE all) where LINE ID = 17 retrieve (POLYGON all) where POLYGON id=10

create OBJECT(name=char[10],obj=postquel,display=cproc)

# POSTQUEL

---

**Time Varying Data** (To save and query historical data)

- Retrieve (E all) from E in EMP["7 January 1985"]
- Discard EMP before "1 week"
- Retrieve (E all) from E in EMP[] where E name = "Smith"

**Three level memories –**

- Main memory
- Secondary memory (Magnetic Disk)
- Tertiary memory (Optical Disk)

# POSTQUEL

---

## Iteration Queries, Alerters, Triggers and Rules

### Alerter

retrieve always (EMP all)  
where emp name = "Bill"

### Trigger

delete always DEPT  
where count(EMP.name by DEPT.dname  
where EMP.dept = DEPT.dname) = 0



# Programming Language Interface

---

## **Hitching Post Objectives**

Simplify the development of browsing style applications

Powerful enough so that all programs including adhoc terminal monitor could be written with the interface

Application developer can tune the program performance (to trade flexibility and reliability for performance)

# Programming Language Interface

---

## Portals

- Portals are similar to cursors
- The concept of portal is based on buffers

Example if user wants to scroll down half a screen

move P forward 10

fetch 20 into P

# Programming Language Interface

---

## Compilation

CODE - System catalog which can store queries to compile

append to CODE(id=1, owner="browser",  
command="replace EMP(salary=\$1) where EMP.name=\$2")

execute (CODE.command)  
with (*NewSalary*, "Smith")  
where CODE.id=1 and CODE.owner="browser"

## Fast Path

exec-fp(1, "browser", *NewSalary*, "Smith")

# Summary

---

- Support for complex objects is simple, easy to use and extendible
- Triggers and alerts prove to be vital for data integrity
- Programming interface supports many embedded query languages
- Achieved desired goals with minimal changes to the existing relations system

# Thank you

---

Questions?