Extensible Systems

- Need to support new data types/operators
  - GIS, CAD, document, time-series data, image databases
  - ...
  - More recently: XML
    - Simple XML can be easily supported using OR facilities
    - See “What goes around comes around”; Chapter 1, Redbook

- Trying to fit everything into SQL can be tricky
  - Intuitive
  - Hard to write queries
  - Need support for new types of queries
  - Can’t optimize or index etc..
Extensible Systems: Needs

- **User interface level**
  - Adding new data types and new operations on the data
  - Extend query language with new set of operators

- **Lower level**
  - New implementations of relational operators
  - New index structures (GiST)
  - Defining new data storage methods

- **Also:**
  - Query optimizer must understand how all these things interact, affect execution etc.
    - Not just for finding the optimal cost plan
    - But also for coming up with query plans that use these operators
Object-oriented vs Object-relational

Object-oriented

- Shrink the impedance mismatch for app programmers
- Semantically richer data models
- New language features (e.g. complex programming)
- Niche Market
Object-oriented vs Object-relational

Object-oriented
- Shrink the impedance mismatch for app programmers
- Semantically richer data models
- New language features (e.g. complex programming)
- Niche Market

Object-relational
- Storage and querying over complex data types

<table>
<thead>
<tr>
<th></th>
<th>Simple Data</th>
<th>Complex Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query</strong></td>
<td>RDBMS</td>
<td>ORDBMS</td>
</tr>
<tr>
<td><strong>No Query</strong></td>
<td>File System</td>
<td>OODBMS</td>
</tr>
</tbody>
</table>
Discussion from: “Extensible Database Management Systems”; Carey and Haas

Approach 1: Support for ADTs, UDFs, user-defined indexes etc...
- Postgres the most well-known (others: ADT-INGRES, RAD, PROBE, $R^2D^2$)
- Support full, end-user DBMS with “hooks”

Approach 2: Layered, toolkit approach
- Provide a database “kernel” with storage and transaction management, and a library of components
- Build your own database on top
- Need more work to extend
- Examples: DASDBS, GENESIS, EXODUS
Outline

1. Overview
2. Postgres
3. Other Issues
INGRES-ADT → Post-INGRES

Query language: “quel”

Ability to add new data type. Type defined by:

- storage size
- input and output methods (functions to convert to-and-from strings)
- other methods

Dynamic linking vs static linking

Trusted code ???
INGRES-ADT → Post-INGRES

Query language: “quel”

Ability to add new data type. Type defined by:
- storage size
- input and output methods (functions to convert to-and-from strings)
- other methods

Ability to add new operators
- input/output types, precedence etc.

Methods called via “function pointers”

Dynamic linking vs static linking

Trusted code ???
Extensibility in PostgreSQL
Types
User-defined Functions
Indexes and GiST
Access Methods

- B+-Trees can be used for any object that supports a set of functions
- Similarly, hashing can be used if $H(\text{key})$ is supported
- Requirements, different data types, selectivities all stored in relations
- GiST!

Interactions with transaction management

- The hardest part for access methods
- Recall that concurrency/recovery highly customized for B+-Trees
- Somewhat okay if you only physical logging
- Otherwise, can expose some of the API
Query Processing

- must be able to compute selectivities
- need to know if we can use sort-merge join, hash join
- figure out which access methods can be used for executing a query
- rules to specify these things
- fairly easy to make dynamic programming (Selinger et al) work with it
ORDBMS

- Postgres → Illustra → Informix → IBM
  - **Datablades**: Packages that supported everything for a new data type
  - From IBM webpage: C-ISAM, Text, Geospatial data, Image data, Spatial data, Time-series, Video, Web
Postgres $\rightarrow$ Illustra $\rightarrow$ Informix $\rightarrow$ IBM

- **Datablades**: Packages that supported everything for a new data type
- From IBM webpage: C-ISAM, Text, Geospatial data, Image data, Spatial data, Time-series, Video, Web

Postgres $\rightarrow$ PostgreSQL (around 1995)

- Named so because it supported SQL (as opposed to only Quel)
- Postgres was BSD license
Postgres → Illustra → Informix → IBM

**Datablades**: Packages that supported everything for a new data type

From IBM webpage: C-ISAM, Text, Geospatial data, Image data, Spatial data, Time-series, Video, Web

Postgres → PostgreSQL (around 1995)

Named so because it supported SQL (as opposed to only Quel)

Postgres was BSD license

Very successful

Most database systems support it in one form or other

SQL:99 has support for it
Query Optimization

- Postgres
  - Largely relational, so no significant new issues
  - The optimizer needs to be able to estimate selectivities

- EXODUS
  - Query optimizer was top-down
    - Repeatedly apply algebraic transformations to query plans to create new plans
  - Had a *query optimizer generator*, which generated an optimizer given:
    - operations of the query algebra
    - the available operators
    - a set of query transformation rules
    - a set of implementation rules
  - Also used a set of C support routines – for cost estimation etc.
Starburst (IBM)
- Also rule-based
- However, optimized for relational data
- So dynamic programming was already built-in

Starburst also had some interesting ideas for extensible storage and access methods