Basis Data Non Relasional
Object Relational Systems

Tricya Widagdo
Program Studi Teknik Informatika
Institut Teknologi Bandung

Requirements of A Third Generation DBMS

[Stonebraker *et al*, 1990]

1. Provide traditional database services plus richer object structures and rules
   - Rich type system
   - Inheritance
   - Functions and encapsulation
   - Optional system-assigned tuple IDs
   - Rules (e.g. integrity rules), not tied to specific objects
2. Subsume second generation DBMSs
   - Navigation only as a last resort
   - Intensional and extensional set definitions
   - Updatable views
   - Clustering, indexes, etc., hidden from the user
3. Support open systems
   - Multiple language support
   - Persistence orthogonal to type
   - SQL
   - Queries and results must be the lowest level of client/server communication

Why Subsuming Second Generation DBMSs?

- Second generation systems made a major contribution in two areas:
  - Non procedural data access
  - Data independence
- Second generation systems are not *ad hoc*, i.e. they rest on a solid theoretical foundation
  - ➢ Enhancing the relational systems to incorporate the good features of object technology, i.e. proper data type support (including type inheritance)
    - ➢ DOMAIN. No need to do anything to the relational model in order to achieve object functionality
- This opinion is *not* shared by some of the object products, nor by some object writers

Problem with 2nd Generation DBMSs - Example

```
CREATE TABLE RECTANGLE
    ( X1…, Y1…, X2…, Y2…, …
      UNIQUE (X1, Y1, X2, Y2) );
```

"Get all rectangles that overlap the unit square (0,0,1,1,)"

Problem with 2nd Generation DBMSs – Example (2)

```
SELECT …
   FROM RECTANGLE
   WHERE (X1>=0 AND X1<=1 AND Y1>=0 AND Y1<=1)
      OR (X2>=0 AND X2<=1 AND Y2>=0 AND Y2<=1)
      OR (X1>=0 AND X1<=1 AND Y2>=0 AND Y2<=1)
      OR (X2>=0 AND X2<=1 AND Y1>=0 AND Y1<=1)
      OR (X1<=0 AND X2>=1 AND Y1<=0 AND Y2>=1)
      OR (X1<=0 AND X2>=1 AND Y1>=0 AND Y1<=0)
      OR (X1>=0 AND X1<=1 AND Y1<=0 AND Y2>=1)
      OR (X2>=0 AND X2<=1 AND Y1<=0 AND Y2>=0)
      OR (X1<=0 AND X2>=1 AND Y2>=0 AND Y2<=1);
```

Try to find the "non obvious" one!

Solution by Using O/R DBMSs

- Define a *rectangle type* (RECT, say)

```
TYPE RECT POSSREP ( X1 RATIONAL, Y1 RATIONAL,
                    X2 RATIONAL, Y2 RATIONAL );
```

- Define an operator to test whether two given rectangles overlap

```
OPERATOR OVERLAP ( R1 RECT, R2 RECT)
   RETURNS ( BOOLEAN );
   RETURN ( THE_X1(R1) ≤ THE_X2(R2) AND
            THE_Y1(R1) ≤ THE_Y2(R2) AND
            THE_X2(R1) ≥ THE_X1(R2) AND
            THE_Y2(R1) ≥ THE_Y1(R2) );
END OPERATOR;
```

Solution by Using O/R DBMSs (2)

- Create a base relvar with an attribute of type RECT

```
VAR RECTANGLE RELATION { R RECT, … }
   KEY { R };
```

- The query “Get all rectangles that overlap the unit square” become:

```
RECTANGLE WHERE
   OVERLAP ( R,
         RECT ( 0.0, 0.0, 1.0, 1.0 ) );
```

The Blunders

- The crucial preliminary question:
  - " What concept is it in the relational world that is the counterpart to the concept *object class* in the object world? "
  - – Possible answers:
    - Domain = object class (√)
    - Relvar = object class (X)
      - – Relvars are variables and classes are types
      - – Relvars are not domains
    - ➢Many people and some products have embraced the second equation ⇒ The first great blunder [Date, 2000]

The Blunders (2)

- Mixing pointers and relations ⇒ The second great blunder [Date, 2000]
  - This blunder might be the logical consequence of the first one, but can be committed in its own right
  - If relvar *R1* is allowed to have an attribute whose values are pointers into relvar *R2*
    - Those pointers point to tuple variables, not to tuple values (since pointers point to variables, not values)
    - There is no tuple variable in the relational model
      - The only variable is relation variable
  - ➢ A major departure from the relational model

O/R System

Should be able to support [Date, 2000]:

- *Ad hoc* query, view definition, and declarative integrity constraints
- Methods that span classes
- Dynamically defined classes
- Dual mode access
- Deferred (COMMIT-time) integrity checking
- Transition constraints
- Semantic optimization
- Relationships of degree greater than two
- Foreign key rules
- Optimizability

O/R System – In Addition

- OIDs and pointer chasing are now totally hidden from the user
- "Difficult" object questions go away
- The benefits of encapsulation still apply, but to scalar values within relations, not to relations
- Relational systems can now handle "complex" application areas
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations while extending modeling power.
- Upward compatibility with existing relational languages.

Stonebraker's Object Relational Database – The Third Wave

Klasifikasi Aplikasi

	Simple Data	Complex Data
Query		
No Query		

- Bagian Kiri Bawah:
  - Contoh: paket teks editor standar
  - Proses: buka file, sunting objek di memori, tutup file
  - Satu-satunya query: ambil file
  - Satu-satunya update: simpan file
  - Model data: File System

Klasifikasi Aplikasi (2)

- Bagian Kiri Atas:
  - Contoh: business data processing
  - Data disimpan dalam bentuk record terstruktur
  - Query untuk mengambil record-record tersimpan
  - Client tools yang memungkinkan pembuatan form untuk data entri dan tampilan (= 4GLs)
  - Pemrosesan transaksi oleh banyak pemakai, biasanya dalam bentuk perintah SQL
  - Kebutuhan: data tidak pernah hilang
  - Security penting: arsitektur client-server
  - Model data: Relational

Klasifikasi Aplikasi (3)

- Bagian Kanan Bawah:
  - Contoh: Electronic CAD (ECAD)
  - Dibutuhkan proses loading, converting, unloading, dan reconverting data yang rumit.
  - Solusi: penyimpanan persistent yang memiliki rutin kompaksi.
  - Fokus: menyediakan integrasi yang erat dengan bahasa pemrograman, performansi yang tinggi bagi update terhadap variabel persistent
  - Model data: Object-oriented

Klasifikasi Aplikasi (4)

- Bagian Kanan Atas:
  - Contoh: librari untuk foto digital, medical imaging, digital library, dan kebanyakan aplikasi basis data ilmiah.
  - Query dilakukan terhadap data dengan struktur yang kompleks —› dibutuhkan bahasa query yang memungkinkan fungsi dan operator user-defined
  - Performansi sangat dipengaruhi oleh proses optimasi yang dilakukan
  - Model data: Object-relational (= kombinasi SQL dan primitif-primitif pemodelan)