OODBMS: Introduction and Logical Database Design

Why OO?

- Relational Systems are limited:
  - Structural restrictions on data
  - Missing semantics (value-based relationships)
  - Linguistic limitations (SQL and Algebra)
- PL community's OO work is appealing:
  - More "realistic" data structures
  - Explicit relationships and behavior modeling
  - "Tighter" interface between DBMS and PL
- New applications:
  - CAD, OIS, hypertext, geograph. data, multimedia, medical data, music, hierarchical data, ...

Fundamental OO Concepts

- Complex object structure
- Explicit relationships
- Object identity: globally unique OIDs
- Methods (behavior) an inherent part of model
  - used to model integrity constraints!
  - written in a "real" programming language
- Subclasses and inheritance
  - structure (attributes) and behavior (methods)
- Private vs. public attributes and methods

OODBMS Required Features

- Complex Objects (set, tuple, list)
- OID (value-independent, permanent)
- Encapsulation (overriding it?)
- Classes/Types (maintain extents?)
- Subclasses (multiple superclasses?)
- Late binding for overridden methods
- Turing-complete host language
- Seamless type extensibility

OODBMS Required Features (cont)

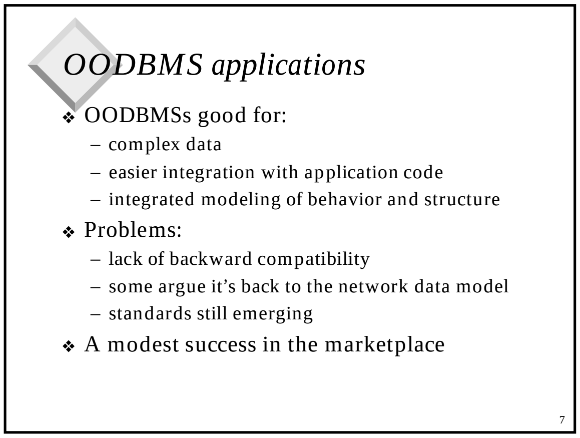
- Persistence enforced by system
- Handle large DBs (indexing, buffering, etc.)
- Concurrency support
- Recovery support
- Must provide a simple (declarative, optimizable) query language
- Separate constraint mechanisms?
- Views?

Solution 1: Object-Oriented DBMS

- Idea: Take an OO language like C++, add persistence & collections.

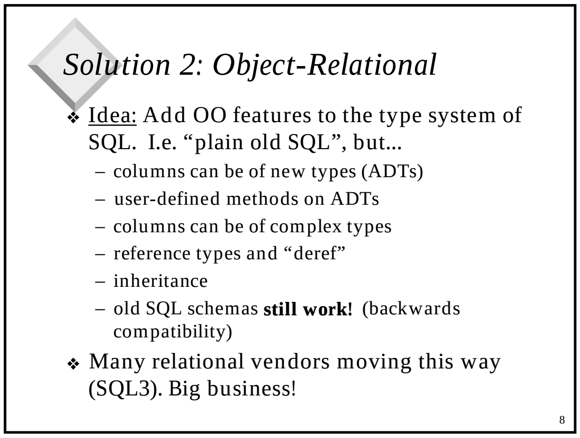
```
class frame {
  int frameno;
  jpeg *image;
  int category;
}
persistent set <frame *> frames;
foreach (frame *f, frames)
  return f->image->thumbnail();
```

- Shut down the program. Start it up again. Persistent vars (e.g. frames) retain values!



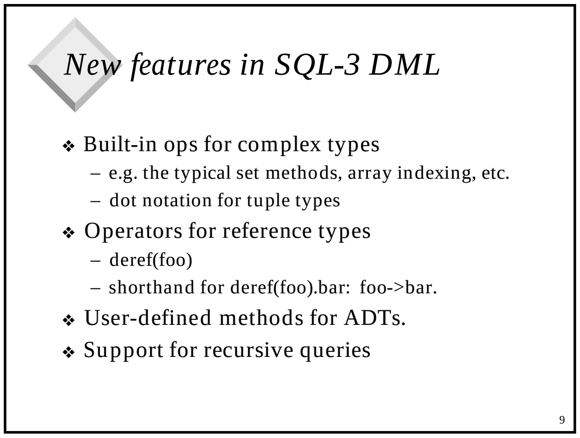
OODBMS applications

- OODBMSs good for:
  - complex data
  - easier integration with application code
  - integrated modeling of behavior and structure
- Problems:
  - lack of backward compatibility
  - some argue it's back to the network data model
  - standards still emerging
- A modest success in the marketplace



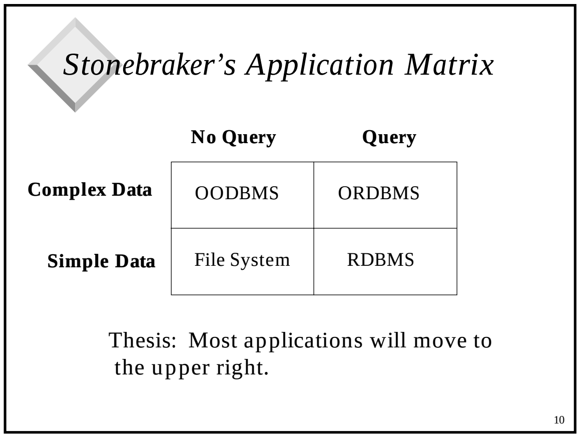
Solution 2: Object-Relational

- Idea: Add OO features to the type system of SQL. I.e. "plain old SQL", but...
  - columns can be of new types (ADTs)
  - user-defined methods on ADTs
  - columns can be of complex types
  - reference types and "deref"
  - inheritance
  - old SQL schemas **still work!** (backwards compatibility)
- Many relational vendors moving this way (SQL3). Big business!



New features in SQL-3 DML

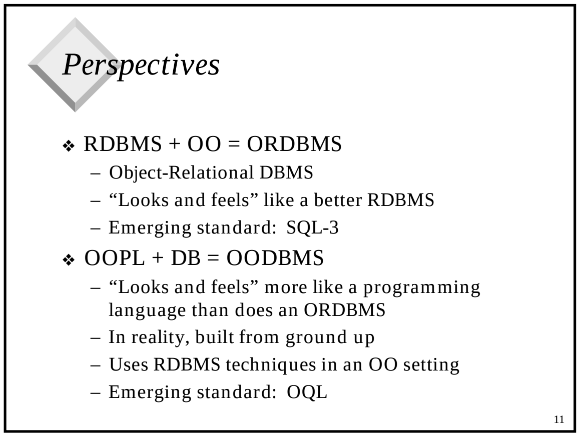
- Built-in ops for complex types
  - e.g. the typical set methods, array indexing, etc.
  - dot notation for tuple types
- Operators for reference types
  - deref(foo)
  - shorthand for deref(foo).bar: foo->bar.
- User-defined methods for ADTs.
- Support for recursive queries



Stonebraker's Application Matrix

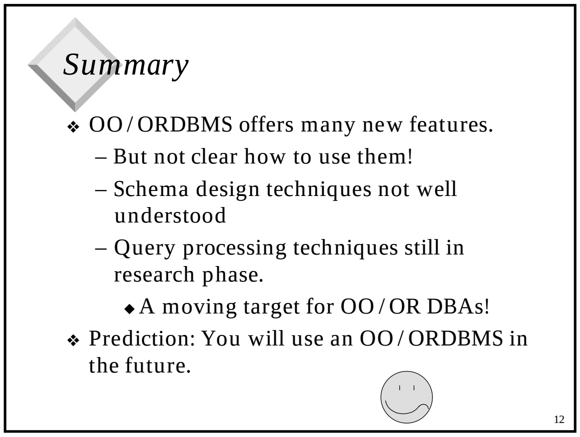
	No Query	**Query**
Complex Data	OODBMS	ORDBMS
Simple Data	File System	RDBMS

Thesis: Most applications will move to the upper right.



Perspectives

- RDBMS + OO = ORDBMS
  - Object-Relational DBMS
  - "Looks and feels" like a better RDBMS
  - Emerging standard: SQL-3
- OOPL + DB = OODBMS
  - "Looks and feels" more like a programming language than does an ORDBMS
  - In reality, built from ground up
  - Uses RDBMS techniques in an OO setting
  - Emerging standard: OQL



Summary

- OO / ORDBMS offers many new features.
  - But not clear how to use them!
  - Schema design techniques not well understood
  - Query processing techniques still in research phase.
    - A moving target for OO / OR DBAs!
- Prediction: You will use an OO / ORDBMS in the future.

Current Products

- Some OR features supported in:
  - Oracle 8
  - IBM DB2
  - Informix UDS
  - UniSQL
- Some OODBMS products:
  - O2
  - ObjectStore
  - Objectivity
  - Versant, Jasmine, Titanium, Poet, …

State of the Art (general OO/OR)

- Incorporating new data types
- Modeling ordered data
- Querying ordered data
- Indexing techniques
- Mapping objects to relations
- OO/OR benchmarks
- Garbage collection techniques

NEXT WEEK: Object Modeling; Object Querying