Object-Relational DBMSs (ORDBMSs)

Example:
California Department of Water Resources
- 500,000 photographs
- Captions: “picture of Auburn Dam taken during scaffold construction”
Natural Representation: 
Schema with Complex Data

```sql
create table slides (  
id int,  
date date,  
caption document,  
picture jpeg_image);
create table landmarks(  
name varchar(30),  
location point);
```

Example Query over our Schema

```sql
select id  
from slides P, landmarks L S  
where sunset(P.picture) and  
contains (P.caption, L.name) and  
L.location |20| S.location and  
S.name = ‘Sacramento’;
```
What’s Special about the Example?

- **Query language**: user-defined functions (e.g., \textit{sunset}) and operators (e.g., \textit{|20|})
- **Data types**: relatively “complex” types (e.g., \textit{document}, \textit{point})
- **Performance**: a query optimizer aware of “expensive functions” like \textit{sunset}

Example: The Dinky Entertainment Company

- **Location**: Hollywood, CA
- **Main assets**: cartoon characters (e.g., \textit{Herbert the Worm})
- **Products**: film shows; image, voice, and video-footage licenses (e.g., for action figures, for video games)

\textbf{DBMS manages everything!}
An “Old-Fashioned” RDBMS is not Adequate for this Application

CREATE TABLE Frames (  
    frameno INTEGER,  
    image BLOB,  
    category INTEGER)

• Application code has to handle BLOBs  
• BLOBs transmitted to client unprocessed  
  (efficiency!)

“Old-Fashioned” Relational DBMSs

• Small, fixed set of data types  
  (e.g., integers, dates, strings)  

• Fixed set of operators and functions

Good for administrative data processing  
Bad for more complex kinds of data
New Data Types Required

- User-defined ADTs: image, sound, video, with functions and operators
- Type constructors: sets, tuples, arrays
- Inheritance: low- and high-resolution images are images

ORDBMS DDL Statements

CREATE TABLE Frames (frame_no integer, image jpeg_image, category integer)
CREATE TABLE Categories (cid integer, name text, lease_price float, comments text)
CREATE TYPE theater_t AS
  ROW(no integer, name text, address text, phone text)
  REF IS SYSTEM GENERATED
CREATE TABLE Theaters OF theater_t REF is tid SYSTEM GENERATED
CREATE TABLE Nowshowing (film integer, theater ref(theater_t) SCOPE Theaters, start date, end date)
CREATE TABLE Films (filmno integer, title text, stars VARCHAR(25) ARRAY [10], director text, budget float)
CREATE TABLE Countries (name text, boundary polygon, population integer, language text)
ORDBMSs: Overview

“Extended” SQL:
- User-defined data types possible for attributes
- Complex attributes possible (not 1NF)
- Reference types
- Inheritance

Defining New ADTs

- Define how to read in and how to output objects of the new type
- Define the size of the objects of the new type
- Define new methods for the new type
“Packaged” ADTs Common and Useful

- DataBlades (IBM’s Informix)
  - time series, spatial extensions
- Data Cartridges (Oracle)
  - Oracle Text
- DB2 Extenders (IBM)
- CREATE TYPE (PostgreSQL)
- …

Inheritance

- To reuse and refine type definitions
- To create hierarchies of collections with similar but not identical objects
Example

```
CREATE TYPE theatercafe_t
UNDER theater_t (menu text)

CREATE TABLE Theater_Cafes
  OF TYPE theatercafe_t
  UNDER Theaters

• By default, queries over Theaters are also evaluated over Theater_cafes
• “ONLY”
```

Substitution Principle

Given a supertype A, and a subtype B, it is always possible to substitute an object of type B into a legal expression written for objects of type A
Binding

When defining a subtype, we can redefine some of the supertype’s methods (overloading)