

From ODL/OO-DBMS Design
to Relational Designs
&
Object-Relational Database Systems (ORDBS)

INF3100/INF4100, V'2005, W5L2
Chapter 4, Sections 1-5 and 7

Last edited By M. Naci Akkøk 25/2-2003, 20/2-2004 and 14/2-2005.

Based upon slides by Pål Halvorsen (26/2-2002).

Contains slides made by Arthur M. Keller and Vera Goebel.

Overview

- From ODL/OO-DBMS design to relational design (very briefly)
- Object-relational DBMS (OR-DBMS)

From ODL/OO-DBMS Design to Relational Design

Translating ODL to Relations

Similar to converting ER-diagrams to relations:

- Classes without relationships are like entity classes (or sets), but some new problems arise:
 - Keys are optional in ODL
(Some situations may require inventing new attributes to serve as a key)
 - ODL may have non-atomic attributes
(Resulting relations may be non-normalized and must be redesigned)
 - ODL allows methods as part of design
- Classes with relationships:
 - Treat the relationship separately, as in E/R.
 - Attach a many-one relationship to the relation for the “many.”

Translating ODL to Relations:

Attributes and Keys

- Each *atomic* ODL attribute translates to a relational attribute.
- ODL allows non-atomic attributes (structures and collection types):
 - **Structure:** make one attribute for each field.
 - **SET:** make one tuple for each member of the set.
(More than one set attribute? Make tuples for all combinations.)
 - **BAG:** make only distinct tuples, add a count attribute
 - **LIST:** make one member for each member of the list, add a position attribute
 - **ARRAY:** fixed length, add each field in the array as own attributes
 - **DICTIONARY:** represent as set, but with a tuple for all pairs that are member of the dictionary
- ODL class may have no key, but we should have one in the relation to represent “OID.”

Translating ODL to Relations:

Attributes and Keys - Example

ODL class:

```
class Person (extent persons key name) {  
    attribute string name;  
    attribute Struct Addr{string street, string city} address;  
    attribute Set<string> phone;  
}
```

Relation schema:

```
persons (name, street, city, phone)
```

| <u>name</u> | street | city | <u>phone</u> |
|-------------|--------|-------|--------------|
| n_1 | s_1 | c_1 | p_1 |
| n_1 | s_1 | c_1 | p_2 |

- “Surprise”: the key for the class (name) is not the key for the relation (name, phone):
 - name in the class determines a unique object, including a *set* of phone numbers.
 - name in the relation does not determine a unique tuple.
 - Since tuples are not identical to objects, there is no inconsistency!
- BCNF violation: name \rightarrow (street, city), name not superkey, separate out name-phone.

Relationships

- Can create a new relation for each relationship
- If the relationship is many-to-one from A to B , put key of B attributes in the relation for class A .
- If relationship is many-many, we'll have to duplicate A -tuples as in ODL with set-valued attributes.
 - Wouldn't you really rather create a separate relation for a many-many-relationship?
 - You'll wind up separating it anyway, during BCNF decomposition.

Relationships - Example

ODL class:

```
class Person (extent persons key name) {
    attribute string name;
    attribute Struct Addr{string street, string city} address;
    relationship Set<Cars> ownCar inverse Cars::ownedBy;
    relationship Person spouse inverse spouse;
    relationship Set<Person> buddies inverse buddies;
}
```

Relation schema:

```
persons (name, street, city, ownCar, spouse, buddies)
```

- **BCNF violation:** name \rightarrow (street, city, spouse)
- **4NF violation:** name \twoheadrightarrow ownCar buddies

- **Decomposition into 4NF:**

```
persons          (name, street, city, spouse)
persons_cars     (name, ownCar)
persons_buddies  (name, buddies)
```

Object-Relational Database Systems (ORDBS)

Data Models & Database System Architectures

- Chronological Overview -

- Network Data Models (1964)
- Hierarchical Data Models (1968)
- Relational Data Models (1970)
- Object-oriented Data Models (~ 1985)
- **Object-relational Data Models (~ 1990)**
- Semistructured Data Models (XML 1.0) (~1998)

Object-Relational Database Systems (ORDBS)

- Motivations

- Allow DBMS to deal with specialized types – maps, signals, images, etc. – with their own specialized methods.
- Supports specialized methods even on conventional relational data.
- Supports structure more complex than “flat files.”
- ...

-  Object-oriented ideas enter the relational world

- Keep the *relation as the fundamental abstraction* whereas the OODBS use the class as the fundamental abstraction.

New Features

- **Structured types**
Not only atomic types. ODL-like type system.
(Also: BLOB, CLOB, ADT, BFILE)
- **Methods**
Special operations can be defined for a type.
- **Identifiers**
Allowing unique IDs for each tuple.
- **References**
Pointers to tuples.

ORDBS:

Nested Relations

- Attributes may have non-atomic types
 - Nested-relational data models give up 1NF (atomic values)
 - A relation's type can be any schema consisting one or more attributes. An attribute may even have an own schema as type.

- Example:

```
moviestar(name, address(street,city), birth, movies(title,year))
```

| name | address | | birth | movie | |
|-------------|----------------|-------------|--------------|--------------|-------------|
| Fisher | street | city | 9/9/1950 | title | year |
| | Maple | Hollywood | | Star Wars | 1977 |
| | 5. Avenue | New York | | Empire | 1980 |
| Hamill | street | city | 8/8/1962 | title | year |
| | Sunset Blvd | LA | | Star Wars | 1977 |
| | | | | Return | 1983 |

References - I

- Non-normalized relation
- Introduce references to allow a tuple t refer to a tuple s rather than including s in t .

| name | address | | birth | movie | |
|--------|-------------|-----------|----------|-----------|------|
| | street | city | | title | year |
| Fisher | Maple | Hollywood | 9/9/1950 | Star Wars | 1977 |
| | 5. Avenue | New York | | Empire | 1980 |
| Hamill | Sunset Blvd | LA | 8/8/1962 | Star Wars | 1977 |
| | | | | Return | 1983 |

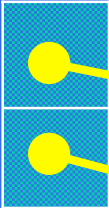
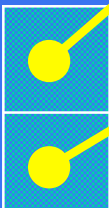
References - II

- If attribute A has a type that is a reference to a relation with schema R , we denote A as $A(*R)$
- If A is a set of references, we denote A as $A(\{*R\})$
- Example:

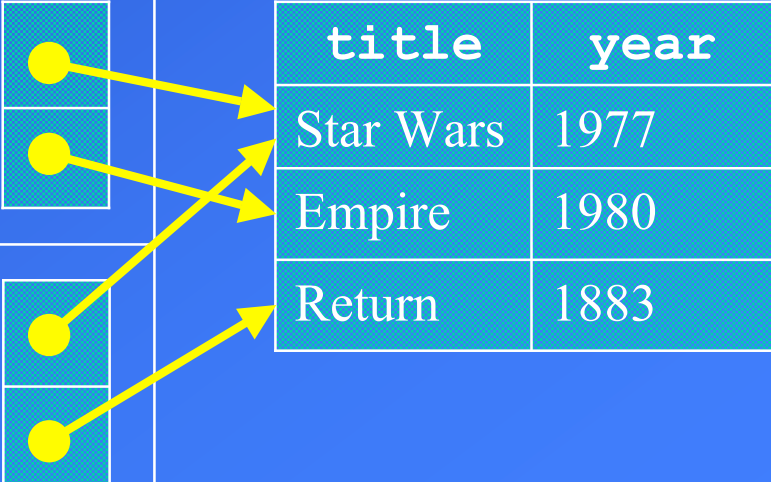
```

moviestar(name, address(street,city), birth, movies(*movies))
movies(title,year)

```

| name | address | | birth | movie |
|--------|-------------|-----------|----------|---|
| Fisher | street | city | 9/9/1950 |  |
| | Maple | Hollywood | | |
| Hamil | street | city | 8/8/1962 |  |
| | Sunset Blvd | LA | | |

| title | year |
|-----------|------|
| Star Wars | 1977 |
| Empire | 1980 |
| Return | 1883 |



OODBS vs. ORDBS - I

two ways to integrate object-orientation into DBS
both directions (OODBS and ORDBS) are also reflected
in the standard developments

Several vendors:

commercial OODBS:

- GemStone
- O2 (now: Ardent)
- ObjectivityDB
- ObjectStore
- ONTOS
- POET
- Versant
- ...

commercial ORDBS:

- ORACLE
- Sybase
- Illustra
- UNISQL
- ...

OODBS vs. ORDBS - II

- **Objects/tuples:**
Both objects and tuples are structs with components for attributes and relationships
- **Extents/relations:**
Both may share the same declaration among several collections
- **Methods:**
Both has the same ability to declare and define methods associated with a type
- **Type systems:**
Both are based on atomic types and constructions of new types by structs and collection types
- **References/OID:**
OODBS OID hidden – ORDBS ID visible (may be part of type)
- **Backwards Compatibility:**
Migrating existing applications to an OODBS require extensive rewriting, but ORDBSes have maintained backward compatibility

OODBS vs. ORDBS - III

OODBS:

- simpler way for programmer to use DBS (familiar with OOPLs)
- “seamlessness”, no “impedance mismatch”
- OO functionality + DBS functionality
◇ higher performance for specific applications
- “revolutionary” approach, no legacy problems
- ...

ORDBS:

- substantial investment in SQL-based rel. DBSs ◇ evolutionary approach
- systems are more robust due to many years of usage and experience
- application development tools
- transaction processing performance
- ...

prediction: both kinds of systems will exist, used for different kinds of applications

NEXT TIME

- Next time is week 6 (22. Feb. 2005)
- We'll look at:
 - An introduction to semi-structured data and XML
 - The Object Query Language (OQL)