

# Object-Relational Database Systems (ORDBS)

Contains slides made by Naci Akkøk, Pål Halvorsen, Arthur M. Keller and Vera Goebel.

# 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 of one or more attributes. An attribute may even have an own schema as type.
- Example:  
`moviestar(name, address(street,city), birth, movies(title,year))`

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

## ORDBS:

# 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

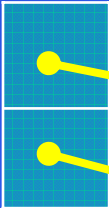
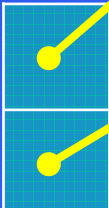
# References - II

- 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:

```

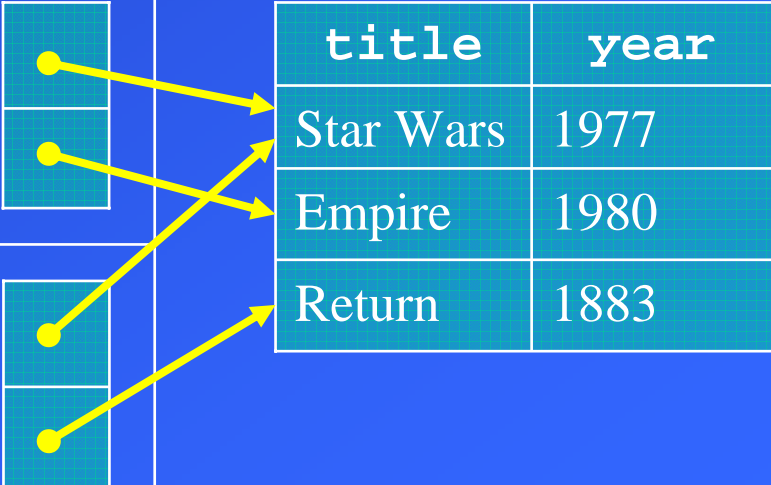
moviestar(name, address(street,city), birth, movie(*movies))
movies(title,year)

```

name	address		birth	movie
Fisher	street	city	9/9/1950	
	Maple	Hollywood		
	5. Avenue	New York		
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