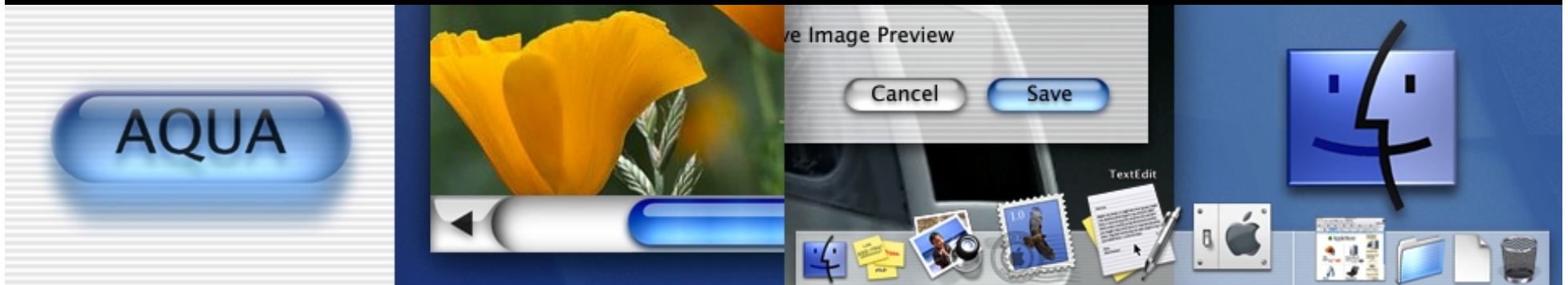




Session 403

Persistent Object Modeling with EOModeler



Mike Gobbi

Senior Developer Trainer, Apple iServices

Introduction

- WebObjects is a powerful technology for building dynamic web sites
- Most dynamic web sites are backed by relational databases
- EOF allows you to access persistent data from within your object-oriented application



What You'll Learn

- Basic EOF and WOF concepts
- How to use EOModeler to
 - Work with the database schema
 - Define validation rules and fetch specifications for use at runtime



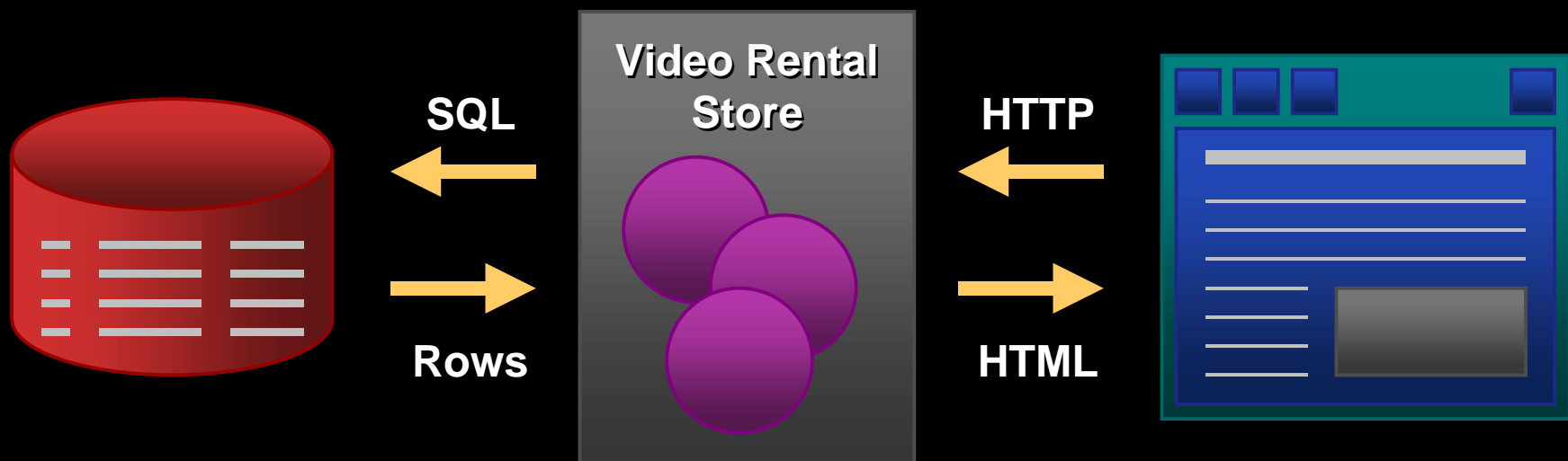
Part 1: Basic Concepts

- Structure of a Web application
- WOF/EOF architecture
- Entity-Relationship modeling concepts



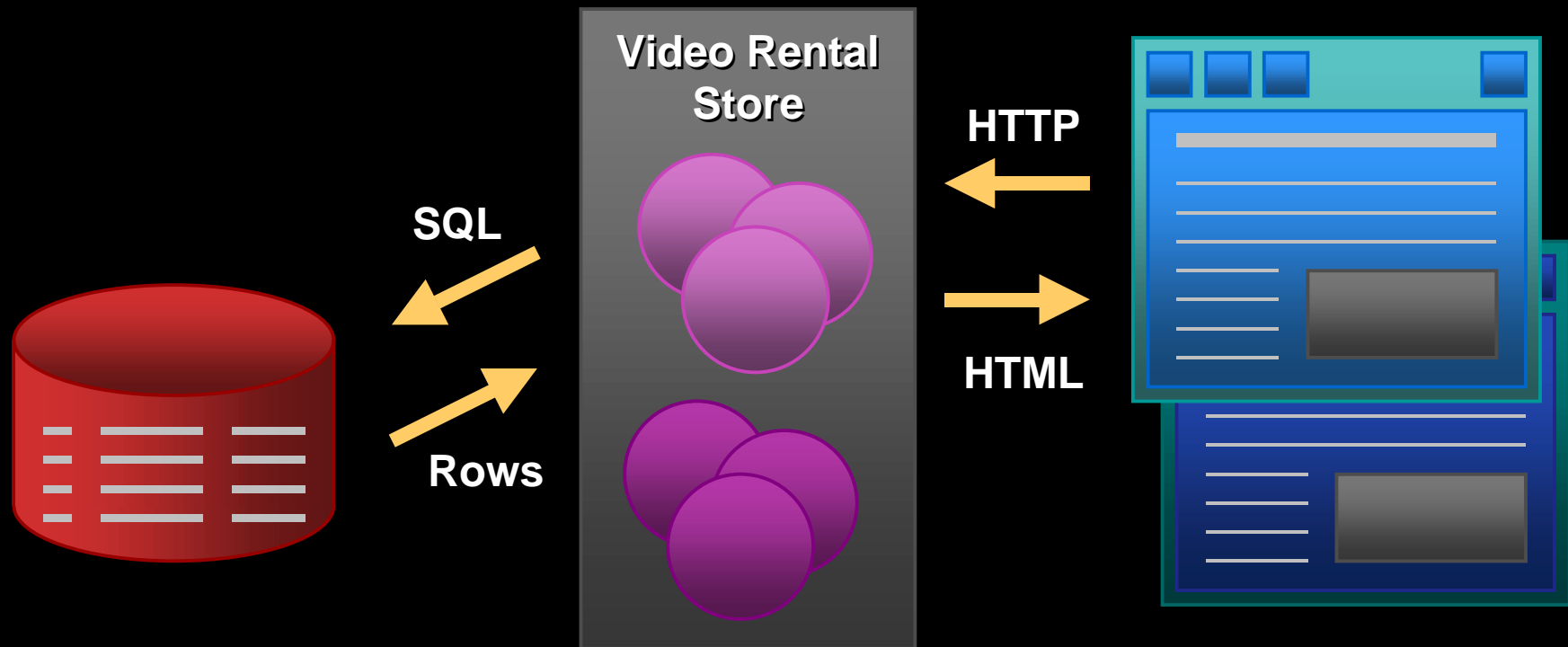
Web Applications

- Service web requests using information stored in relational databases



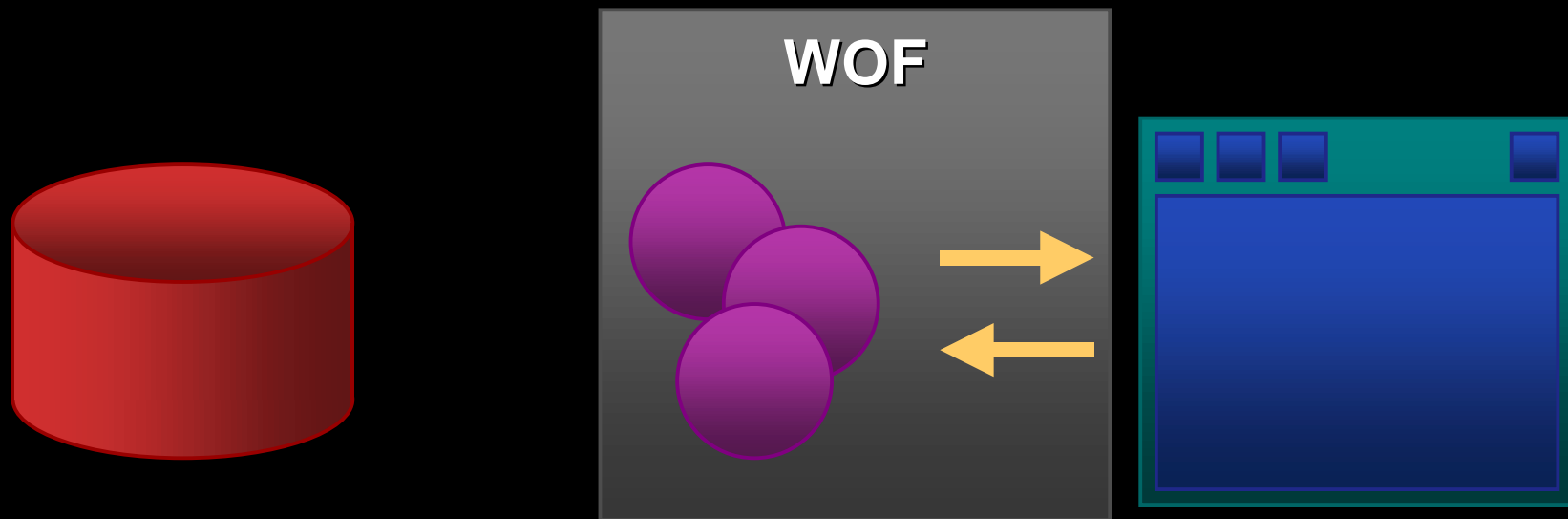
Web Applications

- Maintain a unique editing context for each user



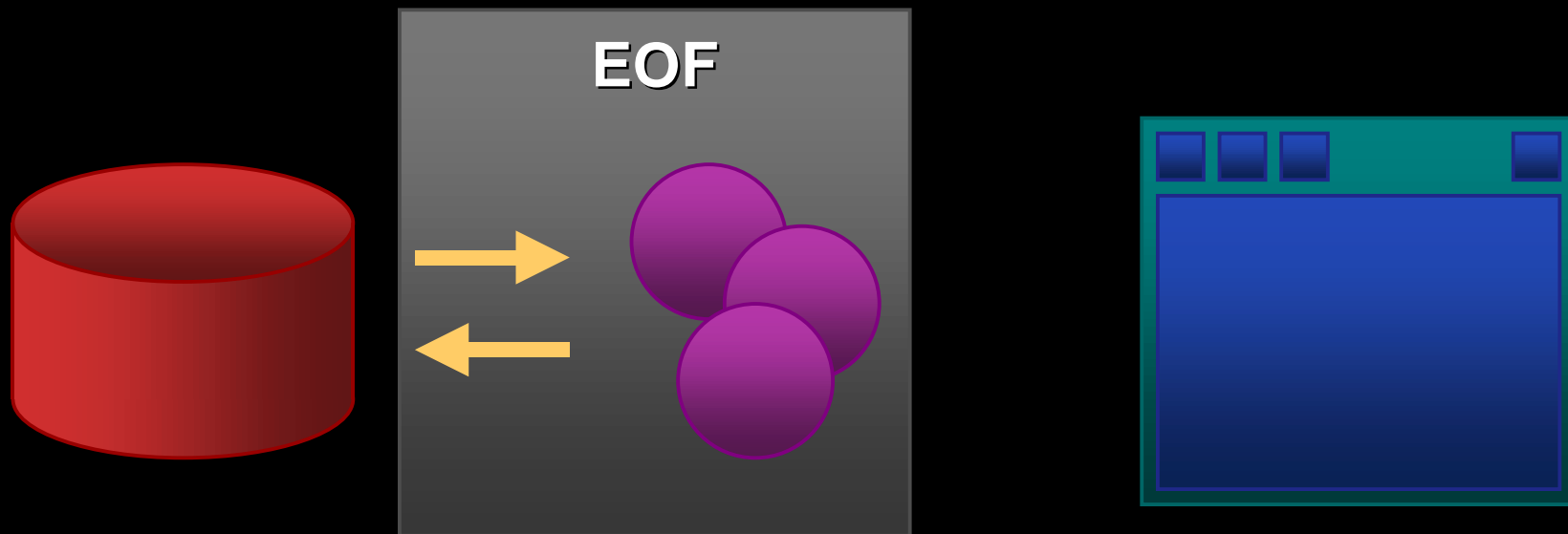
WebObjects Framework

- Creates HTML pages from your business objects

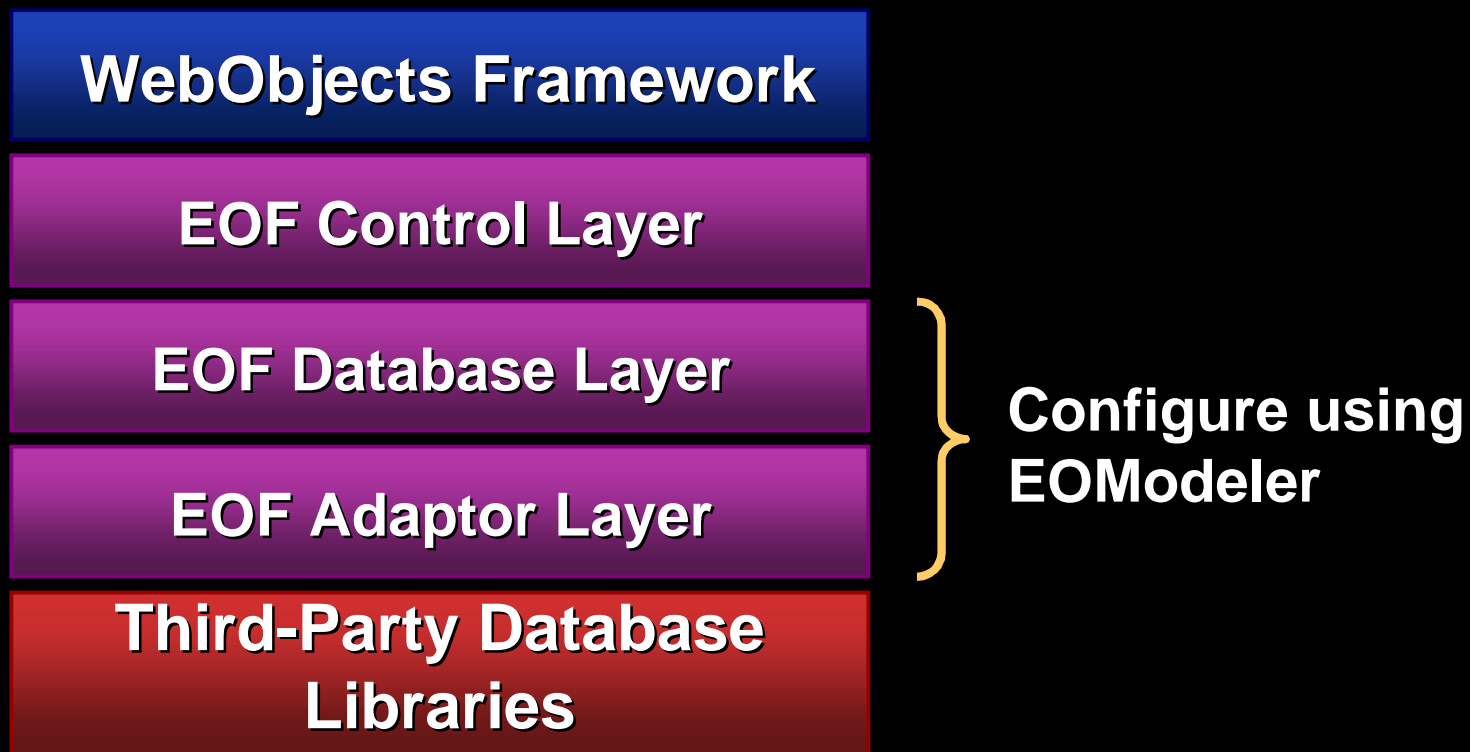


Enterprise Objects Framework

- Creates business objects from database records

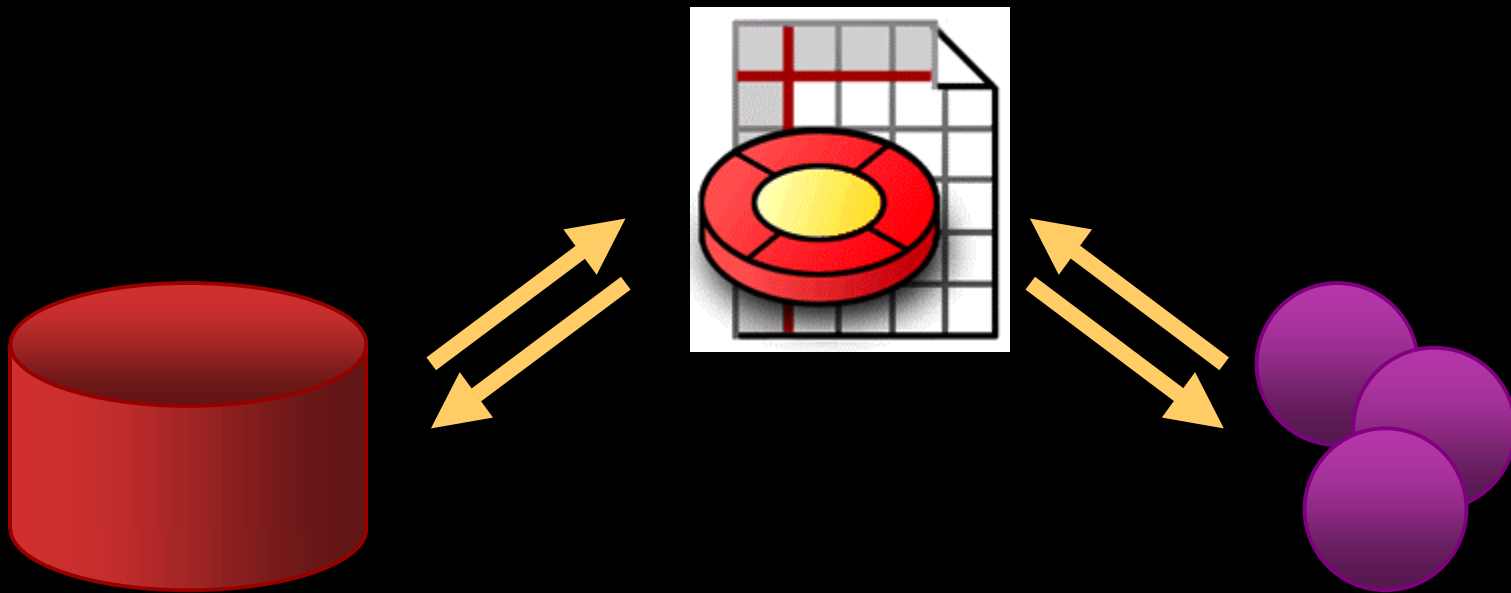


WebObjects Application Architecture



Model File

- Describes the mapping between database tables and object-oriented classes

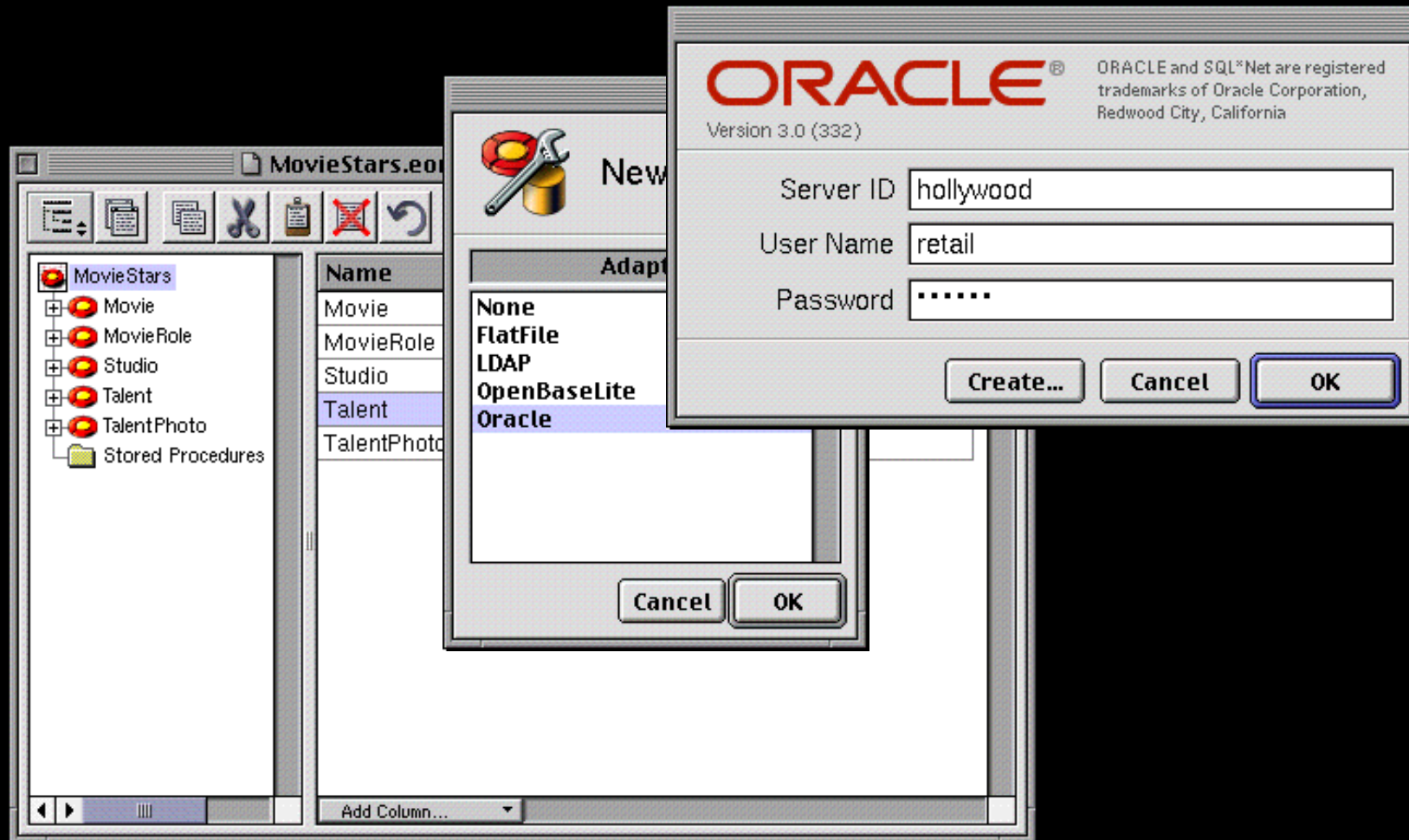


EOModeler

- A tool for editing model files:
 - Configure database adaptor
 - Work with schema:
 - Entities
 - Attributes
 - Relationships
 - Identify stored procedures
 - Create fetch specifications



Configuring the Database Adaptor



Entities: Table and Class

STUDIO		MOVIE			
NAME	STD_ID	TITLE	CATEGORY	MOV_ID	STD_ID
Warner Brothers	112	Casablanca	Drama	345	112
20th Century Fox	113	Alien	Horror	456	113
		Star Wars	Action	567	113

Movie
title "Casablanca"
category "Drama"
studio

Movie
title "Alien"
category "Horror"
studio

Movie
title "Star Wars"
category "Action"
studio

Studio
name "Warner Brothers"
movies

Studio
name "20th Century Fox"
movies



Creating an Entity

The screenshot shows a database modeling application window titled "MovieStars.eomodeld - ~/Library/Models". The main window contains a tree view on the left with "MovieStars" expanded to show "Director", "Movie", "MovieRole", "Studio", "Talent", "TalentPhoto", and "Stored Procedures". A table view in the center lists the following data:

Name	Table	Class Name
Director	DIRECTOR	EOGenericRecord
Movie	MOVIE	Movie
MovieRole	MOVIE_ROLE	MovieRole
Studio	STUDIO	Studio
Talent	TALENT	Talent
TalentPhoto	TALENT_PHOTO	TalentPhoto

The "Entity Inspector" window is open on the right, showing the configuration for the "Director" entity:

- Name: Director
- Table Name: DIRECTOR
- Class: EOGenericRecord

The "Properties" section is empty. A legend at the bottom indicates:

- Primary Key (key icon)
- Used For Locking (lock icon)
- Class Property (diamond icon)
- Not applicable (dash icon)



Attributes: Column and Instance Variable

STUDIO		MOVIE			
NAME	STD_ID	TITLE	CATEGORY	MOV_ID	STD_ID
Warner Brothers	112	Casablanca	Drama	345	112
20th Century Fox	113	Alien	Horror	456	113
		Star Wars	Action	567	113

Movie
title "Casablanca"
category "Drama"
studio

Studio
name "Warner Brothers"
movies

Movie
title "Alien"
category "Horror"
studio

Studio
name "20th Century Fox"
movies

Movie
title "Star Wars"
category "Action"
studio



Simple Attributes

- Represent simple data types
 - Strings
 - Integers
 - Floating-point numbers
 - Dates
- Are automatically converted to objects by EOF



Blobs

- Represent arbitrary data types:
 - Images
 - Sounds
 - QuickTime movies
- Are read from the database as NSData objects
- You specify class and factory methods



Primary and Foreign Keys

- Are columns in the database
- Generally do *not* appear as instance variables in the application
- Are used to define relationships



Editing Attributes

MovieStars.eomodeld - ~/Library/Models

Director Attributes

Name	Value Class	Extern
movieId	NSNumber	long
talentId	NSNumber	long

Director Relationships

Name	Destination	Source
------	-------------	--------

Add Column... Add column...

Attribute Inspector

Name: talentId

Column: TALENT_ID

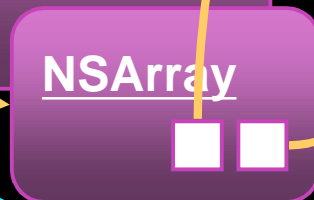
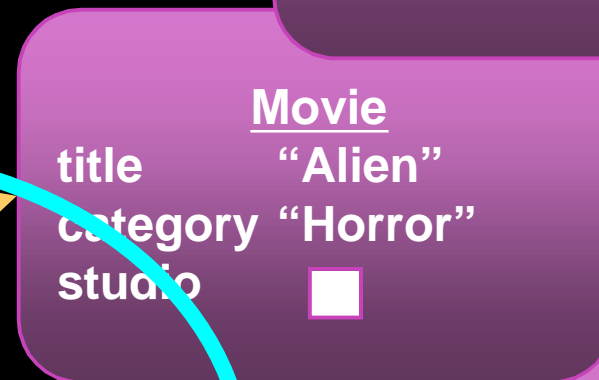
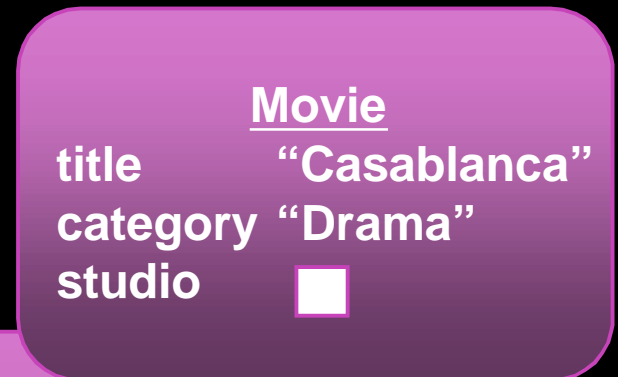
External Type: long

Internal Data Type: Integer



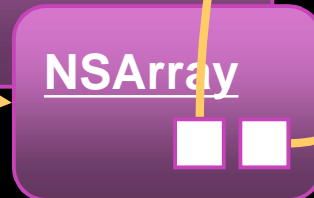
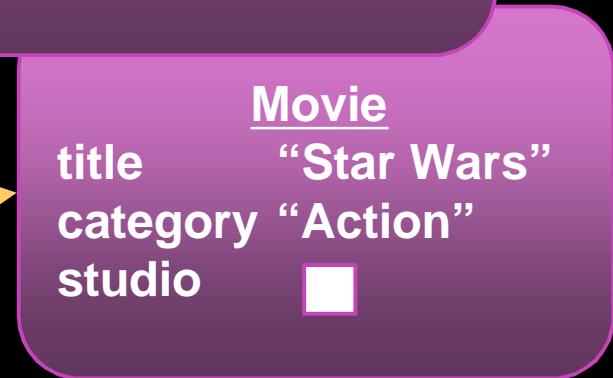
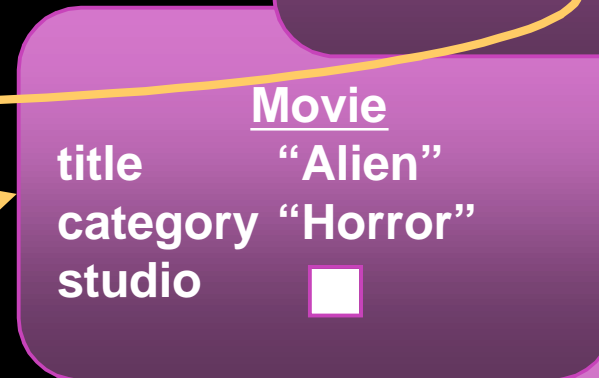
Relationships: Join and Object Reference

STUDIO		MOVIE			
NAME	STD_ID	TITLE	CATEGORY	MOV_ID	STD_ID
Warner Brothers	112	Casablanca	Drama	345	112
20th Century Fox	113	Alien	Horror	456	113
		Star Wars	Action	567	113



To-One vs. To-Many Relationships

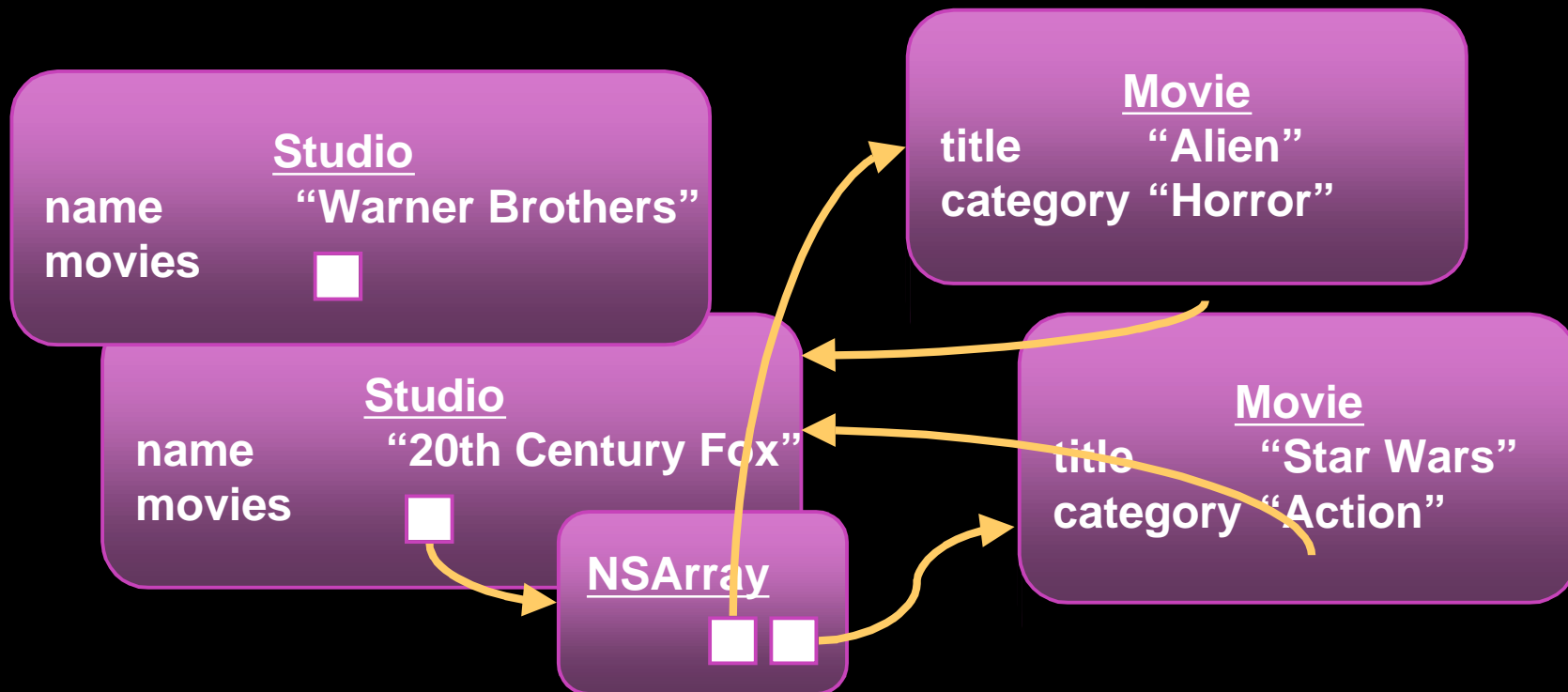
STUDIO		MOVIE			
NAME	STD_ID	TITLE	CATEGORY	MOV_ID	STD_ID
Warner Brothers	112	Casablanca	Drama	345	112
20th Century Fox	113	Alien	Horror	456	113
		Star Wars	Action	567	113



Two-Way vs. One-Way Relationships

STUDIO		MOVIE			
NAME	STD_ID	TITLE	CATEGORY	MOV_ID	STD_ID
Warner Brothers	112	Casablanca	Drama	345	112
20th Century Fox	113	Alien	Horror	456	113
		Star Wars	Action	567	113

Movie
title "Casablanca"
category "Drama"



Editing Relationships

MovieStars.eomodeld - ~/Library/Models

Director Attributes

Name	Value Class	Extern
movieId	NSNumber	long
talentId	NSNumber	long

Director Relationships

Name	Destination	Source
movie	Movie	mov

Relationship Inspector

Name:

Destination

Model: **MovieStars**

Entity: Director, **Movie**, MovieRole, Studio, Talent, TalentPhoto

To One
 To Many

Inner

Joins

Source Attributes	Destination Attributes
movieId	category
talentId	dateReleased
	movieId
	posterName
	rated
	revenue
	studioId

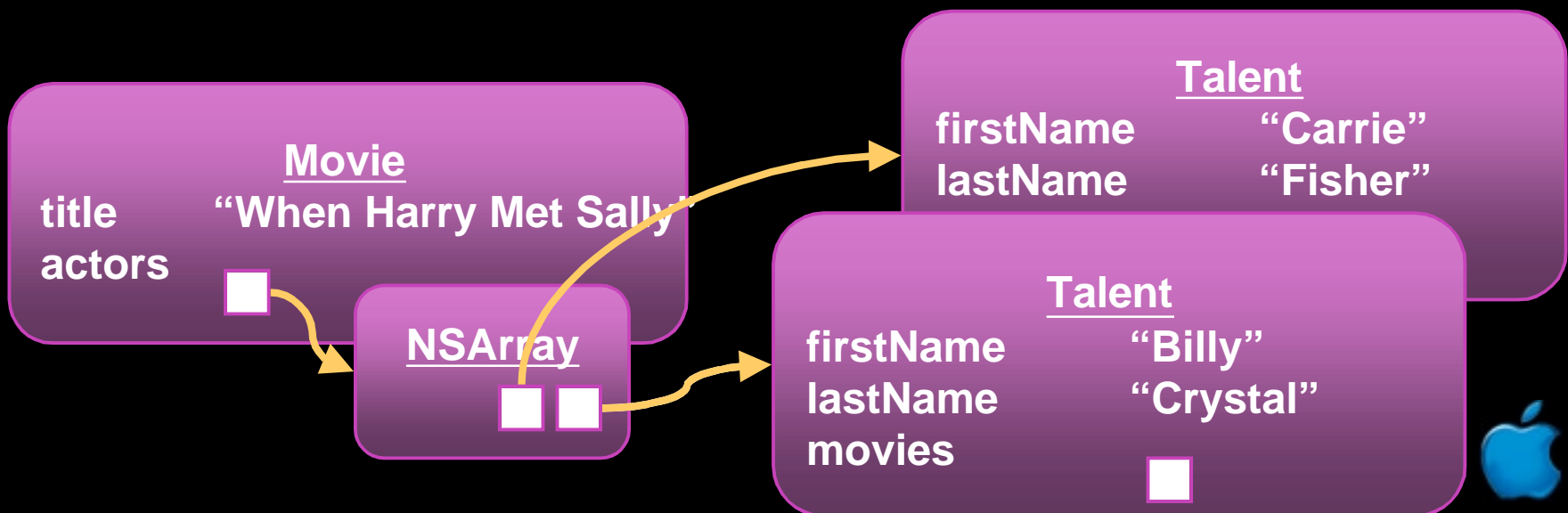
Connect



Many-to-Many Relationships

- Join table is not in application

MOVIE		ACTED_IN		TALENT		
TITLE	MOV_ID	MOV_ID	TLT_ID	FIRST	LAST	TLT_ID
Star Wars	567	567	33	Carrie	Fisher	33
When Harry Met Sally	789	789	33	Billy	Crystal	44
		789	44			



Many-to-Many in EOModeler

The screenshot displays the EOModeler application interface. The 'Property' menu is open, with 'Join in Many-to-Many' selected. The 'Flattened Relationship Inspector' window is also open, showing the following details:

- Name: talents
- Definition: movieTalents.talent
- Entity: Talent

The background shows a project tree with 'Studio', 'Talent', and 'TalentPhoto' entities, and a table view for 'Movie Relationships' with columns for 'Name' and 'Type'.

Name	Type
movie	
roles	
studio	
talents	



Relationships Between Different Databases

The screenshot displays a database modeling application with two main windows: 'MovieStars.eomodeld' and 'VideoRentals.eomodeld'. The 'MovieStars' window shows a tree view with entities: Movie, MovieRole, Studio, Talent, TalentPhoto, and Stored Procedures. The 'VideoRentals' window shows a tree view with entities: Rental, Review, Unit, Video, and Stored Procedures. Below the tree views are two tables: 'Video Attributes' and 'Video Relationships'.

Key	Name
◆	moviold
◆	rentalTermsId
◆	videold

Name
>◆ movie
>>◆ unit

The 'Relationship Inspector' window is open, showing a relationship named 'movie'. The 'Destination' section is set to 'Model: MovieStars' and 'Entity: Movie'. The relationship type is 'To One' and 'Inner'. The 'Joins' section shows 'Source Attributes' (moviold, rentalTermsId, videold) and 'Destination Attributes' (category, dateReleased, moviold, posterName, rating, revenue, studiold). A 'Connect' button is at the bottom.

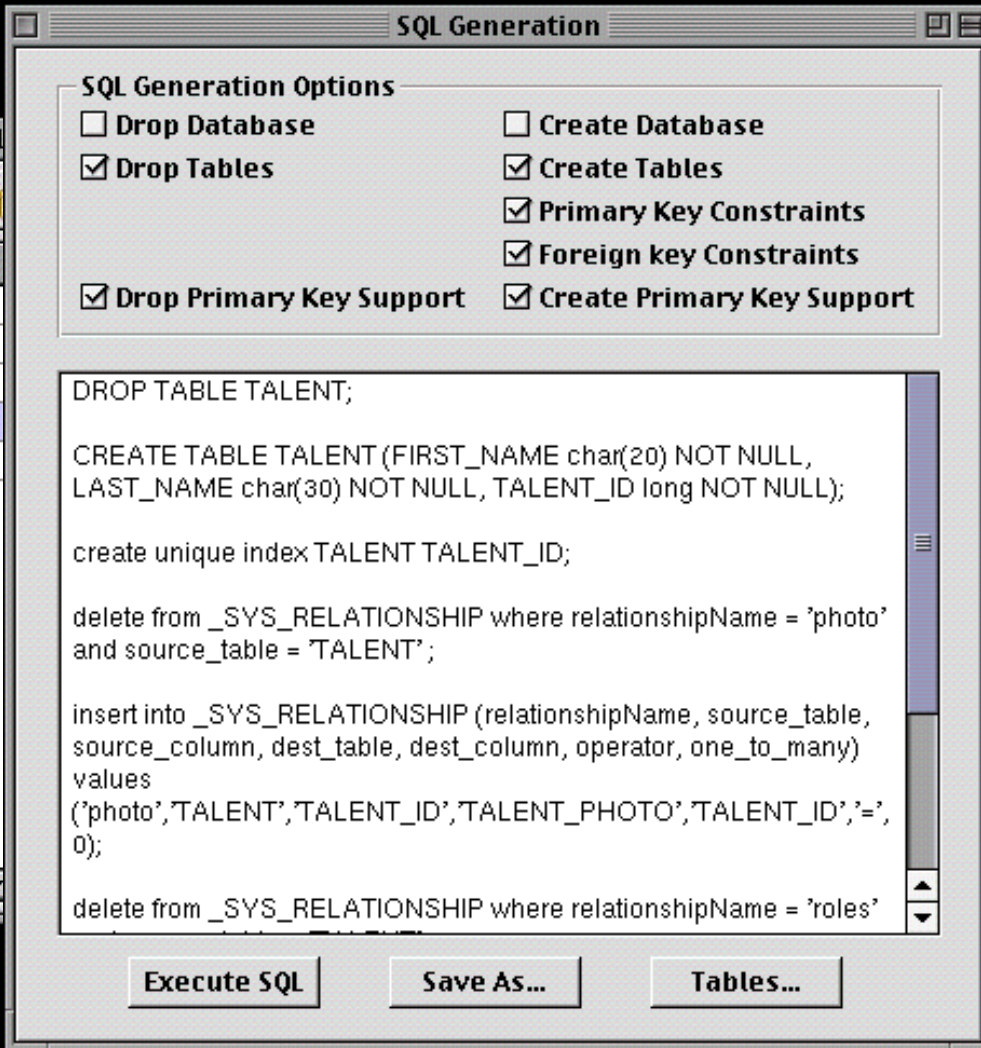
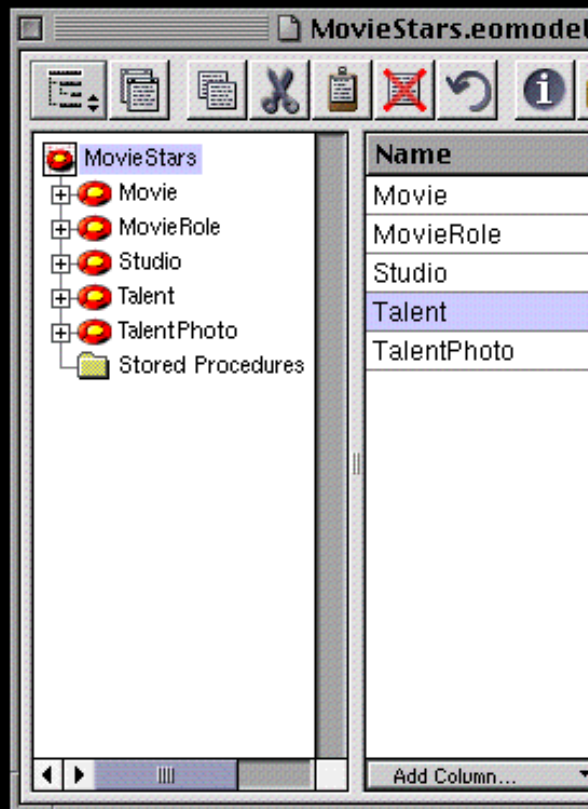


Part 2: Working With the Entities

- Generating SQL
- Browsing data
- Importing an existing database
- Generating Java classes



Generating SQL



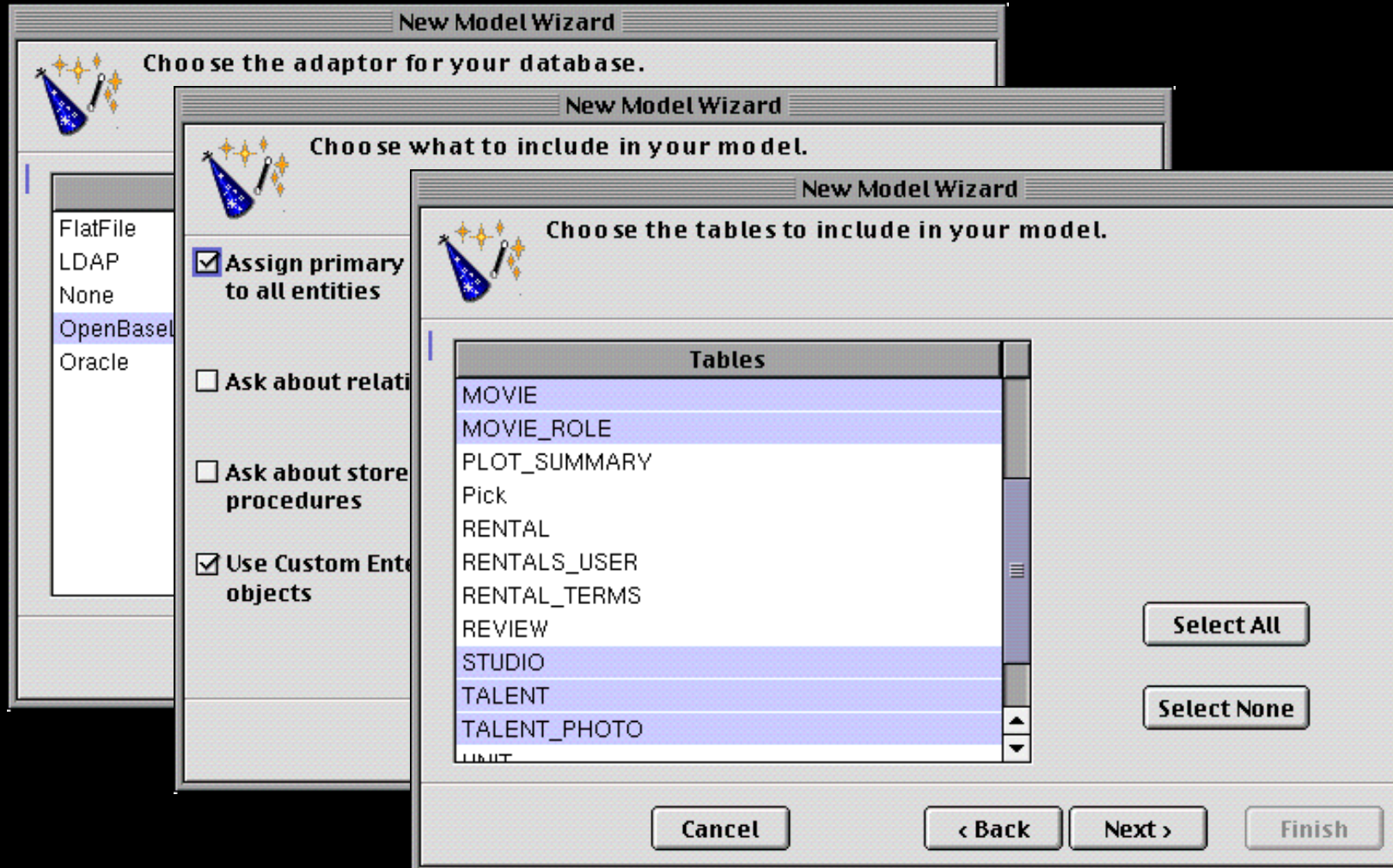
Browsing Data

The screenshot shows a database browser interface. On the left, a tree view displays the 'MovieStars' database structure with entities: Movie, MovieRole, Studio, Talent, TalentPhoto, and Stored Procedures. The 'Talent' entity is selected. The main pane shows a table with columns 'Name' and 'Table'. The 'Data Browser' window is open, showing the 'Talent' entity with a filter 'lastName like 'F*'' applied. The resulting data is shown in a table with columns 'firstName', 'lastName', and 'talentId'. A 'Refetch' button is visible at the bottom of the 'Data Browser' window.

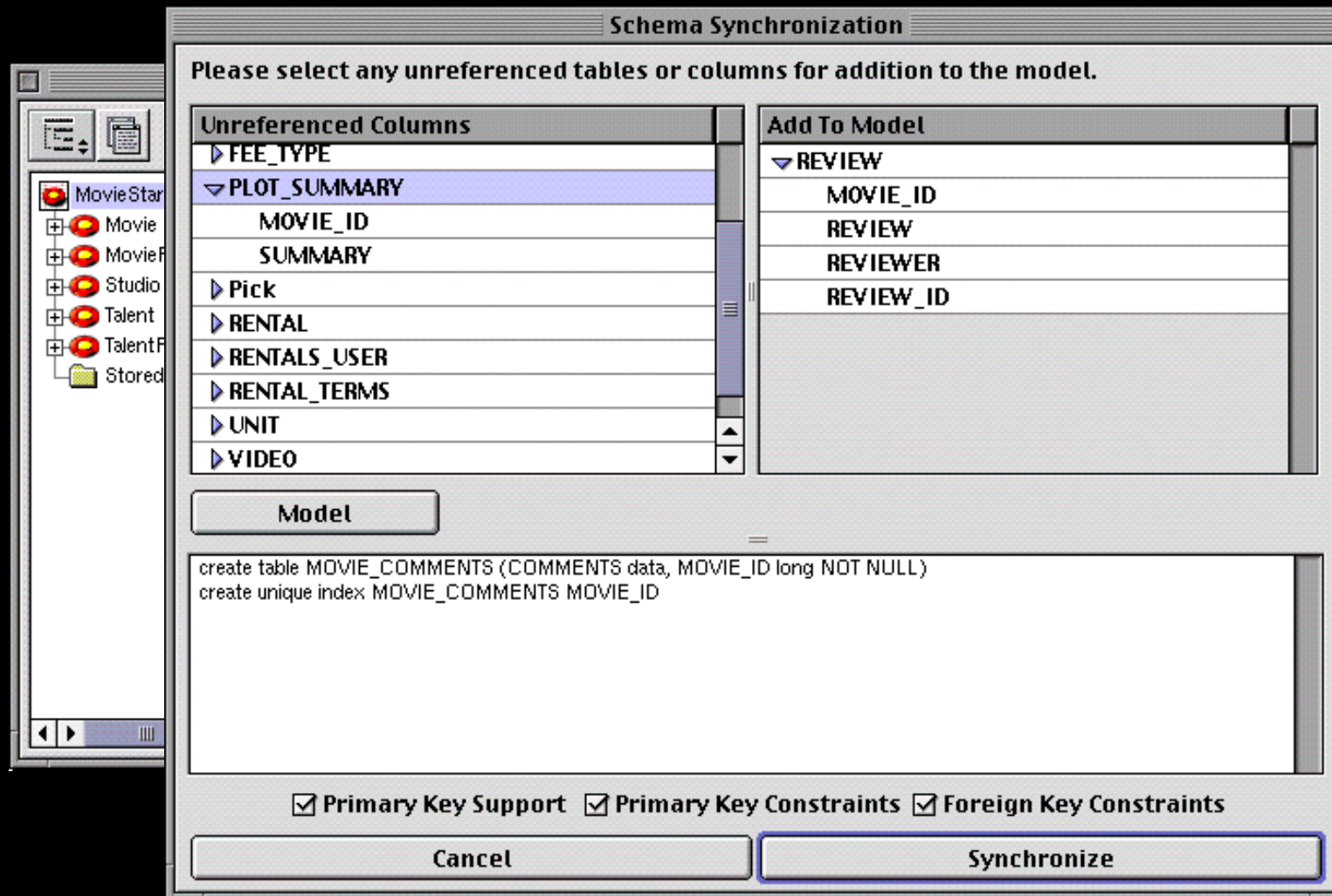
Entity:	Talent	Qualifier:	lastName like 'F*'
firstName	lastName	talentId	
Aretha	Franklin	291	
Carrie	Fisher	120	
Chuck	Fleming	618	
Corey	Feldman	473	
Dennis	Farina	226	
Federico	Fellini	484	
Harrison	Ford	87	
Henry	Fonda	564	
Jacques	François	145	
Jane	Fonda	526	
Jodie	Foster	538	



Reverse Engineering



Synchronizing the Schema



Generating Java Classes

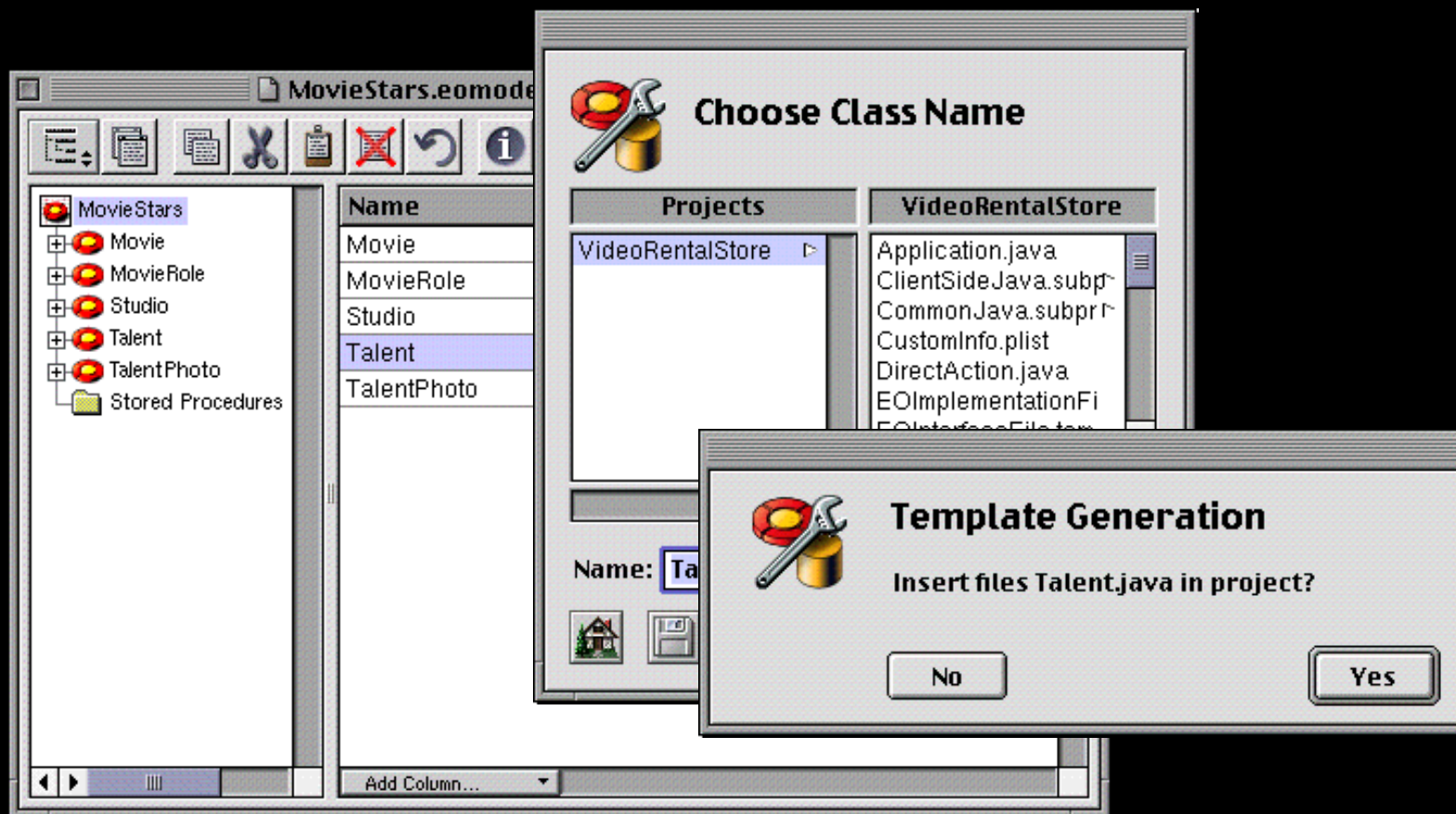
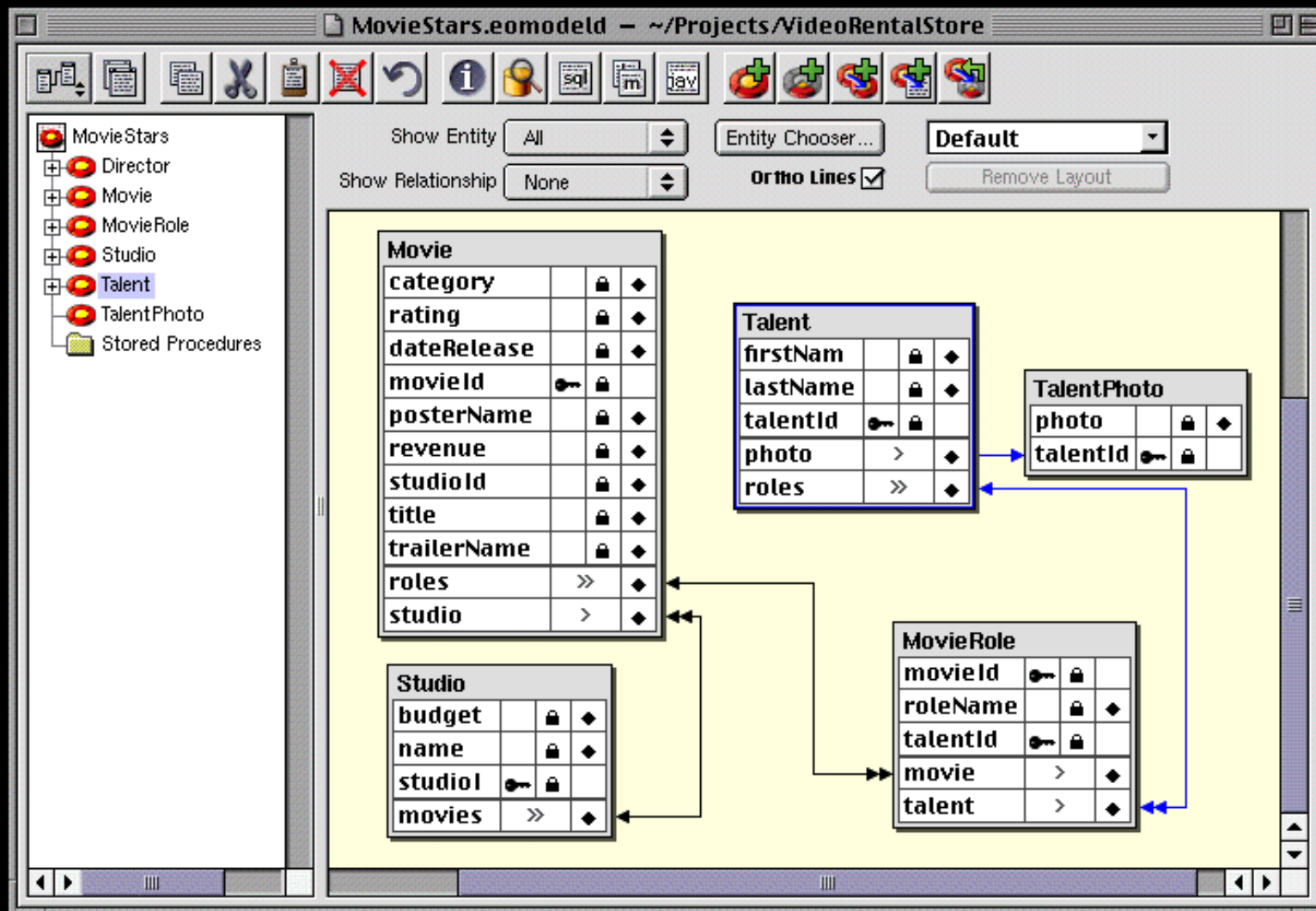


Diagram View

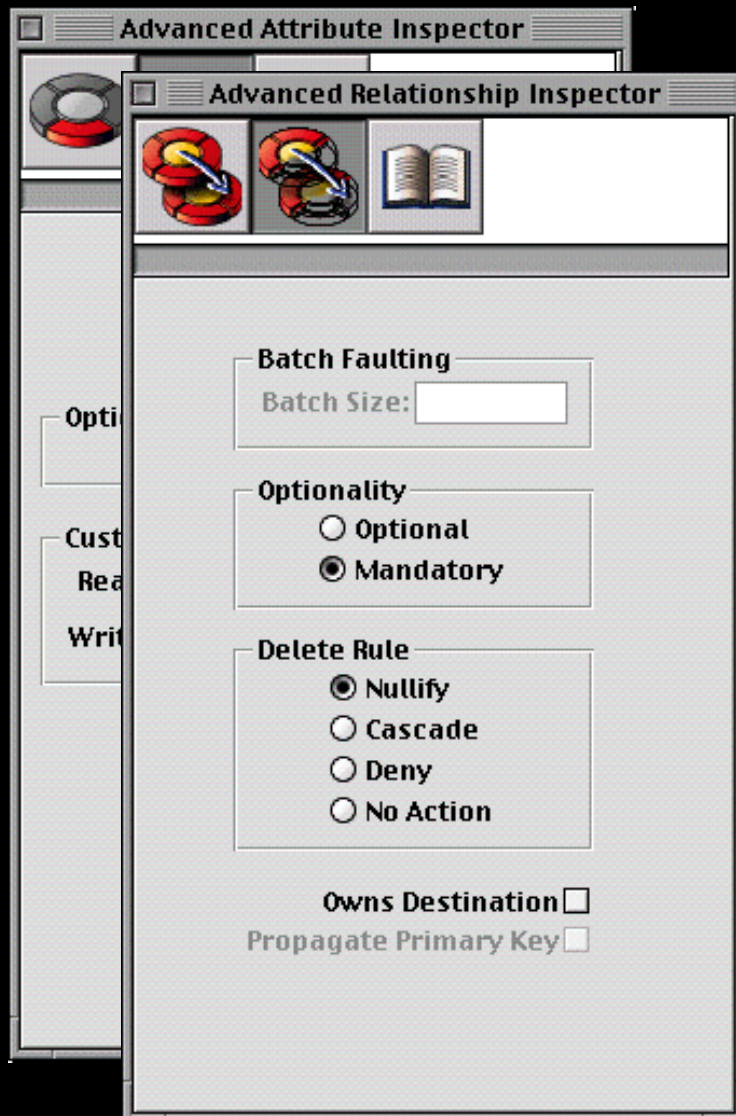


Part 3: Beyond the Schema

- Validation
- Locking and conflict detection
- Shared entities
- Stored procedures
- Fetch specifications



Validation in EOModeler



- Prohibit NULL values
- Limit string lengths
- Make relationships mandatory
- Specify whether deletions should cascade










Validation in Code

- Specify more complex validation rules:

```
public void validateForSave()
throws EOValidation.Exception
{
    if (age < 16 && hasDriversLicence)
        throw new EOValidation.Exception
            ("Too young to drive");
}
```



Locking and Conflict Detection

			Name
			employeeID
			firstName
			lastName
			payGrade

- Pessimistic locking blocks access to the database row
- Optimistic locking allows shared access

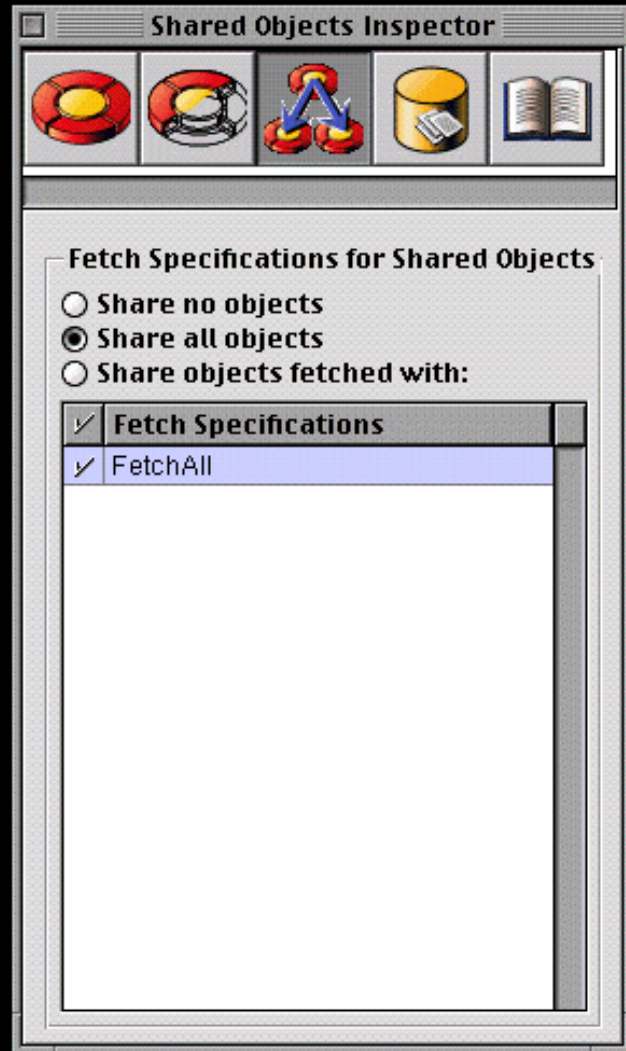


Catching Exceptions at Runtime

```
public WOComponent saveChanges()
{
    EOEditingContext ec =
        session().defaultEditingContext();
    try
    {
        ec.saveChanges();
    }
    catch (EOValidation.Exception e)
    {
        return createErrorPage(e);
    }
    return null;
}
```



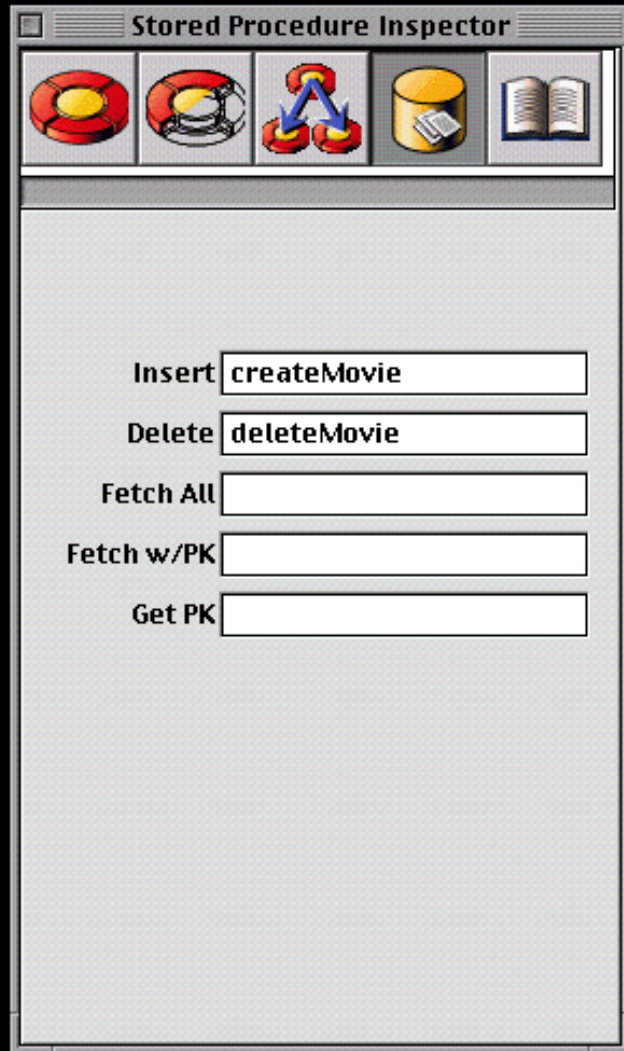
Shared Objects



- Usually each user has their own copy of the objects
- This can be very memory-intensive
- Read-only entities can be safely shared



Stored Procedures



- Can be used to perform common operations:
 - Insert and delete
 - Fetch
 - Generate Primary Key
- You can also call arbitrary stored procedures



Fetch Specifications

- Build complex queries once in EOModeler
- Use variable placeholders in query
- Provide actual query values at runtime



Fetch Specifications in EOModeler

The screenshot shows the EOModeler application window titled "MovieStars.eomodeld - ~/Projects/VideoRentalStore". The interface includes a toolbar with various icons, a left-hand sidebar with a tree view of the model classes, and a main workspace for defining fetch specifications.

The sidebar tree view shows the following structure:

- MovieStars
 - Director
 - Movie
 - MovieRole
 - Studio
 - Talent
 - photo
 - roles
 - Actor By Movie Revenue
 - TalentPhoto
 - Stored Procedures

The main workspace is titled "Fetch Specification Name: ActorByMovieRevenue". It features several tabs: "Qualifier", "Sort Ordering", "Prefetching", "Raw Fetch", "Options", and "SQL". The "Qualifier" tab is active, showing two columns: "Talent" and "roles".

Talent	roles
firstName	movie
lastName	movieId
photo	roleName
roles	talent
talentId	talentId

Below the table, there are several comparison operators: =, <>, <=, >=, <, >, and like. The "AND" operator is selected, and the following conditions are entered:

```
(roles.movie.revenue >= $revenue)
AND
(roles.movie.dateReleased >= $afterDate)
(roles.movie.dateReleased < $beforeDate)
```

The resulting SQL query is displayed at the bottom:

```
((roles.movie.revenue >= $revenue) and (roles.movie.dateReleased >= $afterDa
```

At the bottom of the workspace, there are buttons for "And", "Or", "Not", and "Remove".



Using Fetch Specifications in Code

```
public NSArray performQuery()
{
    EOEditingContext ec = session().
        defaultEditingContext();
    NSArray results;

    results = EOUtilities.
        objectsWithFetchSpecificationAndBindings
        (ec, "Customers",
         "ActorByMovieRevenue", dictionary);
    return results;
}
```

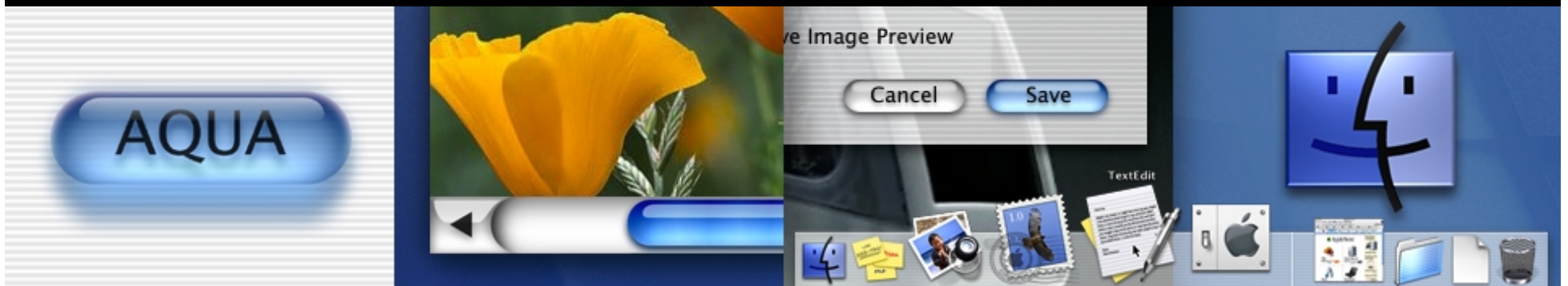


"revenue"	500,000
"beforeDate"	1/1/1975





DEMO



Mike Gobbi
Senior Developer Trainer, Apple iServices

Roadmap

407 WebObjects: EJB

The wonderful fruit...

Room J2
Wed., 2:00 p.m.

410 WebObjects: Optimization

Tuning your applications

Room J2
Thur., 9:00 a.m.

412 EOF Caching & Synchronization

Client-Server done right

Room J2
Thur., 2:00 p.m.

415 Advanced EOF

Everything you ever wanted to know...

Room J2
Fri., 9:00 a.m.

417 Building Large Applications

Welcome to the real world

Room J2
Fri., 2:00 p.m.



For More Information

<http://www.apple.com/webobjects>

Visit the WebObjects lab downstairs!
Everyday from 11:00 a.m.–2:00 p.m.

Try out your WebObjects 4.5 Evaluation CD!

WebObjects Community BOF
Wed., 6:30 p.m.–8:00 p.m.



Who to Contact

Toni Trujillo Vian

Director, WebObjects Engineering
wofeedback@group.apple.com

Ernest Prabhakar

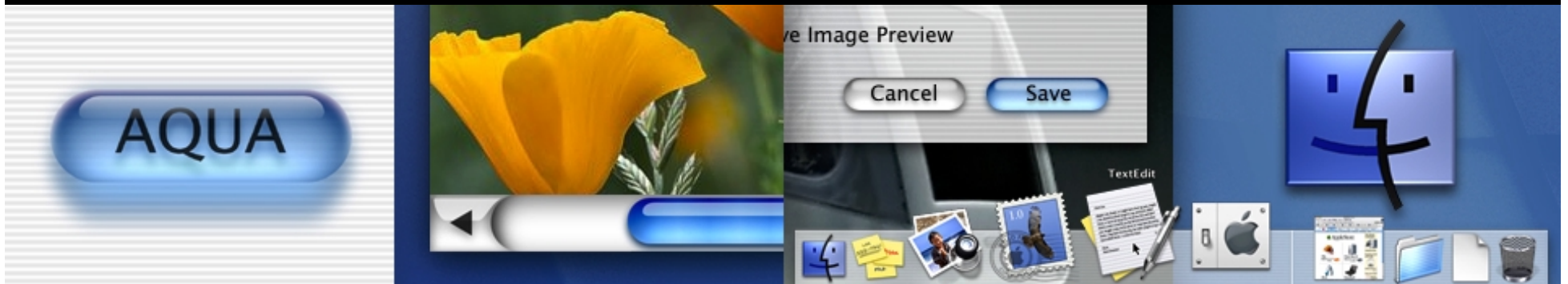
Product Line Manager, WebObjects
wofeedback@group.apple.com





Session 403

Q&A



Steve Miner
WebObjects Engineer



WWDC

Worldwide Developers Conference 2000



Think different.