Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

WebObjects

"This year's Software Product of the Year is Apple Computer's WebObjects 4.0. . . Along with substantial power and flflexibility, WebObjects brings an ease of development rarely seen in industrial-strength development environments."

EOModeler

David Scheck
and Nader Nafissi

Senior Systems Engineers

Agenda

- The Problem/The Solution (EOF)
- EOModeler Features
- Entities
- Attributes
- Relationships
- Fetch Specifications
- Interacting with DBMS
- Advanced Features
- Extending EOModeler

The Problem

"It is easy to lose the benefits of Objects when accessing relational databases."

Database Perspective

Client

clientId	last	birth	empId
521	Jones	4/30/69	42
522	Smith	3/7/75	45

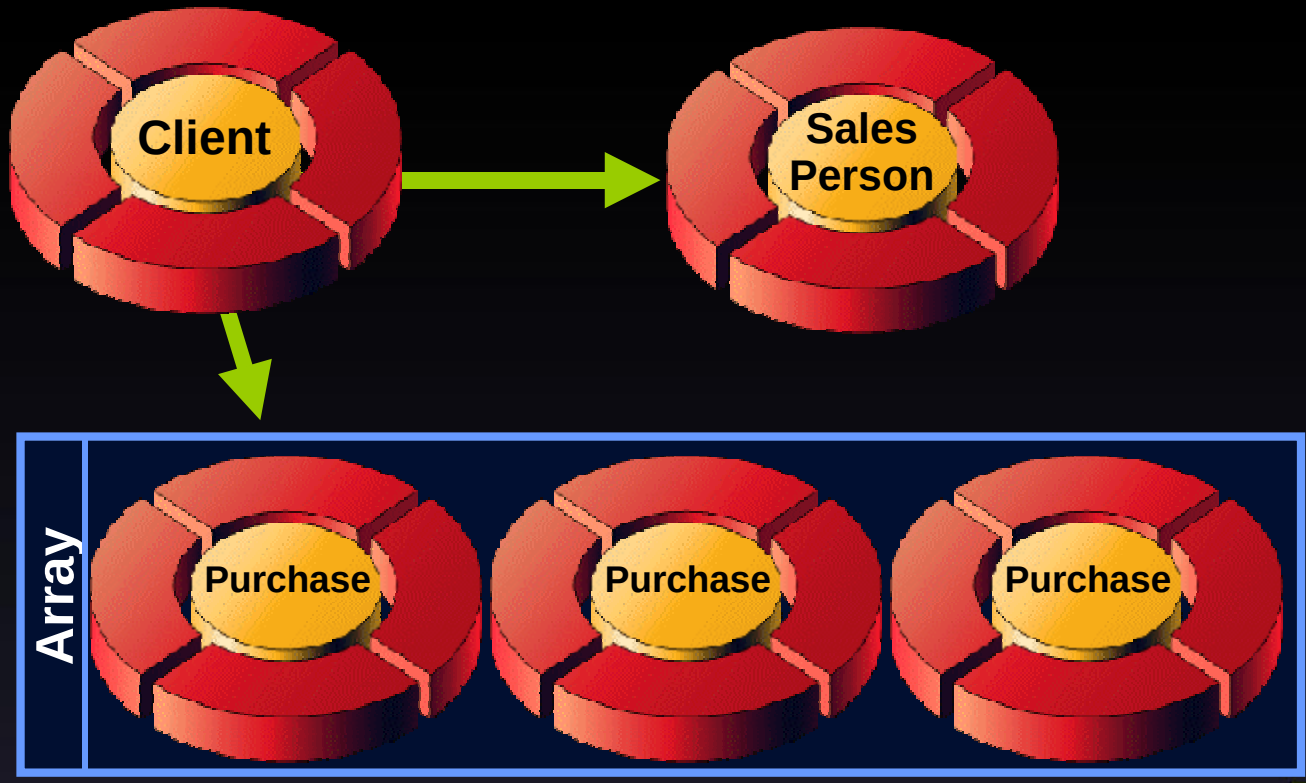
SalesPerson

empId	name
42	Don Ber
43	Ric Vee

Purchase

purchaseId	prodId	clientId
1556	630	521
1594	720	521

Object Perspective



Goals

- Work with business objects, not rows and columns
- Automatic synching of rows and objects
- Keep objects free of database access plumbing code
- Can relate objects between different databases
- No code generation for database access

Goals (Cont.)

- Not SQL centric—reusable qualifiers and sort orders
- Portable across different database vendors
- Configurable parameters for optimizing performance
- Hooks everywhere for special cases
- Graphical tools for modeling

The Solution: EOF

- EOF is a collection of classes and tools that allow you to get your enterprise objects into and out of a relational database
- EOF allows you to keep your objects free of information about the database
- Basically, EOF maps instances of an object class into rows of a database table

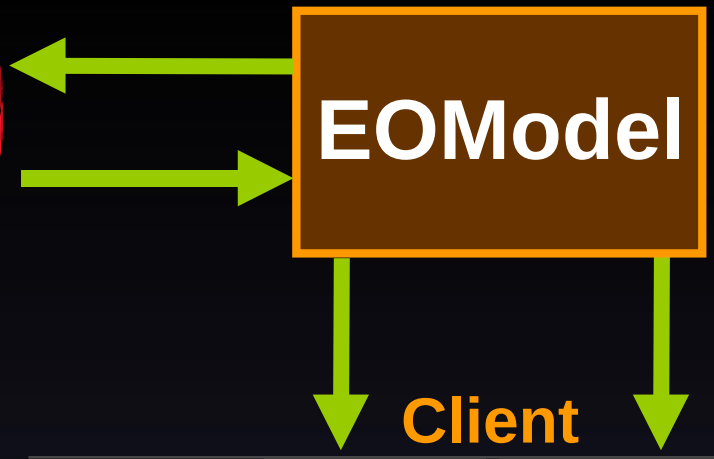
EOF: Key Features

- Uniques rows into business objects
- Saves row snapshot for locking and caching
- Provides relationship traversal with faults
- Many-to-Many table hiding
- Remembers changes and provides undo

EOF: Key Features (Cont.)

- Generates appropriate SQL at runtime
- Uses open adaptor design to support different data sources using their native client libraries (Not just odbc/jdbc)
- EOF aware tools to model and use business objects

EOModel = Mapping



Client

clientId	last	birth	empId
521	Jones	4/30/69	42
522	Smith	3/7/75	45

Modeler Features

- Reverse engineer model from database
- Generate and execute SQL to create database
- Switch model to different database
- Add Entities, Attributes, Relationships, FetchSpecifications, Store Procedures
- Flattening attributes and relationships
- Defining Entity inheritance

Modeler Features (Cont.)

- Defining Entity object class
- Generating class sub files in Java or ObjC
- Generating Java Client sub files
- Defining referential integrity rules
- User defined data on all model objects
- EOModel prototypes
- Creating EOModeler bundle extensions

Creating a New Model

- Select database adaptor
- Connect to database
- Answer Questions
    - Primary Keys
    - Referential Integrity
    - Stored Procedures
    - Object class for row

What Is an Entity?

- Maps a table to an object
- Contains attributes (columns)
- Can have relationships to other entities
- Can have predefined fetch specifications
- Can have stored procedure call mappings

Entity Options

- Object server and java client class name
- Primary keys, locking keys, class props
- Is abstract and is read only
- Cache in memory
- Batch faulting size
- External query
- Limiting qualifier

What Is an Attribute?

- Defines a mapping to a data element
- Can be column or derived
- Can be flattened from a relationship

Attribute Options

- Is primary key, used for locking, class property
- Is read only
- Allows NULL value
- Specifies external datatype
- Specifies value class
- Can use attribute prototypes

What Is a Relationship?

- Defines a mapping between two Entities
- Can be either toOne or toMany
- Can be defined across model files (dbs)
- Can be used to hide Many-to-Many tables

Relationship Options

- toMany batch faulting size
- Referential Integrity rules
    - Owns Destination
    - Propagate primary key
    - Delete Rules (Cascade, Nullify, Deny, None)
- Is optional

App Demo

- New db wizard app (Employees entity)
  - Selected Record/Matching Record
  - FirstName,LastName,mailStop,email
  - lastName
  - FirstName,LastName
- Compile and Run it
- Add firstName to hyperlink
- Add toJobTitle.jobTitleShortName

App Demo (Cont.)

- Add departments picklist to qualifier
  - Add WOPopUpButton
  - Drag Department entity from EOM
  - Add department variable (dept)
- Bind popup
  - list = deptDG.allObjects
  - item = dept
  - displayString = dept.departmentName
  - selection = empDG.queryMatch.toDepartment
  - noSelectionString = “No Choice”
- Show generated SQL (no batch faulting)

SQL Output

- SELECT t0.department_Id, t0.department_Name, t0.department_Short_Name, t0.internal_Code FROM Department t0"
- SELECT t0.department_Id, t0.email, t0.employee_Id, t0.employee_Status_Id, t0.first_Name, t0.job_Title_Id, t0.last_Name, t0.mail_Stop, t0.manager_Id, t0.office_Phone, t0.other_Phone, t0.site_Id FROM Employee t0 ORDER BY t0.last_Name asc, t0.first_Name asc

Notice that each job title is fetched separately, no batch faulting set

- SELECT t0.internal_Code, t0.job_Title_Id, t0.job_Title_Name, t0.job_Title_Short_Name FROM Job_Title t0 WHERE t0.job_Title_Id = :job_Title_Id0 " withBindings:{job_Title_Id0 = 2; }
- SELECT t0.internal_Code, t0.job_Title_Id, t0.job_Title_Name, t0.job_Title_Short_Name FROM Job_Title t0 WHERE t0.job_Title_Id = :job_Title_Id0 " withBindings:{job_Title_Id0 = 1; }
- SELECT t0.internal_Code, t0.job_Title_Id, t0.job_Title_Name, t0.job_Title_Short_Name FROM Job_Title t0 WHERE t0.job_Title_Id = :job_Title_Id0 " withBindings:{job_Title_Id0 = 4; }

What Is a FetchSpec?

- Describes a fetch
  - Entity to fetch
  - Qualifiers to apply
  - Sort Orderings to apply
  - Distinct, FetchLimits, CustomRawSQL
  - Hints: Prefetch Relationships, etc
- Predefined fetchspec defined in model

Why Create FetchSpecs?

- Predefined FetchSpecs can be assigned to WODisplayGroups or accessed in code
- WODisplayGroup's query match bindings are always AND'd together
- Predefined FetchSpecs can be complex; Subsitition variables are in Display Group's query bindings

Creating a FetchSpec

- Create new fetchSpec
- Define qualifiers with substitution vars
- Setup sort ordering
- Options
    - Prefetch relationships
    - Distinct, Fetch Limits, Locking, etc
    - Raw rows, Custom SQL

Demo: FetchSpec

- firstNameOrLastNameNotInDept
  - Qualifier
    - firstName like $firstName
    - lastName like $lastName
    - AND
    - NOT toDepartment = $deptObject
  - Sort Orderings
    - firstName, LastName

Demo: FetchSpec

- Bring up the WODisplayGroup inspector
- Select our fetchSpec in the popup
- Bind display group query bindings
    - firstName to First Name text field
    - lastName to Last Name text field
    - deptObject to Dept popup selection
- Compile and Run

SQL Output

Notice the complex qualifier for:

- **FirstName = Da***
- **LastName = Sc***
- **Not In Department = Engineering Technical Support**

```
SELECT  t0.department_Id, t0.email, t0.employee_Id,
t0.employee_Status_Id, t0.first_Name, t0.job_Title_Id,
t0.last_Name, t0.mail_Stop, t0.manager_Id, t0.office_Phone,
t0.other_Phone, t0.site_Id FROM Employee t0 WHERE (not
(t0.department_Id = :department_Id0 ) AND (t0.first_Name
like :first_Name1 ESCAPE '\' OR t0.last_Name like
:last_Name2 ESCAPE '\')) ORDER BY t0.last_Name asc,
t0.first_Name asc"

withBindings:{department_Id0 = 2; first_Name1 = "Da%";
last_Name2 = "Sc%"
```

Interacting with DBMS

- Import/Export of data
- Stored procedure support
- Generate SQL to create database schema
- Switching between databases

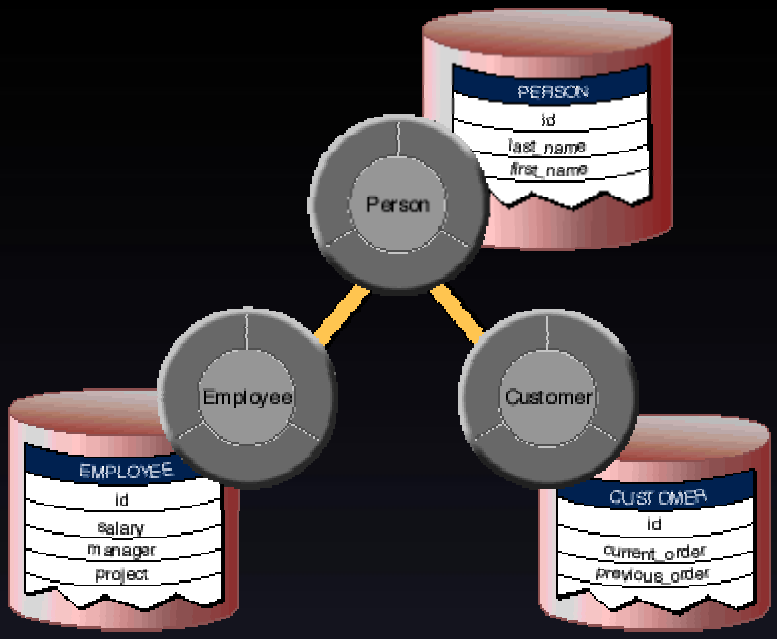
Advanced Features

- Entity Inheritance
- Cross model/dbms relationships
- Generating class stub files (Objc, Java)
- Client side Java
- User Info Dictionary on model objects
- Attribute Prototypes
- Overriding defaults

Entity Inheritance

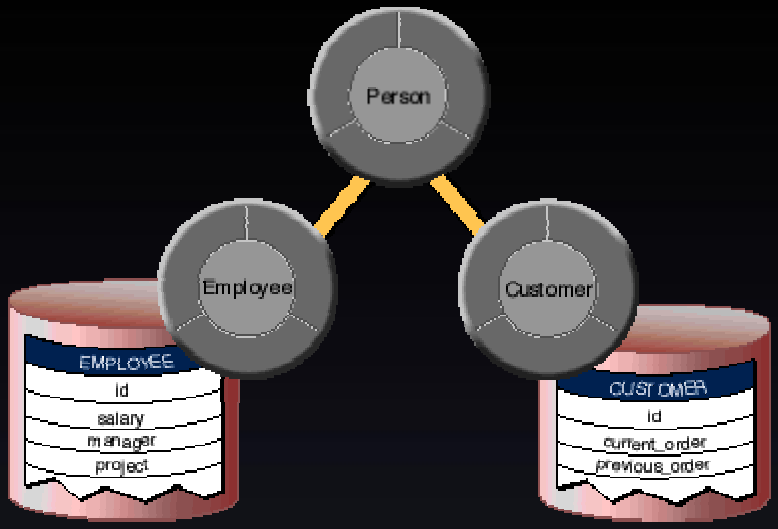
- Different than EO inheritance
- Vertical Mapping
  - Storage directly reflects the class hierarchy
  - Least efficient approach
- Horizontal Mapping
  - Does not need to join to resolve relationships
  - Altering the root is painful
- Single Table Mapping
  - Fastest approach
  - Requires having lots of NULLs in DBMS

Vertical Mapping



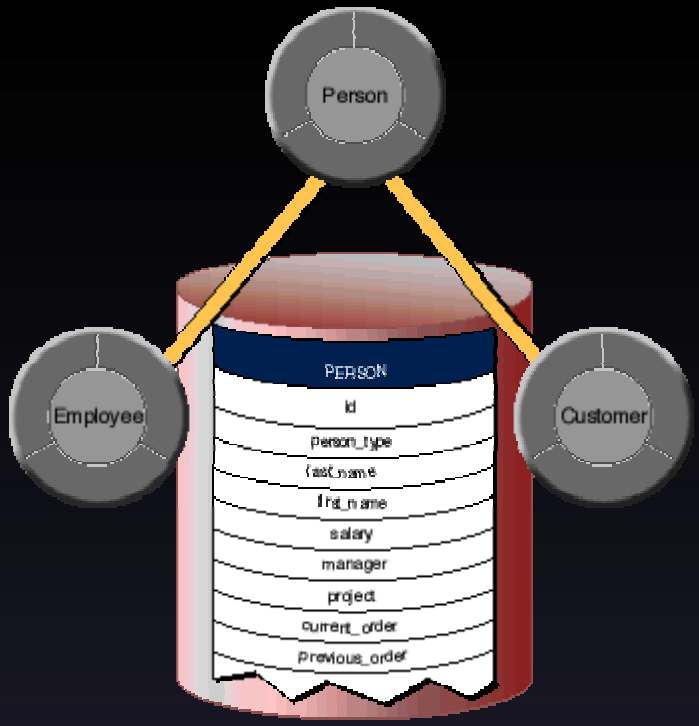
- A subclass can be added at any time without modifying the Person table
- Vertical mapping is the least efficient of all of the approaches
- Every layer of the class hierarchy requires a join to resolve the relationships

Horizontal Mapping



- A subclass can be added at any time without modifying other tables
- Works well for deep class hierarchies, as long as the fetch occurs against the leaves of the class hierarchy

Single Table Mapping



- This approach is faster than the other two methods for deep fetches
- Unlike vertical or horizontal mapping, you can retrieve superclass objects with a single fetch, without performing joins
- Adding a subclass or modifying the superclass requires changes to just one table

Inheritance at a Glance

	Fetches from Leaves	Fetches from Root
Vertical Mapping	1 fetch using join	n fetches using join
Horizontal Mapping	1 fetch	n fetches
Single Table Mapping	1 fetch	1 fetch

In the table,
"n" represents the number of entities involved in a deep fetch

- In Java you can not have a to-one relationship to an ambiguous to-one relationship, unless you implement a workaround
- Ambiguous to-one relationships are not possible in Java because of strong typing in the language

Joining Models/dbms

- Finds all models using PB.project
- Creates EOModelGroup containing
    - All models in Resources
    - All models in Frameworks
- Relationship inspector shows all known models in popup list

Extending App Demo

- Close EOMWWDC model
- Add EOMWWDCSite model to project
- Open both models
- Show site table in data browser
- Add relationship from Employee to Site
- Add toSite.siteName in GUI
- Add batch fault size 20 to Site Entity

Extend App Demo (Cont.)

- Flatten Many-to-Many relationship
  Employees.toGeographies
- Drag in Geography entity for new DG
- Add WOToManyRelationship
  dataSource = geographyDisplayGroup.dataSource
  destinationDisplayKey = "geographyName"
  relationshipKey = "toGeographies"
  sourceEntityName = "Employee"
  sourceObject = employeeDisplayGroup.selectedObject

Extend App Demo (Cont.)

- Create custom Employee class
- Add fullName method

```
public String fullName() {
        return firstName()+" "+ lastName();
}
```

- Replace fullName and lastName with fullName *(Notice, WOB picks up the new method immediately)*
- Notice Batch Faulting, show SQL

SQL Output

Notice the batch fault size of 20 being fired

```
SELECT  t0.ADDRESS1, t0.ADDRESS2, t0.CITY,
t0.COUNTRY, t0.FAXPHONE, t0.INTERNALCODE,
t0.OFFICEPHONE, t0.POSTALCODE, t0.REGION, t0.SITEID,
t0.SITENAME FROM SITE t0 WHERE (t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ? OR t0.SITEID = ? OR t0.SITEID = ? OR
t0.SITEID = ?)"

withBindings:(1:28(siteid), 2:22(siteid), 3:21(siteid),
4:38(siteid), 5:83(siteid), 6:42(siteid), 7:84(siteid),
8:86(siteid), 9:40(siteid), 10:19(siteid), 11:43(siteid),
12:39(siteid), 13:13(siteid), 14:72(siteid), 15:88(siteid),
16:9(siteid), 17:14(siteid), 18:44(siteid), 19:6(siteid),
20:81(siteid))
```

Overriding Defaults

```
defaults write NSGlobalDomain
   EOAdaptorDebugEnabled YES

defaults write NSGlobalDomain
   EOOracleTableNamesSQL “SELECT TABLE_NAME
   FROM USER_TABLES ORDER BY TABLE_NAME”

defaults write NSGlobalDomain
   EOSybaseTableNamesSQL “select name from
   sysobjects where type = 'U' or type = 'V'”

defaults write EOModeler
   SkipBeautifyNamesOnModelCreation YES
```

Extending EOModeler

- EOModeler.framework
- Example Bundles
    - /Developer/Examples/EnterpriseObjects /AppKit/ModelerBundle
    - PDF Reporting
    - OracleBeautifier
    - Direct To Web Launcher

Demo

Example Bundles

Summary

- Fetched objects from two dbms
- Related objects in object graph
- Added popup list to qualifier
- Qualified on predefined fetchSpec
- Flattened Many-to-Many
- Create custom business object
- Without really writing any code!

WebObjects Sessions

Deploying WebObjects	Hall B **Wed., 10:15am**
Generating Reports in WebObjects Apps	Hall B **Wed., 1:00pm**
Debugging Techniques in WebObjects	Hall B **Wed., 2:30pm**
Fine-Tuning the Performance of WebObjects	Hall B **Wed., 4:00pm**

WebObjects Sessions

Complex WebObjects Application Design	Hall B **Thur., 9:00am**
WebObjects: Tips from the Experts	Hall B **Thur., 10:15am**
Advanced HTML/DHTML/ JavaScript Tips and Tricks	Hall B **Thur., 4:00pm**

Think different.™

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys