Database Design: SQL (cont.), Conceptual Model, UML and The Logical Model

University of California, Berkeley

School of Information

INFO 257: Database Management

Announcements

- Questions about A1 and 2a?
- A1 and A2a due next week
  - Late Policy updated
- Lecture
- Lab – Project Ideas and Classmate Introductions

SQL Concepts Continued

- Joins
- Data Integrity Controls
- Row Value & Aggregates
- Query Design and Tips
- Subqueries
- Views

Processing Multiple Tables

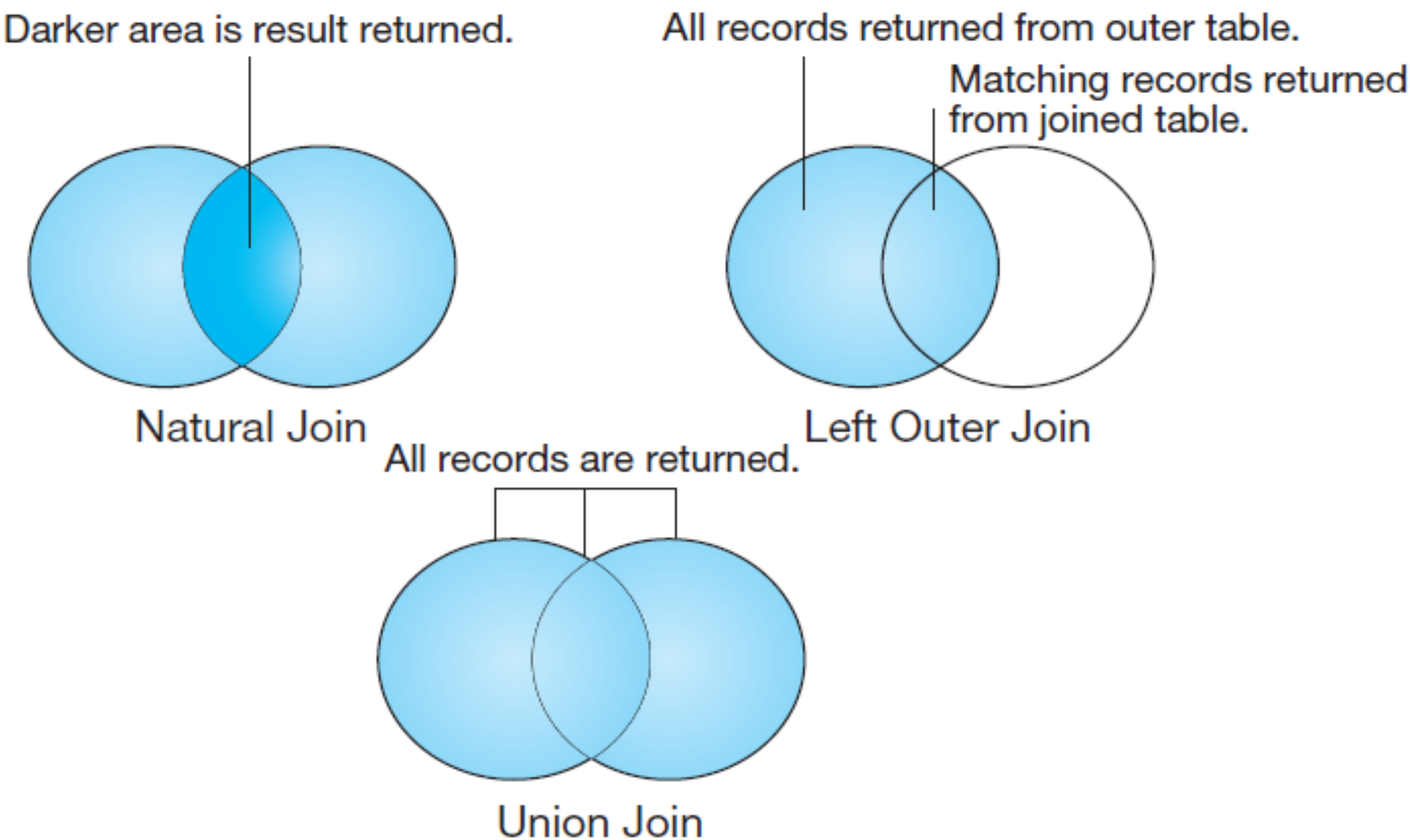
- **Join**–a relational operation that causes two or more tables with a common domain to be combined into a single table or view
- **Equi-join**–a join in which the joining condition is based on equality between values in the common columns; common columns appear redundantly in the result table
- **Natural join**–an equi-join in which one of the duplicate columns is eliminated in the result table

The common columns in joined tables are usually the primary key of the dominant table and the foreign key of the dependent table in 1:M relationships

Processing Multiple Tables–Joins

- Outer join - a join in which
  - rows that do not have matching values in common columns are nonetheless included in the result table
  - (as opposed to *inner* join, in which rows must have matching values in order to appear in the result table)
- Union join–includes all columns from each table in the join, and an instance for each row of each table

Visualization of different join types



"Golden Rules" of Outer join

- Rule 1:
- OUTER JOIN **can <u>only join 2 tables</u>**
  - *IF* attempt to join more than two tables in outer join, unexpected outcomes would result.
- Rule 2:
- Do NOT abuse outer join: UNLESS you truly want ALL rows from one table (including unmatched rows), should you use outer join

Subqueries

- Subquery – placing an inner query (SELECT statement) inside an outer query
- Options:
  - In a condition of the WHERE clause
  - As a “table” of the FROM clause
  - Returning a field for the SELECT clause
  - Within the HAVING clause
- Subqueries can be:
  - Noncorrelated – executed once for the entire outer query
  - Correlated – executed once for each row returned by the outer query

Subquery Example

What are the name and address of the customer who placed order number 1008?

```
SELECT CustomerName, CustomerAddress, CustomerCity, CustomerState,
CustomerPostalCode
  FROM Customer_T
    WHERE Customer_T.CustomerID =
      (SELECT Order_T.CustomerID
        FROM Order_T
          WHERE OrderID = 1008);
```

Alternative Approach, Using a Join

What are the name and address of the customer who placed order number 1008?

```
SELECT CustomerName, CustomerAddress, CustomerCity,
   CustomerState, CustomerPostalCode
FROM Customer_T, Order_T
WHERE Customer_T.CustomerID = Order_T. CustomerID
   AND OrderID = 1008;
```

Correlated versus. Noncorrelated Subqueries

- Noncorrelated subqueries:
  - Do not depend on data from the outer query
  - Execute once for the entire outer query
- Correlated subqueries:
  - Make use of data from the outer query
  - Execute once for each row of the outer query
  - Can use the EXISTS and ALL operators

Example of a Correlated Subquery

List the details about the product with the highest standard price.

```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
FROM Product_T PA
  WHERE PA.ProductStandardPrice > ALL
    (SELECT ProductStandardPrice FROM Product_T PB
      WHERE PB.ProductID ! = PA.ProductID);
```

Result:

PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
Dining Table	Natural Ash	800

Another Correlated Subquery

What are the order IDs for all orders that have included furniture finished in natural ash?

```
SELECT DISTINCT OrderID FROM OrderLine_T
WHERE EXISTS
        (SELECT *
          FROM Product_T
                 WHERE ProductID = OrderLine_T.ProductID
                       AND Productfinish = 'Natural Ash');
```

A correlated subquery always refers to an attribute from a table referenced in the outer query.

Figure 6-8 Subquery Processing
(1 of 2)

a) Processing a noncorrelated subquery

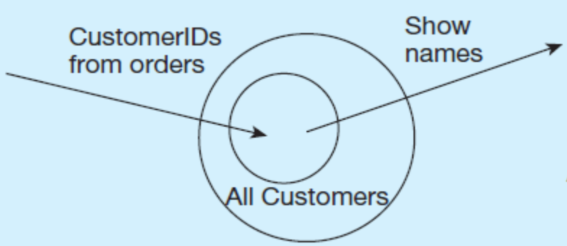
What are the names of customers who have placed orders?

```
SELECT CustomerName
        FROM Customer_T
              WHERE CustomerID IN
                                  (SELECT DISTINCT CustomerID
                                   FROM Order_T);
```

1. The subquery (shown in the box) is processed first and an intermediate results table created:

CUSTOMERID
1
8
15
5
3
2
11
12
4

9 rows selected.



2. The outer query returns the requested customer information for each customer included in the intermediate results table:

CUSTOMERNAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

9 rows selected.

Figure 6-8 Subquery Processing
(2 of 2)

b) Processing a correlated subquery

What are the order IDs for all orders that have included furniture finished in natural ash?

```
SELECT DISTINCT OrderID FROM OrderLine_T
WHERE EXISTS
        (SELECT *
          FROM Product _T
                WHERE ProductID = OrderLine_T.ProductID
                   AND Productfinish = 'Natural Ash');
```

	OrderID	ProductID	OrderedQuantity
1 →	1001	1	1
	1001	2	2
	1001	4	1
3 →	1002	3	5
	1003	3	3
	1004	6	2
	1004	8	2
	1005	4	4
	1006	4	1
	1006	5	2
	1007	1	3
	1007	2	2
	1008	3	3
	1008	8	3
	1009	4	2
	1009	7	3
	1010	8	10
*	0	0	0

	ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID
	1	End Table	Cherry	$175.00	10001
2 →	2	Coffee Table	Natural Ash	$200.00	20001
4 →	3	Computer Desk	Natural Ash	$375.00	20001
	4	Entertainment Center	Natural Maple	$650.00	30001
	5	Writer's Desk	Cherry	$325.00	10001
	6	8-Drawer Dresser	White Ash	$750.00	20001
	7	Dining Table	Natural Ash	$800.00	20001
	8	Computer Desk	Walnut	$250.00	30001
*	(AutoNumber)			$0.00	

Derived Table (Subquery in the FROM Clause of the Outer Query)

What are the order IDs for all orders that have included furniture finished in natural ash?

```
SELECT ProductDescription, ProductStandardPrice, AvgPrice
   FROM
      (SELECT AVG(ProductStandardPrice) AvgPrice FROM Product_T),
         Product_T
   WHERE ProductStandardPrice > AvgPrice;
```

Here, the subquery forms the derived table used in the FROM clause of the outer query. The AvgPrice column from the subquery is used in the SELECT clause of the outer query.

More Complicated SQL Queries

- Production databases contain hundreds or even thousands of tables, and tables could include hundreds of columns.
- So, sometimes query requirements can be very complex.
- Sometimes it’s useful to combine queries, through the use of Views.
- If you use a view (which is a query), you could have another query that uses the view as if it were a table.

Using a View in Your Query

For each salesperson, list his or her biggest-selling product.

The view:

```
CREATE VIEW TSales AS
SELECT SalespersonName,
       ProductDescription,
       SUM(OrderedQuantity) AS Totorders
FROM Salesperson_T, OrderLine_T, Product_T, Order_T
       WHERE Salesperson_T.SalespersonID=Order_T.SalespersonID
       AND Order_T.OrderID=OrderLine_T.OrderID
       AND OrderLine_T.ProductID=Product_T.ProductID
       GROUP BY SalespersonName, ProductDescription;
```

The query using the view:

```
SELECT SalespersonName, ProductDescription
       FROM TSales AS A
         WHERE Totorders = (SELECT MAX(Totorders) FROM TSales B
         WHERE B.SalesperssonName = A.SalespersonName);
```

Using and Defining Views

- Dynamic View
  - A “virtual table” created dynamically upon request by a user
  - No data actually stored; instead data from base table made available to user
  - Based on SQL SELECT statement on base tables or other views
- Materialized View
  - Copy or replication of data, data actually stored
  - Must be refreshed periodically to match corresponding base tables

A Sample Create View Command

```
CREATE VIEW ExpensiveStuff_V
  AS
    SELECT ProductID, ProductDescription, ProductStandardPrice
      FROM Product_T
        WHERE ProductStandardPrice > 300
        WITH CHECK OPTION;
```

- View has a name
- View is based on a SELECT statement
- CHECK_OPTION works only for updateable views and prevents updates that would create rows not included in the view

Advantages of Dynamic Views (1 of 2)

- Simplify query commands
- Assist with data security
- Enhance programming productivity
- Contain most current base table data
- Use little storage space
- Provide customized view for user
- Establish physical data independence

Advantages of Dynamic Views (2 of 2)

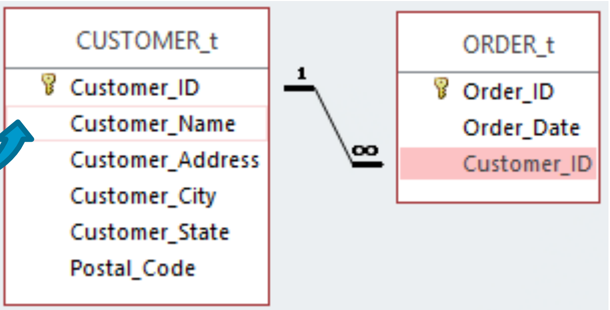
- Use processing time each time view is referenced
- May or may not be directly updateable
- As with all SQL constructs, you should use views with discretion

Issues about row value & aggregates

- SQL **cannot return both** a row value (such as Product_ID) and a set value (such as COUNT/AVG/SUM of a group);
  – users must run two separate queries, one that returns row info and one that returns set info

1. SELECT S_ID FROM STUDENT - Row
2. SELECT AVG(GPA) FROM STUDENT - Set
3. SELECT S_ID, AVG(GPA) FROM STUDENT – row & set

Tips for Developing Queries



- Be familiar with the data model (entities and relationships)
- Understand the desired results
- Know the attributes desired in result
- Identify the entities that contain desired attributes
- Review ERD
- Construct a WHERE equality for each table join
- Fine tune with GROUP BY and HAVING clauses if needed
- Consider the effect on unusual data

Query Efficiency Considerations

- Instead of SELECT *, identify the specific attributes in the SELECT clause; this helps reduce network traffic of result set
- Limit the number of subqueries; try to make everything done in a single query if possible
- If data is to be used many times, make a separate query and store its results rather than performing the query repeatedly

Guidelines for Better Query Design

- Understand how indexes are used in query processing
- Use compatible data types for fields and literals
- Write simple queries
- Break complex queries into multiple simple parts
- Don't nest one query inside another query
- Don't combine a query with itself (if possible avoid self-joins)

Guidelines for Better Query Design

- Create temporary tables for groups of queries
- Combine update operations
- Retrieve only the data you need
- Don’t have the DBMS sort without an index
- Learn!
- Consider the total query processing time for ad hoc queries

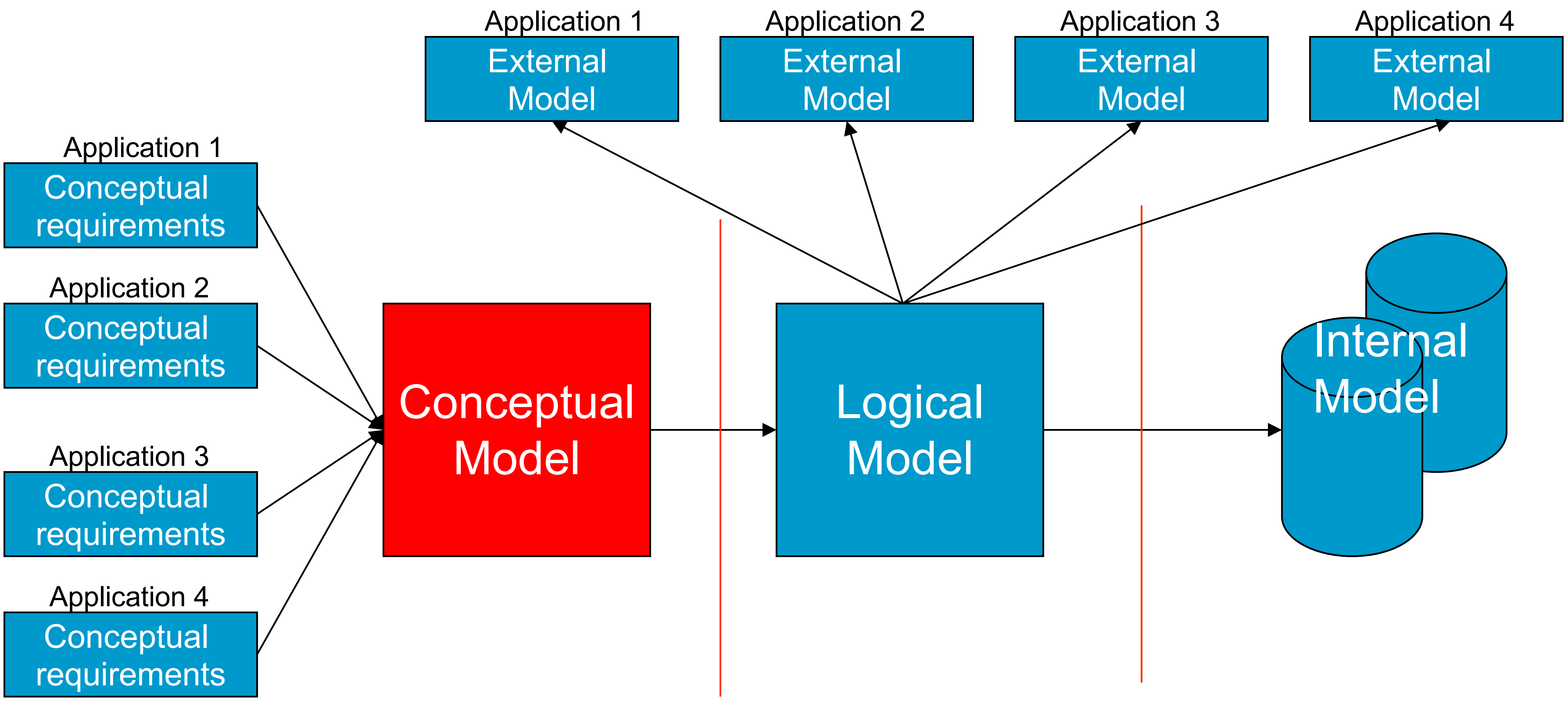
Lecture Outline

- Review (and continuation)
  - Database Design, Conceptual Model
- Assignment 2b – Personal Database Conceptual Design
- Object-Oriented Modeling in UML
- The Logical Model

Lecture Outline

- Review (and continuation)
  - Database Design, Conceptual Model
- Assignment 2 – Personal Database Conceptual Design
- Object-Oriented Modeling in UML
- The Logical Model

Database Design Process



Developing a Conceptual Model

- Overall view of the database that integrates all the needed information discovered during the requirements analysis.
- Elements of the Conceptual Model are represented by diagrams, *Entity-Relationship or ER Diagrams*, that show the meanings and relationships of those elements independent of any particular database systems or implementation details.
- Can also be represented using other modeling tools (such as UML – more later)

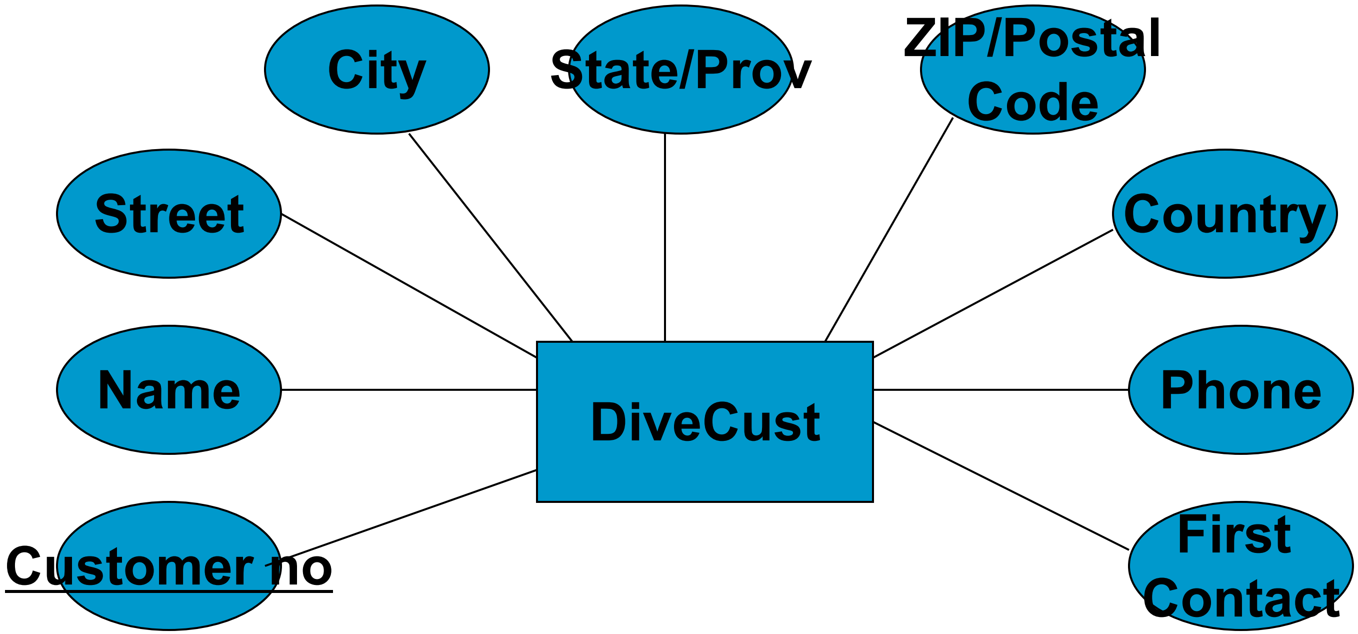
Developing a Conceptual Model

- We will look at a small business -- a diveshop that offers diving adventure vacations
- Assume that we have done interviews with the business and found out the following information about the forms used and types of information kept in files and used for business operations...

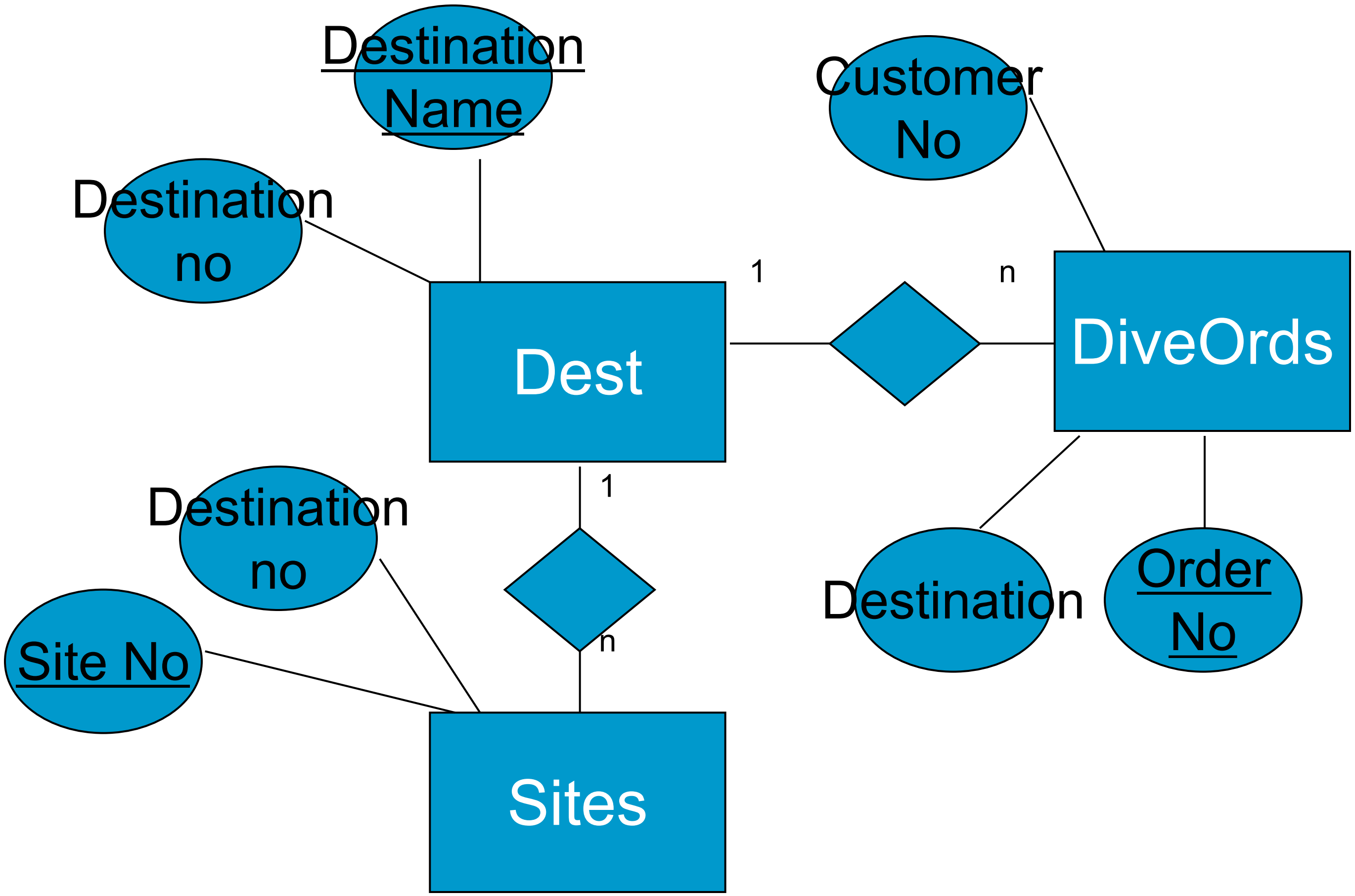
Entities

- Customer
- Dive Order
- Line item
- Shipping information
- Dive Equipment/ Stock/Inventory
- Dive Locations
- Dive Sites
- Sea Life
- Shipwrecks

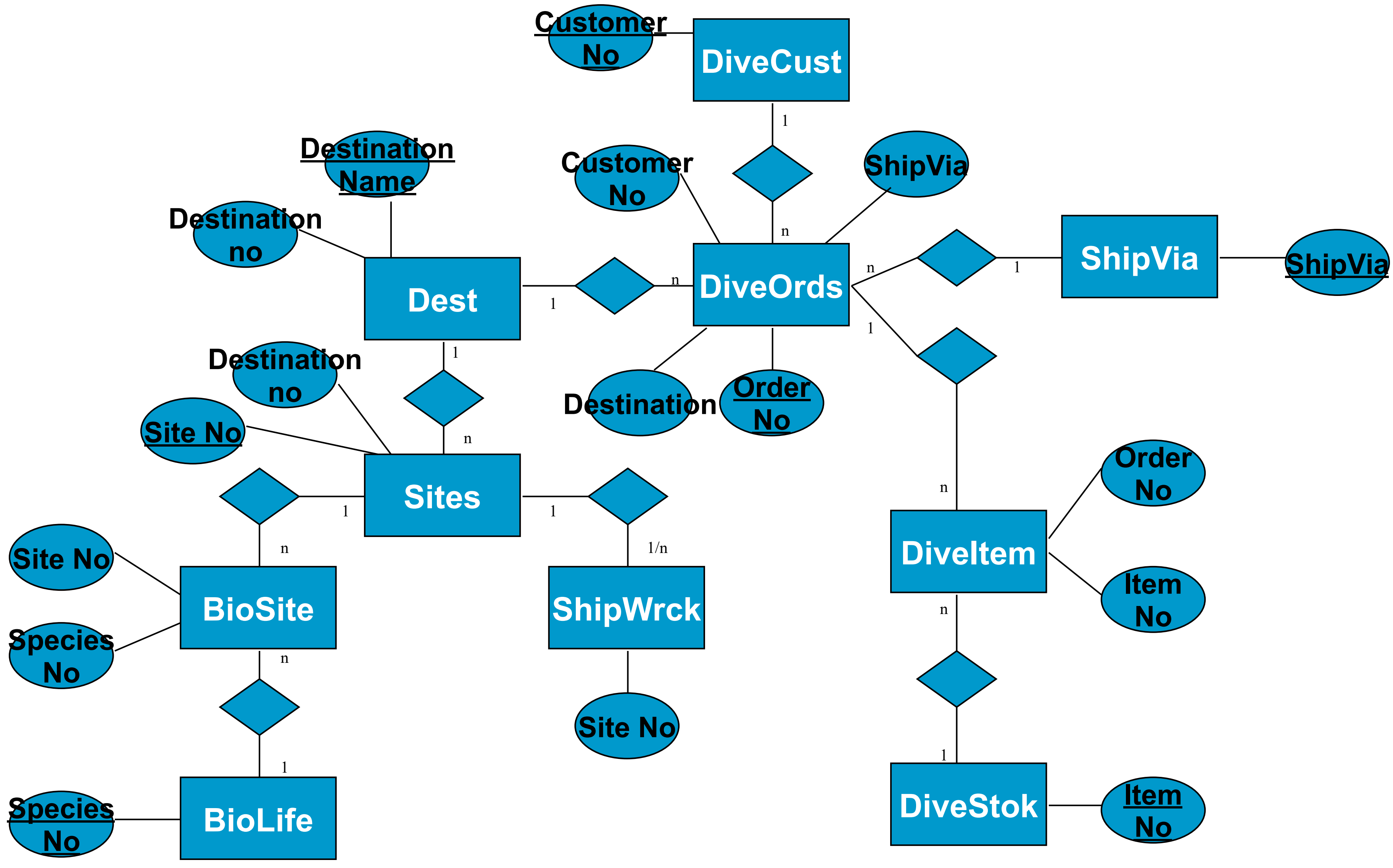
Diveshop Entities: DIVECUST



Destination/ Sites



DiveShop ER Diagram



Another ERD Example (Crows Foot)

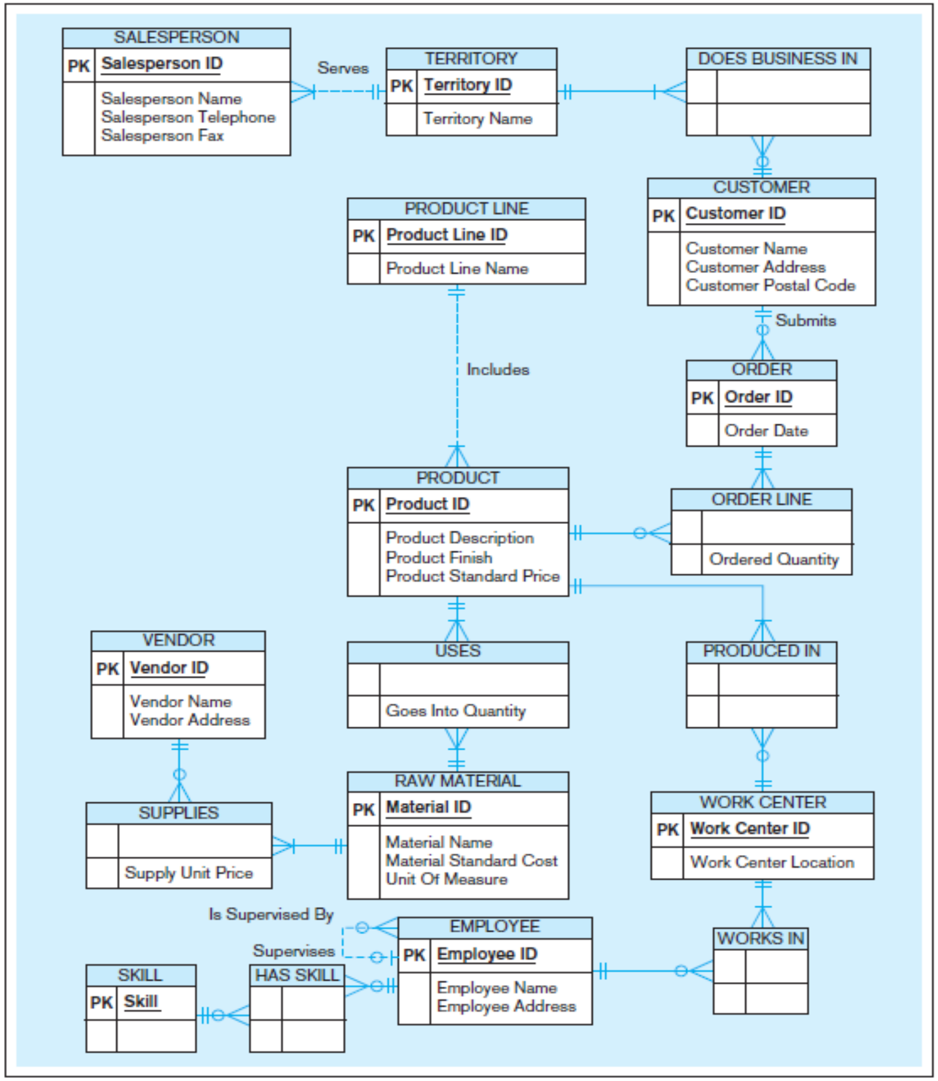


Figure 2-22
Data model for Pine Valley Furniture Company in Microsoft Visio notation

Different modeling software tools may have different notation for the same constructs.

Online references of E-R diagram

A concise but fairly complete E-R model quick reference:

- http://creately.com/blog/diagrams/er-diagrams-tutorial/

E-R diagram elements (Chen and crow's foot), with examples

- http://www.conceptdraw.com/solution-park/diagramming-ERD

Lecture Outline

- Review (and continuation)
  - Database Design, Conceptual Model
- Assignment 2 – Personal Database Conceptual Design
- Object-Oriented Modeling
- The Logical Model

Assignment 2b

- Due Friday March 12th
- Personal Database Project Design
- **Note: decide groups by February 20th**

- The following information should be turned in for the preliminary design of your personal database project.

1. A written description of the data you will be using for the database, and what uses you might expect the database to have. (2-4 pages)
2. A preliminary data dictionary for the entities and attributes and format of the data elements of the database. You should have *at least 5 entities with some logical connections between them*. The data dictionary consists of all of the attributes that you have identified for each entity, along with indication of whether the attribute is a primary key (or part of a primary key), and what format the data will be (e.g.: text, decimal number, integer, etc.)
3. Produce an entity-relationship diagram of the database *OR* a UML diagram.
   – These will be preliminary design specifications, so do not feel that you must follow everything that you describe here in the final database design.
   – The report should be in PDF format

Assignment 2b – ERD Development

Steps of ERD development (logic flow)

1. Identify/define entity types
2. Identify/define attributes for each entity type
3. Identify/define relationships
4. State business rules (that contains meaning of cardinality – “for each instance in A, 1 (or M) instances of B can be related)
5. Based on the above, draw the ER diagram

Tools for ER (and UML) diagrams

- **Microsoft Visio** has various sets of diagramming templates for databases
- For Macs **OmniGraffle** has UML or spreadsheet templates that can be used for ER diagrams
- **Lucidchart**
- **DrawIO**
- More sophisticated (and open source) CASE tools are available such as:
  - **MySQLWorkbench** (for MySQL only)
- Many other drawing packages have ERD available (sometimes as add-ons)

Lecture Outline

- Review (and continuation)
  - Database Design, Conceptual Model
- Assignment 2 – Personal Database Conceptual Design
- Object-Oriented Modeling in UML
- The Logical Model

Object-Oriented Modeling

- Becoming increasingly important as
  - Object-Oriented and Object-Relational DBMS continue to proliferate
  - Databases become more complex and have more complex relationships than are easily captured in ER or EER diagrams

- *(Most UML examples based on McFadden, "Modern Database Management", 5th edition)*

Object Benefits

- Encapsulate both data and behavior
- Object-oriented modeling methods can be used for both database design and process design
  - Real-World applications have more than just the data in the database they also involve the processes, calculations, etc performed on that data to get real tasks done
  - OOM can be used for more challenging and complex problems

Unified Modeling Language (UML)

- Combined three competing methods
- Can be used for graphically depicting
  - Software designs and interaction
  - Database
  - Processes

CLASS

- A class is a named description of a set of objects that share the same attributes, operations, relationships, and semantics.
  - An object is an instance of a class that encapsulates state and behavior.
    - These objects can represent real-world things or conceptual things.
  - An attribute is a named property of a class that describes a range of values that instances of that class might hold.
  - An operation is a named specification of a service that can be requested from any of a class's objects to affect behavior in some way or to return a value without affecting behavior

UML Relationships

- An relationship is a connection between or among model elements.
- The UML defines four basic kinds of relationships:
  - Association
  - Dependency
  - Generalization
  - Realization

UML Diagrams

- The UML defines nine types of diagrams:
  - activity diagram
  - class diagram
    - Describes the data and some behavioral (operations) of a system
  - collaboration diagram
  - component diagram
  - deployment diagram
  - object diagram
  - sequence diagram
  - statechart diagram
  - use case diagram

Class Diagrams

- A class diagram is a diagram that shows a set of classes, interfaces, and/or collaborations and the relationships among these elements.

UML Class Diagram

DIVEORDS
Order No Customer No Sale Date Shipvia PaymentMethod CCNumber No of People Depart Date Return Date Destination Vacation Cost
CalcTotalInvoice() CalcEquipment()

Class Name

List of Attributes

List of operations

Object Diagrams

307:DIVORDS
Order No = 307 Customer No = 1480 Sale Date = 9/1/99 Ship Via = UPS PaymentMethod = Visa CCNumber = 12345 678 90 CCExpDate = 1/1/01 No of People = 2 Depart Date = 11/8/00 Return Date = 11/15/00 Destination = Fiji Vacation Cost = 10000

Differences from Entities in ER

- Entities can be represented by Class diagrams
- But Classes of objects also have additional operations associated with them

Operations

- Three basic types for database
  - Constructor
  - Query
  - Update

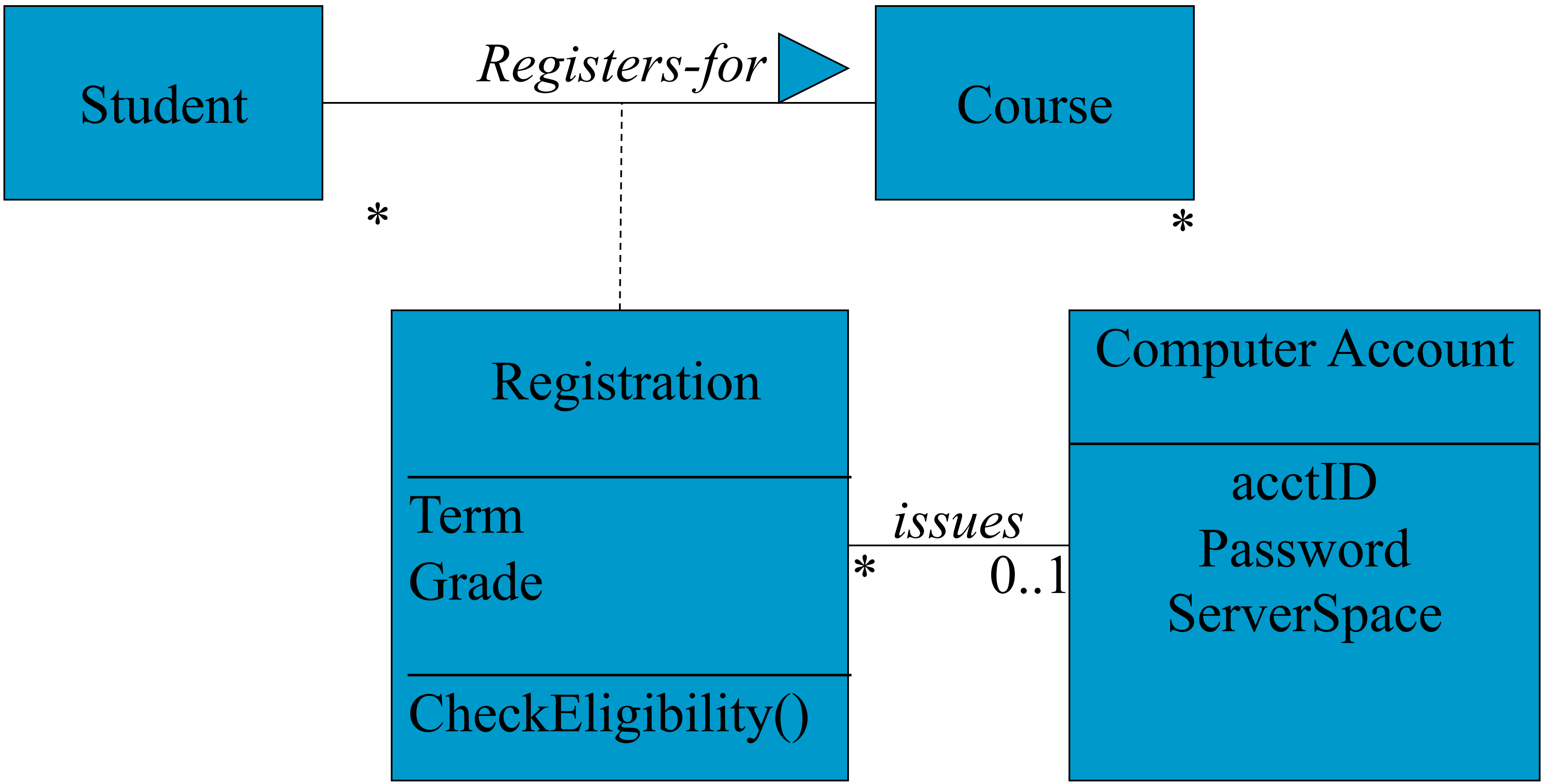
Associations

- An association is a relationship that describes a set of links between or among objects.
- An association can have a name that describes the nature of this relationship. You can put a triangle next to this name to indicate the direction in which the name should be read.

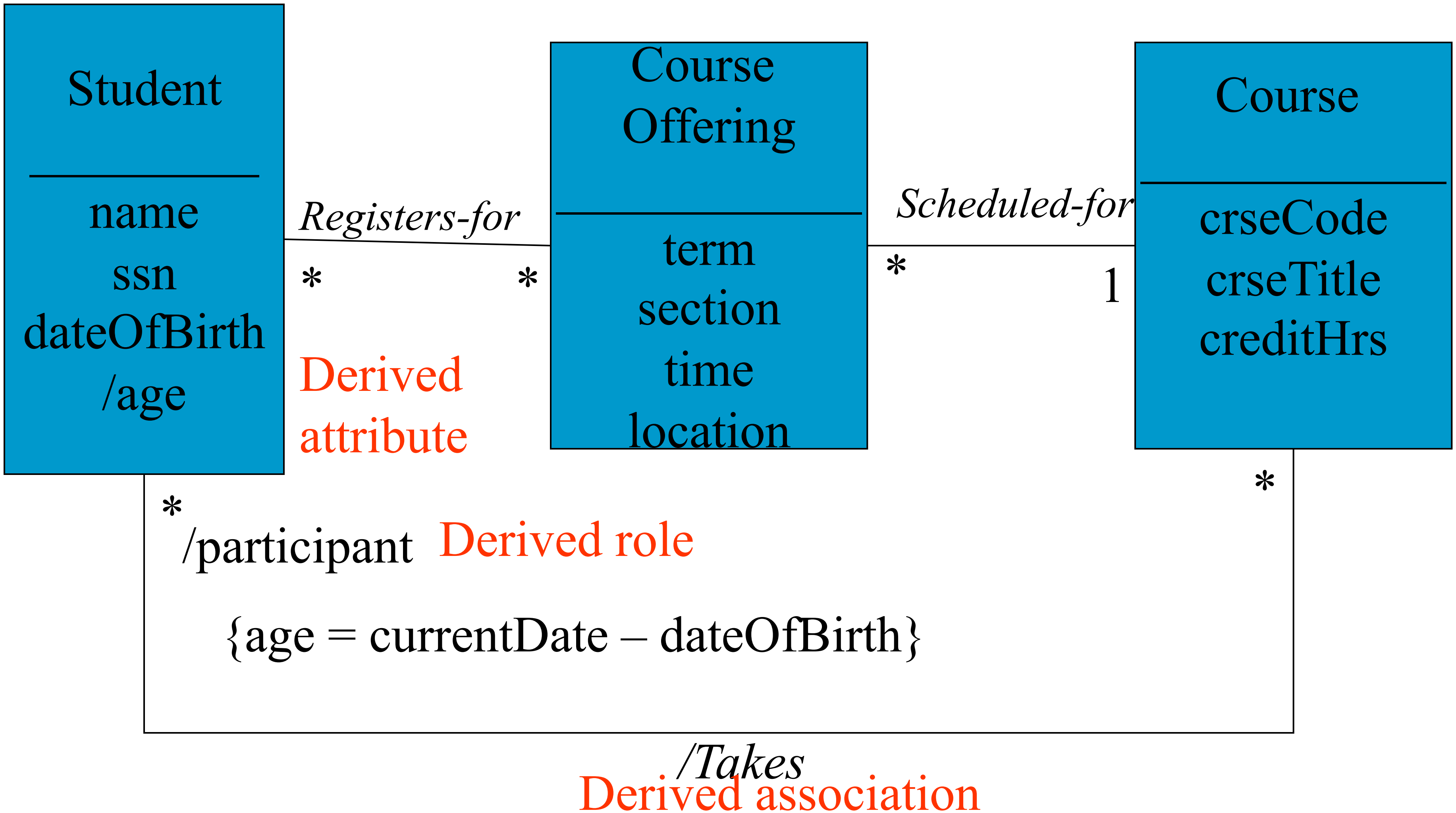
Associations

- An association contains an ordered list of association ends.
  – An association with exactly two association ends is called a binary association
  – An association with more than two ends is called an n-ary association.

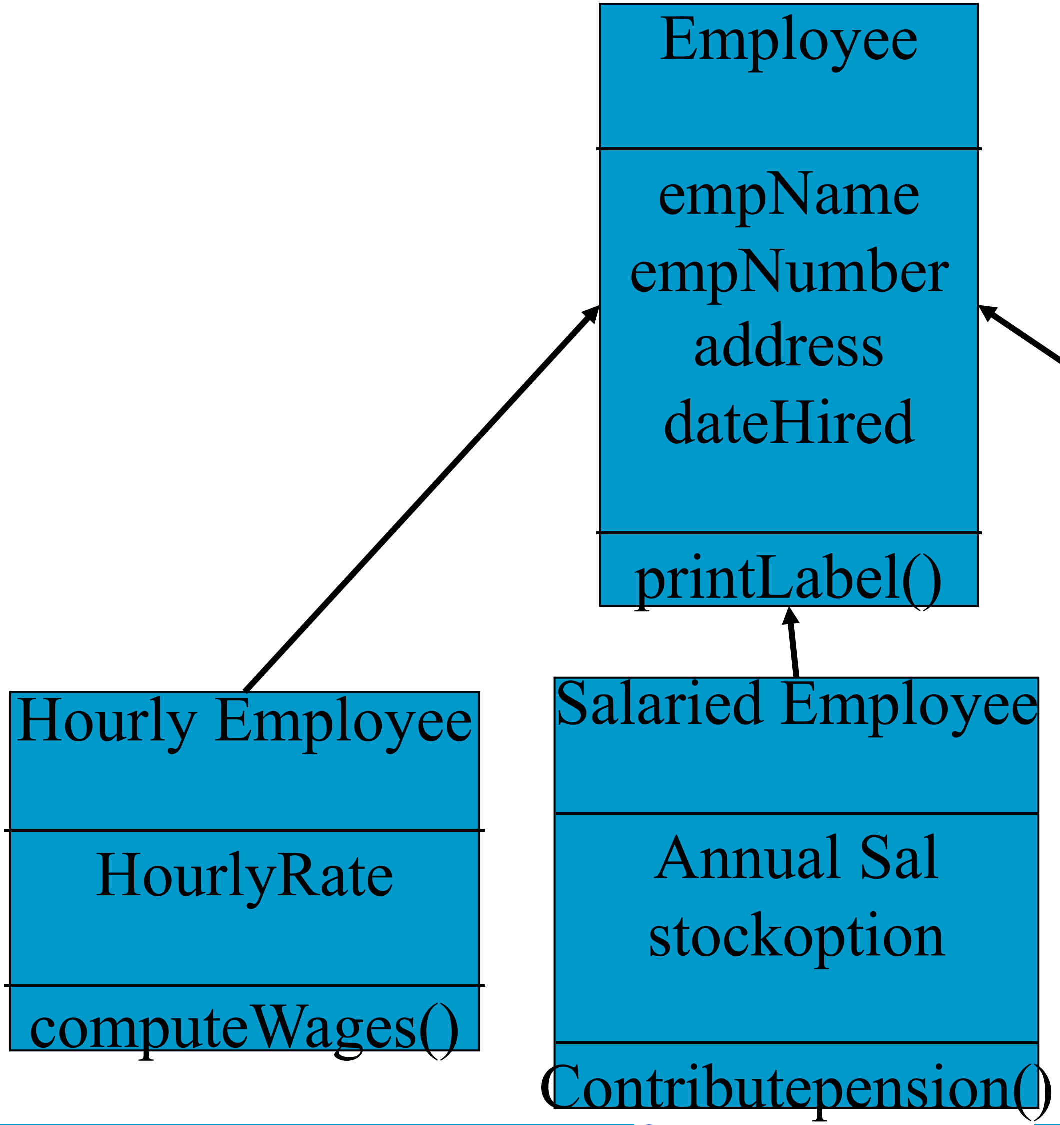
Association Classes



Derived Attributes, Associations, and Roles



Generalization



Other Diagramming methods

- SOM (Semantic Object Model)
- Object Definition Language (ODL)
  - Not really diagramming
- Access relationships display
- Hybrids

Application of SOM to Diveshop

DIVECUST

Name 1.1

Address

Street 1.1

City 1.1

StateProvince 1.1

ZIPPostalCode 1.1

Country 1.1

1.1

Phone 1.1

FirstContact 1.1

DIVEORDS 1.N

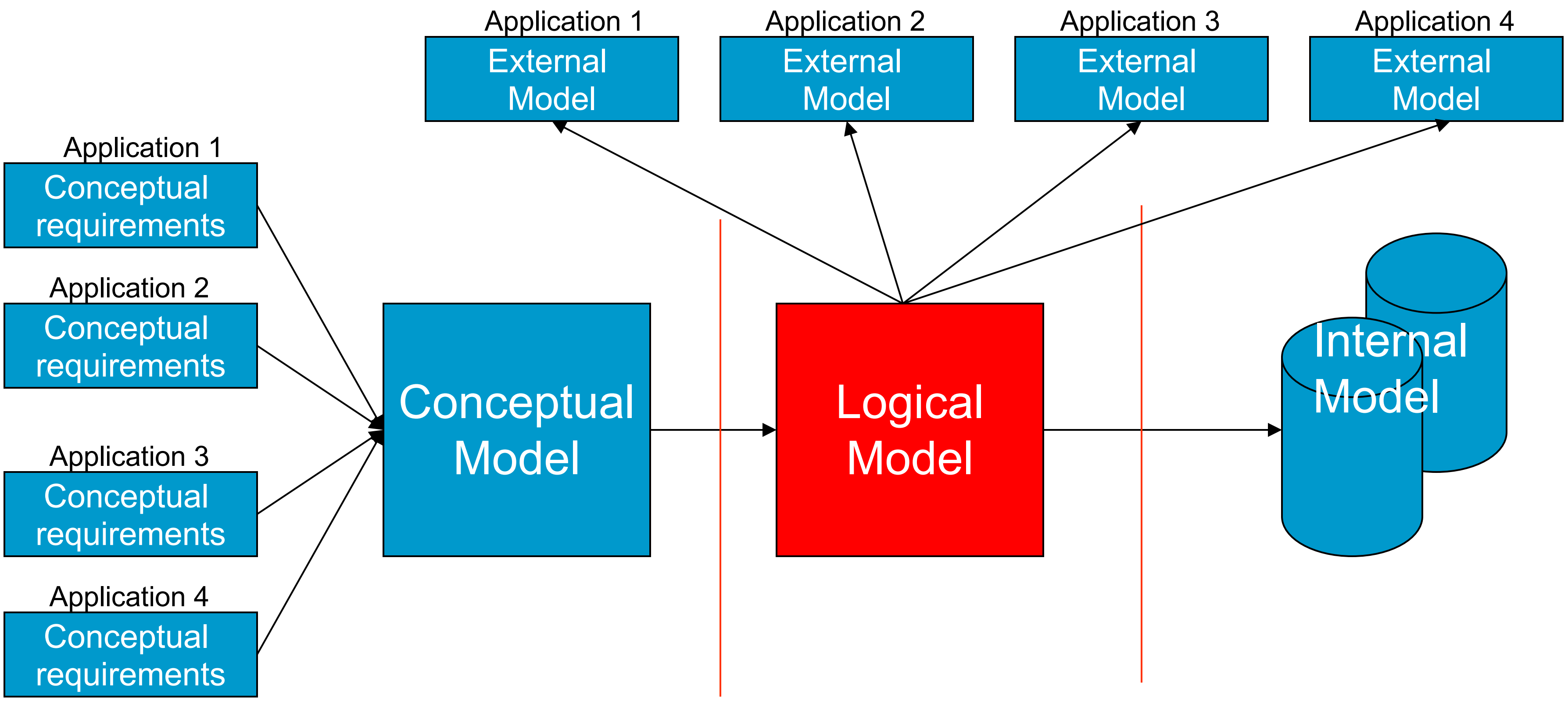
DIVEORDS

DIVEORDS
id OrderNo
SaleDate
DIVECUST
SHIPVIA
DESTINATION
DIVEITEM
PaymentMethod
CCNumber
CCExpDate
NoOfPeople
DepartDate
ReturnDate
VacationCost

Lecture Outline

- Review (and continuation)
  - Database Design, Conceptual Model
- Assignment 2 – Personal Database Conceptual Design
- Object-Oriented Modeling in UML
- The Logical Model

Database Design Process



Logical Model: Mapping to a Relational Model

- Each entity in the ER Diagram becomes a relation.
- A properly **normalized** ER diagram will indicate where intersection relations for *many-to-many* mappings are needed.
- Relationships are indicated by common columns (or domains) in tables that are related.
- We will examine the tables for the Diveshop derived from the ER diagram

Components of Relational Model

- Data structure
  - Tables (relations), rows, columns
- Data manipulation
  - Powerful SQL operations for retrieving and modifying data
- Data integrity
  - Mechanisms for implementing business rules that maintain integrity of manipulated data

Relation

- A relation is a named, two-dimensional table of data.
- Consists of rows (records) and columns (attribute or field)
- Requirements for a table to qualify as a relation:
  - It must have a unique name.
  - Every attribute value must be atomic (not multivalued, not composite).
  - Every row must be unique (can't have two rows with exactly the same values for all their fields).
  - Attributes (columns) in tables must have unique names.
  - The order of the columns must be irrelevant.
  - The order of the rows must be irrelevant.

Note: All **relations** are in **1st Normal form.**

Key Fields

- Keys are special fields that serve two main purposes:
  - **Primary keys** are **unique** identifiers of the relation. Examples include employee numbers, social security numbers, etc. **This guarantees that all rows are unique.**
  - **Foreign keys** are identifiers that enable a **dependent** relation (on the many side of a relationship) to refer to its **parent** relation (on the one side of the relationship).
- Keys can be **simple** (a single field) or **composite** (more than one field).
- Keys are usually used as indexes to speed up the response to user queries.

Integrity Constraints (1 of 2)

- Domain Constraints
  - Allowable values for an attribute (includes data types and restrictions on values)
- Entity Integrity
  - No primary key attribute may be null. All primary key fields **MUST** contain data values.
- Referential Integrity
  - Rules that maintain consistency between the rows of two related tables.

Integrity Constraints (2 of 2)

- **Referential Integrity** – rule states that any foreign key value (on the relation of the many side) MUST match a primary key value in the relation of the one side.(Or the foreign key can be null.)
  - For example: Delete Rules
    - **Restrict** – don’t allow delete of “parent” side if related rows exist in “dependent” side
    - **Cascade** – automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted
    - **Set-to-Null** – set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities

Figure 4-5 Referential Integrity Constraints (Pine Valley Furniture)

Referential integrity constraints are drawn via arrows from dependent to parent table

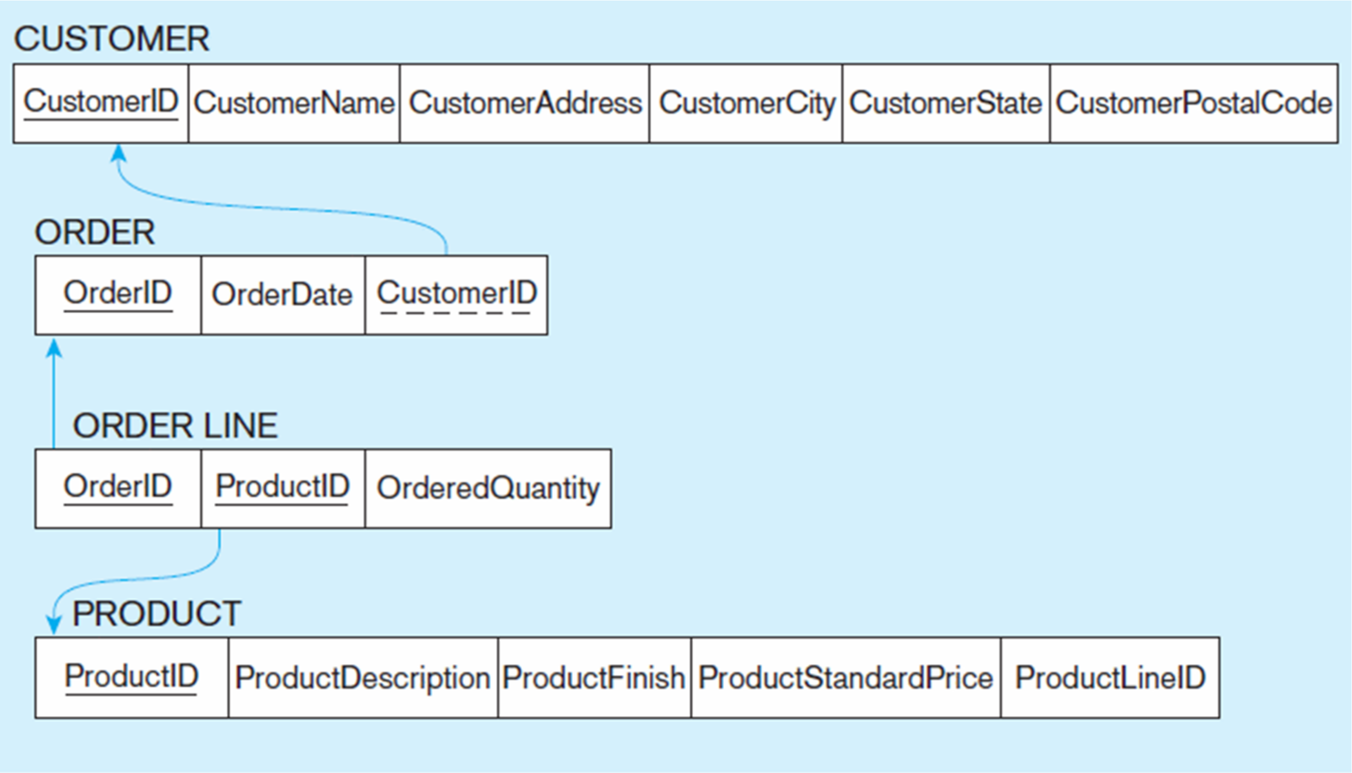


Figure 4-6 SQL Table Definitions

Referential integrity constraints are implemented with foreign key to primary key references.

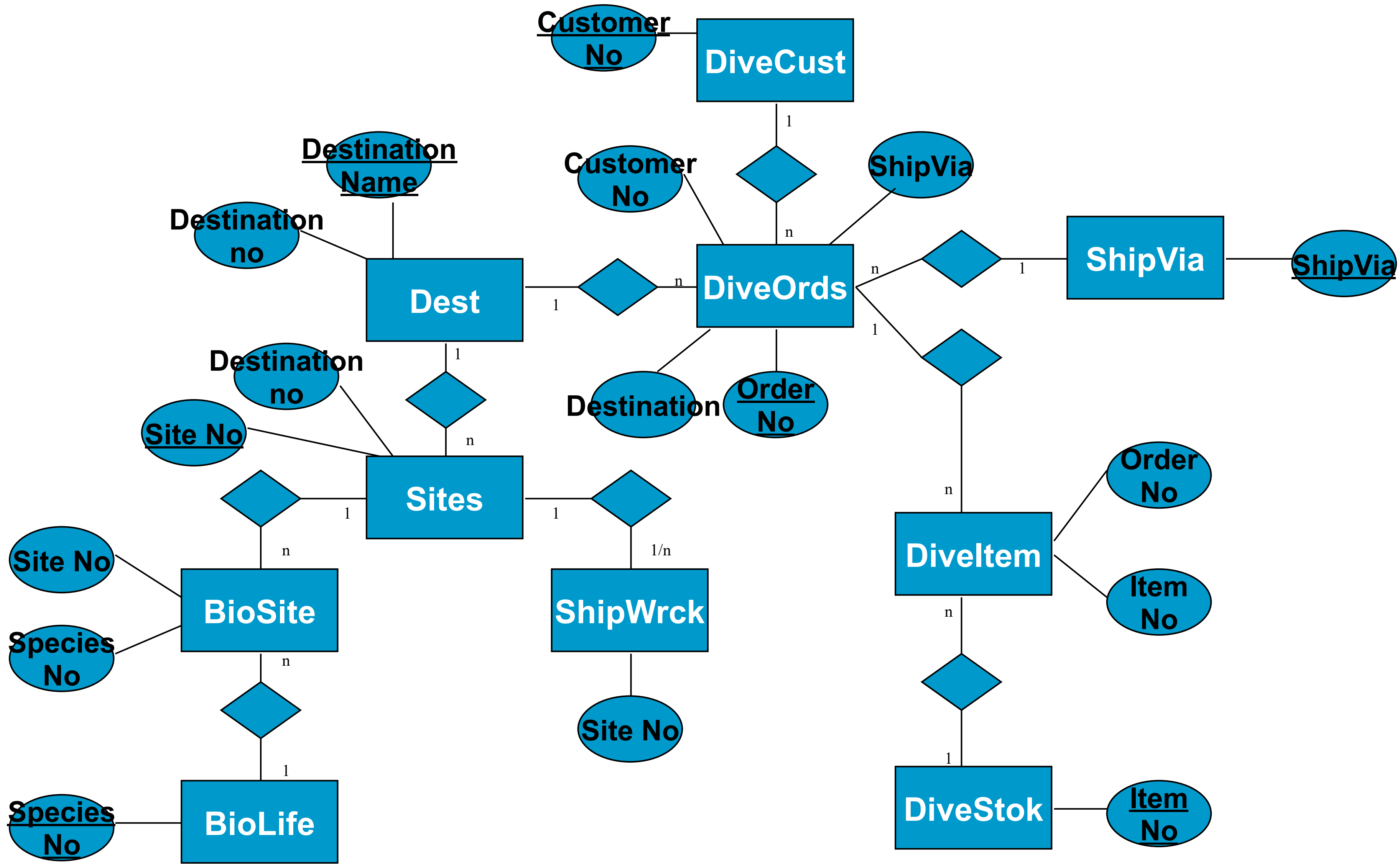
```
CREATE TABLE Customer_T
        (CustomerID                NUMBER(11,0)     NOT NULL,
        CustomerName               VARCHAR2(25)     NOT NULL,
        CustomerAddress            VARCHAR2(30),
        CustomerCity               VARCHAR2(20),
        CustomerState              CHAR(2),
        CustomerPostalCode         VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                   NUMBER(11,0)     NOT NULL,
        OrderDate                  DATE DEFAULT SYSDATE,
        CustomerID                 NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
        (ProductID                 NUMBER(11,0)     NOT NULL,
        ProductDescription         VARCHAR2(50),
        ProductFinish              VARCHAR2(20),
        ProductStandardPrice       DECIMAL(6,2),
        ProductLineID              NUMBER(11,0),
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
        (OrderID                   NUMBER(11,0)     NOT NULL,
        ProductID                  NUMBER(11,0)     NOT NULL,
        OrderedQuantity            NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```

DiveShop ER Diagram



Customer = DIVECUST

Customer	Name	Street	City	State/Prov	Zip/Postal	Country	Phone	First Conta
1480	Louis Jazd	2501 O'Co	New Orlea	LA	60332	U.S.A.	(902) 555-	1/29/95
1481	Barbara W	6344 W. F	San Franc	CA	95031	U.S.A.	(415) 555-	2/2/93
1909	Stephen B	559 N.E. 1	Indianapoli	IN	46241	U.S.A.	(317) 555-	1/5/93
1913	Phillip Dav	123 First S	Berkeley	CA	94704	U.S.A.	(415) 555-	3/9/98
1969	David Burg	320 Montg	Seattle	WA	98105	U.S.A.	(206) 555-	3/12/99
2001	Mary Rioux	1701 Gate	Pueblo	CO	81002	U.S.A.	(719) 555-	3/15/97
2306	Kim Lopez	14134 Nott	Honolulu	HI	96826	U.S.A.	(808) 555-	1/29/99
2589	Hiram Mar	7233 Mill F	San Franc	CA	94123	U.S.A.	(415) 555-	2/18/99
3154	Tanya Kule	505 S. Flo	New York	NY	10032	U.S.A.	(212) 555-	1/30/99
3333	Charles Se	110 East F	Miller	SD	57362	U.S.A.	(613) 555-	3/16/98
3684	Lowell Lutz	915 E. Fes	Dallas	TX	75043	U.S.A.	(214) 555-	2/15/99
4158	Keith Luca	56 South E	Chicago	IL	60542	U.S.A.	(312) 555-	3/17/98
4175	Karen Ng	2134 Elmh	Klamath F	OR	97603	U.S.A.	(503) 555-	3/20/99

Dive Order = DIVEORDS

Order No	Customer N	Sale Date	Ship Via	PaymentMe	CcNumber	CcExpDate	No Of Peopl	Depart Date	Return Date	Destination	VacationCos
307	1480	9/1/99	UPS	Visa	12345 678 9	1/1/01	2	11/8/00	11/15/00	Fiji	10000
310	1481	9/1/99	FedEx	Check			1	4/4/00	4/18/00	Santa Barba	6000
313	1909	9/1/99	Walk In	Visa	456456456	9/11/00	4	6/27/00	7/11/00	Cozumel	8000
314	1913	9/1/99	FedEx	Check			3	2/7/00	2/14/00	Monterey	6000
317	1969	9/1/99	FedEx	AmEx	432432432	12/31/02	4	5/9/00	5/16/00	Fiji	20000
320	2001	9/1/99	Walk In	Cash			1	10/10/00	10/17/00	Santa Barba	3000
321	2306	9/1/99	Emery	Master Carc	1112223334	8/12/00	1	3/15/00	4/12/00	New Jersey	8000
325	2589	9/1/99	Emery	AmEx	332332332	12/10/99	1	3/15/00	4/12/00	New Jersey	8000
326	3333	9/1/99	FedEx	Money Order			2	2/10/00	2/17/00	Monterey	4000
327	3684	9/1/99	DHL	Master Carc	122122321	11/9/99	4	3/10/00	3/23/00	Florida	24000
329	4158	9/1/99	Walk In	Cash			1	5/4/00	5/15/00	Cozumel	1571
330	4175	9/1/99	FedEx	Check			2	7/3/00	7/10/00	Florida	6000
331	5510	9/1/99	FedEx	Money Order			6	6/20/00	6/30/00	Santa Barba	36000
333	5926	9/1/99	DHL	Discover	123123123	12/21/00	2	6/10/00	6/17/00	Fiji	10000

Line item = DIVEITEM

Order No	Item No	Rental/Sal	Qty	Line Note	
307	90010	Rental	4		
307	90020	Rental	1	This is our most popul	
307	90021	Rental	1		
307	90030	Rental	2	These are our best se	
307	90051	Rental	2		
310	90011	Rental	1		
310	90045	Rental	1		
310	90059	Rental	1	A good weight belt for	
310	90074	Rental	1		
310	90078	Rental	1		
313	90127	Sale	1	Holds 10 cubic feet of	
314	90072	Rental	3		
314	90094	Rental	3		
314	90100	Rental	3		

Shipping information = SHIPVIA

Ship Via	Ship Cost
DHL	8
Emery	11
FedEx	12
UPS	10
US Mail	6

Dive Equipment Stock= DIVESTOK

Item No	Descriptior	Equipment	On Hand	Reorder P	Cost	Sale Price	Rental Pric
90010	Shotgun 2	Snorkel	12	2	$18.00	$30.00	$2.00
90011	Shotgun 2	Snorkel	12	2	$18.00	$30.00	$2.00
90012	Shotgun 2	Snorkel	11	2	$18.00	$30.00	$2.00
90020	Tri-Vent M	Mask	14	2	$62.50	$100.00	$5.00
90021	Tri-Vent M	Mask	10	2	$62.50	$100.00	$5.00
90022	Tri-Vent M	Mask	14	2	$62.50	$100.00	$7.00
90023	Quad Visic	Mask	11	2	$48.25	$80.00	$7.00
90024	Quad Visic	Mask	13	2	$48.25	$80.00	$7.00
90025	Quad Visic	Mask	10	2	$48.25	$80.00	$10.00
90030	Sea Wing	Fins	12	2	$60.00	$100.00	$12.00
90031	Sea Wing	Fins	11	2	$60.00	$100.00	$12.00
90032	Sea Wing	Fins	12	2	$60.00	$100.00	$12.00
90033	Jet Fin - Bl	Fins	14	2	$30.00	$60.00	$10.00
90040	D350 Secc	Regulator	11	1	$162.50	$270.00	$20.00
90041	G250 Secc	Regulator	13	1	$144.50	$240.00	$20.00

Dive Locations = DEST

Destinatior	Destinatior	Avg Temp	Avg Temp	Spring Ten	Spring Ten	Summer Te	Summer Te	Fall Temp (	Fall Temp (	Winter Ten	Winter Ter	Accomoda	Night Life	Body of Wa	Travel Cost
1	Cozumel	78	25.556	76	24.444	84	28.889	78	25.556	74	23.333	Cheap	Sleepy	Caribbean	1000
2	Great Barrie	80	26.667	76	24.444	84	28.889	78	25.556	76	24.444	Moderate	Pleasant	Coral Sea	5000
3	Monterey	60	15.556	62	16.667	64	17.778	64	17.778	58	14.444	Expensive	Wild	Pacific	2000
4	Santa Barb	75	23.889	73	22.777	78	25.556	72	22.222	70	21.111	Expensive	Wild	Pacific	3000
5	Florida	77	25	75	23.889	85	29.444	78	25.556	70	21.111	Moderate	Pleasant	Caribbean	3000
6	Fiji	75	23.889	76	24.444	80	26.667	74	23.333	70	21.111	Expensive	Sleepy	South Paci	5000
7	New Jersey	57	13.889	57	13.89	60	15.556	58	14.444	53	11.667	Expensive	Pleasant	Atlantic	2000

Dive Sites = SITE

Site No	Destinatior	Site Name	Site Highligh	Sit	Distan	Distanc	Depth (	Depth (m)	Visibility (f	Visibility (m	Current	Skill Level
1001	1	Palancar Reef	Reef		10	16.09	100	30.48	150	45.72	Strong	Intermediat
1002	1	Santa Rosa Reef	Reef		8	12.87	80	24.384	150	45.72	Strong	Intermediat
1003	1	Chancanab Reef	Reef		4	6.437	60	18.288	100	30.48	Mild	Beginning
1004	1	Punta Sur	Reef		13	20.92	120	36.576	175	53.34	Strong	Advanced
1005	1	Yocab Reef	Reef		6	9.656	50	15.24	100	30.48	Mild	Beginning
2001	2	Heron Island	Reef		50	80.47	90	27.432	150	45.72	Mild	Intermediat
2002	2	Cod Hole	Fish		45	72.42	50	15.24	150	45.72	Mild	Beginning
2003	2	Butterfly Bay	Caves		20	32.19	70	21.336	70	21.336	None	Advanced
2004	2	Wheeler Reef	Marine Life		30	48.28	50	15.24	125	38.1	Mild	Beginning
2005	2	Watanabe	Marine Life		130	209.2	150	45.72	200	60.96	None	Intermediat
3001	3	Point Lobos	Marine Life		3	4.828	60	18.288	75	22.86	None	Beginning
3002	3	Macabee Beach	Marine Life		0.1	0.161	40	12.192	40	12.192	None	Beginning
3003	3	Pinnacles	Pinnacle		1	1.609	60	18.288	50	15.24	Mild	Beginning
3004	3	Monastery Beach	Marine Life		3	4.828	50	15.24	40	12.192	Surge	Beginning

Sea Life = BIOLIFE

Species No	Category	Common Name	Species Name	Length (c	Length (i	Notes	Graphic
90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillu	50	19.685		
90030	Snapper	Red Emperor	Lutjanus sebae	60	23.622		
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus	229	90.157		
90070	Angelfish	Blue Angelfish	Pomacanthus nauarch	30	11.811		
90080	Cod	Lunartail Rockcod	Variola louti	80	31.496		
90090	Scorpionfish	Firefish	Pterois volitans	38	14.961		
90100	Butterflyfish	Ornate Butterflyfish	Chaetodon Ornatissim	19	7.4803		
90110	Shark	Swell Shark	Cephaloscyllium ventr	102	40.157		
90120	Ray	Bat Ray	Myliobatis californica	56	22.047		
90130	Eel	California Moray	Gymnothorax mordax	150	59.055		
90140	Cod	Lingcod	Ophiodon elongatus	150	59.055		

BIOSITE – linking relation

Species No	Site No
90010	2001
90010	2002
90010	2003
90010	2004
90010	2005
90010	6001
90010	6003
90010	6004
90010	6005
90020	2001
90020	2002

Shipwrecks = SHIPWRK

Ship Name	Site No	Category	Type	Interest	Tonnage	Length	Length (m	Beam (	Beam (m)	Cause	Date Sunk	Com	Passer	Survivo	Condition	Gr
Delaware	7007	Commercial	Steam Frei	Treasure	1646	252	76.8096	37	11.2776	Fire			66	66	Broken	
F.S.Loop	4004	Commercial	Steam Sch	Machinery	794	193	58.8264	39	11.8872	Deliberate	1/1/47		0		Scattered	
Gosford	4001	Commercial	Barque Rig	Fixture	2250	280	85.344	42	12.8016	Fire					Intact	
Great Isaac	7002	Commercial	Seagoing	Fixture	1117	185	56.388	37	11.2776	Collision	4/16/47		27	27	Intact	
Lizzie D	7001	Commercial	Tug/Rumr	Treasure	122	84	25.6032	21	6.4008	Unknown	10/19/22		8	0	Intact	
Mohawk	7004	Passenger	Ocean Line	Treasure	8140	402	122.5296	54	16.4592	Collision	1/25/35		163	118	Scattered	
R.P. Resor	7006	Commercial	Oil Tanker	Treasure	7450	435	132.588	66.8	20.36064	Military	2/28/42		50	2	Broken	
Star of Scotl	4002	Passenger	British Q-B	Treasure	1250	263	80.1624	35	10.668	Weather	1/22/42		5	4	Broken	
Tolten	7008	Commercial	Freighter	Fixture	1858	280	85.344	43	13.1064	Military	3/13/42		28	1	Intact	
USS Moody	4006	Military	WWI Destr	Treasure	1308	314	95.7072	31	9.4488	Deliberate	1/1/33		0		Intact	
Valiant	4003	Passenger	Luxury Mo	Treasure	444	[illegible]	[illegible]	36	[illegible]	Fire	12/17/30		25	25	Intact	

Mapping to Other Models

- Hierarchical
  - Need to make decisions about access paths
- Network
  - Need to pre-specify all of the links and sets
- Object-Oriented
  - What are the objects, datatypes, their methods and the access points for them
- Object-Relational
  - Same as relational, but what new datatypes might be needed or useful (more on OR later)

Next Lecture

- Normalization and the relational model