Logical Database Design
Mapping ER Models to Relational Models

Also called “transformation” of conceptual models to logical models.

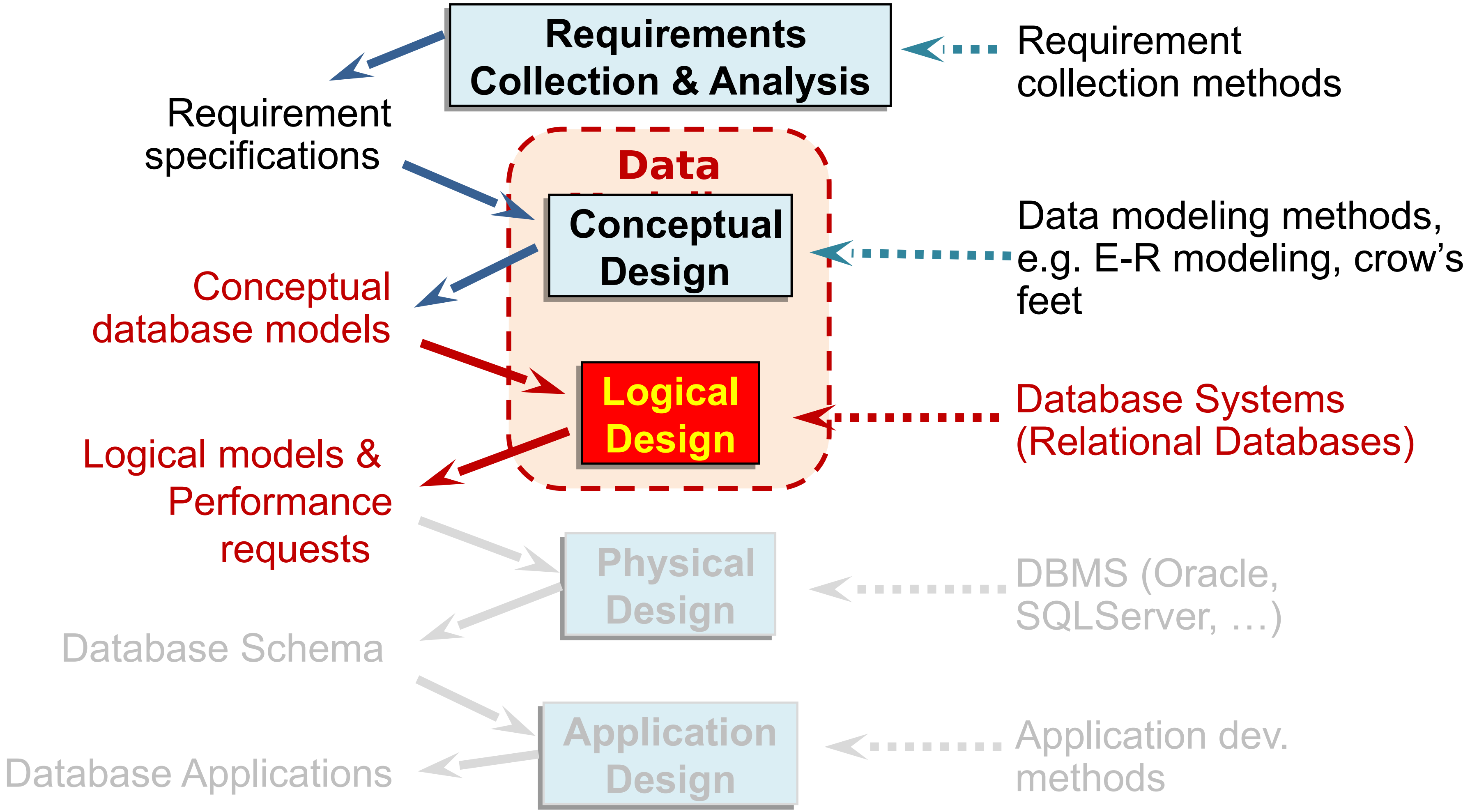
Entity-relationship model -> relational model

Acknowledgement: Dr. S. Hong provided some of the material for this set of lecture notes.

Objectives

- Understand how to transform an ER model into a relational model
- Constructs of an entity-relationship model
  - Entity types
    - attributes (key, non-key)
  - Relationship types
    - 1:N
    - M:N
    - 1:1
- Construct of relational model
  - Relation

Data modeling is part of database design methodology



Requirements Collection & Analysis

Requirement collection methods

Requirement specifications

Data

Conceptual Design

Data modeling methods, e.g. E-R modeling, crow's feet

Conceptual database models

Logical Design

Database Systems (Relational Databases)

Logical models & Performance requests

Physical Design

DBMS (Oracle, SQLServer, …)

Database Schema

Application Design

Application dev. methods

Database Applications

The Relational Model

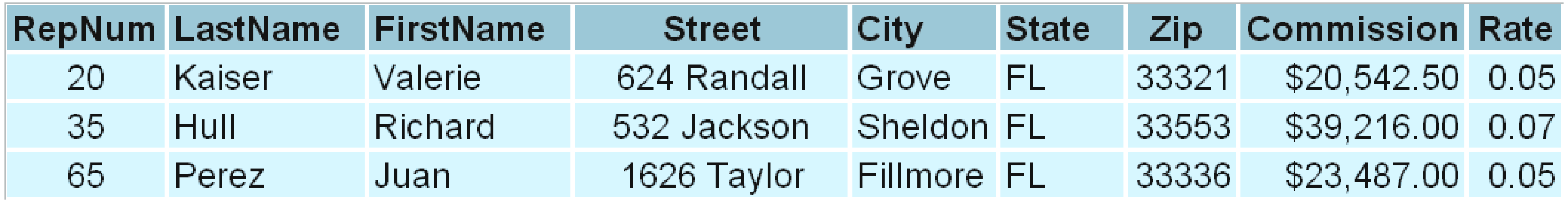
- Collection of tables, called relations, stored in a database
  - Codd 1970
  - Simplicity

- Key concepts:
  - A **table** (relation) refers to a list of data arranged in columns (fields) and rows (records)
    - Structured to minimize redundancy
  - A database is either a single table or a collection of related tables
  - A column (**field**) defines the data that a table can hold
  - A row (**record**) represents a single instance of whatever the table keeps track of
  - A **key** is the field(s) used to uniquely identify each record in a table and to relate tables in a database

- SQL (Structured Query Language) – for creating and querying a database

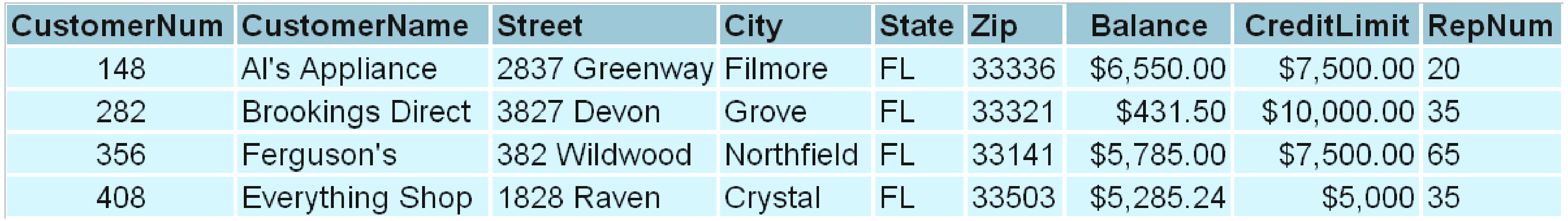
Simple Database (Revisited)
Customers of sales representatives

Sales Representative (i.e. Employee)



RepNum	LastName	FirstName	Street	City	State	Zip	Commission	Rate
20	Kaiser	Valerie	624 Randall	Grove	FL	33321	$20,542.50	0.05
35	Hull	Richard	532 Jackson	Sheldon	FL	33553	$39,216.00	0.07
65	Perez	Juan	1626 Taylor	Fillmore	FL	33336	$23,487.00	0.05

Customer



CustomerNum	CustomerName	Street	City	State	Zip	Balance	CreditLimit	RepNum
148	Al's Appliance	2837 Greenway	Filmore	FL	33336	$6,550.00	$7,500.00	20
282	Brookings Direct	3827 Devon	Grove	FL	33321	$431.50	$10,000.00	35
356	Ferguson's	382 Wildwood	Northfield	FL	33141	$5,785.00	$7,500.00	65
408	Everything Shop	1828 Raven	Crystal	FL	33503	$5,285.24	$5,000	35

What queries might you ask of this database? Note: minimize redundancy.

Logical design:
Transform conceptual model into a logical model to be implemented in a relational database

- Relational model has one construct (relation/table)

- Conceptual ER models are very descriptive (semantics rich)

- ER modeling concepts must be converted to relational concepts
  - Entities -> relations
  - Relationships -> foreign key in existing relation; or new relation
  - Depends on min/max cardinalities

Conceptual ER Model	Relational Model
Entity types	Relation (table)
Single valued attributes	Single-valued attributes
Relationships (1-to-1, 1-to-N)	Foreign keys
Many-to-many (N:M) relationships (w/ relationship attributes possible)	Relation (table)

Translation to Logical Model

Transformation / Conversion / Mapping Rules – Entity

[Note: Do not confuse *mapping rules* with *mapping ratios*]

Each entity becomes a relation (table) with same key

Transformation / Conversion / Mapping Rules -- Relationship

- 1:N – Foreign key
  - (0,1) and (0/1,N) – foreign key (can have null value)
  - (1,1) and (0/1,N) – foreign key (cannot have null value)
- N:M – New relation
  - (0/1,N) and (0/1,M)
  - Key is concatenation (joining together) of keys of the two entities
  - Relationship attributes become non-keys

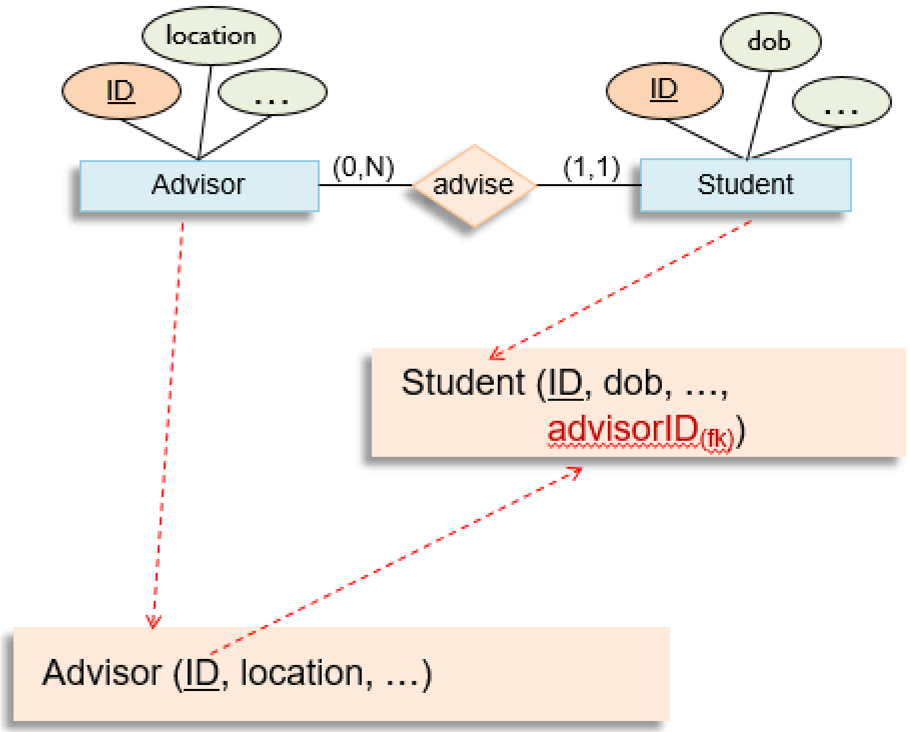
1:1 – foreign key

- (1,1) and (1,1) – foreign key in either relation (decide based upon usage)
- (0,1) and (1,1) [or (1,1) and (0,1)] – use foreign key such that there are no null values allowed.

Binary (1-to-N) relationship represented as foreign keys

(1) Map entity types and attributes

(2) 1-to-N relationship: insert the PK of the entity with the many max cardinality into another entity's table as a FK



Student (ID, dob, …, advisorID(fk))

Advisor (ID, location, …)

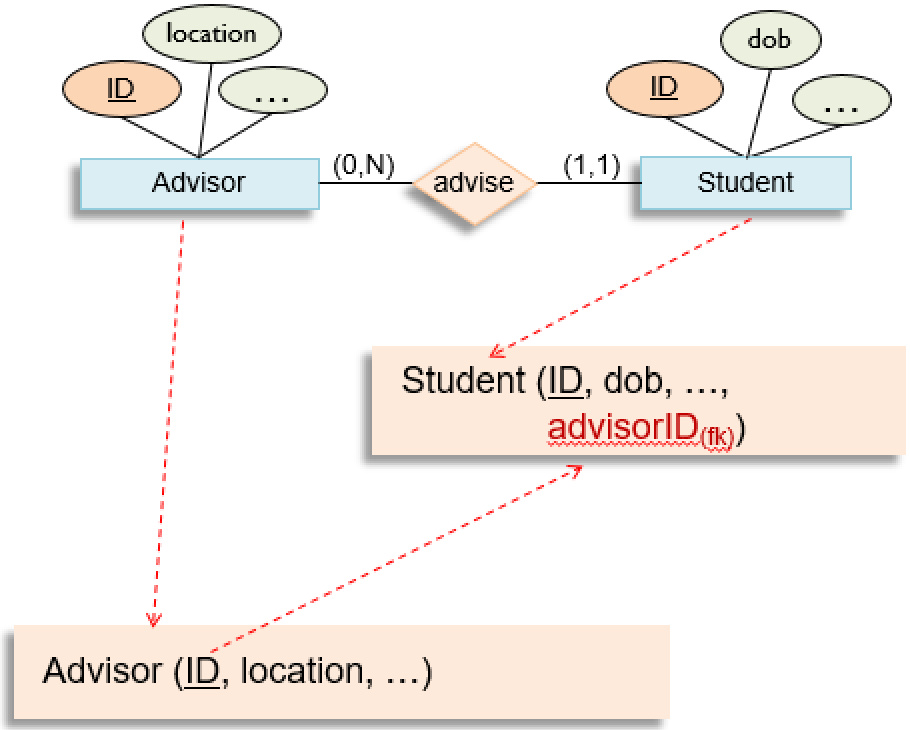
Note: 1:N relationship does not have relationship attributes

Binary (1-to-N) relationship represented by foreign keys

Relations:

Advisor: [AdvisorID, location, …]

Student: [StudentID, dob, …. advisorID]

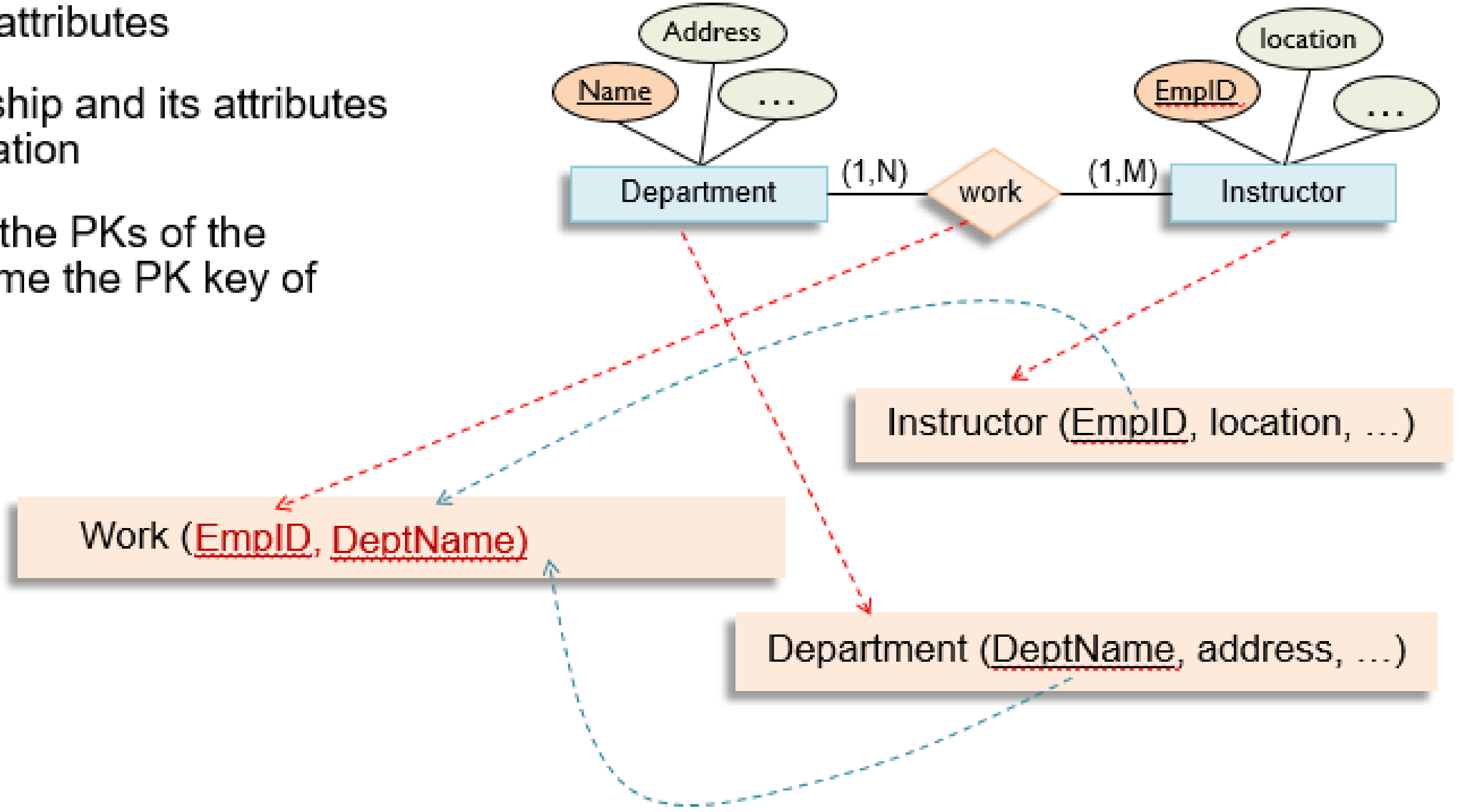


Binary N:M relationship represented as new relation

(1) Map entity types and attributes

(2) Map the N:M relationship and its attributes (if any) into a new relation

(3) The concatenation of the PKs of the involved entities become the PK key of the new relation.



Instructor (EmpID, location, ...)

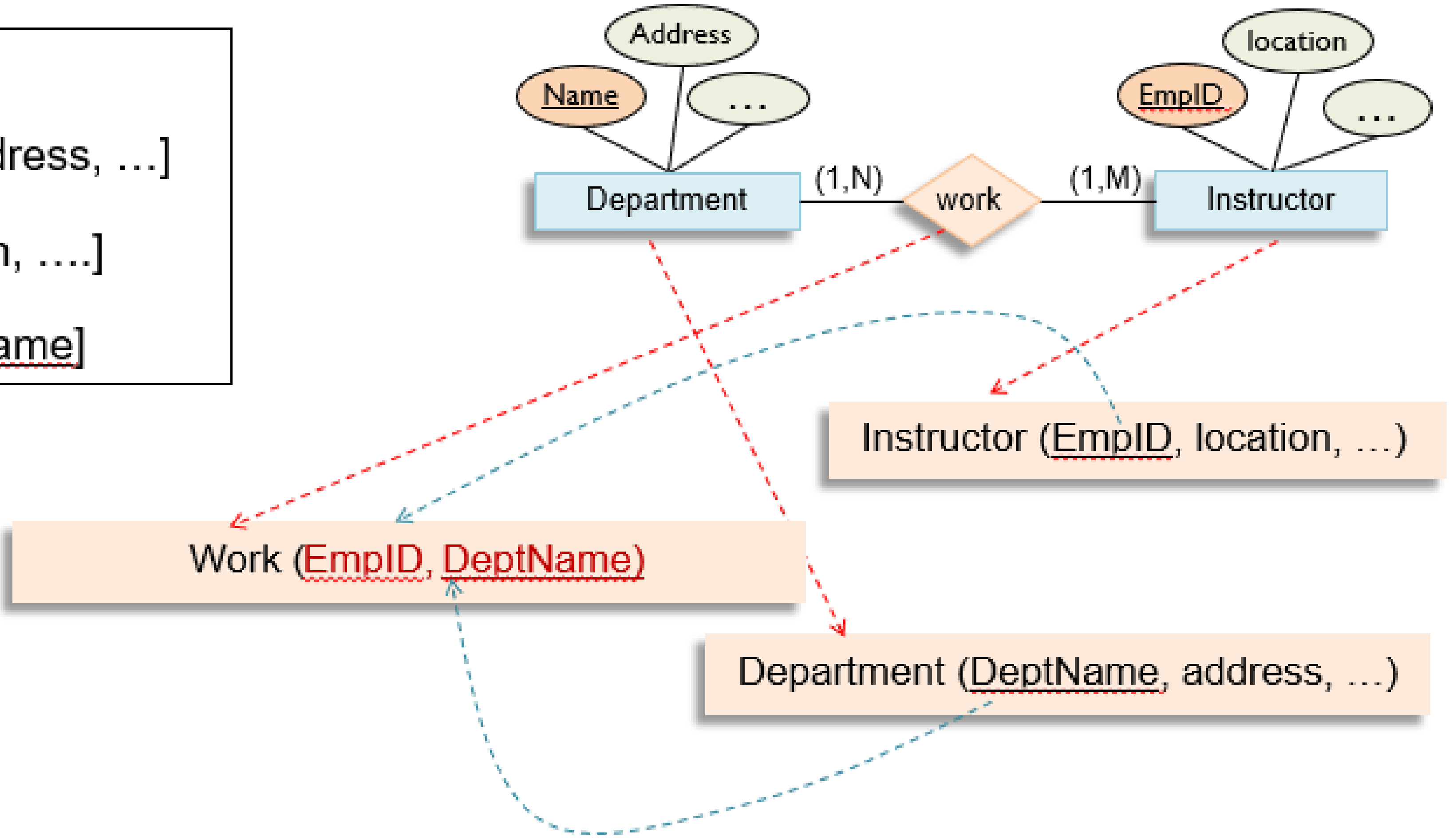
Work (EmpID, DeptName)

Department (DeptName, address, ...)

Binary N:M relationship represented as new relation

Relations:

Department:	[DeptName, address, …]
Instructor:	[EmpID, location, ….]
Work:	[EmpID, DeptName]



Instructor (EmpID, location, …)

Work (EmpID, DeptName)

Department (DeptName, address, …)

Binary N:M relationship with relationship attributes

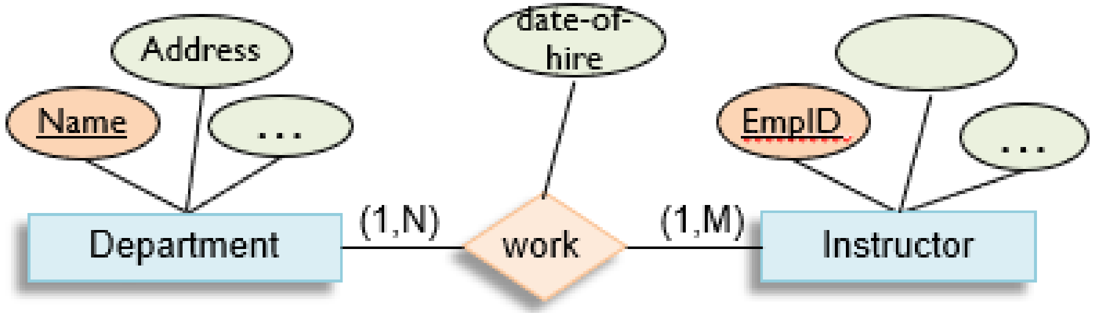
(1) Map entity types and attributes

(2) Map the N:M relationship and its attributes into a new relationship table

(3) The concatenation of the PKs of the involved entities become the PK key of the new relation

(4) The relationship attribute(s) become the non-key(s).

.



Work (<u>EmpID</u>, <u>DeptName</u>, date-of-hire)

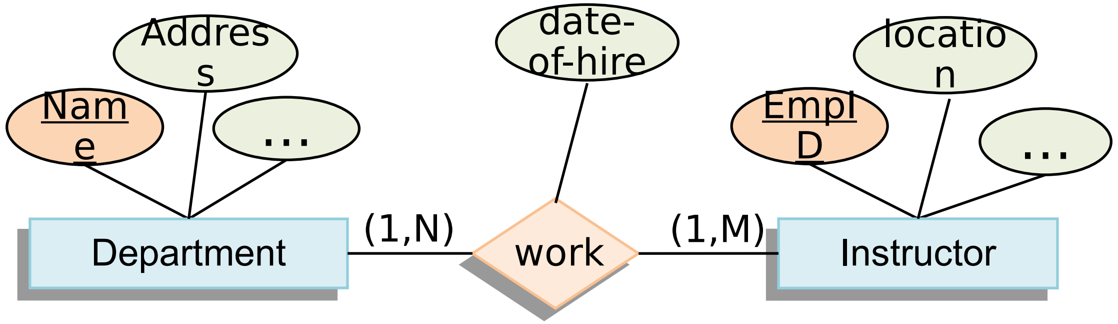
Binary N:M relationship with relationship attributes

Relations:

Department: [Name, address, …]

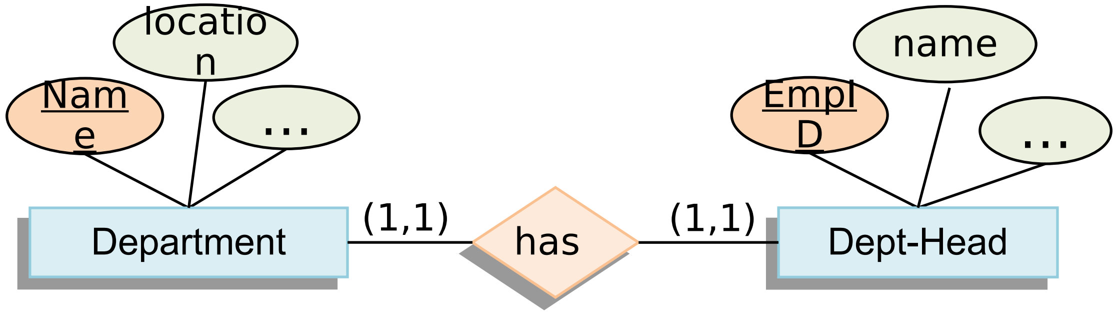
Instructor: [EmpID, location, ….]

Work: [EmpID, DeptName, date-of-hire]



1:1 Relationship

Every dept must have one (and only one) department head.



Options:

-- foreign key in either relation

Department: [DeptName, location, … EmpID]

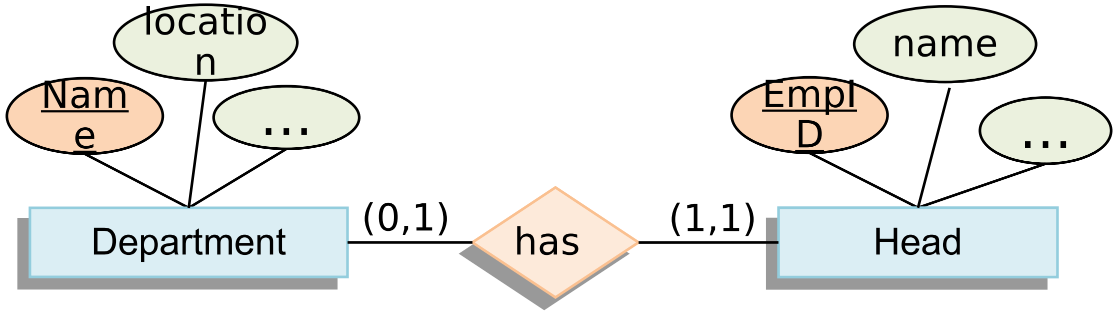
Or:

Dept-Head: [EmpID, name, …., DeptName]

How would you decide?

Optional 1:1 Relationship

Business rule: department can have at most one department head

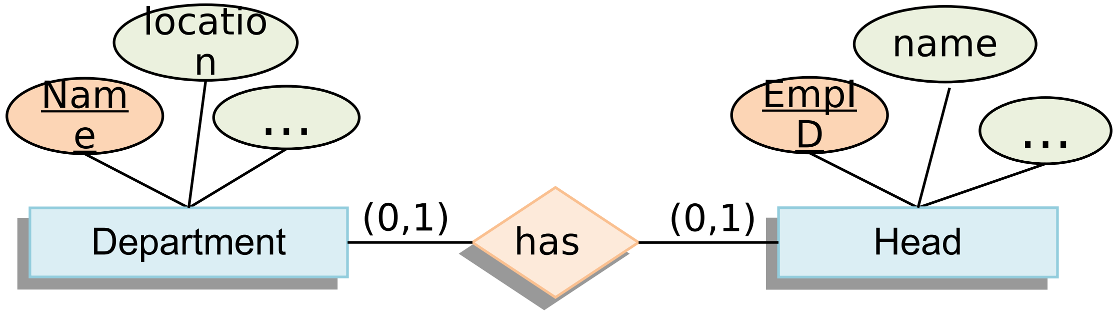


Head: [<u>EmpID</u>, name, …., DeptName]

Where “DeptName” is not allowed to have a null value.

Optional 1:1 Relationship in both directions

What if both optional?



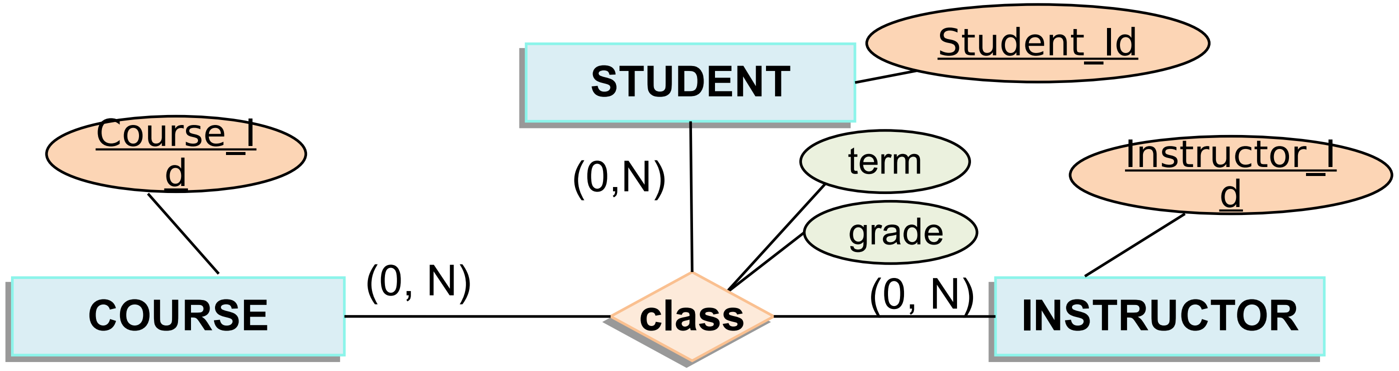
Dept-Head: [EmpID, DeptName]

-or- Head: [EmpID, name, …., DeptName]
(null FK allowed)
-or- Dept: [DeptName, … EmpID]
(null FK allowed)

Ternary relationships mapped to separate relation

(1) Map entity types and attributes

(2) Map the ternary relationship and its attributes (if any) into a new relation



Class: [<u>Student_Id</u>, <u>Course_Id</u>, <u>Instructor_Id</u>, term, grade]

Assumptions inherent in this example?

Summary

- Mapping rules
  - Conceptual data model ☾ relational data model
- Most common
  - 1:1, 1:N, M:N
  - Others – unary, ternary
- Special cases, exceptions
  - Understand application being modelled
- More examples coming