Revision

COMP 1531

Aarthi Natarajan

Week 10

Week 12 Revision
Classes and ERDs

Case Study

- UNSW has several departments. Each department is managed by a chair, and at least one professor.
- Professors must be assigned to one, but possibly more departments. At least one professor teaches each course, but a professor may be on sabbatical and not teach any course.
- Each course may be taught more than once by different professors.
- We know of the department name, the professor name, the professor employee id, the course names, the course schedule, the term/year that the course is taught, the departments the professor is assigned to, the department that offers the course
- Draw a class diagram and an ER model for the above case-study

Steps to drawing the class diagram (contd…)

1. Identify classes
   - Abstract or tangible “things” in our problem domain (nouns and noun phrases) determined from requirement analysis
   - e.g., departments, chair, professor
2. Find associations
   - Verbs that join the nouns e.g., professor (noun) teaches (verb) students (noun)
3. Draw CRC diagram

Defining the CRC cards

Professor	
Assigned to a Department	Department
Teaches Course	Course
Knows Name	
Knows Employee ID	

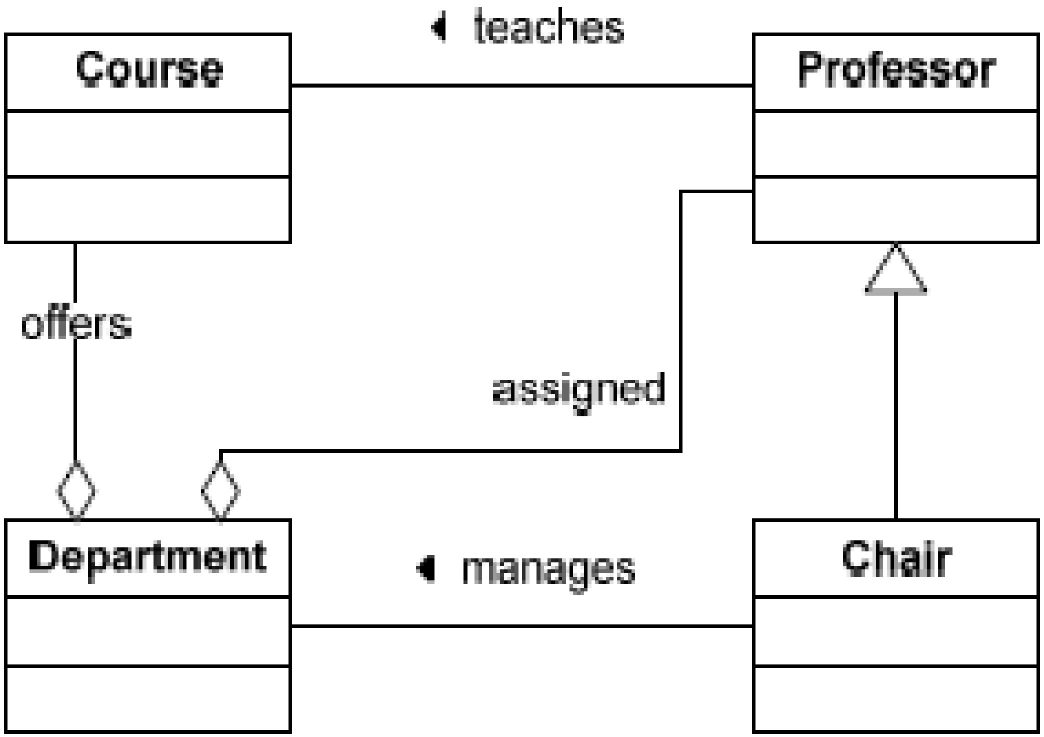
Department	
Managed by a Chair	Chair
Is Assigned Professors	Professor
Offers Courses	Course
Knows Department Name	

Course	
Offered by a Department	Department
Taught by Professor	Professor
Knows schedule	
Knows term/year offered	

Chair	
Manages a Department	Department
Is a Professor	Professor
Knows Department Name	

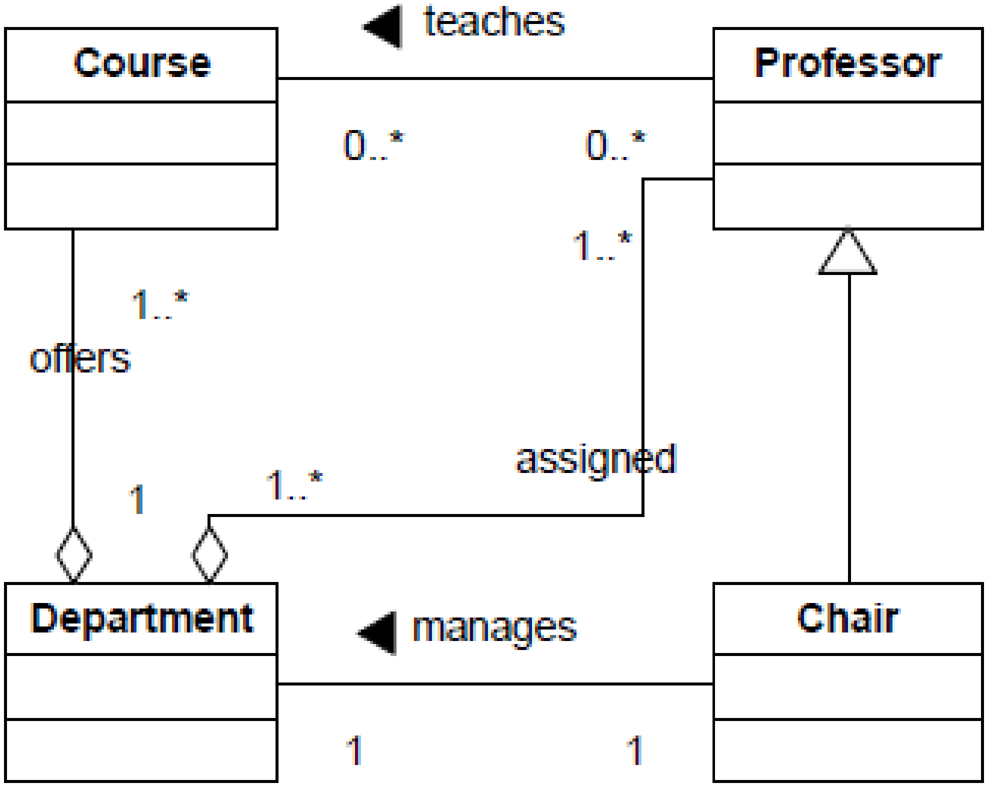
Steps to drawing the class diagram (contd…)

4. Draw the conceptual class diagram



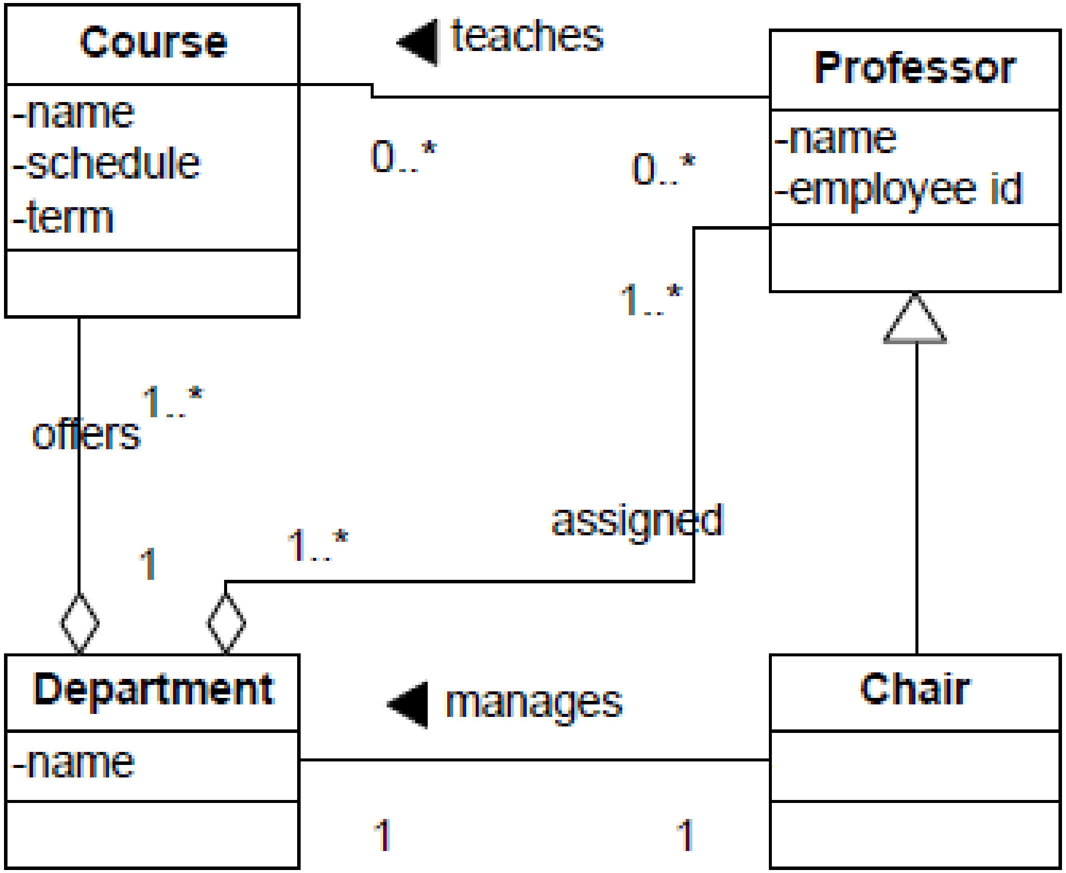
Steps to drawing the class diagram (contd…)

5. Fill in the multiplicity



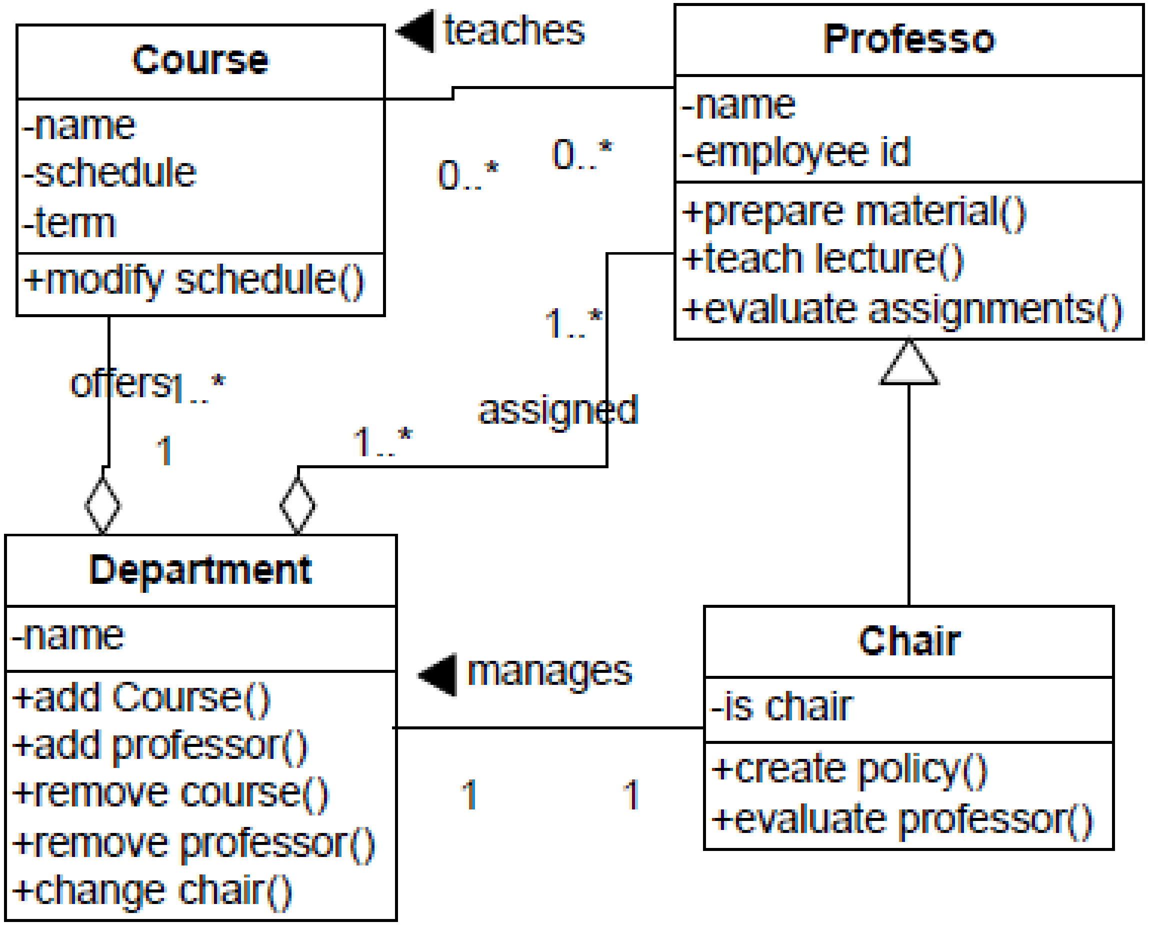
Steps to drawing the class diagram (contd…)

5. Identify attributes



Steps to drawing the class diagram (contd…)

5. Identify behaviours
6. Review class diagram and fine tune it



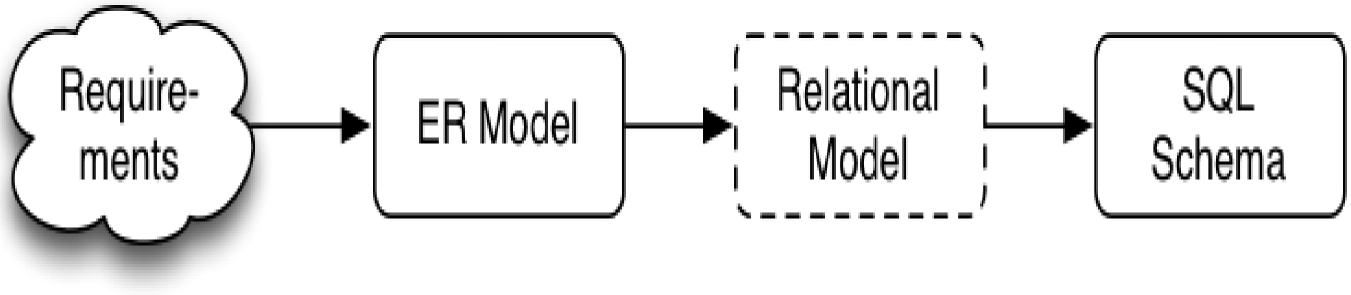
Designing a database

Two data models

- Logical: abstract model e.g., ER Model, OO Model
- Physical: record-based models e.g., relational model

A strategy for designing a database

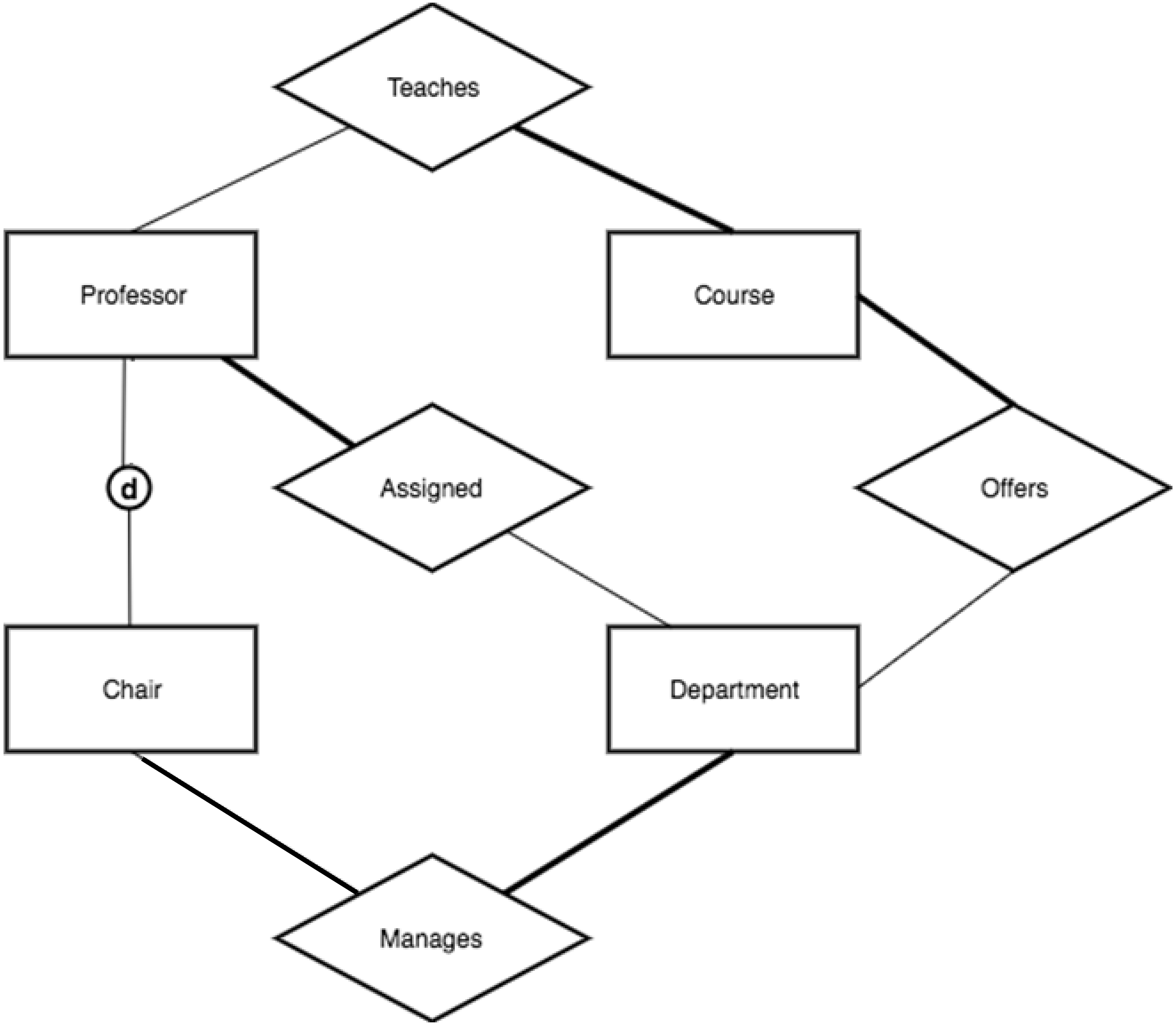
- Design using abstract model (conceptual-level modelling)
- Map to physical model (implementation-level modelling)



Steps to drawing the entity relationship diagram

1. Identify entities
   - Typically the nouns and noun phrases determined from requirement analysis
   - Include only entities relevant to problem domain
   - e.g., departments, chair, professor
2. Find relationships
   - Verbs that join the nouns e.g., professor (noun) teaches (verb) students (noun)
3. Draw conceptual ER diagram

Steps to drawing the class diagram (contd…)

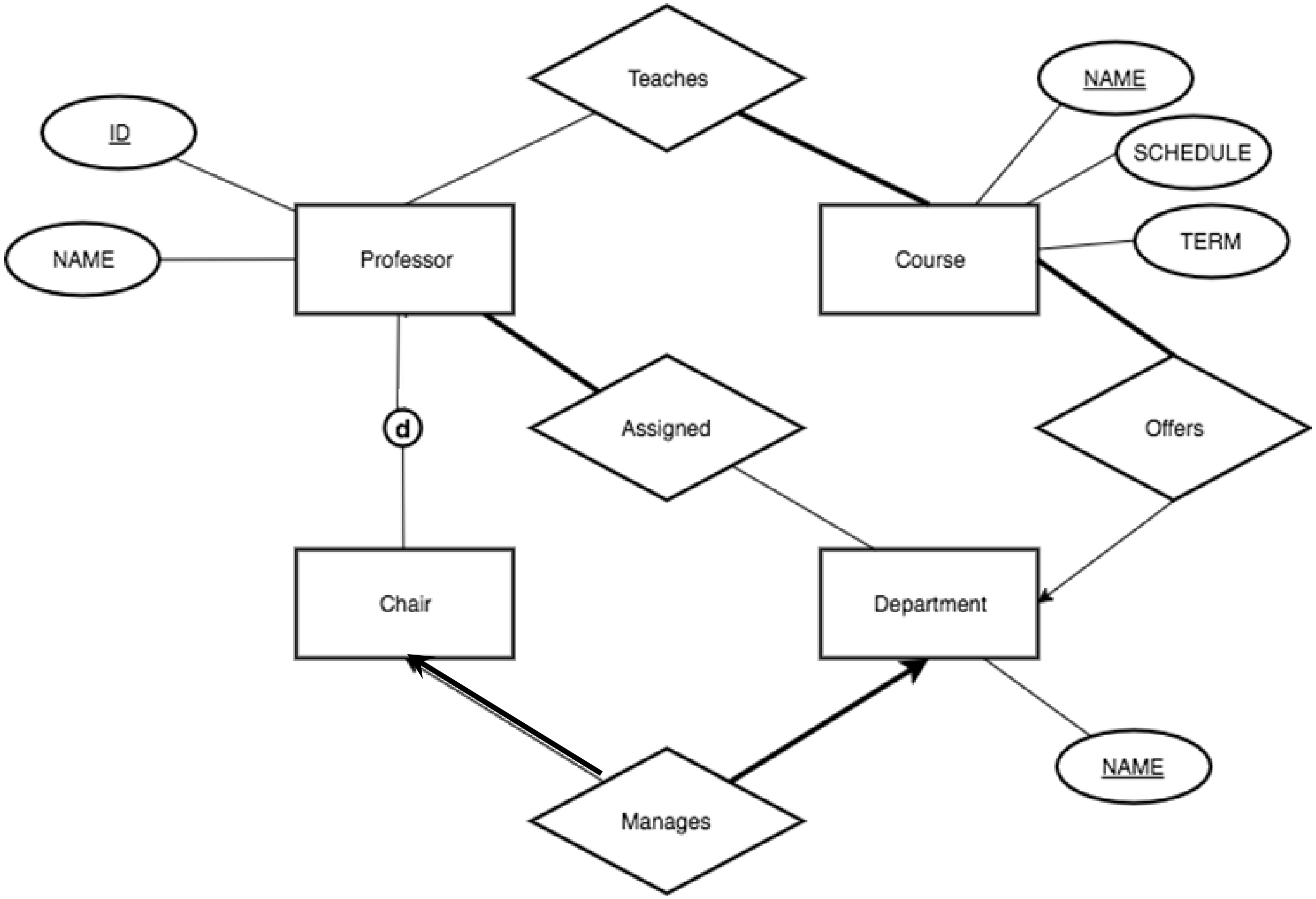


Steps to drawing the entity relationship diagram

5. Add cardinality

6. Identify the entity attributes

7. Identify the primary key



Relational Data Model

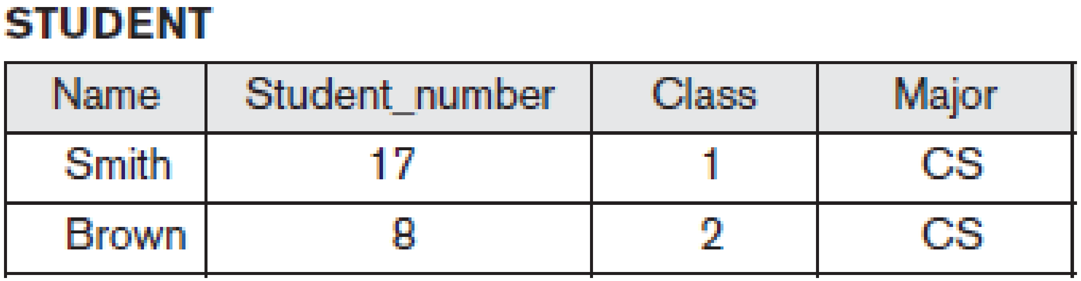
The relational data model describes the world as a collection of inter-connected relations (or tables)

Goal of relational model:

- a simple, general data modelling formalism
- maps easily to file structures (i.e. implementable)

Relational model has two styles of terminology:

mathematical	relation	tuple	attribute
data-oriented	table	record (row)	field (column)

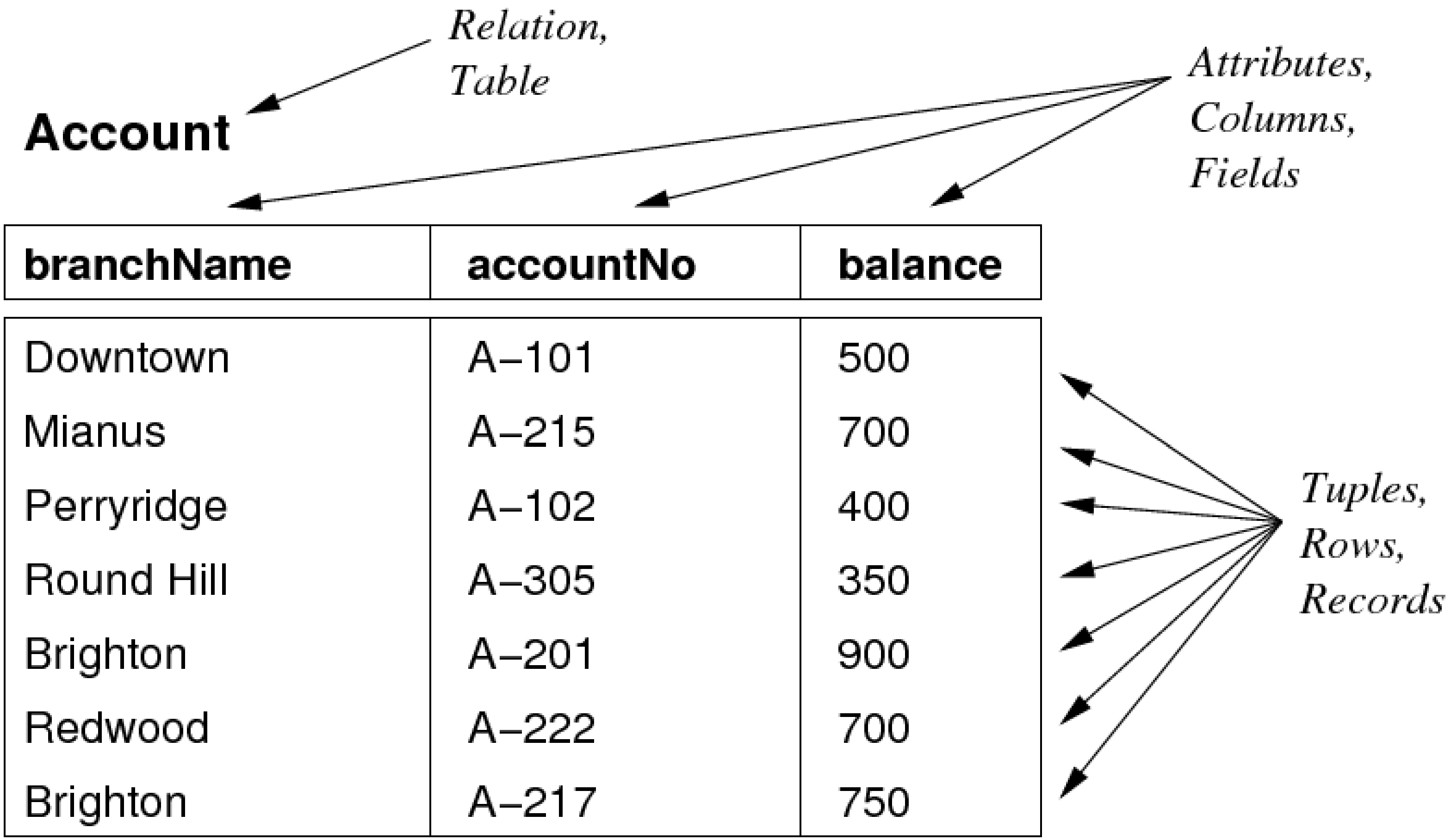


STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

Relational Data Model

Example of a relation (table): Bank Account



Relation, Table

Account

Attributes, Columns, Fields

branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700
Brighton	A–217	750

Tuples, Rows, Records

Constraints

Relations are used to represent real-world *entities* and *relationships* between these *entities*

To represent real-world problems, need to describe

- what values are/are not allowed
- what combinations of values are/are not allowed

Constraints are logical statements that do this:

- Domain constraint
- Key constraint
- Entity constraint
- Referential integrity

Referential Integrity Example

Table DEPARTMENT (Parent Table)

Primary Key in Parent Table

DEPT_NO	DEPT_NAME	CITY
10	MARKETING	SYDNEY
11	SALES	SYDNEY
12	TECH	MELBOURNE

Foreign Key in child table must match a primary key in the parent table

Table EMPLOYEE (Child Table)

EMP_NO	EMP_NAME	ROLE	DEPT
5012	John	CEO	10
5016	Alison	SALESPERSON	11
5018	Cathy	MANAGER	12

Insert Fails due to violates the referential integrity constraint

5015 Mitchell SALESPERSON 30

Relational Model vs Entity Model

Correspondences between relational (R) and ER data models:

- ER attribute → relational attribute
- ER entity → relational tuple
- ER entity-set → relational table (relation)
- ER relationship → relational table (relation)
- ER key → relational primary key

Differences between relational and ER models:

- Relational uses *relations* to model *entities* and *relationships*
- Relational has **no** *composite* or *multi-valued* attributes (only atomic)
- Relational has **no** *object-oriented notions* (e.g. subclasses, inheritance)

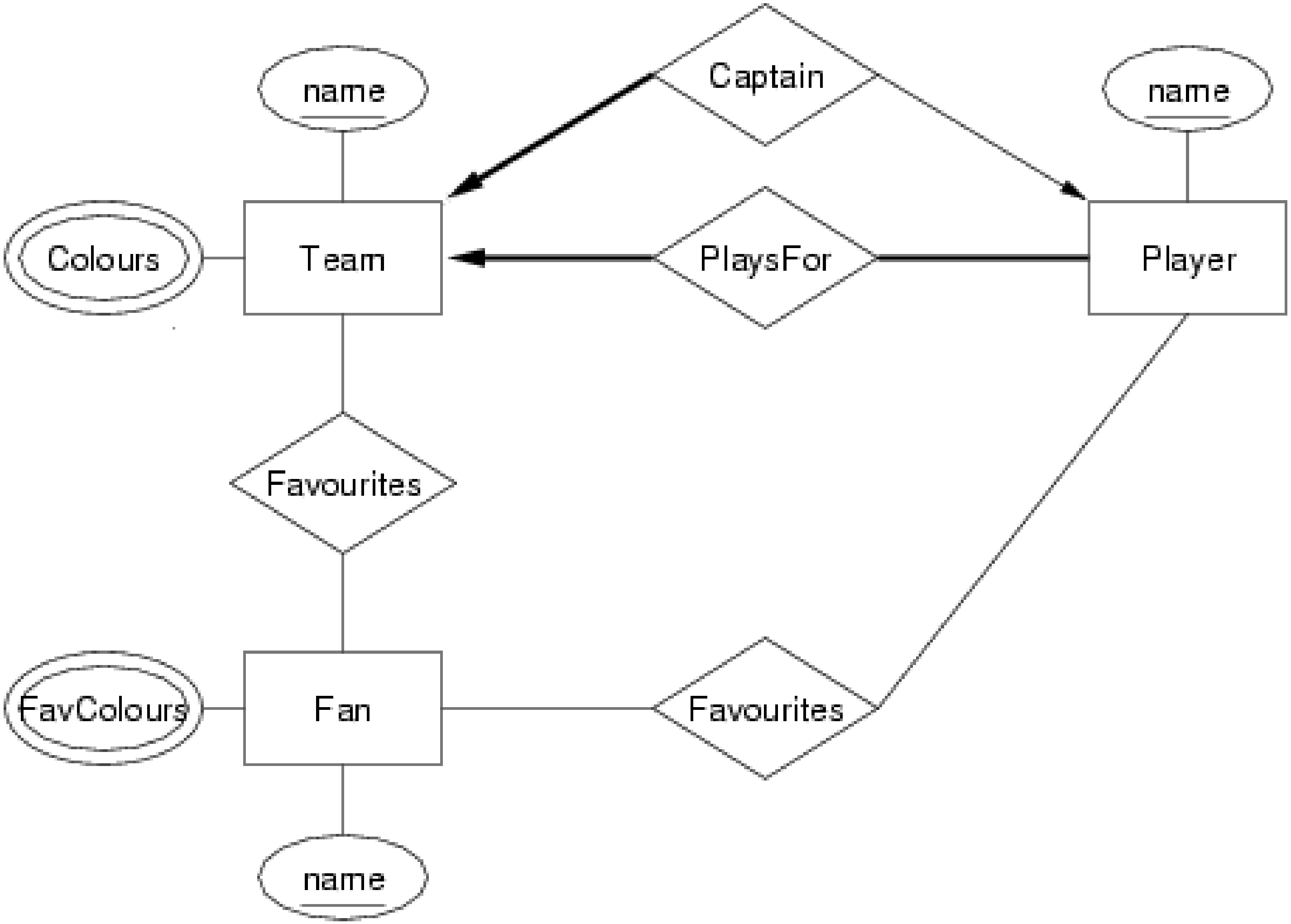
Case Study:

(1) Develop an ER design for the following scenario:

A database records information about teams, players, and their fans, including:

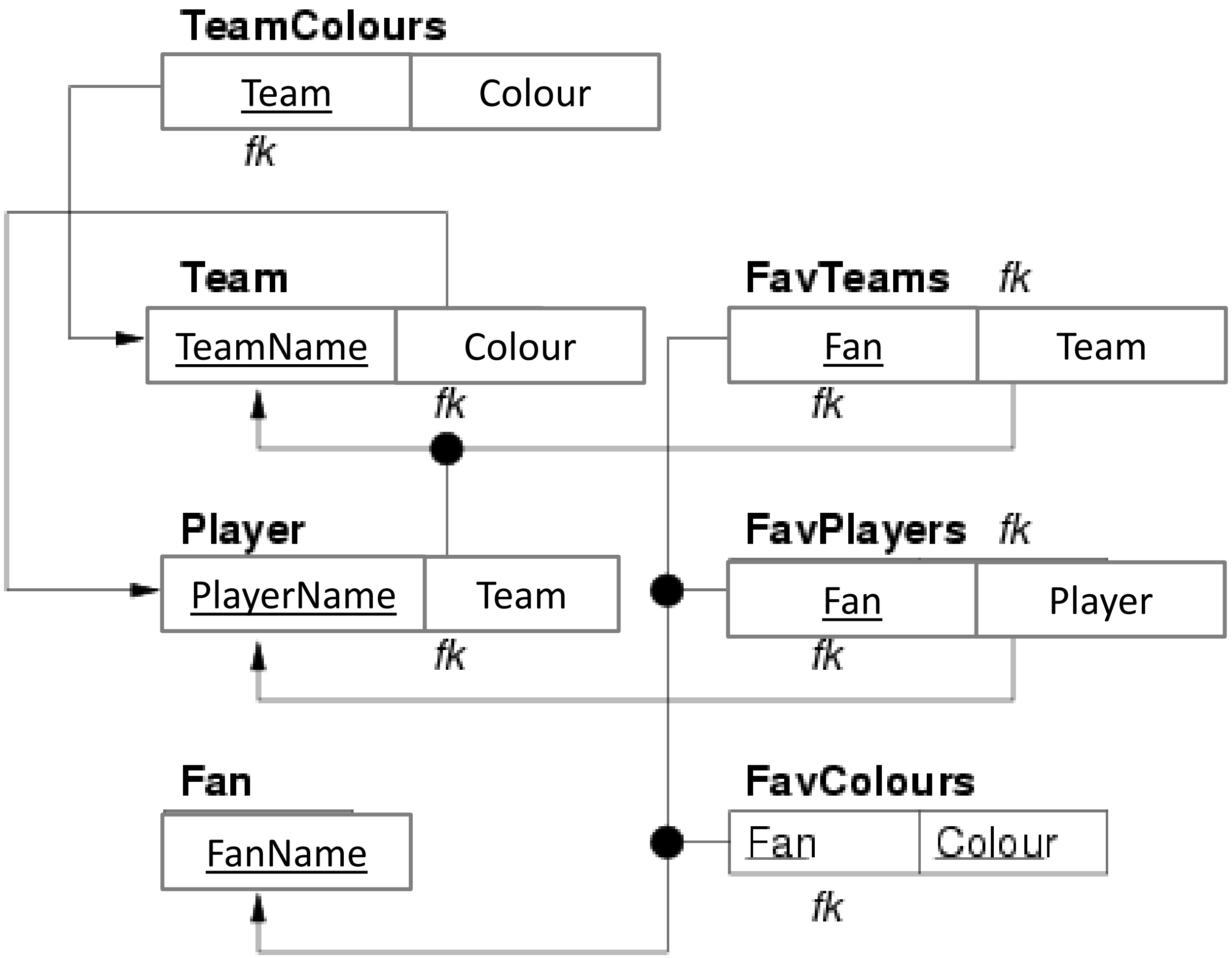
- For each team, its name, its players, its captain (one of its players) and the colours of its uniform.
- For each player, their name and team.
- For each fan, their name, favourite teams, favourite players, and favourite colours.

Solution: ER Design



Solution:

(2) Now, convert the following ER design into a relational data model based on the box schema notation



TeamColours

Team	Colour

Team

TeamName	Colour

FavTeams

Fan	Team

Player

PlayerName	Team

FavPlayers

Fan	Player

Fan

FanName

FavColours

Fan	Colour

Solution:

(3) Which elements of the ER design do not appear in the relational model?

- At a syntactic level, the multi-valued attributes from the E/R design do not appear directly in the relational model, but are replaced by tuples in the `TeamColours` and `FavColours` tables.

- At a semantic level, it doesn't capture the **total participation** of the `Team` entity in the `PlaysFor` relationship. While all players have to play for a team, the diagram does not enforce that each team must have at least one player who plays for it (except indirectly via the fact that it has to have a captain)

- It also doesn't require that a team has any colours or that a fan has any favourite colours.

  (Of course, the E/R diagram doesn't imply this either (non-key attributes are not required to have a value), but if it did state this, the relational model as given could not capture it.)