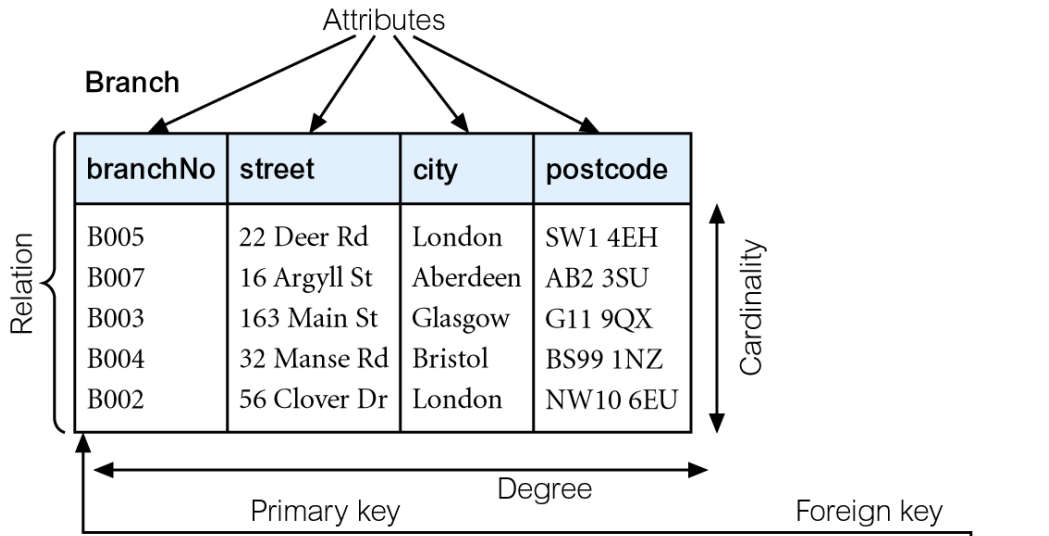
Terminology Questions
Relations, Schemas,

Terminology

- Base relation: A relation we *explicitly define* in our database
  - We will later see *derived relations*, which are built from applying operations to base relations

- Relation schema: A *named* relation defined by a *set of attribute and domain name pairs*: (A1:D1, A2:D2, A3:D3)
  - Defines the structure of one table (one relation)
  - Example: Staff(staffNumber:int, name:string, salary:double)
  - Similar to a class, not to an object
- Relational database schema:
  - A collection of relation schema, where all names are unique
  - This defines the database – i.e. what is the group of tables (relations) that make up the database

Super Keys, Candidate Keys, Primary Keys



Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relation

Superkey: any attribute/set
Of attributes that uniquely identify
Candidate key: a minimal superkey
Primary key: One chosen candidate key

For BRANCH relation:
branchNo, street, city, postcode
branchNo, street, city
branchNo, street, postcode
branchNo, city, postcode
branchNo, street
branchNo, city
branchNo, postcode
branchNo *
street, city, postcode
street, postcode
street, city (?)

Mathematical View of Relations

Definition

- Let D_i represent a domain (a finite source of values)
  - Potentially very large [all integers]
  - Or a narrow definition {1,4,5}
- We can take Cartesian product of two domains D_1 x D_2 resulting in a set of ordered pairs {(d1,d2) | d1 in D_1, d2 in D_2}
  - Extends to arbitrary number of domains

    $\prod_{i=1}^{n} D_i$. Result are tuples of degree *n*.

    -- Number of possible tuples is $\prod_{i=1}^{n} |D_i|$ == product of sizes of each domain

Definition

- Assume that we can have some property of interest that allows us to pick a subset of the tuples from our CP of domains:
  - {(d1,d2) | d1 in D_1 and d1 is even, d2 in D_2 and d2 is odd}
  - This gives us a subset of all possible tuples, termed a *relation*

Properties

- Since D1.. DN are sets themselves (unique elements), their CP cannot generate repeated tuples [no repeats in tables]

- Since the subset of CP defining our relation is itself a set, its internal order is not important [table rows can be re-arranged]

Properties

- Tuples are ordered. [suggests that table columns can't be re-ordered]
  - Assume we re-order domains in schema though?

- Cardinality is a term often used to represent the size of a set [can employ on a table as well]

Translating E/R to Relational (Chapter 17, pages 442-449) "Logical Database Design Methodology for the Relational Model – Step 2.1"

Entities

- Each entity in E/R gets its own table in entity relationship, with columns for each attribute
  - Primary key for E/R model is primary key for relational model

Student

studentID	gpa	firstName	lastName

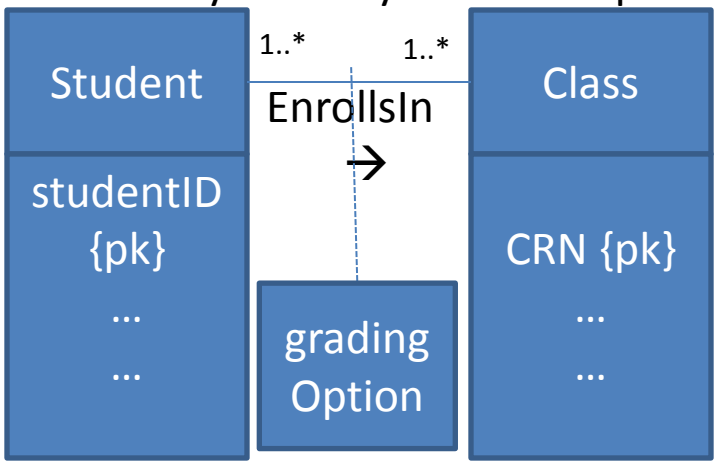
Relationships

- Not as straightforward
  - 1 rule that works for (most) everything
    - Build a new table to represent relationship
  - Shortcut rules
    - Add columns onto an existing table
  - Special cases
  - Think about: what to use as primary key

Relationships: New Table Approach

- General rule: Should work for every E/R relationship
  - Build a table of N (N >= 2) columns where
    - Column 1: primary key of 1st entity
    - Column 2: primary key of 2nd entity
      - Those primary keys may be > 1 attribute though, so expand as needed
    - Columns 3..N: relationship attributes
    - Primary key of new table is (usually)union of primary keys of entities

A many to many relationship



EnrollsIn

studentID	CRN	grading Option

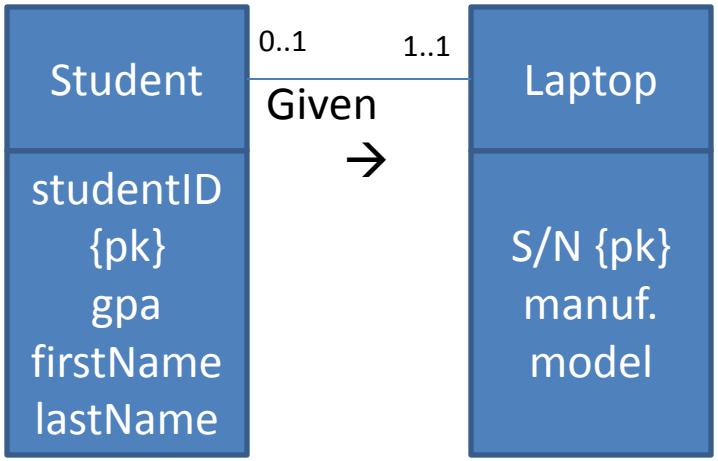
+

Student Table
Class Table

Relationships: Adding Columns Approach

- If you have a 1:1 relationship with partial mandatory participation (one of the 1s in the 1:1) add a column onto the entity table for the entity with mandatory participation with the column representing the 1 it maps to

A 1:1 relationship
Every student given 1 laptop
A laptop can be associated with no students or just 1



Student

student Id	gpa	first Name	last Name	S/N

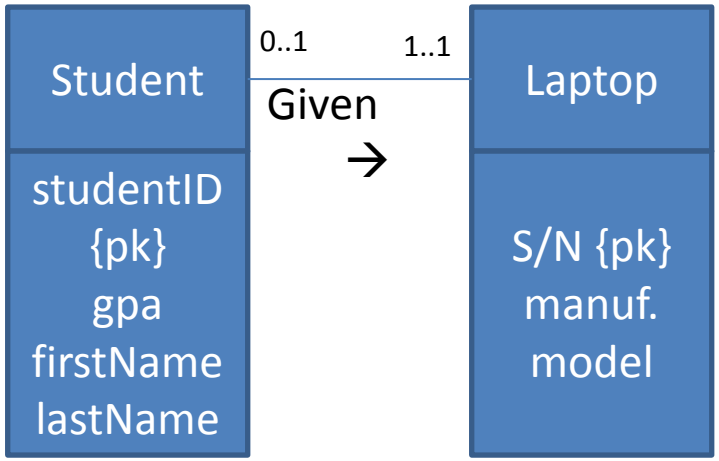
Laptop

S/N	manuf.	model

Does not change PK of original entities

Relationships: Adding Columns Approach

A 1:1 relationship



This also works!

Given

student Id	S/N

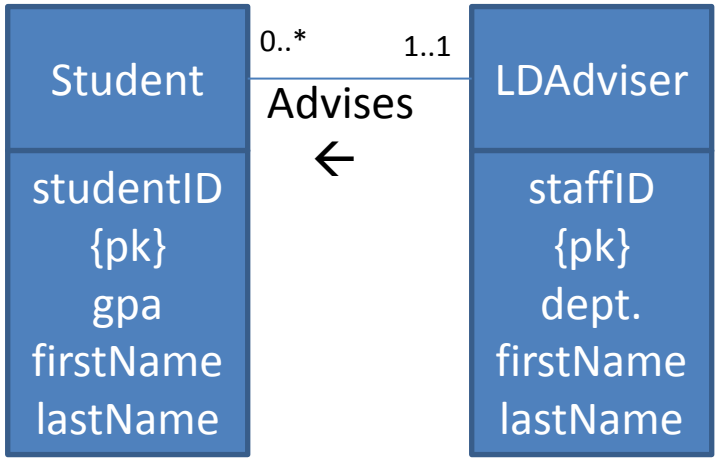
+ original Student relation
+ original Laptop relation

Why might we want to use the first example over the "new table" approach?

Relationships: Adding Columns Approach

If you have a 1:many relationship with partial (on the many side) or full mandatory participation, add a column onto the many, representing the 1

A *1:many* relationship w/
mandatory participation on "many" side
(every student must have exactly 1 LDAdviser,
each LDAdviser works with zero or more students)



Student

student Id	gpa	first Name	last Name	staffID

LDAdviser

staffID	dept.	firstName	lastName

Why don't we want to put this the other way - a studentID on the LDAdviser table?

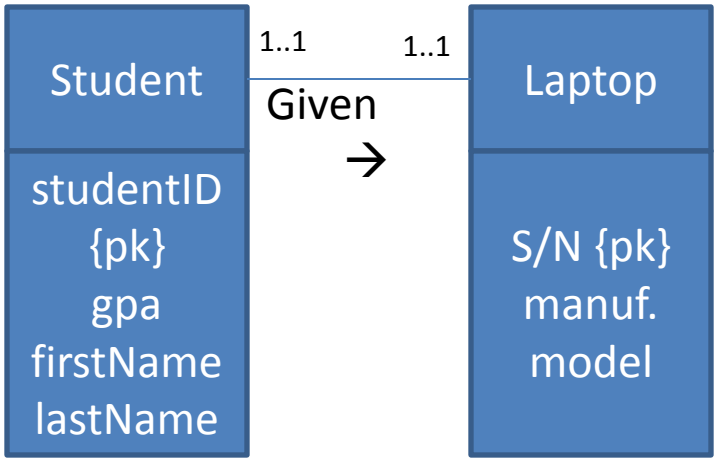
Relationships: Merging

- If you have a 1:1 relationship with mandatory participation on both sides, and NO other relationships between the two entities
  - You can merge the two entity tables

Student

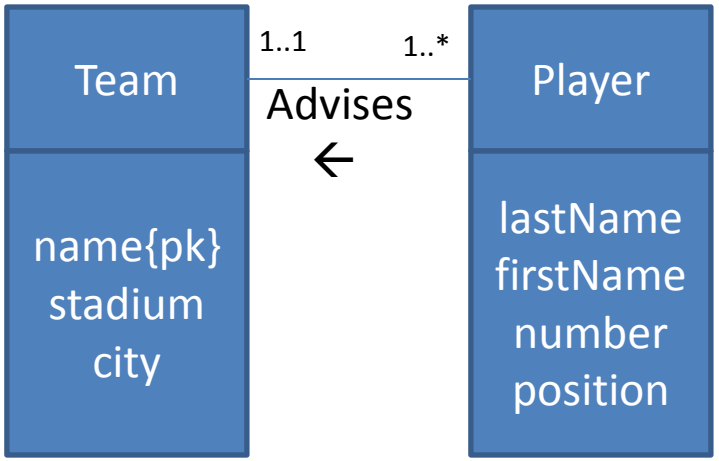
studentId	gpa	first Name	last Name	S/N	manuf	model

A 1:1 relationship
Every student given 1 laptop
Every laptop given out, each to 1 student



Maintain one of the primary keys

Relationships: Special Cases: Weak Entity Sets



- Player is a WeakEntity
  - No proper primary key with just Player attriutsto uniquely identify Player in system (not all assigned “playerID”)
  - When associated with a team, can uniquely identify a player
  - Number is a discriminator within Team
- Special case of one-to-many
  - One team has many players
  - Mandatory participation of players on a team
  - So, add team name to player entity table
  - PK is StrongEntity primary key + WeakEntity discriminator

Player

last Name	first Name	number	position	name (of team)

Team

name	stadium	city

Relationships: Special Cases: Composite Attributes

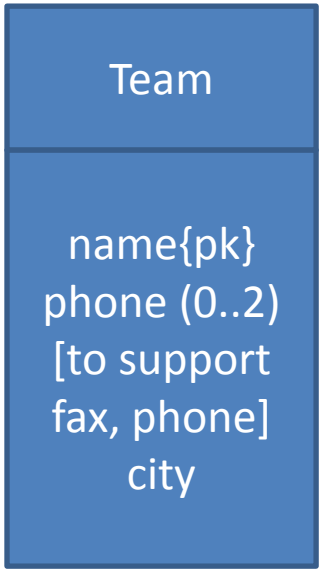
- Decompose a composite attribute into separate attributes and just use those

Team

name	stadium	city	state

| not “location” |

Relationships: Special Cases: Multi-Valued Attributes



- If an entity has a multi-valued attribute (such as two phone numbers), need to represent this with a separate table
- Build a 2-column table
  - 1st column: primary key of entity
  - 2nd column: setting for the multi-valued attribute
  - Primary key is all attributes
- Remove the multi-valued attribute from the original entity tiable

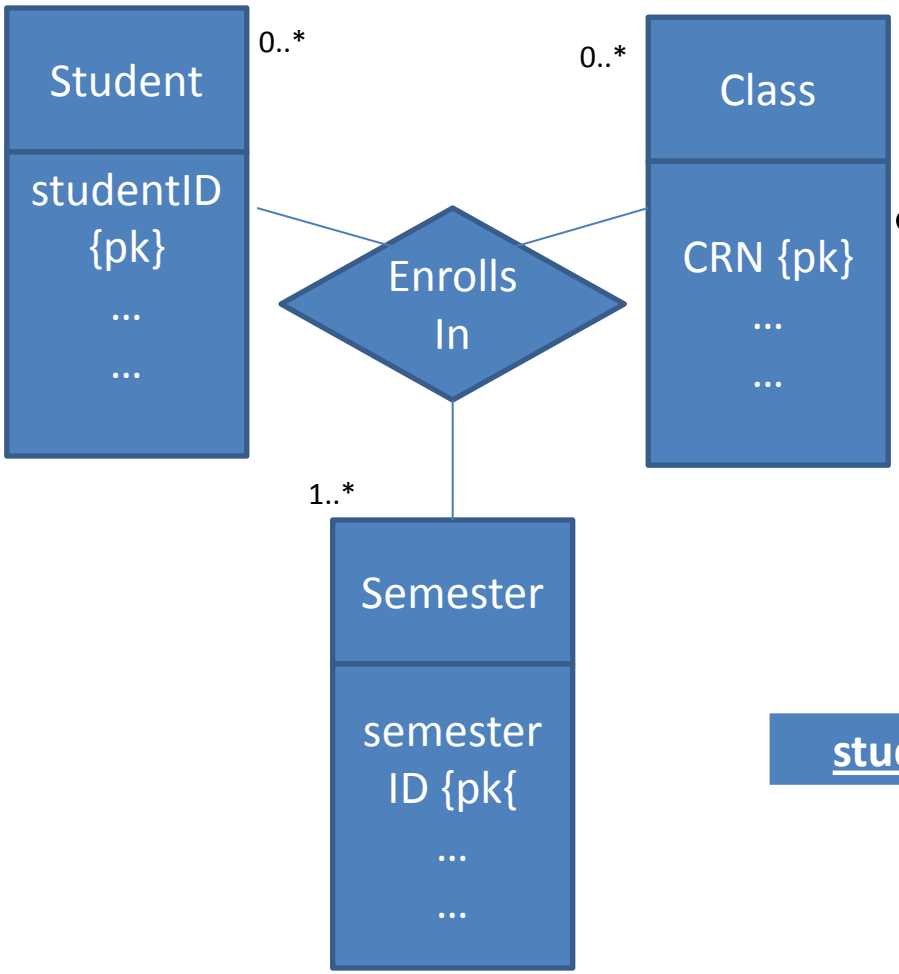
Team

name	city

TeamPhone

name	phone

Relationships: N-ary relationships

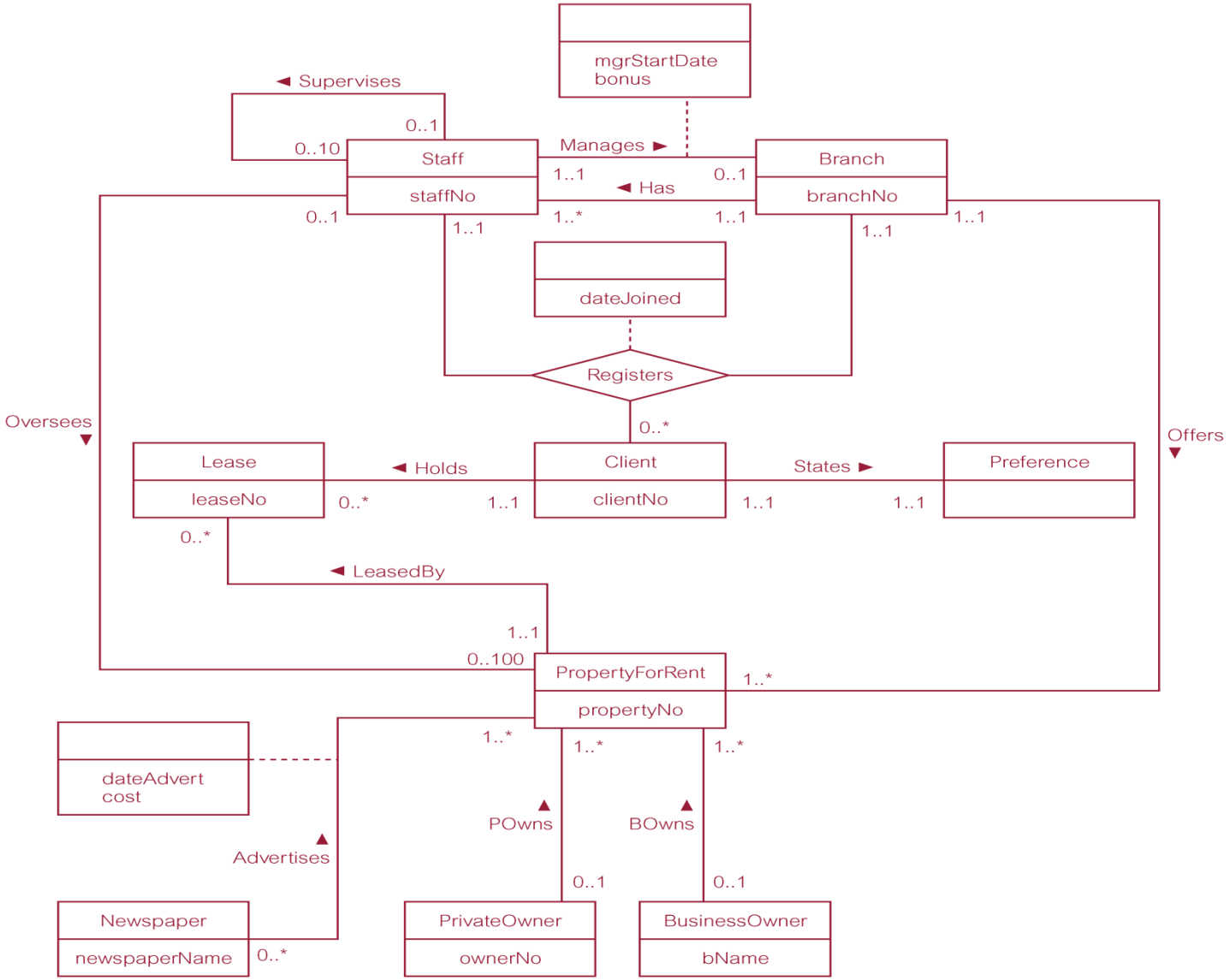


- For (complex) n-ary relationships, just use "new-table" approach
- Primary key is union of all associated entity primary keys

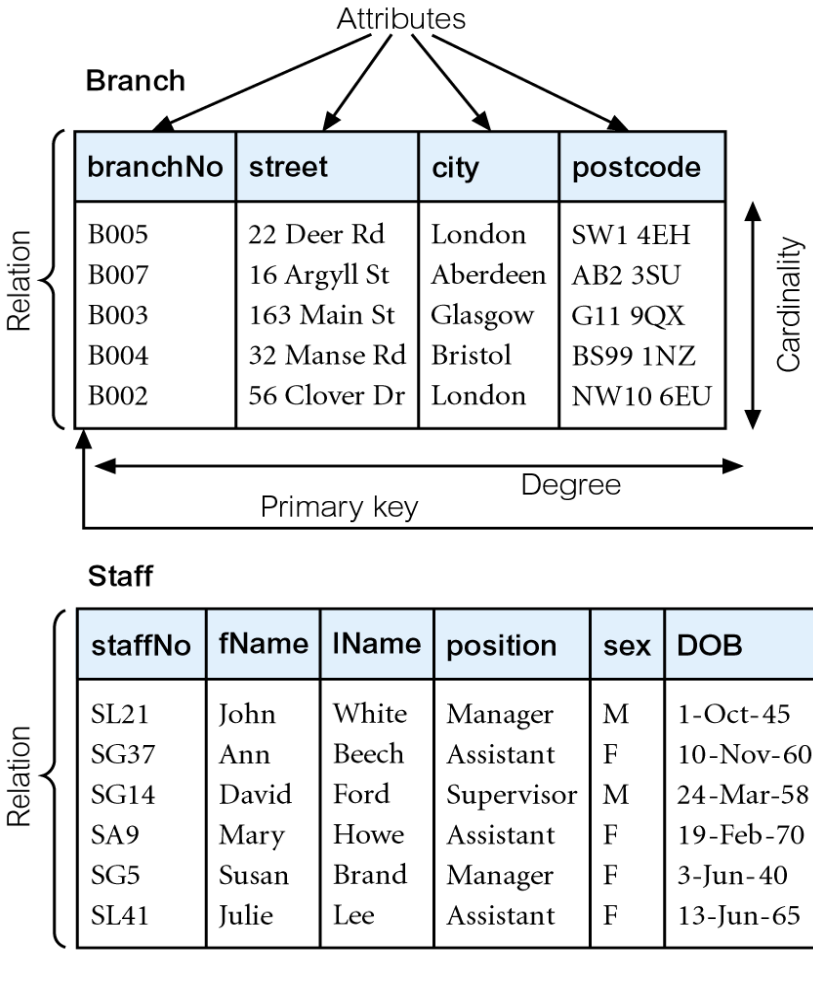
EnrollsIn

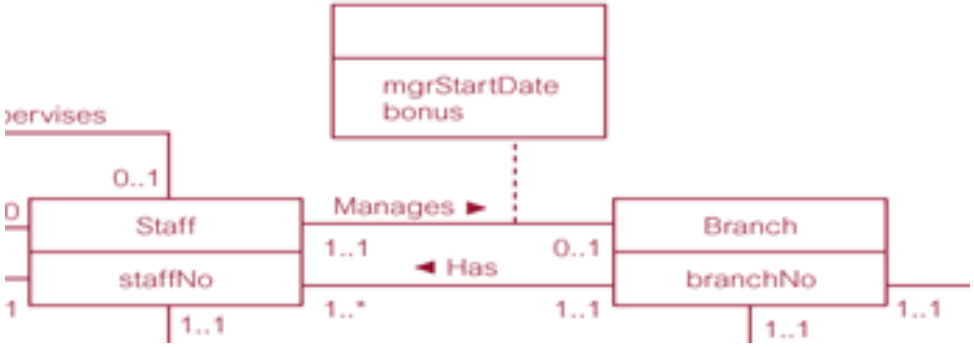
studentID	CRN	semesterID

Validating The Book: DreamHome E/R diagram



Validating The Book: DreamHome Staff & Branch Relationships

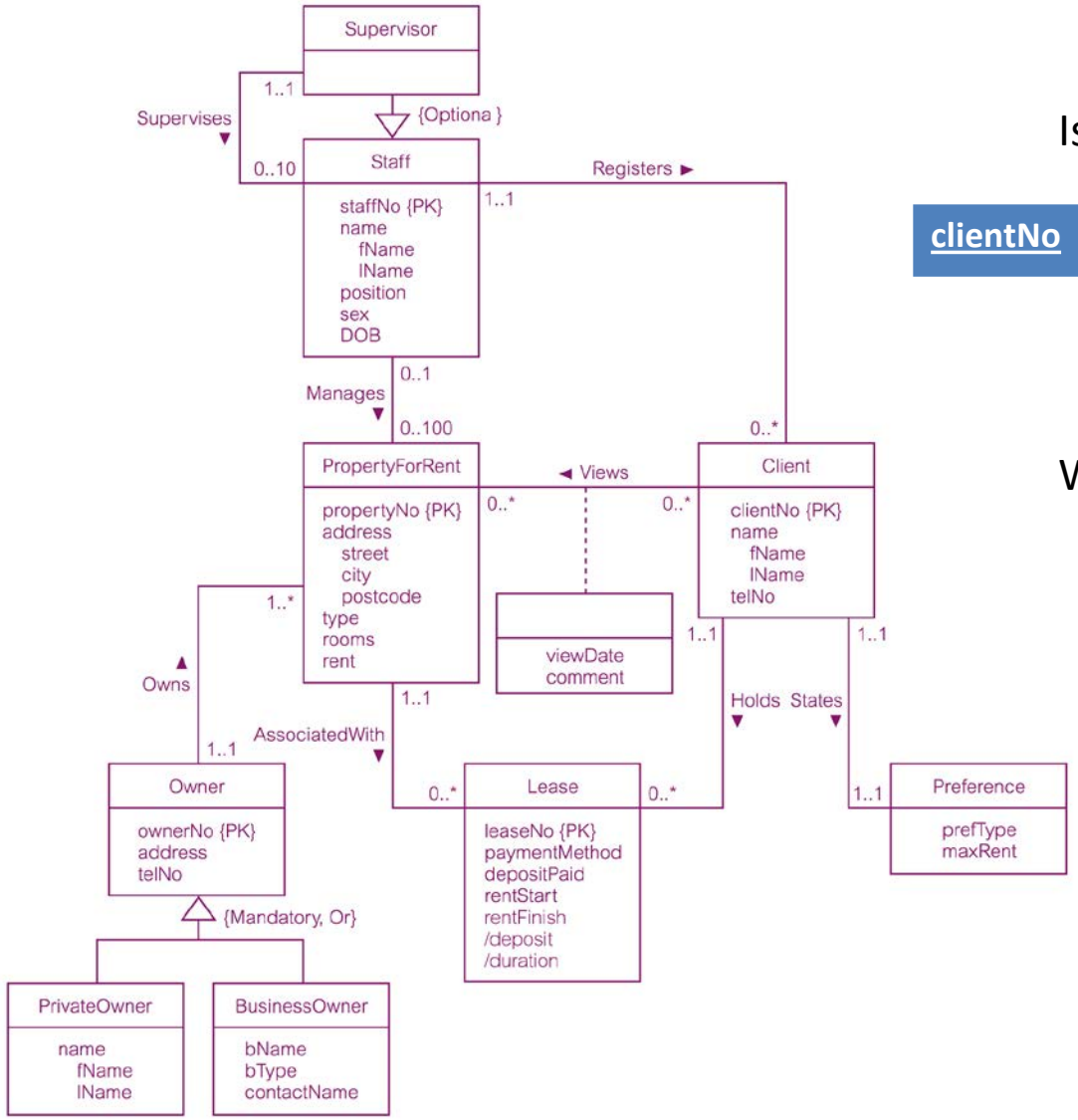




Modeling the "Branch has Staff" relationship

1 to many, mandatory participation by staff (everyone works at a branch)

Validating The Book: Additional DreamHome Suggestions

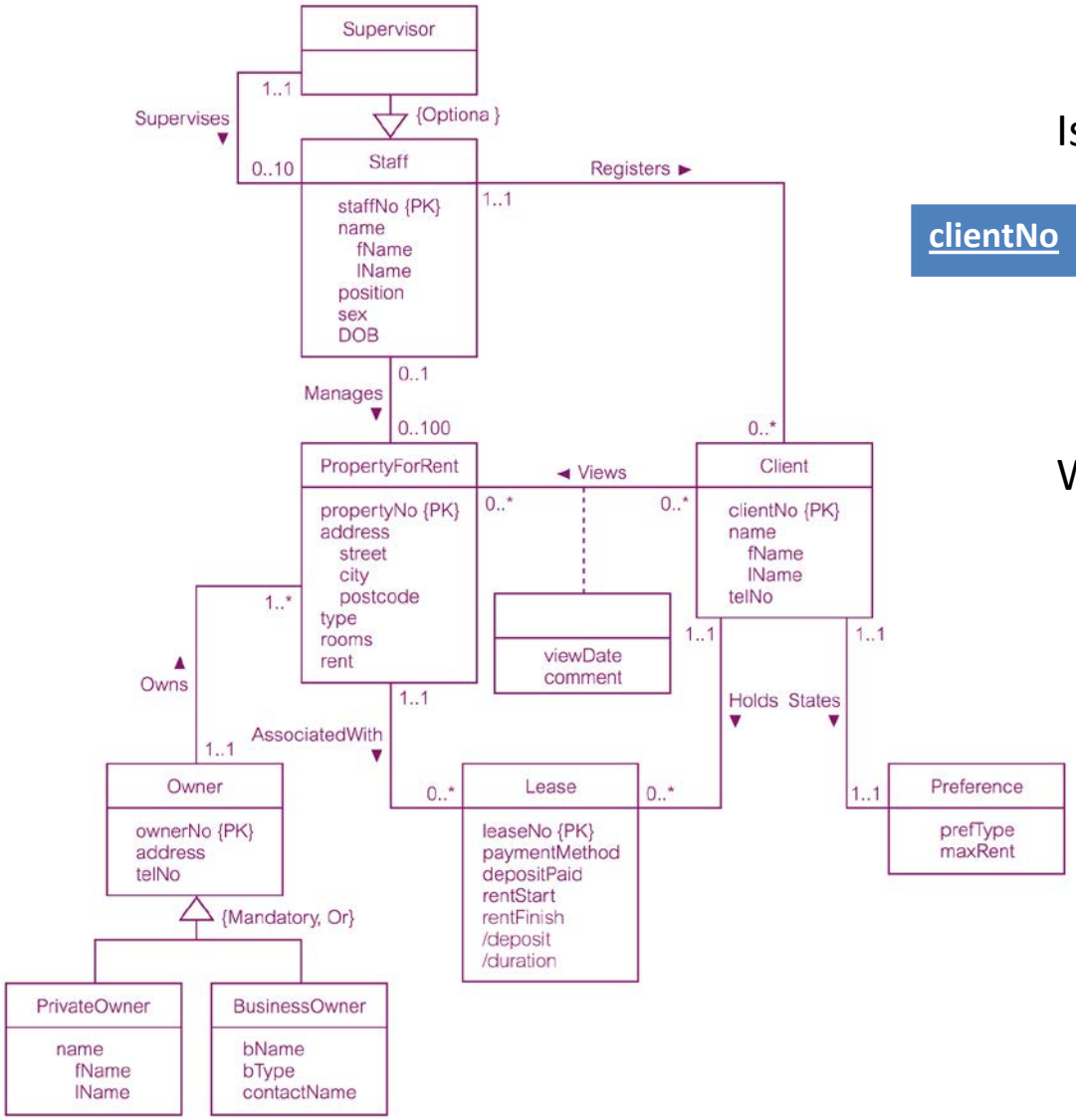


Is it reasonable to have?

clientNo	fName	lName	telNo	staffNo

Why or why not?

Validating The Book: Additional DreamHome Suggestions



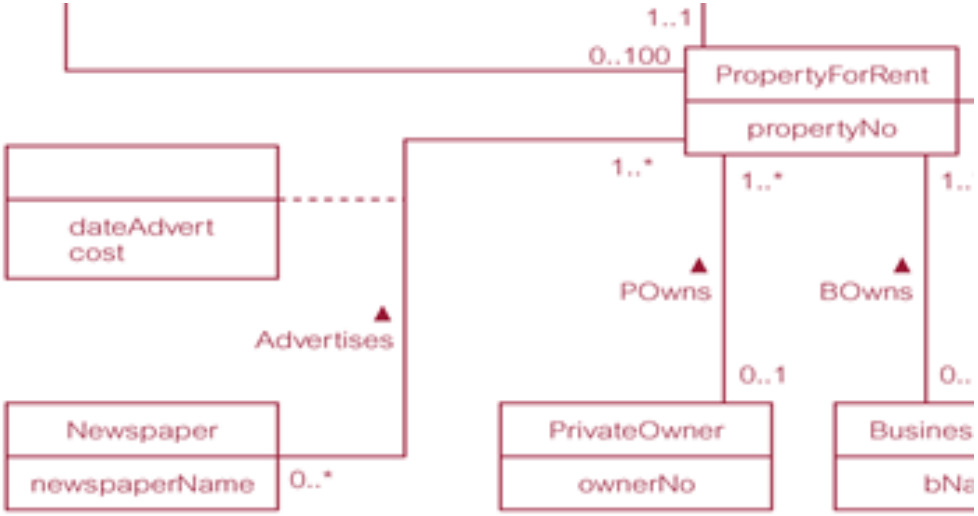
Is it reasonable to have?

clientNo	propertyNo	viewDate	comment

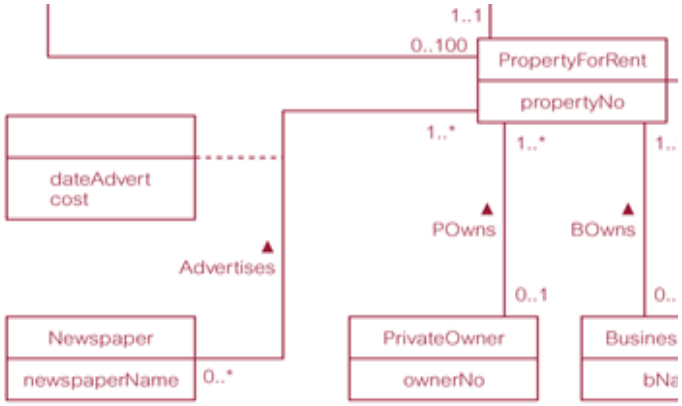
Why or why not?

Practice Problem

Build all the tables necessary to represent the *Advertises* relationship between *Newspaper* and *PropertyForRent* shown in an E/R model below: (ignore other relationships)



Practice Problem



Many to many relationship
Optional PropertyForRent participation
Mandatory Newspaper participation
Relationship-specific attributes

Newspaper

newspaperName

PropertyForRent

propertyNumber

Advertises

newspaperName	**propertyNumber**	**dateAdvert**	**cost**