Systems Design and Security

UNIFIED MODELING LANGUAGE

Part 7: Database Design

http://staffwww.dcs.shef.ac.uk/people/A.Simons/

Home ⇒ Teaching ⇒ Lectures ⇒ COM2008/COM3008

Bibliography

- Database Systems
  - T Connolly and C Begg, Database Systems – a Practical Approach to Design, Implementation and Management, 6th ed., Pearson, 2014.
  - C J Date, An Introduction to Database Systems, 8th ed., Pearson, 2003.
- UML Profile for Databases (not official)
  - D Gornik, UML Data Modelling Profile, IBM/Rational Software TP162, May, 2002.
  - S Ambler, A UML Profile for Data Modelling, Agiledata.org, 2009.

Outline

- Evolution of databases
- Relational data model
- UML profile for databases
- Entity-relationship modelling
- Traditional table normalization
- Normal and pre-normal forms

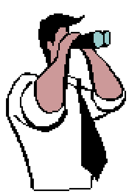
Reading: Date chapters 10, 11, 12, 13;
Connolly and Begg, chapters 11, 12, 13, 14

Business Data

- Valuable
  - data forms the core of business operations
  - customers, suppliers, purchasing and sales, …
  - often, provides the commercial advantage
- Persistent
  - data survives many executions of the program
  - possibly accessed by many programs, many users
- Protected
  - must be kept safe, even if the system fails
  - must be protected from unauthorised access

Database Systems

- Early data storage
  - 1880s – punched cards, influenced CODASYL strategy
  - 1950s – magnetic tape, influenced IBM's IMS
- Navigational model
  - 1960s – CODASYL, networks of records, pointer-following
  - IBM IMS, hierarchical records, influences XML today
  - 1990s – XML databases, hierarchical, text-based
- Relational model
  - 1970s – Edgar Codd, relational algebra, platform-independent
  - Relational databases, table-based, most efficient searching
  - Structured Query Language (SQL) – common query language
  - 1980s – object-oriented model, richer datatypes, procedures
  - 1990s – object-relational mapping, OQL, ...



Storage Issues

- Programs in memory
  - objects have rich datatypes, eg: Account, Holder
  - data structure is an arbitrary connected graph
  - structures may be extended dynamically (lists of pointers)
  - navigation is by object reference (memory pointer)
- Database files on disk
  - simply-typed data, eg: Integer, String, Date, Money
  - data structure is a fixed, predefined set of tables
  - data tables have a fixed width, cannot grow/shrink to accommodate varying lists of references
  - navigation is by searching according to key values

Relational Databases

- Original Purpose
  - to eliminate redundancy of stored information
    - don't store same information in several places
    - eliminate blank fields, repeated groups of fields
  - to minimize dependency between data items
    - easy to search for data – optimal links via keys
    - easy to insert, update, delete single items
    - fewer cascading effects (viz. knock-on updates)

- Structure of Data
  - logically a set of tables, indexed by row $\times$ column
  - physically a set of files containing many (fixed-length) records

Database Tables

Current Account Table

<u>number</u>	balance	overDr
0214537	323.50	-100
0773465	443.97	-100
1334890	-27.68	-500

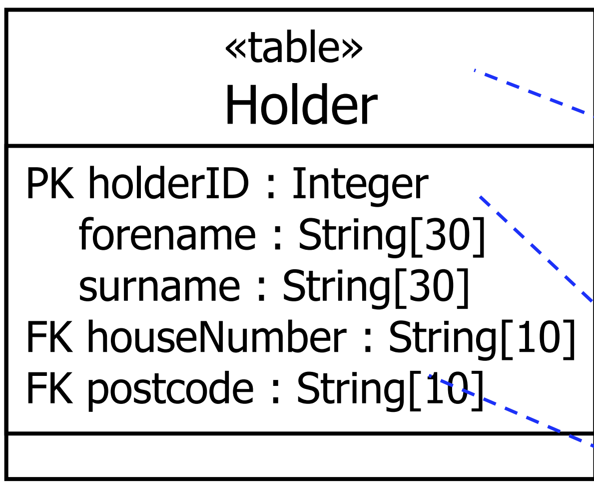
- Table columns
  - represent attributes
  - have simple types
  - a column has one type
  - a type has a fixed size

- Table rows
  - represent objects
  - all attribute values
  - row has mixed types

Holder Table

<u>custID</u>	forename	surname	addrID
235	Inderpal	Singh	S104DP12
673	Sarah	Wilson	S57AA297
589	Tariq	Al Harq	S116SQ40

Database Profile



primary and foreign keys are attributes that serve a special row-identifying purpose

- UML Profiles
  - a profile is an extension to standard UML notation
  - database profile is not yet universal; but widespread
- Data tables
  - use the «table» label to indicate a data table
  - first box defines columns
  - second box is for triggers, integrity check functions
- Key attributes
  - use PK for primary key
  - use FK for foreign key

The University Of Sheffield.

Primary Key

«table»
CurrentAccount

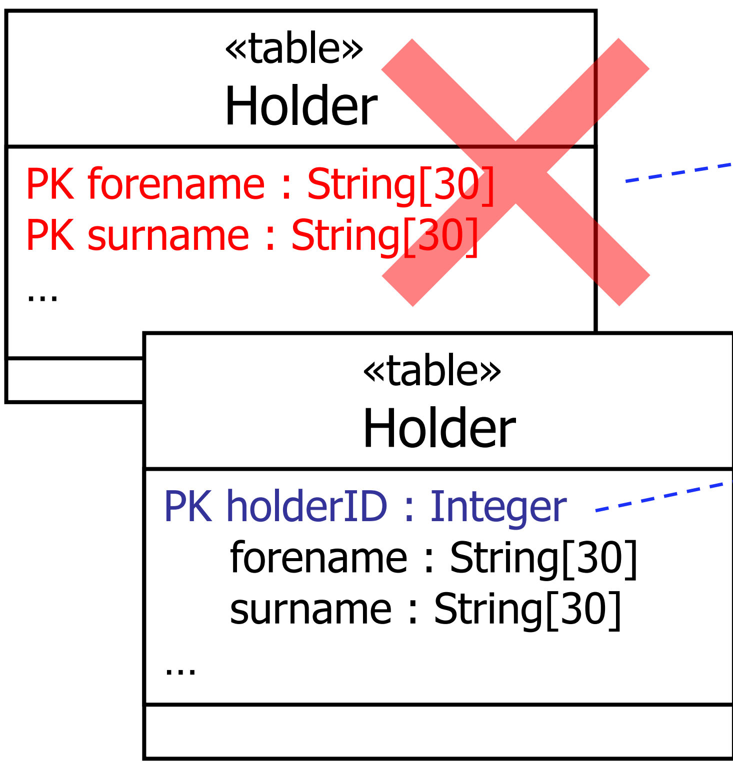
PK number : Integer
balance : Money
overdraftLimit : Money

«table»
Address

PK houseNumber : String[10]
streetName : String[30]
cityName : String[30]
PK postcode : String[10]

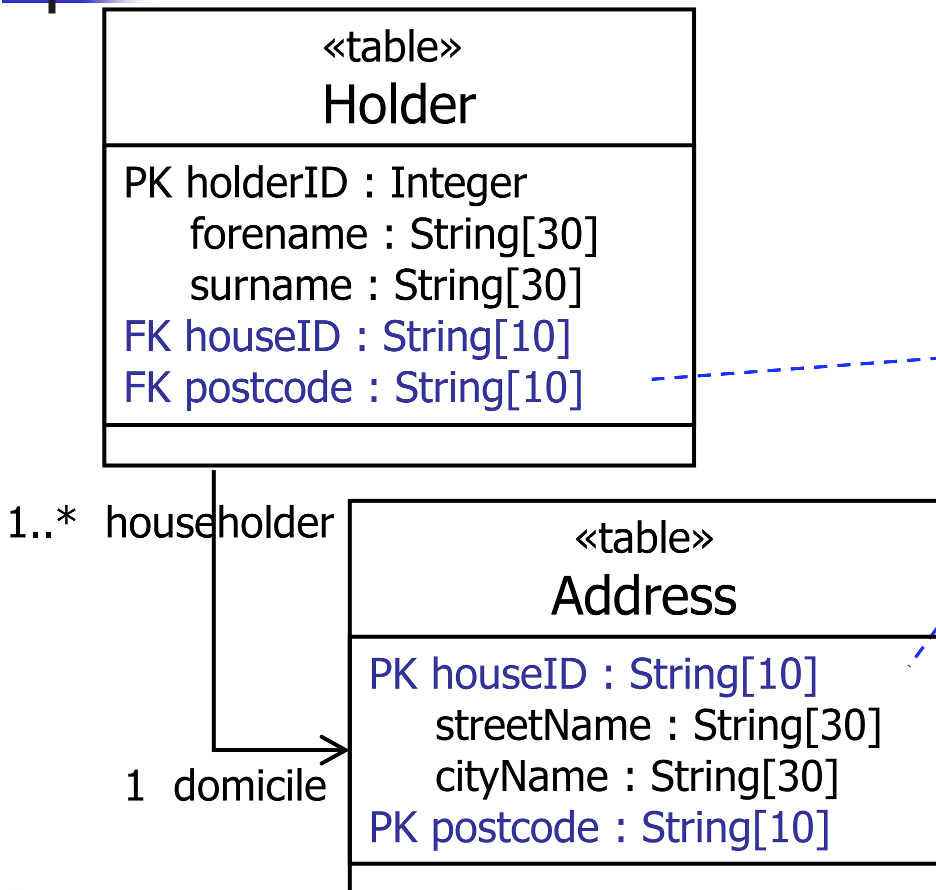
- Simple key
  - a single column, having a
  - unique value for each row
- Compound key
  - several columns, having a
  - unique value-combination for each row
- Entity integrity
  - PK may not be null – must exist for each row
  - PK must identify a single row – no duplicate keys

Surrogate Key



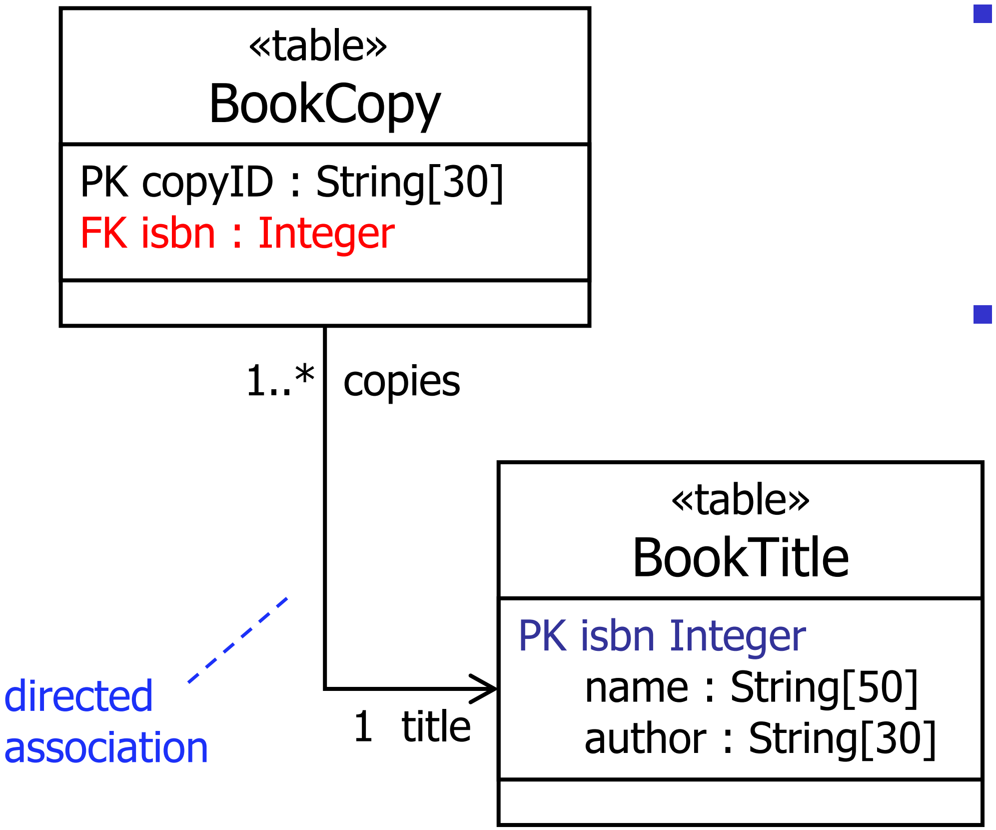
- Problem
  - forename and surname – can't be a compound PK
  - because of the possibility of duplicate names
- Surrogate key
  - create an artificial attribute
  - only if no other suitable key can be found
  - or if compound key is too large, > 3 rows
  - but don't use for all tables! (wasteful)

Foreign Key



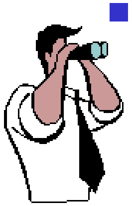
- Foreign key
  - a table has one FK for each related table
  - FK from one table refers to the PK in another table
  - value of FK from one table
  - equals the value of the PK in another table
  - may be null, or repeated
- Referential integrity
  - if FK is not null, then
  - row with same PK must exist in the related table

Navigation

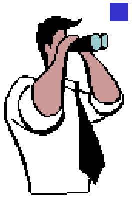


- Linking tables
  - tables are linked in one direction only
  - link from the "many" to the "one" side – why is this?
- Directed association
  - arrow shows direction of navigation in UML
  - arrow points from the "many" to the "one" side
  - the "many" side has the FK
  - refers to PK of the "one"
  - matched PK, FK will have the same values

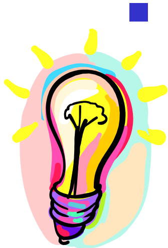
Data Normalization



- Relational Data Analysis
  - Edgar Codd, Ray Boyce – complicated set of rules for transforming arbitrary data tables into Normal Form (NF)
  - based on the detailed analysis of attribute dependency
  - 1NF, 2NF, ... 5NF, 6NF (typically 3NF, 4NF required)



- Entity-Relationship Modelling
  - Peter Chen – simple diagram-based technique for normalization
  - based on simplifying object relationships and multiplicities
  - achieves 3NF and sometimes better



- Event-Driven Design
  - Monique Snoeck, Anthony Simons – event table and graph-building technique linking according to existence dependency
  - constructs a data model already in 3NF, sometimes better

The University Of Sheffield.

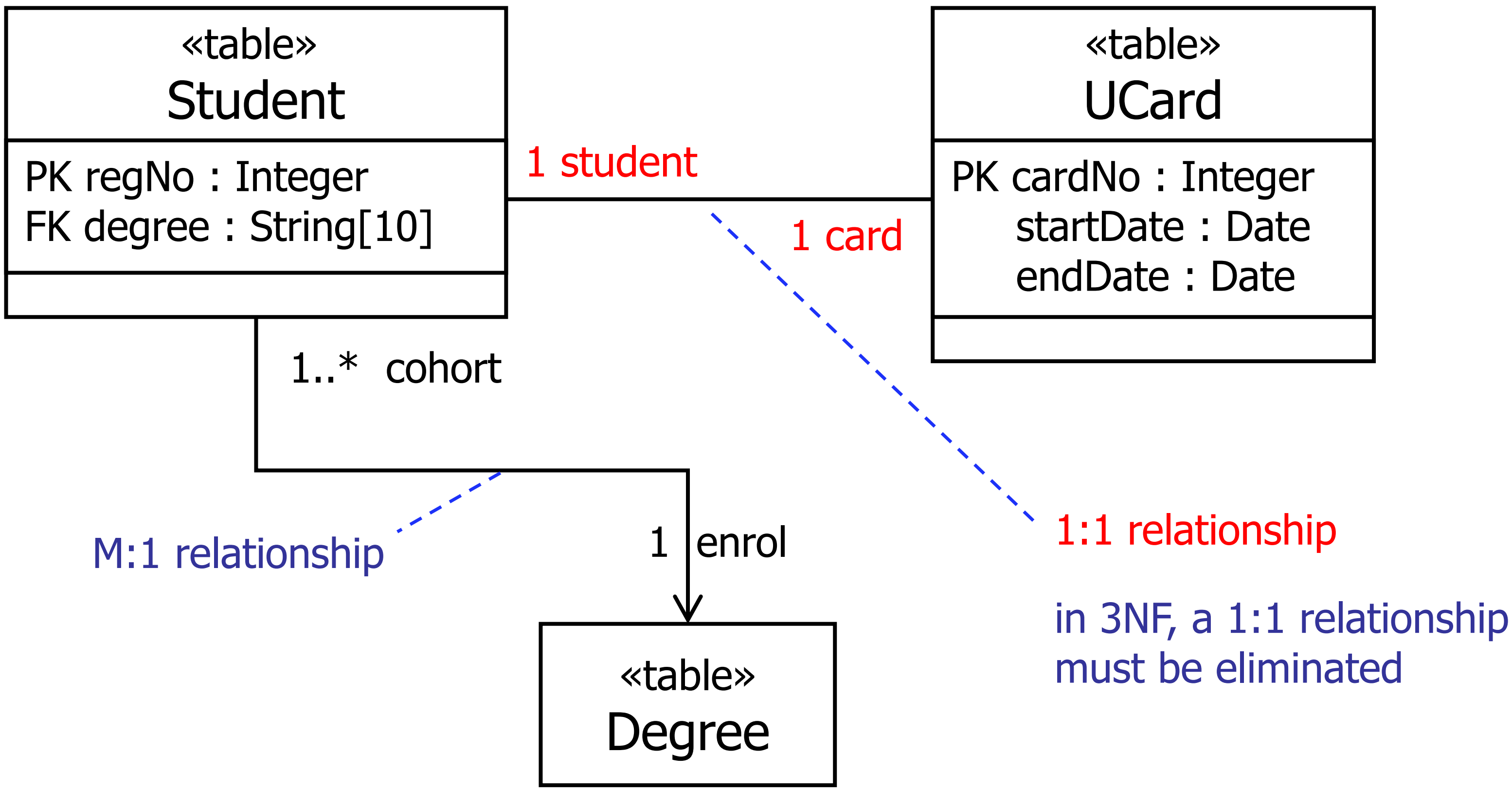
Relationship Types

- One-to-One (1:1)
  - eg: every student has a unique UCard (1:1)
  - eg: each bank branch has a single address (1:1)
- One-to-Many (1:M)
  - eg: an address contains many householders (1:1..*)
  - eg: a person optionally has a driving licence (1:0..1)
  - eg: a woman optionally has many children (1:0..*)
  - eg: a person may optionally also be a student (1:0..1)
- Many-to-Many (M:N)
  - eg: many modules are offered on many degrees (1..*:1..*)
  - eg: books are loaned optionally to a borrower (0..*:0..1)
  - eg: many borrowers optionally reserve many books (0..*:0..*)

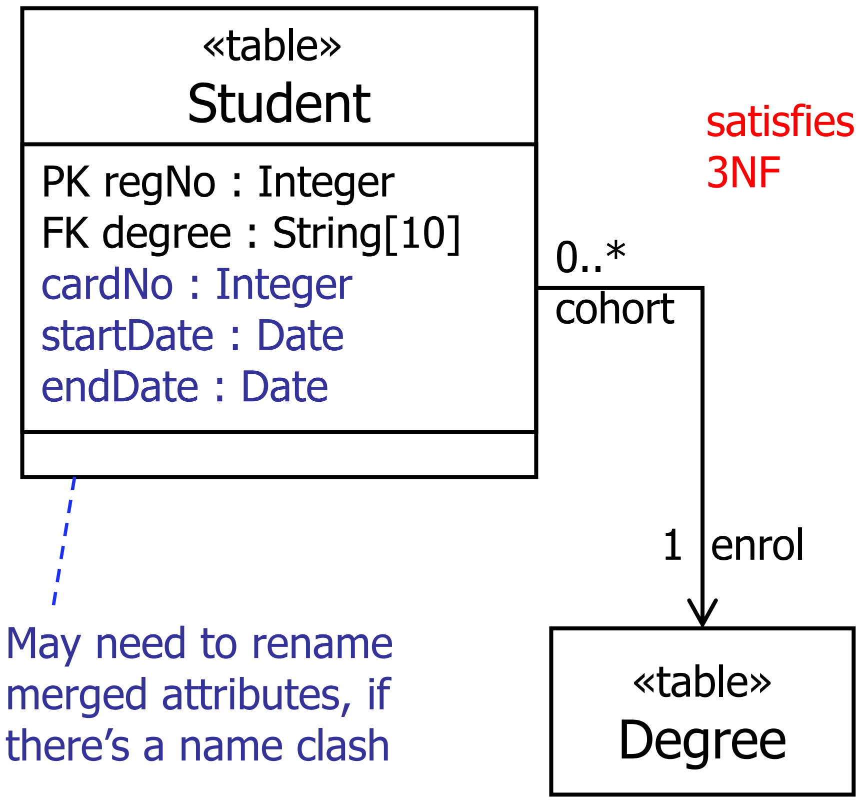
Minimize Data Dependency

- Data design goals
  - remove duplicated data, redundant paths (for space efficiency)
  - optimise table structure for the independent insertion, update and deletion of single data items (reduces cascading effects)
- Minimization technique (ERM)
  - merge tables that are in 1:1 relationship (removes need for join)
  - link tables that are in 1:M relationship (from $M \rightarrow 1$)
  - introduce new linker tables to encapsulate each M:N relationship, yielding two new 1:M relationships (from $M \rightarrow 1$)
  - find redundant search paths navigating to the same sets of rows and delete the shorter path

Merge Required

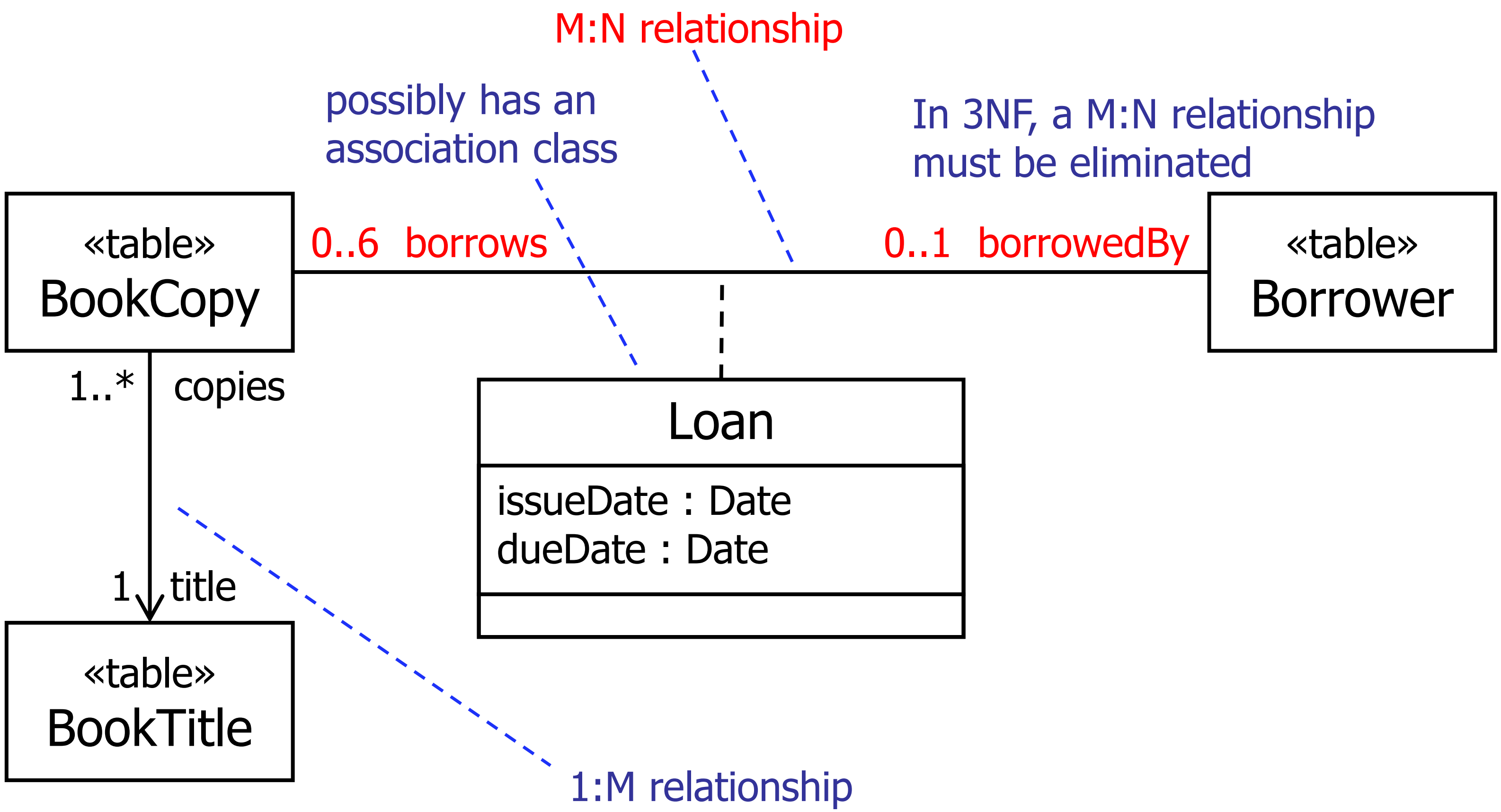


Merged Tables

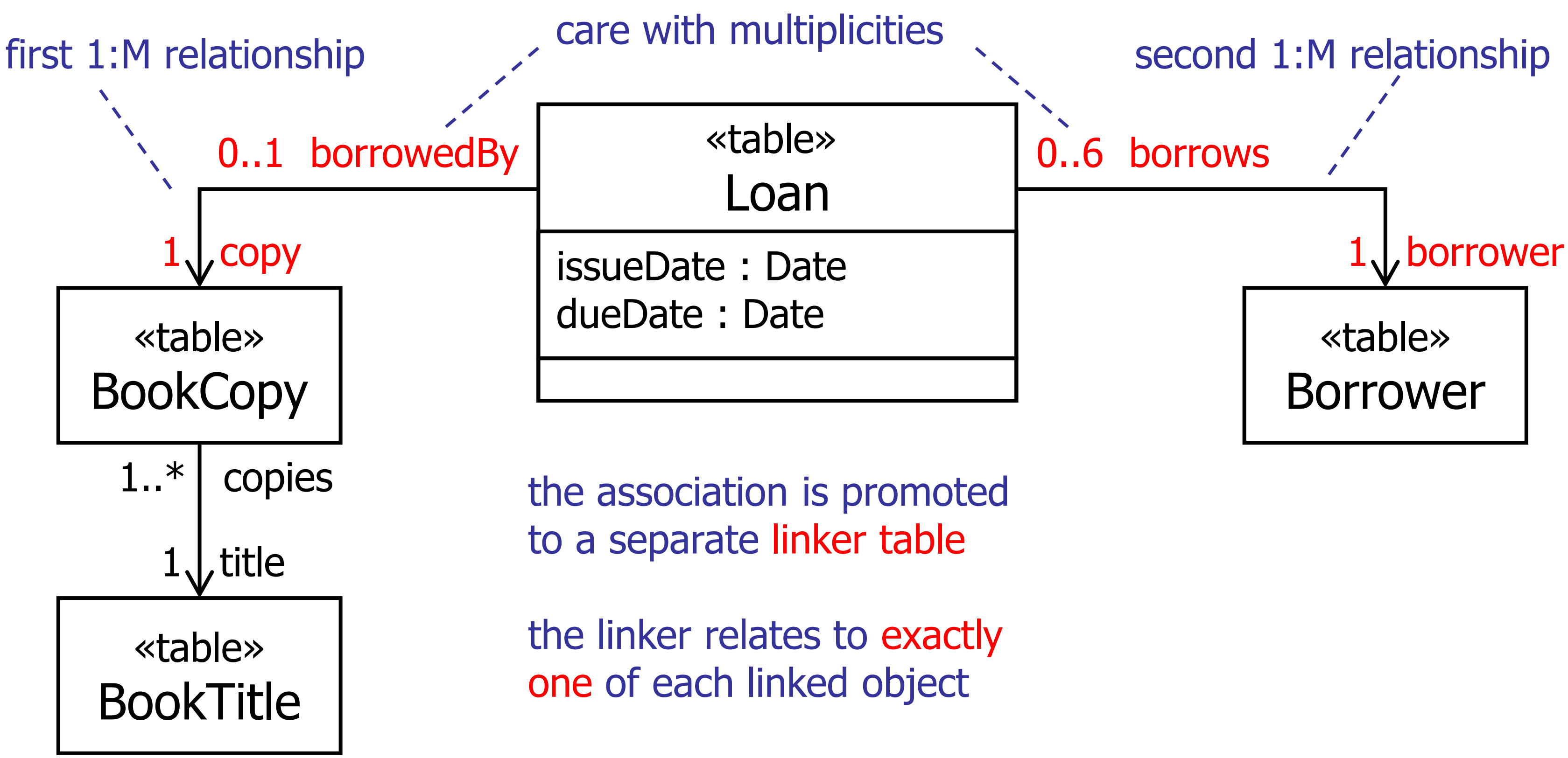


- Which table?
  - keep the stronger concept
  - merge the attributes of the deleted table
  - transfer any associations from the deleted concept
- Which key?
  - possibly several candidate keys for the PK
  - preserve one PK only
  - here: cardNo demoted to a dependent attribute

Linker Required



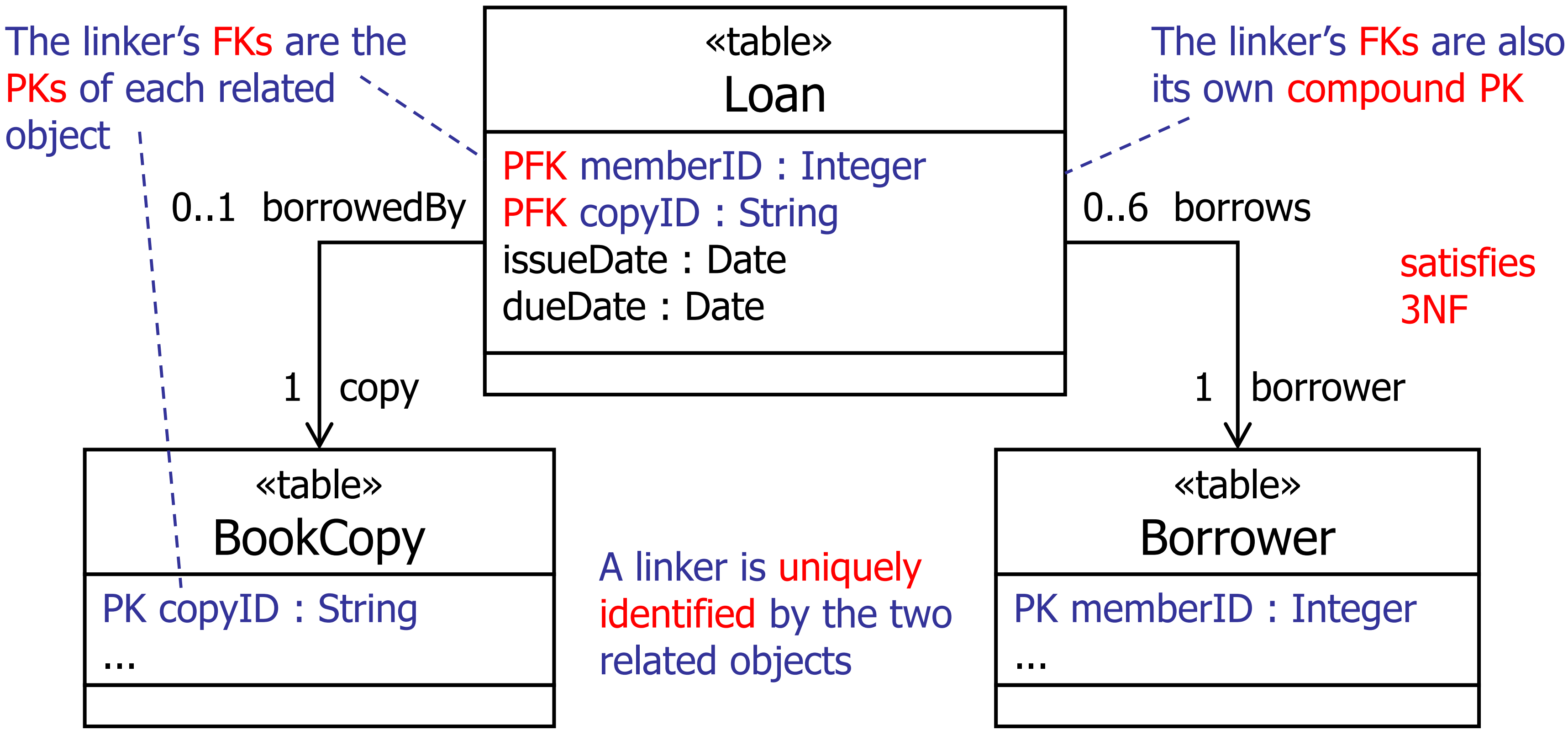
Linker Added



the association is promoted to a separate linker table

the linker relates to exactly one of each linked object

Linker Keys



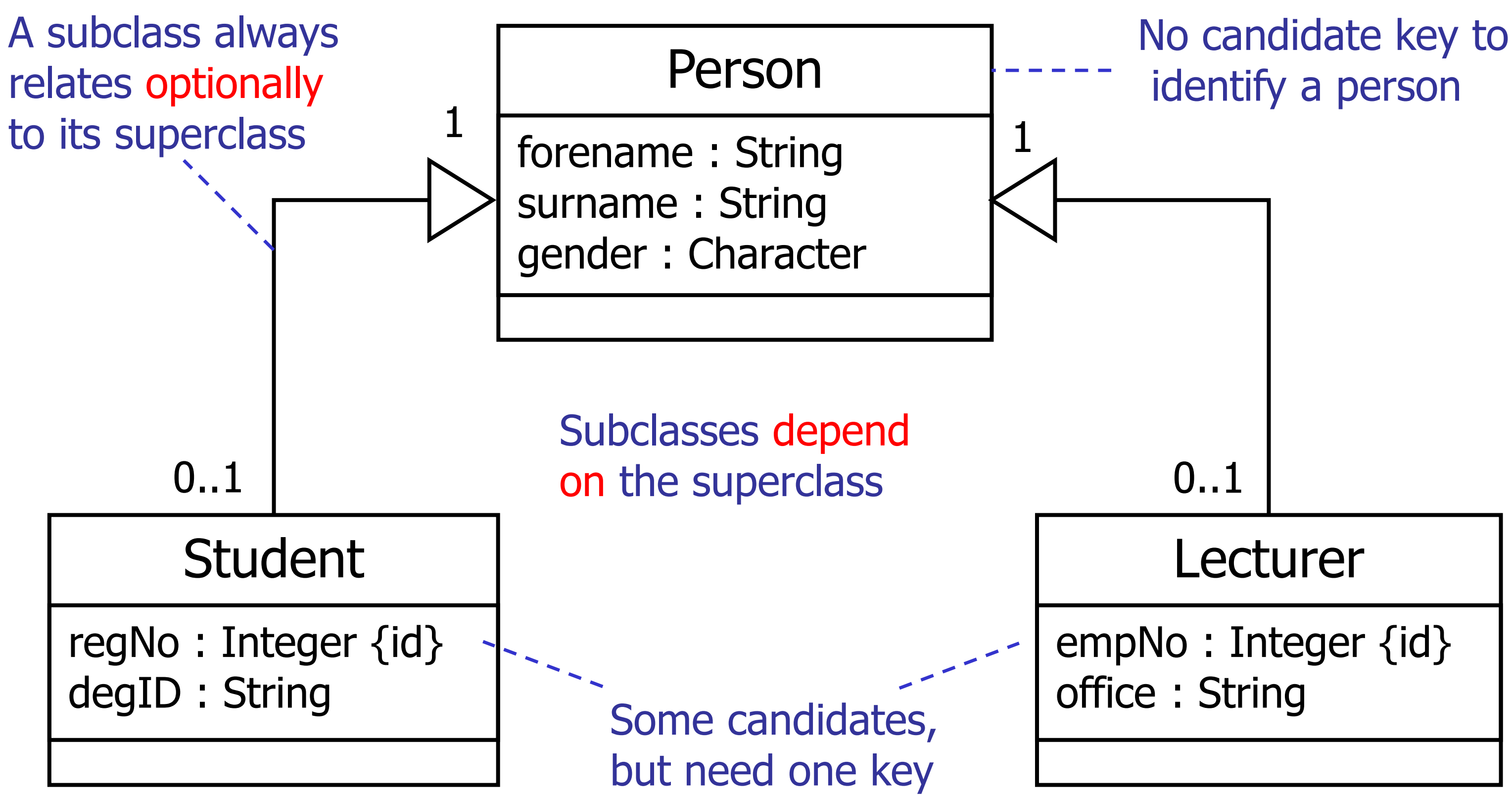
The linker's FKs are the PKs of each related object
«table»
Loan
PFK memberID : Integer
PFK copyID : String
issueDate : Date
dueDate : Date
The linker's FKs are also its own compound PK
0..1 borrowedBy
0..6 borrows
satisfies 3NF
1 copy
1 borrower
«table»
BookCopy
PK copyID : String
...
A linker is uniquely identified by the two related objects
«table»
Borrower
PK memberID : Integer
...

Lab 1: Normalize the Library

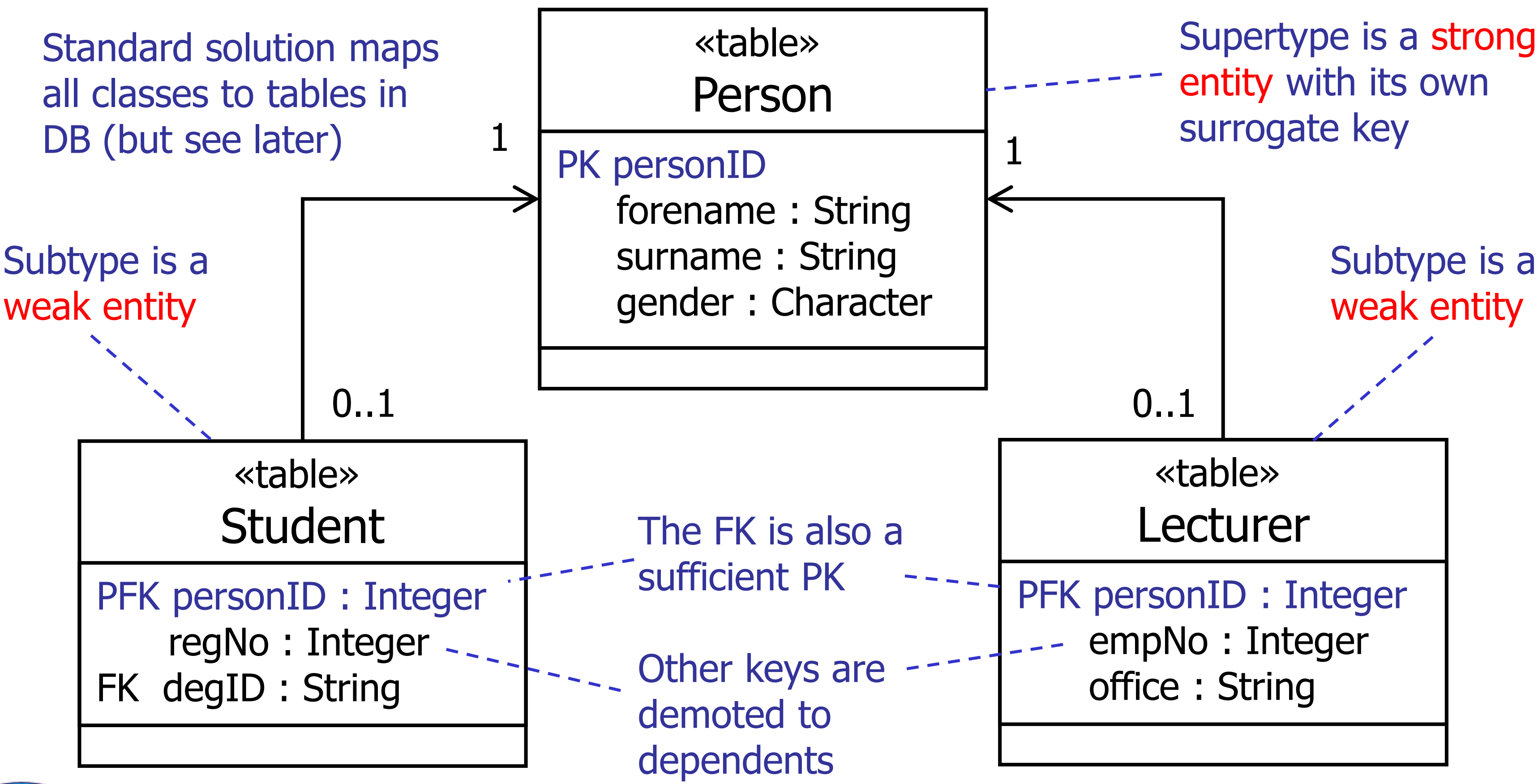
Run a Poll

- Complete the normalization of the tables
  - use the information model developed earlier
  - consider the relationship where borrowers reserve books
  - care with the roles and multiplicities of linkers
- Identify all navigation paths through the data
  - identify all primary keys (compound? surrogate?)
  - supply all required foreign keys (on the many-side)
  - how does this affect generalisation relationships?

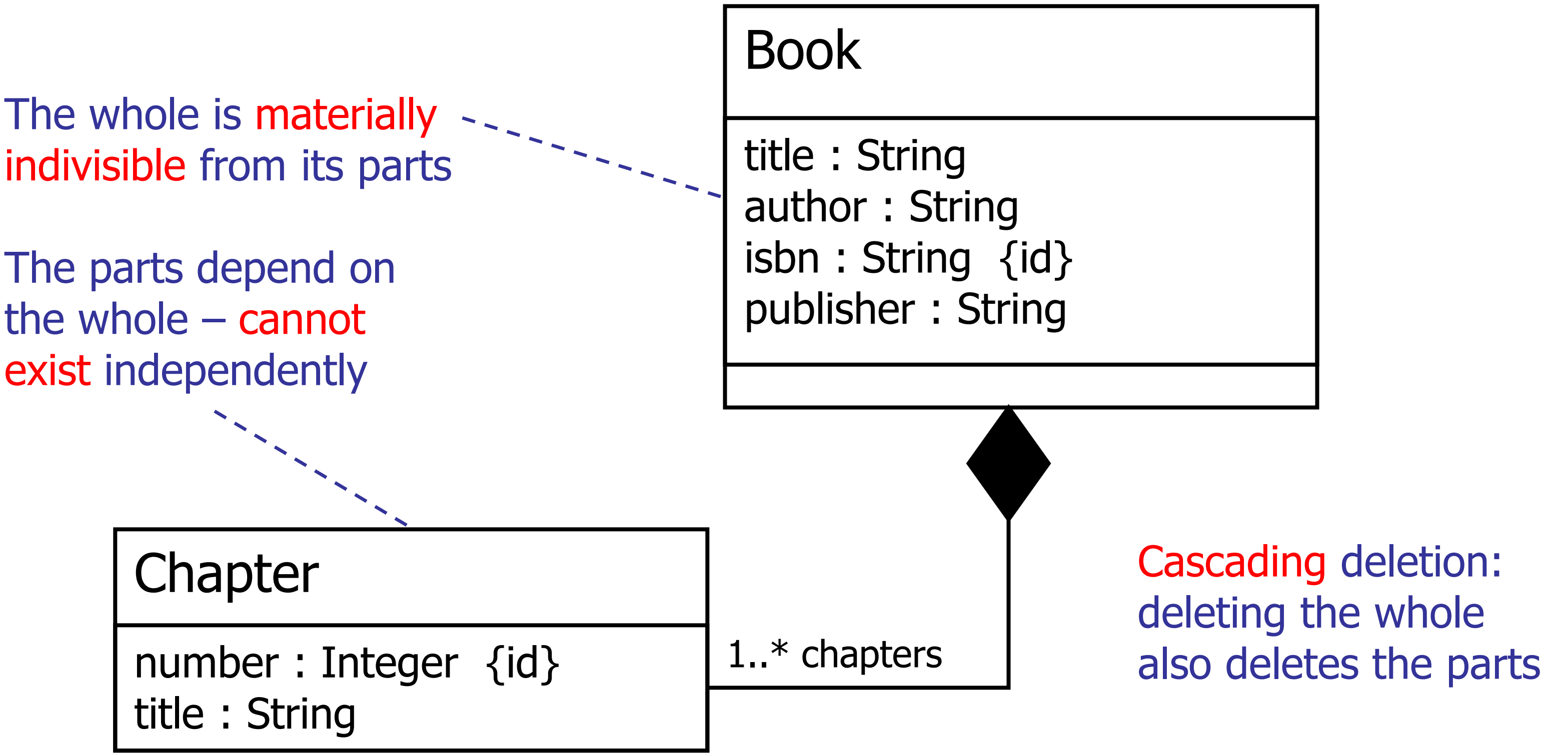
Generalisation



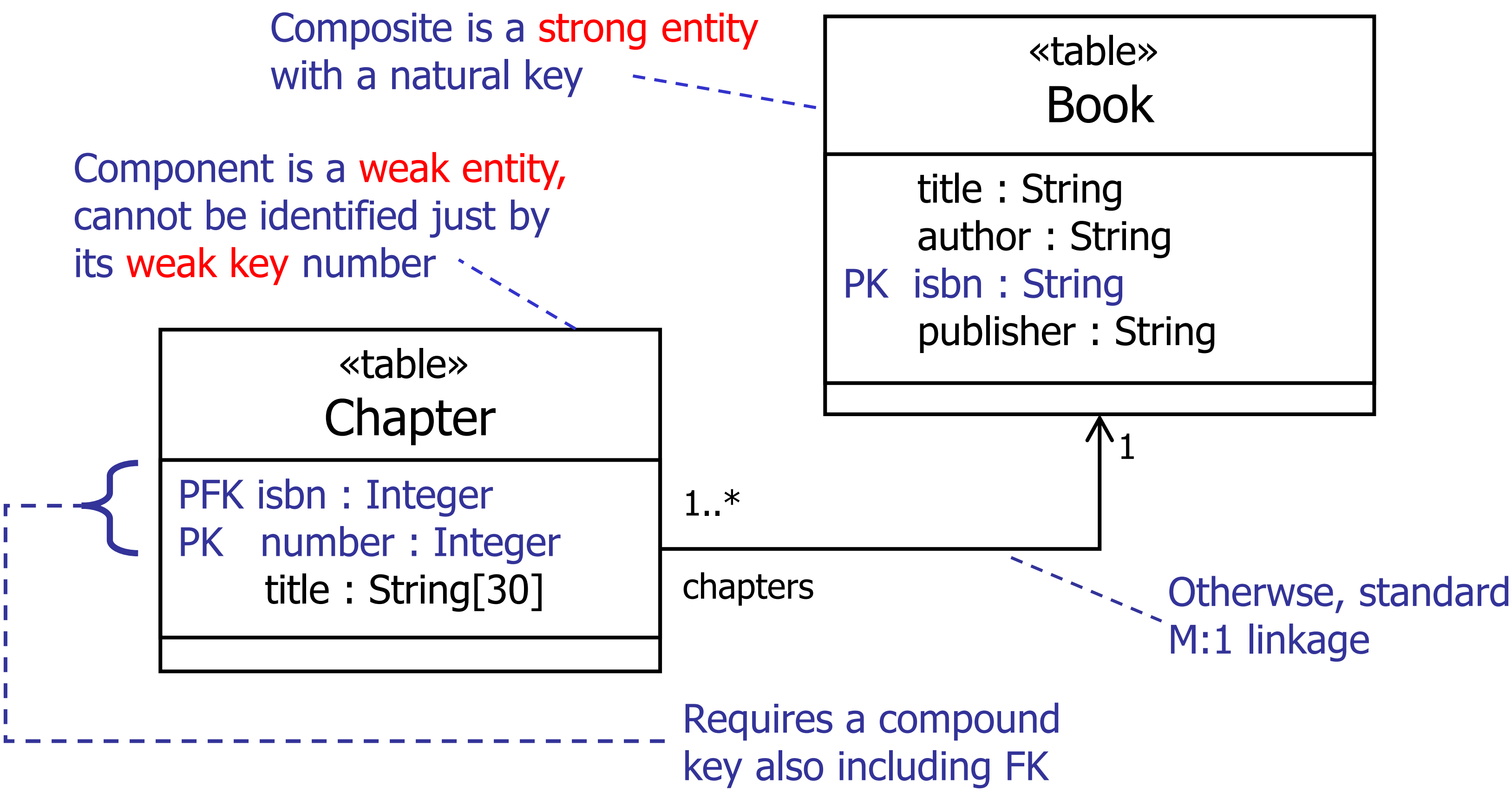
Strong/Weak Entities



Composition



Strong/Weak Entities



Aggregation

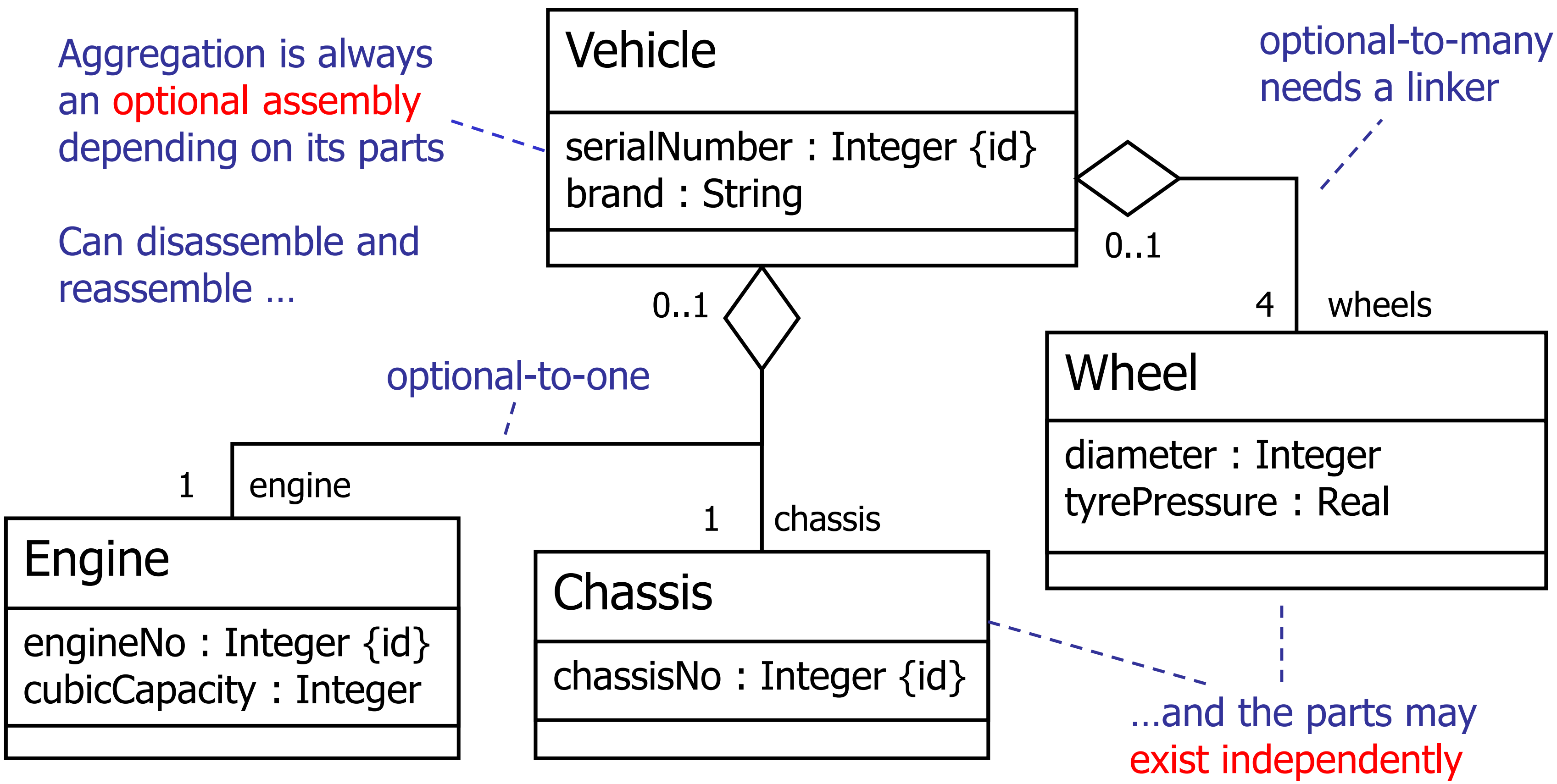
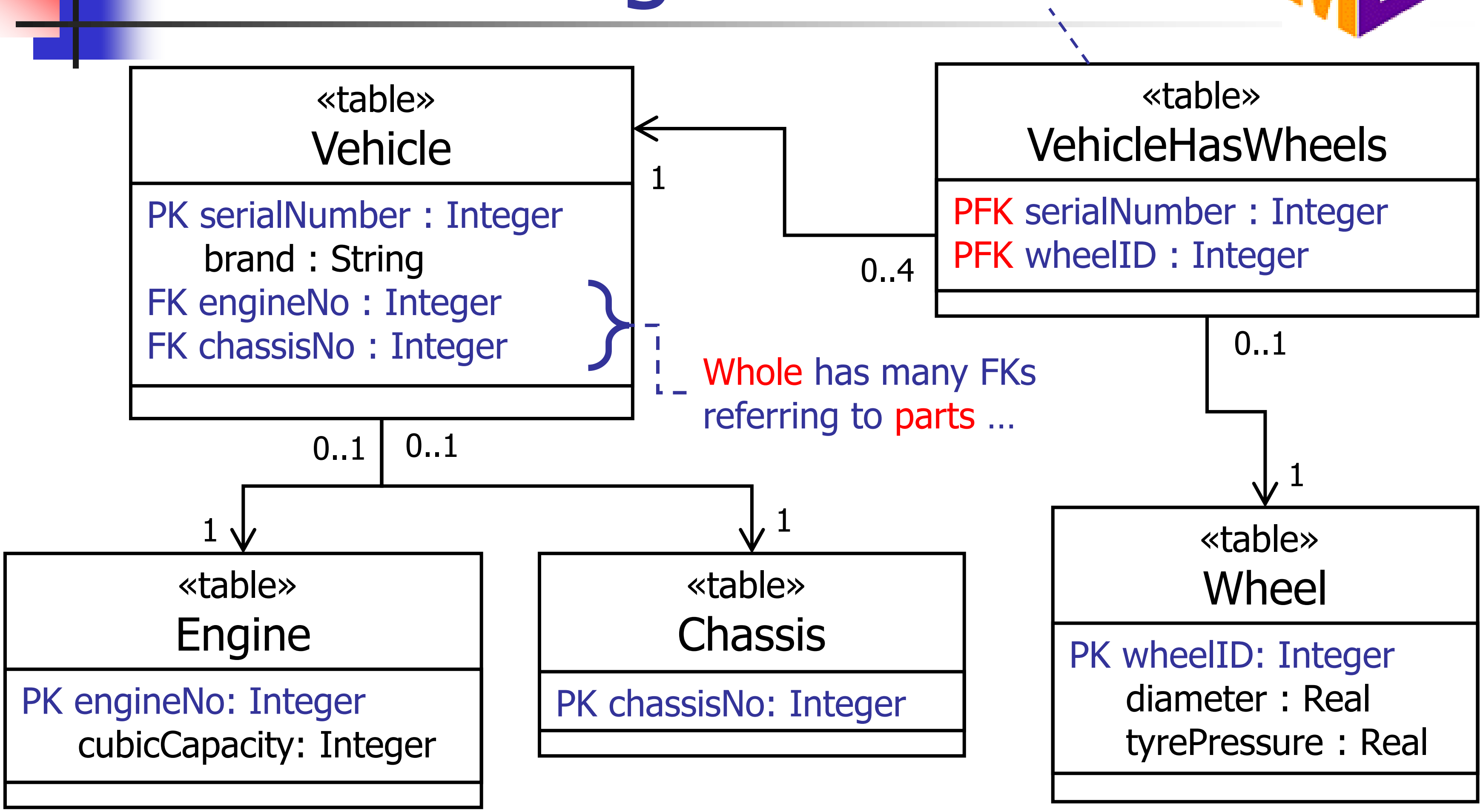


Table Design

...except where a linker is required



Traditional Normalization

- Start with arbitrary table, subdivide
  - start: data in one table, maybe with list-valued columns
  - next: break up lists by replicating atomic data in many rows
  - next: split up into many tables to remove replicated data
  - next: make sure nonkey attributes depend fully on the key
- Driven by functional dependency
  - eg: customer's name, age depend on which customer
  - viz: <u>custID</u> → name, <u>custID</u> → age, but not age → name
  - transitive, eg deptID in: <u>regNo</u> → degID → deptID
  - partial, eg <u>modID</u> → lectID in: R(lectID, <u>modID</u>, <u>regNo</u>)
  - multivalued, eg product set in: <u>company</u> →→ product

Normal Forms

- 1NF (Codd) – no lists or repeating data groups (atomicity rule) and find a primary key for each row
- 2NF (Codd) – and no nonkey attributes depend on part of a compound primary key (full functional dependence on the key)
- 3NF (Codd) – and no nonkey attributes depend on other nonkey attributes (don't depend transitively on the primary key)
  - BCNF (Boyce/Codd; Heath) – and every attribute (or compound) on which other attributes fully depend is also a candidate key
- 4NF (Fagin) – and no multivalued dependency anomalies (no dependent set of values depends on part of a compound key)
- 5NF (Fagin) – and every join dependency is implied by the candidate keys (project/join NF, rare case, hard to explain)

First Normal Form

Enrolment (non-normal)

regNo	degID	modIDs
0214537	G402	COM101, COM102, ACS101...
0773465	G650	COM101, COM103, EEE105, ...
1334890	G600	

non-atomic!
list-valued data!

...at the cost of replicating each regNo and degID

PK = regNo + degID + modID

Enrolment (1NF)

regNo	degID	modID
0214537	G402	COM101
0214537	G402	COM102
0214537	G402	ACS101
0773465	G650	COM101
0773465	G650	COM103

now atomic,
lists split up...

The University Of Sheffield.

Second Normal Form

Examinations (1NF)

regNo	modID	lectID	marks
0214537	COM101	GJB	72
0773465	COM101	GJB	65
1334890	COM103	SDN	67

PK = regNo, modID

while each mark depends on both regNo, modID, the lecturer depends only on modID! ...

...so, split into two tables

Teaches (2NF)

modID	lectID
COM101	GJB
COM103	SDN

regNo	modID	marks
0214537	COM101	72
0773465	COM101	65
1334890	COM103	67

Examinations (2NF)

Third Normal Form

Registration (2NF)

regNo	degID	deptID
0214537	G404	COM
0343662	G404	COM
1754532	H400	MEC

PK = regNo

while each degree taken depends directly on the student regNo, the home deptID depends on the degID, only indirectly on the PK!

Owner (3NF)

degID	deptID
G404	COM
H400	MEC

Registration (3NF)

regNo	degID
0214537	G404
0343662	G404
1754532	H400

...eliminate transitive dependencies in 3NF

Fourth Normal Form

Sells (3NF)

company	product	country
IBM	PC	France
IBM	Server	Italy
DEC	PC	France

PK = company, product, country

multivalued dependency: set of products and set of countries depend on company, but are independent of each other

...so decompose into two tables

Exports (4NF)

company	country
IBM	France
IBM	Italy
DEC	France

Makes (4NF)

company	product
IBM	PC
IBM	Server
DEC	PC

How Normal?

- Typical levels
  - aim for at least 3NF – good separation of data
  - 4NF needed where multivalued dependency exists
- What to check
  - insertion anomaly – can't insert X until Y is also supplied
  - update anomaly – affects some, not all rows (duplication)
  - deletion anomaly – delete X causes unintended loss of Y
- Denormalize?
  - to increase efficiency of searches (fewer joins)
  - to decrease fragmentation of data (fewer tables)

Lab 2: Normalize Orders

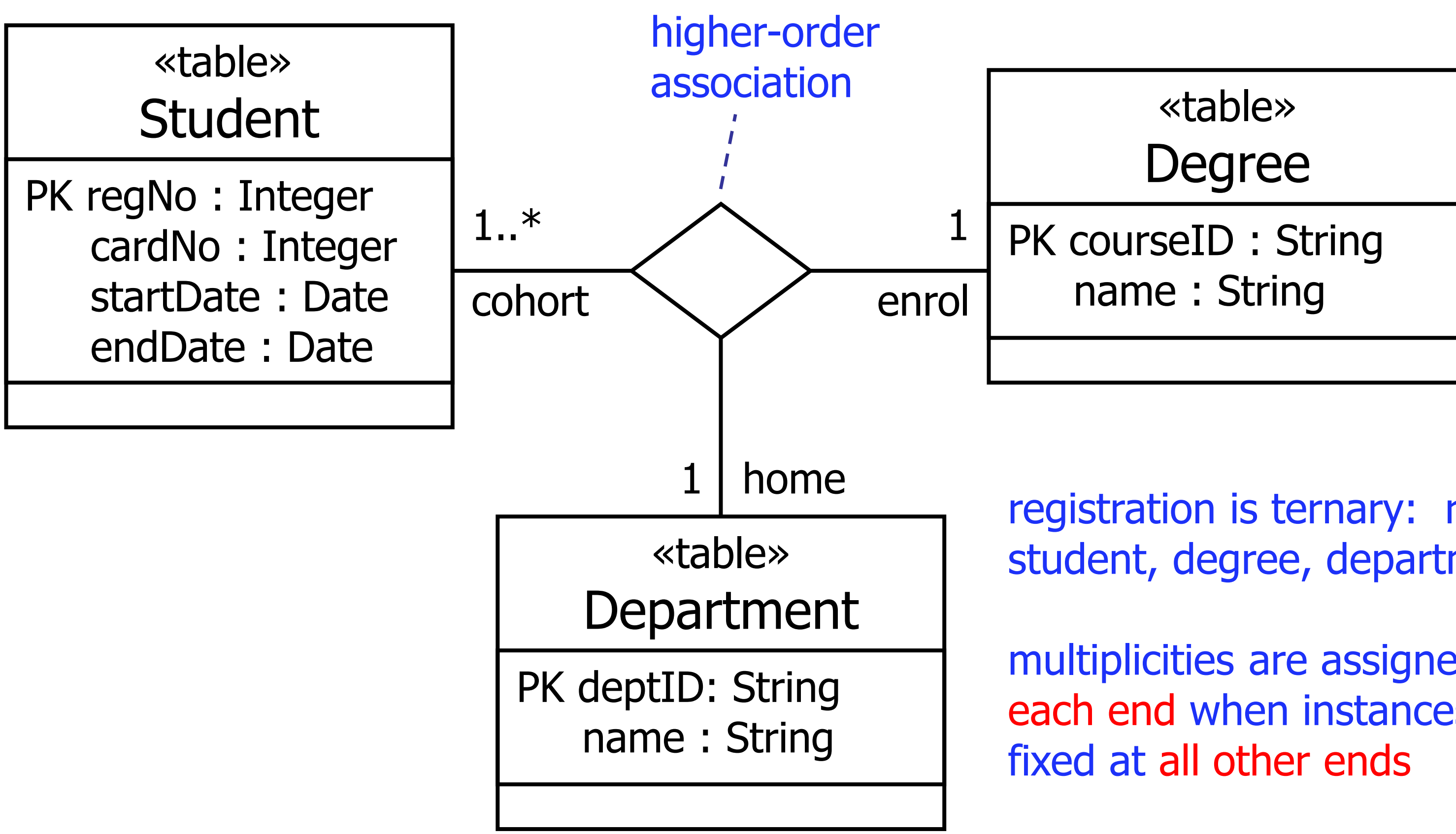
<u>orderID</u>	city	<u>productID</u>	quantity
S1	London	P1	100
S1	London	P2	200
S2	Paris	P1	200
S2	Paris	P2	300
S3	Paris	P2	300
S4	London	P2	400
S4	London	P4	500
S4	London	P5	700

Run a Poll

- First: work out the functional dependencies
- Second: split the tables, aiming for at least 3NF

The University Of Sheffield.

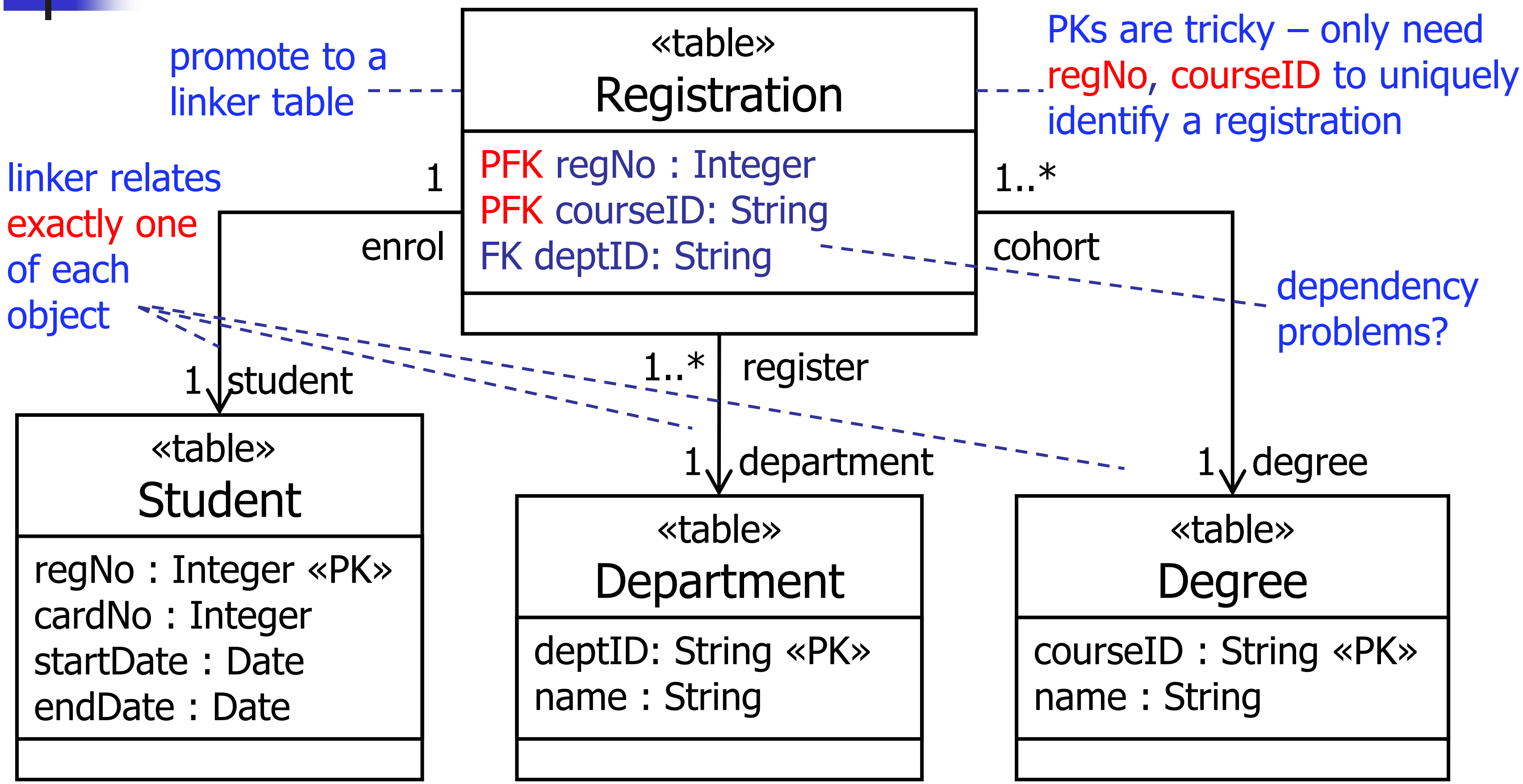
Ternary Association



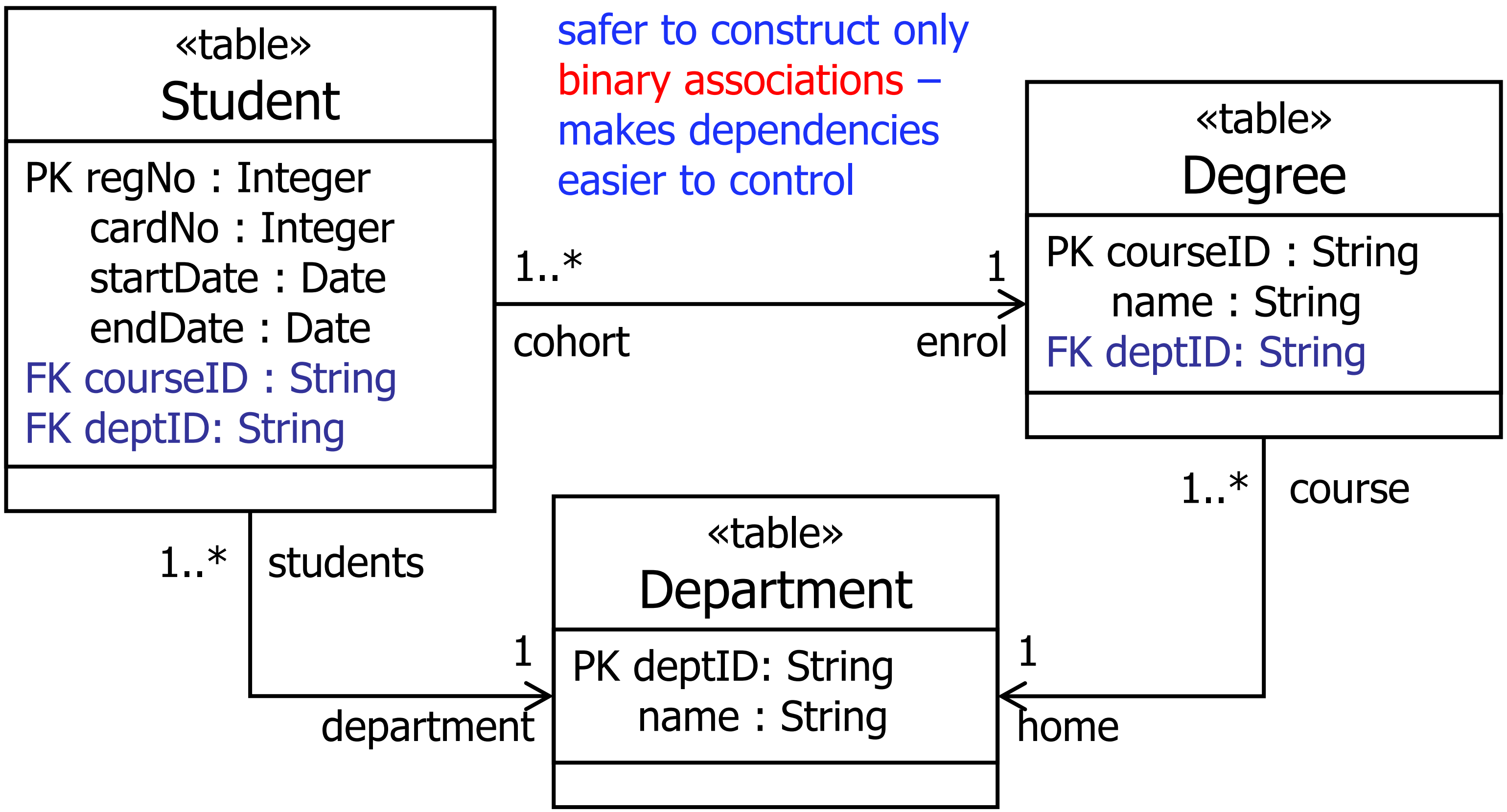
registration is ternary: relates student, degree, department

multiplicities are assigned to each end when instances are fixed at all other ends

Ternary Linker

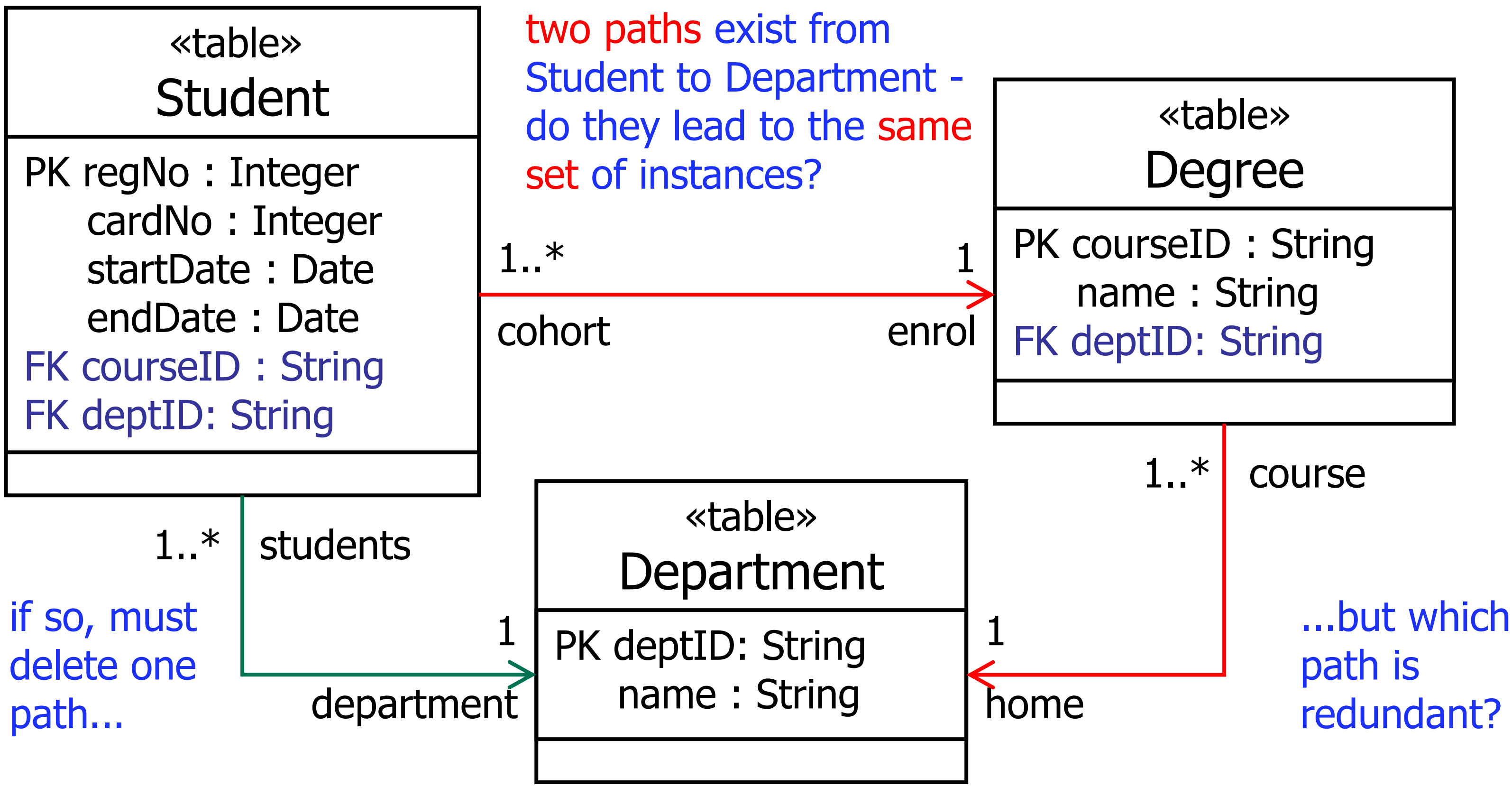


Make Binary

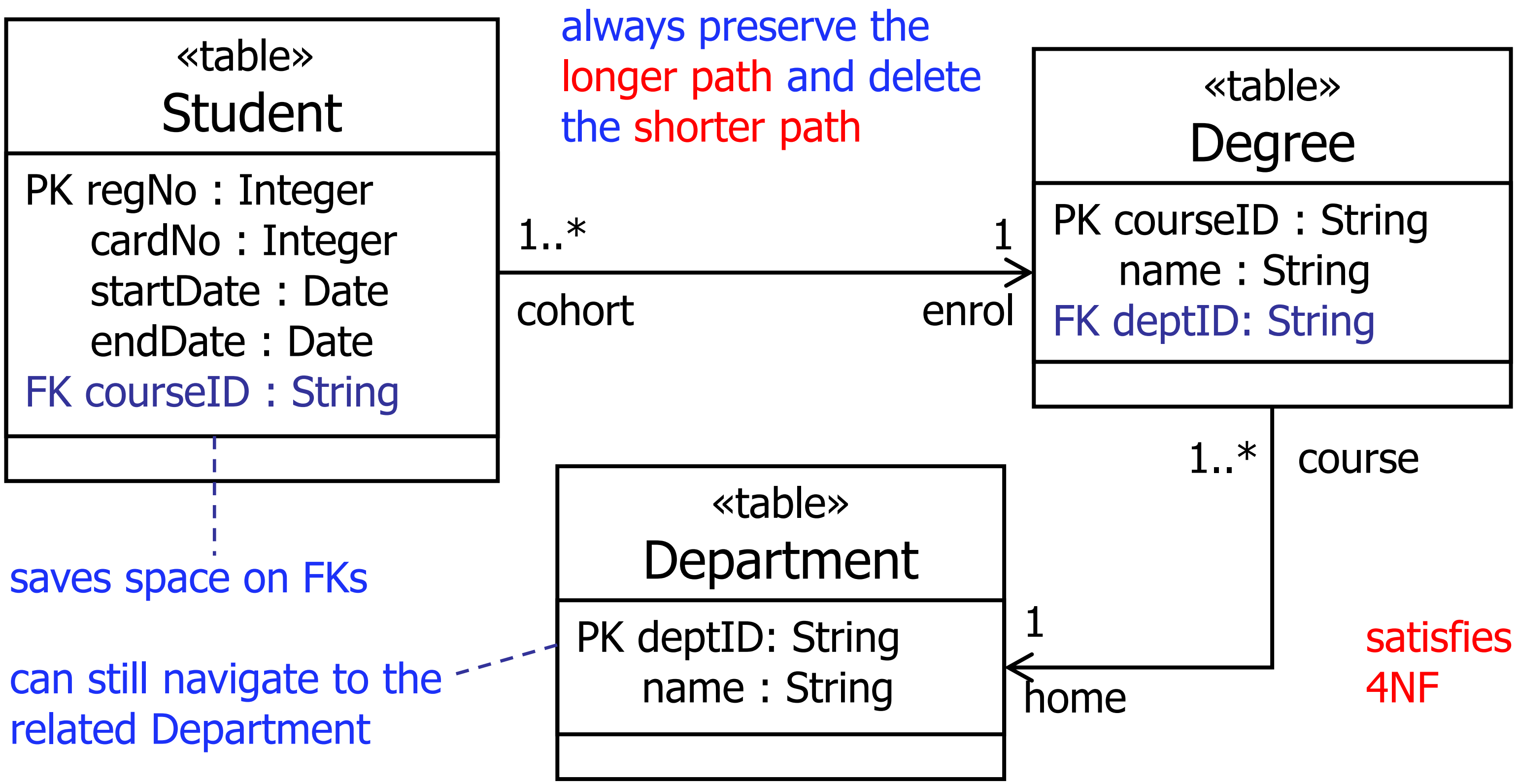


«table»
Student
PK regNo : Integer
cardNo : Integer
startDate : Date
endDate : Date
FK courseID : String
FK deptID: String
safer to construct only
binary associations –
makes dependencies
easier to control
1..*
1
cohort
enrol
«table»
Degree
PK courseID : String
name : String
FK deptID: String
1..*
course
1..*
students
«table»
Department
PK deptID: String
name : String
1
department
1
home

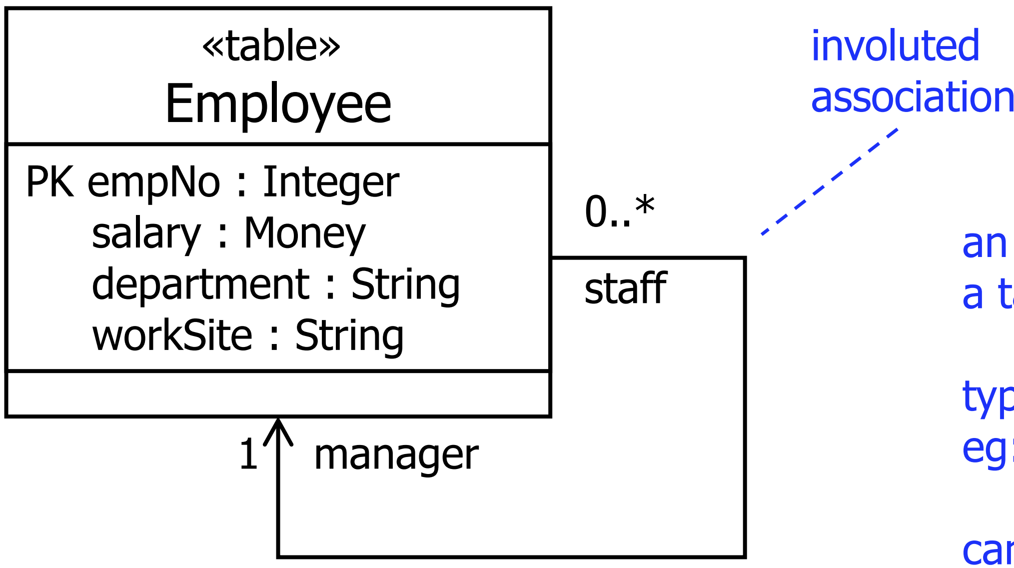
Redundant Paths



Redundancy Eliminated



Involuted Association



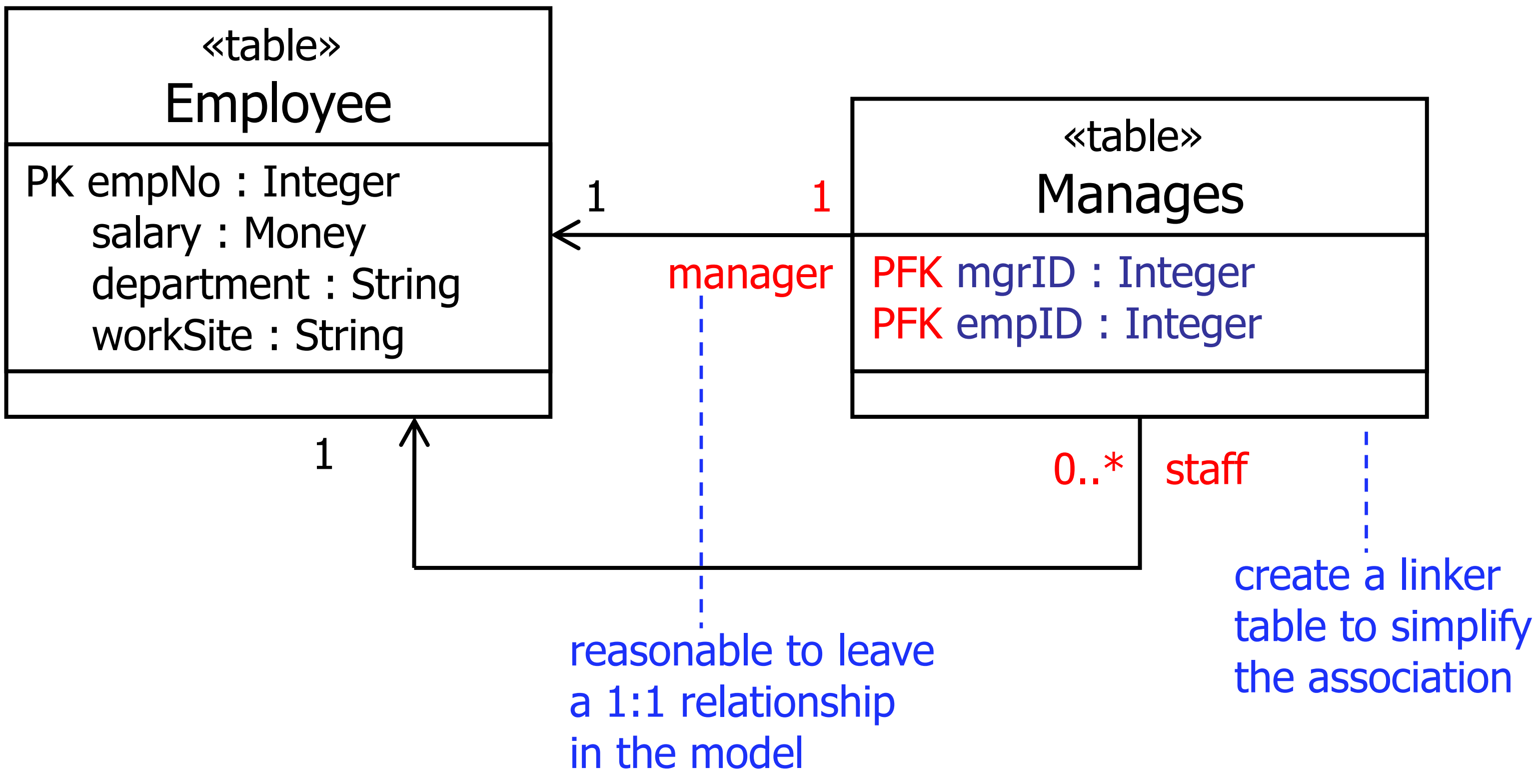
an involuted association relates a table back to itself

typical in human relationships, eg: marriage, employment

cannot easily link through FKs, since not all Employees are managed (null values)

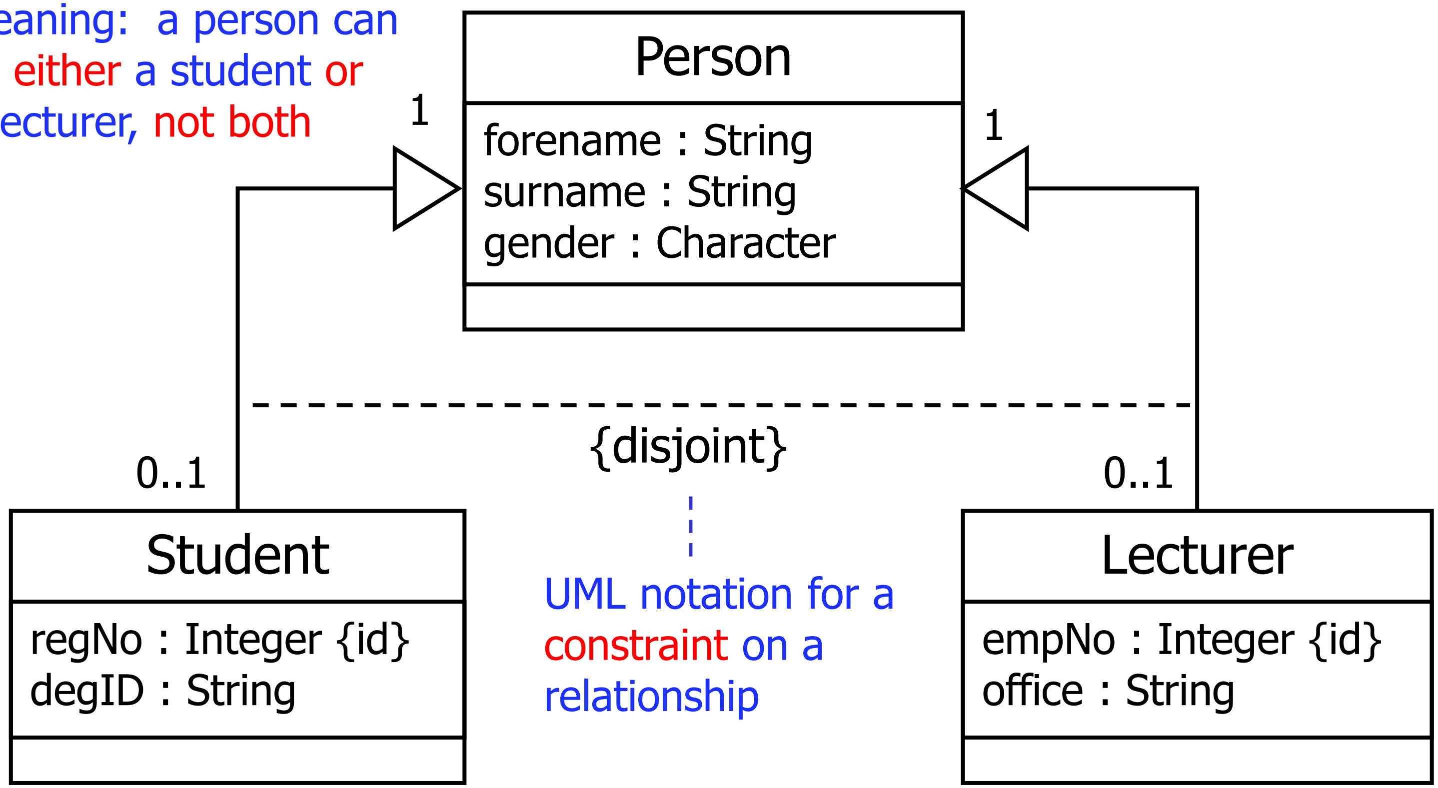
impossible for M:N anyway

Involuted Linker



Disjoint Semantics

meaning: a person can be either a student or a lecturer, not both



Partitioned Tables

if a person can be either a student or a lecturer, not both...

...requires only 2 tables in the DB (reduces joins)...

...but cannot represent a person who is both a student and a lecturer

use the natural candidate keys

«table» Student
PK regNo : Integer
degID : String
forename : String
surname : String
gender : Character

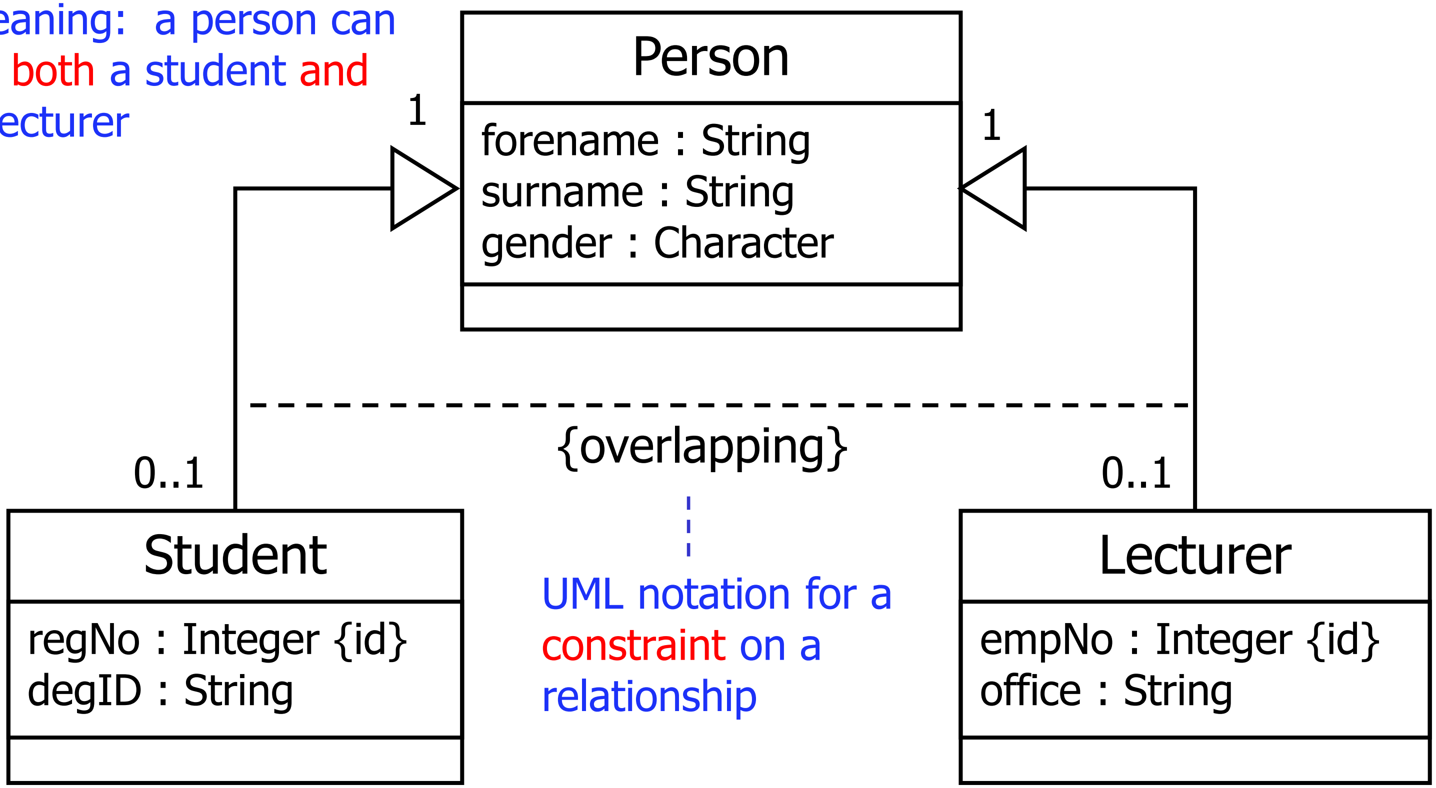
Person attributes replicated in each subclass table

satisfies 3NF, 4NF

«table» Lecturer
PK empNo : Integer
office : String
forename : String
surname : String
gender : Character

Overlapping Semantics

meaning: a person can be both a student and a lecturer



Denormalized Table

surrogate key for Person instances (generated key)

Student attributes may be null!

Lecturer attributes may be null!

«table» Person
PK personID
forename : String
surname : String
gender : Character
regNo : Integer
FK degID : String
empNo : Integer
office : String

folds all subclass attributes into the superclass

reduces joins between tables to reconstruct whole persons...

...but wasteful in space, because of many null fields

satisfies only 2NF due to transitive dependency on regNo, empNo

Denormalization?

- Reduces number of tables
  - eg: flatten generalisation structures into one table – but at the cost of wasted space (null values)
  - eg: flatten composition structures into one table – at the cost of replicating the part-data many times in the whole
  - saves reconstructing whole objects in memory from fragmented table data – fewer joins
- Optimizes access paths
  - eg: retain redundant access paths through the data – but at the cost of additional FKs, risk of inconsistency
  - saves time when searching – again, fewer joins

Summary

- Database systems have evolved from the network model, via the hierarchical model, to the relational model
- The relational model stores data in tables, whose rows denote objects and whose columns denote their simple attributes.
- Each row must be uniquely identifiable by a primary key
- Navigation is unidirectional, from the many to the one, using a foreign key to identify the related row in another table
- Data must be normalised to 3NF, 4NF to support insert, update and delete without duplication, or cascading effects
- Entity-relationship modelling is one approach to normalising data, based on optimising multiplicities
- Traditional table normalisation is another approach based on analysing functional dependency