Database Fundamentals

UML TRANSLATION

UML Translation Process

- UML can often be translated automatically into relational schemas
  - MS-SQL Diagrams feature
  - DBDesigner Fork
  - ArgoUML
- The next lot of slides will show how this can be done manually following some basic rules

UML Translation Process

1. **Classes**
   - Every UML Class becomes its own relation of the same name
   - The Primary Key (PK) becomes the relations PK

Relational Schema1:
Student(StudentID, EmailID, StudentName)
PK = (StudentID)
CK = (EmailID)

Relational Schema2:
Course(CourseID, CourseName)
PK = (CourseID)

UML Translation Process

2. Associations

- **One : One Relationship** (mandatory)
- Associations can be represented by a new relation that contains the PK from **both sides of the association**

Description	
Class1 PK	Class 2 PK

NOT ALWAYS Necessary

Relational Schema:
Class1(Attribute1, Attribute2)
PK(Attribute1)

Class2(Attribute1, Attribute2)
PK(Attribute1)

New Relation Schema:
Description(Class1Attr1, Class2Attr1)
PK (Class1Attr1, Class2Attr2)
FK(Class1Attr1) ~> Class1(Attribute1)
FK(Class2Attr1) ~> Class2(Attribute1)

UML Translation Process

2. Associations

- **One : One Relationship** (mandatory)

Relational Schema:
Class1(Attribute1, Attribute2, Class2Attr1)
PK=(Attribute1)
FK(Class2Attr1) → Class2(Attribute1)

Class2(Attribute1, Attribute2)

Alternatively, store the PK of one of the classes as a FK in the other. Choose it such that empty values are avoided if the relationship is optional

UML Translation Process

2. Associations

- **One : One Relationship** (optional)

Customers

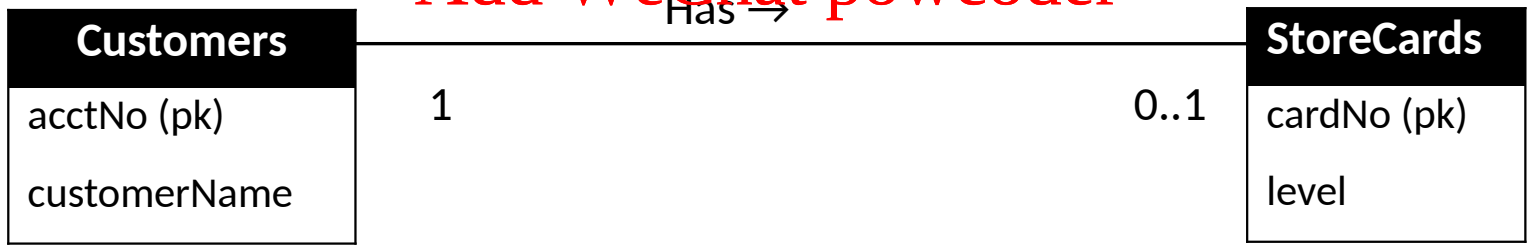
acctNo	custName
001	Bruce
002	Wayne
003	Mitch
004	Melissa

CustomerStoreCard

001	NULL
002	A1
003	A2
004	A3

StoreCards

cardNo	level
A1	1
A2	1
A3	4



Schema Option1:

Schema Option2:

Note: this is from the perspective of a single retailer

UML Translation Process

2. Associations

- **One : One Relationship** (optional)

Customers

acctNo	custName
001	Bruce
002	Wayne
003	Mitch
004	Melissa

CustomerStoreCard

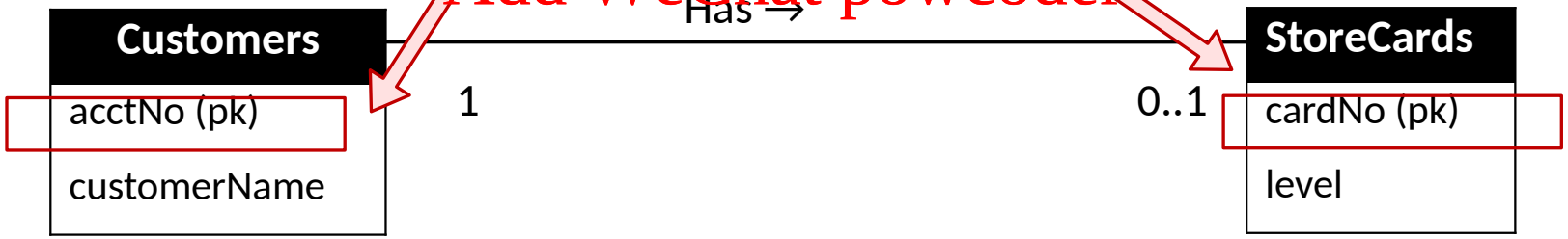
001	NULL
002	A1
003	A2
004	A3

Same PK

Same PK

StoreCards

cardNo	level
A1	1
A2	1
A3	4



Schema Option1:

Schema Option2:

Note: this is from the perspective of a single retailer

UML Translation Process

2. Associations

- **One : One Relationship** (optional)

Customers

acctNo	custName
001	Bruce
002	Wayne
003	Mitch
004	Melissa

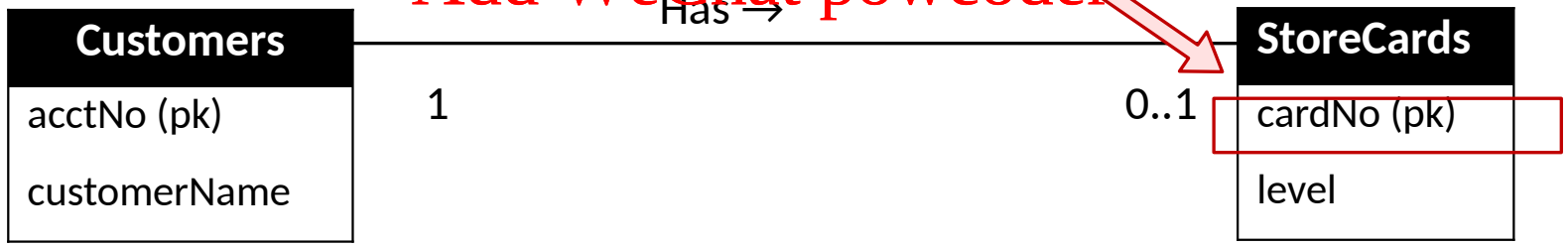
CustomerStoreCard

001	NULL
002	A1
003	A2
004	A3

Same PK

StoreCards

cardNo	level
A1	1
A2	1
A3	4



Schema Option1:

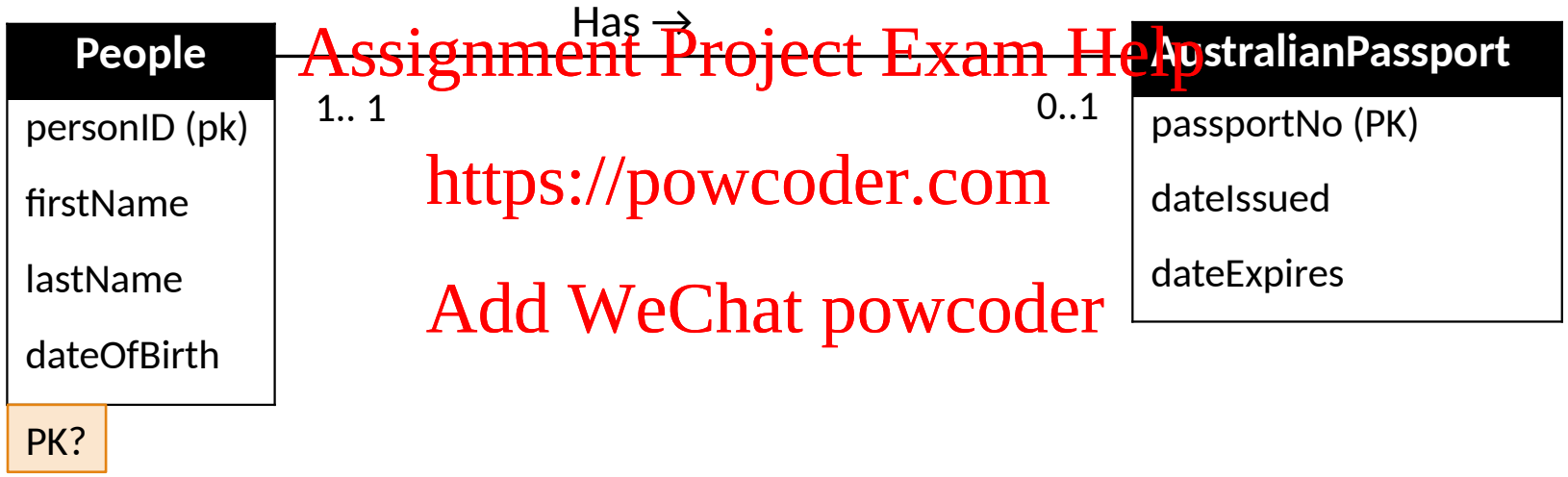
Schema Option2:

Note: this is from the perspective of a single retailer

UML Translation Process

2. Associations

- **One : One Relationship** (optional)



Schema Option1:

Schema Option2:

Note: this is from the perspective of Australian citizens holding standard Australian Passports

UML Translation Process

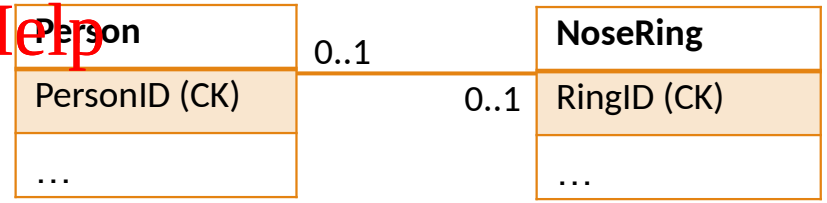
- Rule 1
  - In a optional 0..1 : 0..1 relationship or a mandatory 1:1 relationship, copy the key of **ONE** side and store it in the **OTHER** as a **FOREIGN KEY**
    - There is no need to create a new relation. **Minimise** storage of **empty** values.

Logical Schemas

Person(<u>PersonID</u>, ...,RingID)
PK(PersonID)
FK(RingID) ~> Ring (RingID)

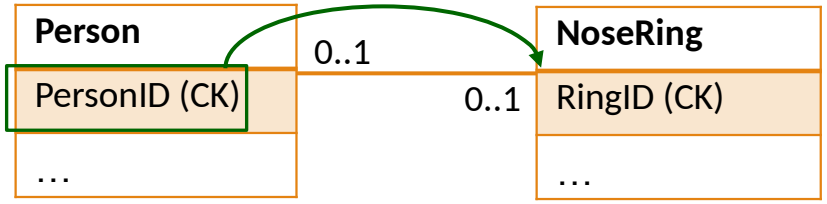
?

NoseRing(<u>RingID</u>, ...)
PK(RingID)



Person(<u>PersonID</u>)
PK(PersonID)

NoseRing(<u>RingID</u>, ..., PersonID)
PK(RingID)
FK(PersonID) ~> Person(PersonID)



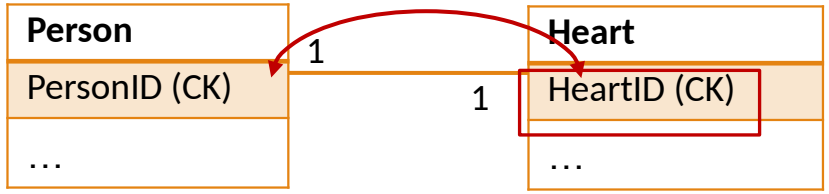
UML Translation Process

- Rule 1
  - In a optional 0..1 : 0..1 relationship or a mandatory 1:1 relationship, copy the key of **ONE** side and store it in the **OTHER** as a **FOREIGN KEY**
    - There is no need to create a new relation. **Minimise** storage of **empty** values.

Logical Schemas

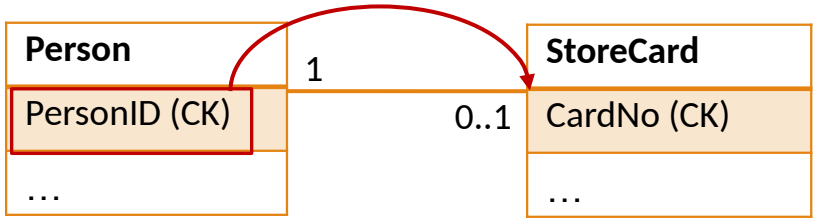
Person(PersonID, ..., HeartID)
PK(PersonID)
FK(HeartID) ~> Heart(HeartID)

Heart(HeartID, ...)
PK(HeartID)



Person(PersonID, ...)
PK(PersonID)

StoreCard(CardNo, ..., PersonID)
PK(CardNo)
FK(PersonID) ~> Person (PersonID)



UML Translation Process

2. Associations

- **One : Many Relationships**
- Does not require an additional relation
- Instead take the PK from the **one side** and apply it to the **many side** as a **FOREIGN KEY**.

Relational Schema:
Class1(Attr1, Attr2)
PK=(Attribute1)

Class2(Class2Attr1, Class2Attr2, Class1Attr1)
PK=(Class2Attr1)
FK(Class1Attr1) ~> Class1(Attr1)

UML Data Modelling – Basic Concepts

- Multiplicity
  - **One-to-Many (1:m) and Many-to-One (m:1)**
    - Many elements of one object are related to at most one of the other object
    - 0..1 → 1..*

Nerd
01
02
03

NerdyToys		
01	T1	(Spock)
01	T4	(Data)
02	T2	(Captain Picard)
03	T3	(Spock)

StarTrekToys	
T1	Spock
T2	Captain Picard
T3	Spock
T4	Data

Nerds
NerdID (PK)
FirstName
LastName
DateOfBirth
CanDoVulcanSalute

Collects →

1.. 1

1..*

StarTrekToys
ToyID (PK)
Character
BoxOpened
Condition

UML Data Modelling – Basic Concepts

- Multiplicity
  - **One-to-Many (1:m) and Many-to-One (m:1)**
    - Many elements of one object are related to at most one of the other object
    - 0..1 → 1..*

Nerd
01
02
03

NerdyToys

01	T1	(Spock)
01	T4	(Data)
02	T2	(Capt. n Picar
03	T3	(Spock)

Duplicate

Same Primary Key

StarTrekToys	
T1	Spock
T2	Captain Picard
T3	Spock
T4	Data

Nerds
NerdID (PK)
FirstName
LastName
DateOfBirth
CanDoVulcanSalute

Collects →

1.. 1

1..*

StarTrekToys
ToyID (PK)
Character
BoxOpened
Condition

If we have 0..1 or 1..1 on One side of the relation, then the other side will be the key

UML Translation Process

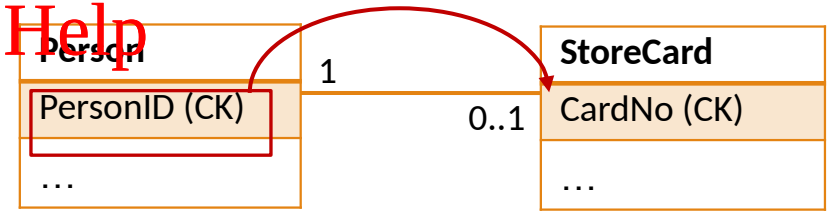
- Rule 2
  - If we have 0..1 or 1..1 one on **only one side** of the relationship, then the other side is the **key** of the new relation
  - This means the new relation is not necessary, instead, create a copy of the key from the one side and store it in the other relation as a **FOREIGN KEY**:

Logical Schemas

Person(PersonID, ...)
PK(PersonID)

+

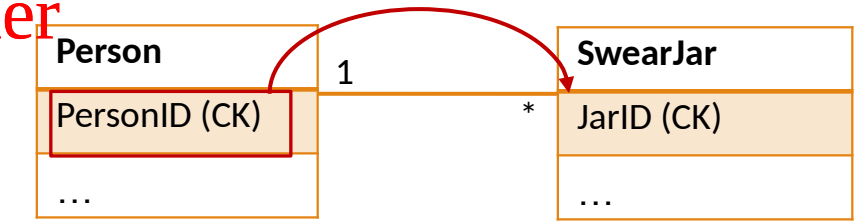
StoreCard(CardNo, ..., PersonID)
PK(CardNo)
FK(PersonID) ~> Person (PersonID)



Person(PersonID, ...)
PK(PersonID)

+

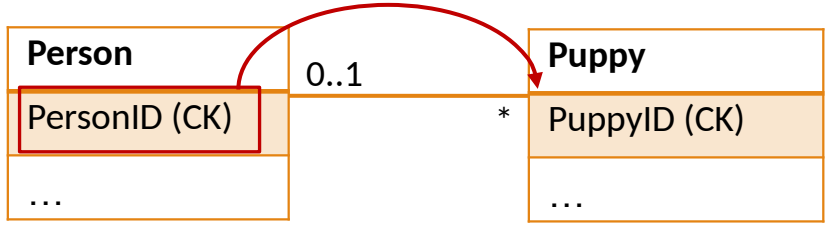
SwearJar(JarID, ..., PersonID)
PK(JarID)
FK(PersonID) ~> Person (PersonID)



Person(PersonID, ...)
PK(PersonID)

+

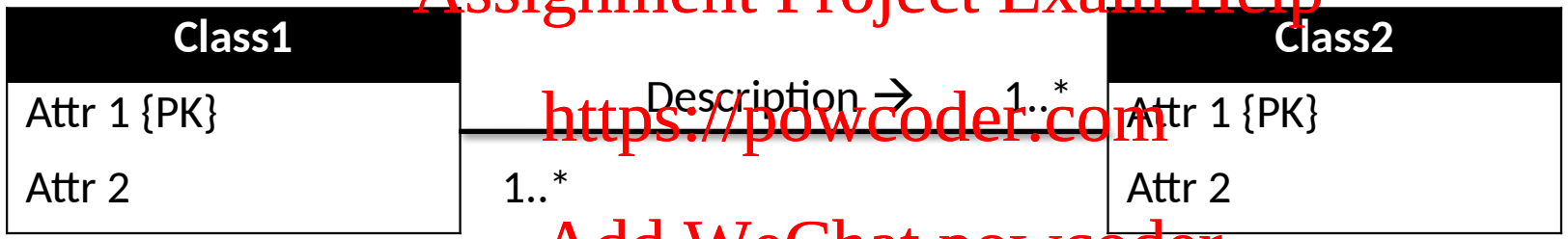
Puppy(PuppyID, ..., PersonID)
PK(PuppyID)
FK(PersonID) ~> Person (PersonID)



UML Translation Process

2. Associations

- **many: many Relationship**
- Results in a **new relation** that contains the key of both classes as the PK



Relational Schema:
Class1(Attr1, Attr2)

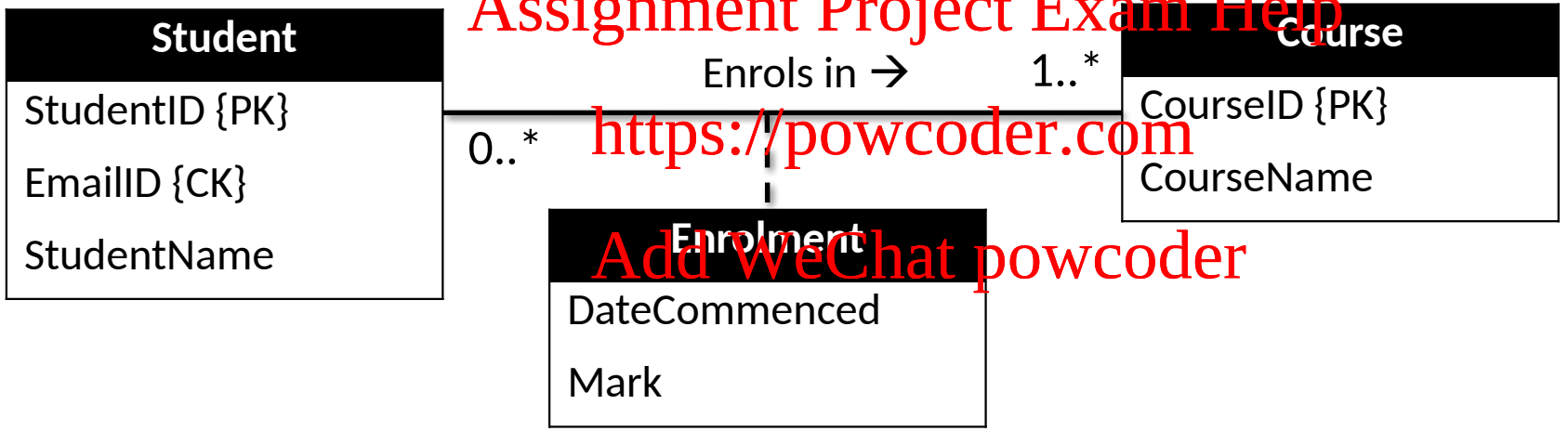
Class2(Attr1, Attr2)

Description(Class1Attr1, Class2Attr1)

UML Translation Process

3. Association Classes

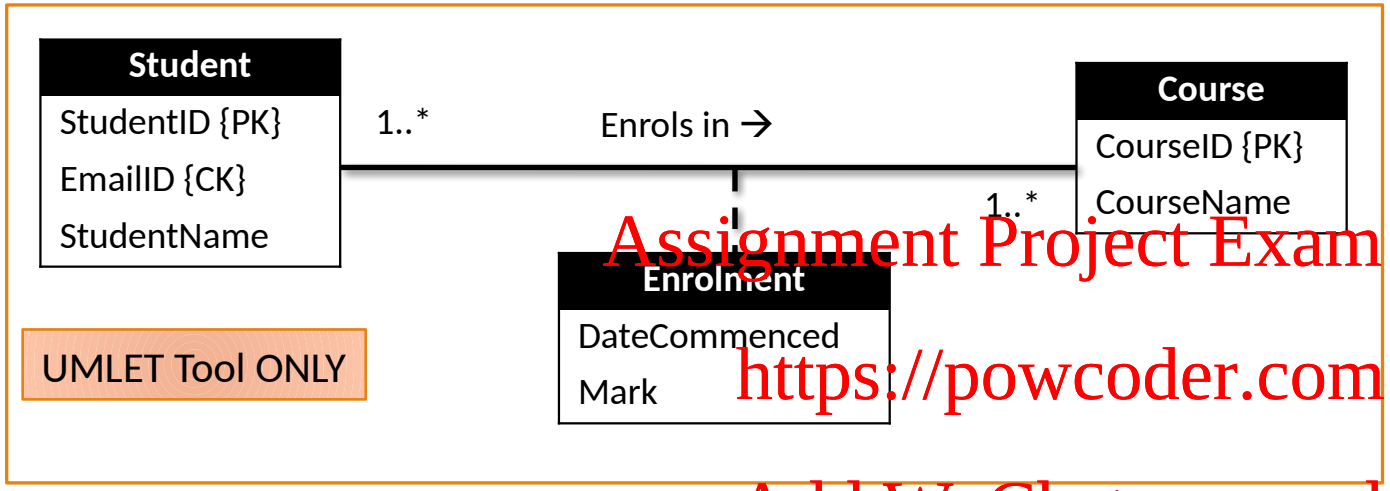
- Results is a new relation containing the **PK** from both sides of the association
- Any Association Class attributes are added to the new relation



Relational Schema:
Enrolment(StudentID, CourseID, DateCommenced, Mark)
PK(StudentID, CourseID)
FK(StudentID) ~> Student(StudentID)
FK(CourseID) ~> Course(CourseID)

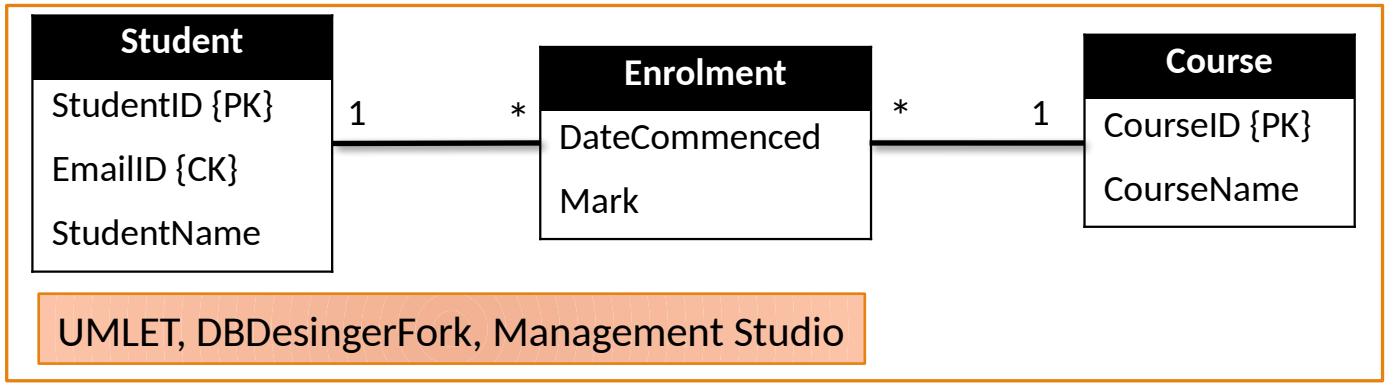
UML Data Modelling – Basic Concepts

- Association Classes vs 1 : many and many :1



Captures that there **WAS** at least **ONE** an interaction

Enrolment(StudentID, CourseID, DateCommenced, Mark)
PK(StudentID, CourseID)
FK(StudentID) ~> Student(StudentID)
FK(CourseID) ~> Course(CourseID)



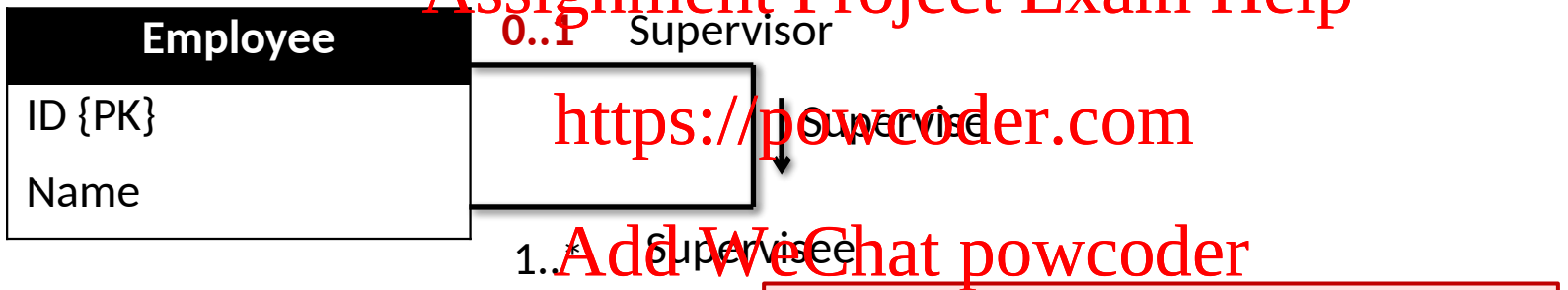
Captures **EVERY** interaction

Enrolment(StudentID, CourseID, DateCommenced, Mark)
PK(StudentID, CourseID, DateCommencd)
FK(StudentID) ~> Student(StudentID)
FK(CourseID) ~> Course(CourseID)

UML Translation Process

4. Recursive/Self Associations

- A Class that associates with itself
  - Results in a relation that contains the key of both classes as the PK
  - Any association class attributes get added to the new relation



Relational Schema:
Employee(ID, Name)
PK = ID

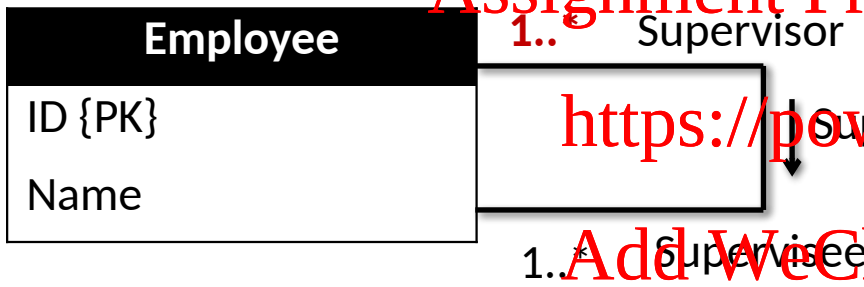
Relational Schema:
Employee(ID, Name, SupervisorID)
PK = ID
FK(SupervisorID) ~> Employee(ID)

Or you can place the EmployeeID and SupervisorID into their own relation

UML Translation Process

4. Recursive/Self Associations

- A Class that associates with itself
  - Results in a relation that contains the key of both classes as the PK
  - Any association class attributes get added to the new relation



Relational Schema:
Employee(ID, Name)
PK = ID

Relational Schema:
Employee(ID, Name)
PK = ID

Supervision(SupervisorID, SuperviseeID)
PK = (SupervisorID, SuperviseeID)
FK (SupervisorID) ~> Employee(ID)
FK (SuperviseeID) ~> Employee(ID)

UML Translation Process

- Classes – Structured (multi-value) Domains
  - An attribute that consists of sub-components
  - Violates basic relational theory (single value attributes)

Student
StudentID {PK}
EmailID {CK}
StudentName
Address Street Suburb Postcode

Approach 1:
Student(StudentID, EmailID, StudentName, Address:String)
PK = (StudentID)

1NF ??

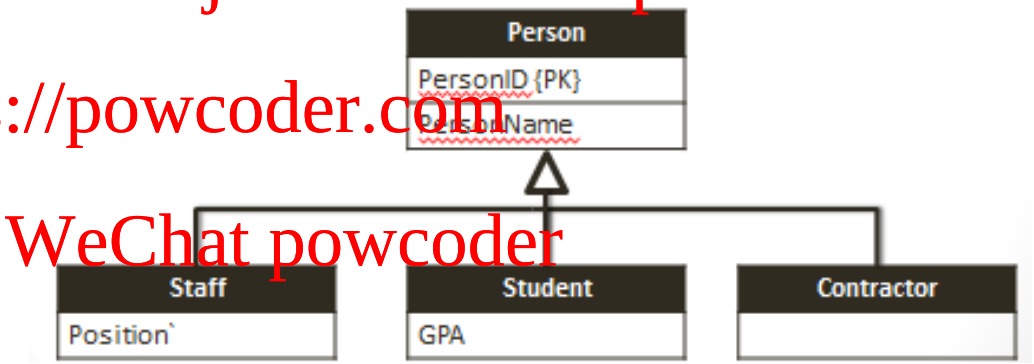
Approach 2:
Student(StudentID, EmailID, StudentName, Street, Suburb, Postcode)
PK = (StudentID)

Approach 3:
Addresses(AddressID, Street, Suburb, Postcode)
PK = (AddressID)

Student(StudentID, EmailID, StudentName, AddressID)
PK = (StudentID)
FK(AddressID) ~> Address(AddressID)

UML Translation Process

- Inheritance/Sub Classes – **Vertical Inheritance**
  - Each class is translated into its **own relation** with the PK of the parent
  - The PK of each relation is the PK of the parent class
    - Suitable for **Disjoint** (OR) where every person is either **one of** a staff member, student or contractor

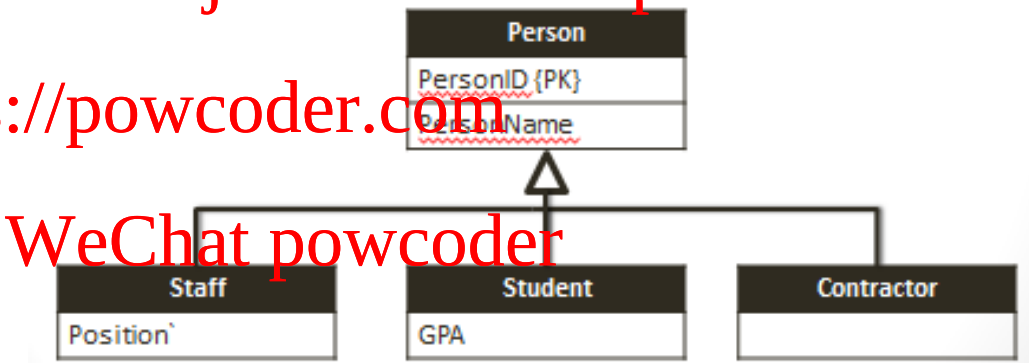


Relational Schema Variation 1:
Person(PersonID, PersonName)

Staff(PersonID, Position)
Student(PersonID, GPA)
Contractor(PersonID)
In each new relation FK(PersonID) ~> Person(PersonID)

UML Translation Process

- Inheritance/Sub Classes – **Horizontal Inheritance**
  - Each class is translated into its own relation with all attributes of the parent
  - The PK of each relation is the PK of the parent class
    - Ok for **Mandatory Disjoint** Participation where Every person is either a student or a staff member
    - Saves re-combining records across tables



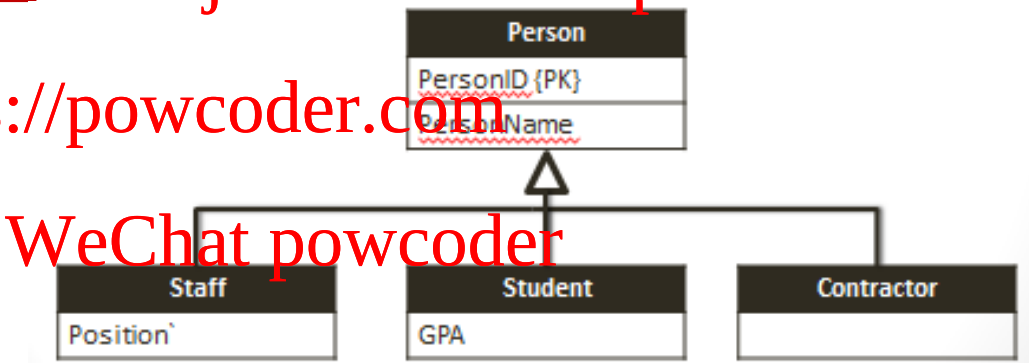
Relational Schema Variation 2:

~~Person(PersonID, PersonName)~~

Staff(PersonID, PersonName, Position)
Student(PersonID, PersonName, GPA)
Contractor(PersonID, PersonName)

UML Translation Process

- Inheritance/Sub Classes – **Horizontal Inheritance Variation**
  - Create a complex relation consisting of all attributes
  - The PK of each relation is the PK of the parent class
    - Suitable for **Non-Disjoint** (AND) – overlapping relationships
    - ~~Every person is represented by at least two of the sub-classes~~

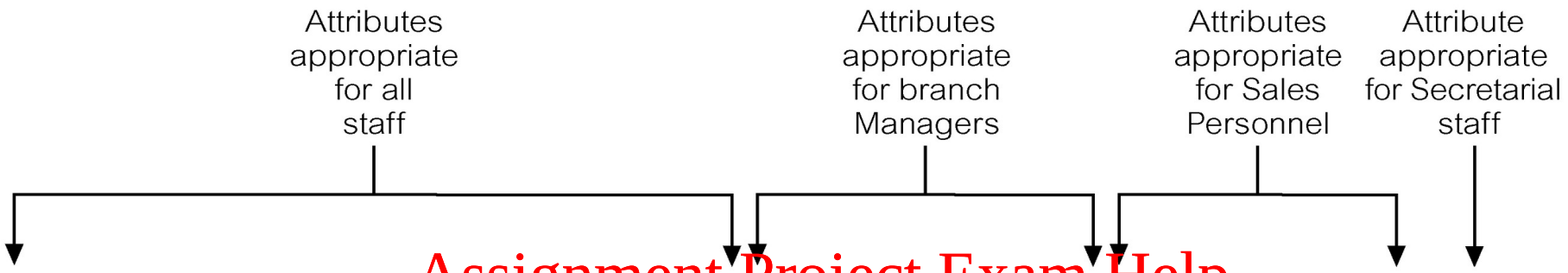


Relational Schema Variation 3:

Person(PersonID, PersonName, Position, GPA, category)

What about our contractors???

Inheritance in One Relation – Poor solution



staffNo	name	position	salary	mgrStartDate	bonus	sales Area	car Allowance	typing Speed
SL21	John White	Manager	30000	01/02/95	2000			
SG37	Ann Beech	Assistant	12000					
SG66	Mary Martinez	Sales Manager	27000			SA1A	5000	
SA9	Mary Howe	Assistant	9000					
SL89	Stuart Stern	Secretary	8500					100
SL31	Robert Chin	Snr Sales Asst	17000			SA2B	3700	
SG5	Susan Brand	Manager	24000	01/06/91	2350			

Using one table (horizontal inheritance) can introduce too many empty (NULL) values.

UML Translation Process

- What about 1:many with an association class. . .
  - Pretty rare but possible!
  - What would be the **best** PK of the association class **?**

Customers

custID	custName
001	Bruce
002	Wayne
003	Mitch
004	Melissa

custID	addID	dateFrom
001	A001	1/1/2000
002	A002	1/1/2005
002	A003	1/1/2015

Addresses

addID	address
A001	…
A002	…
A003	…