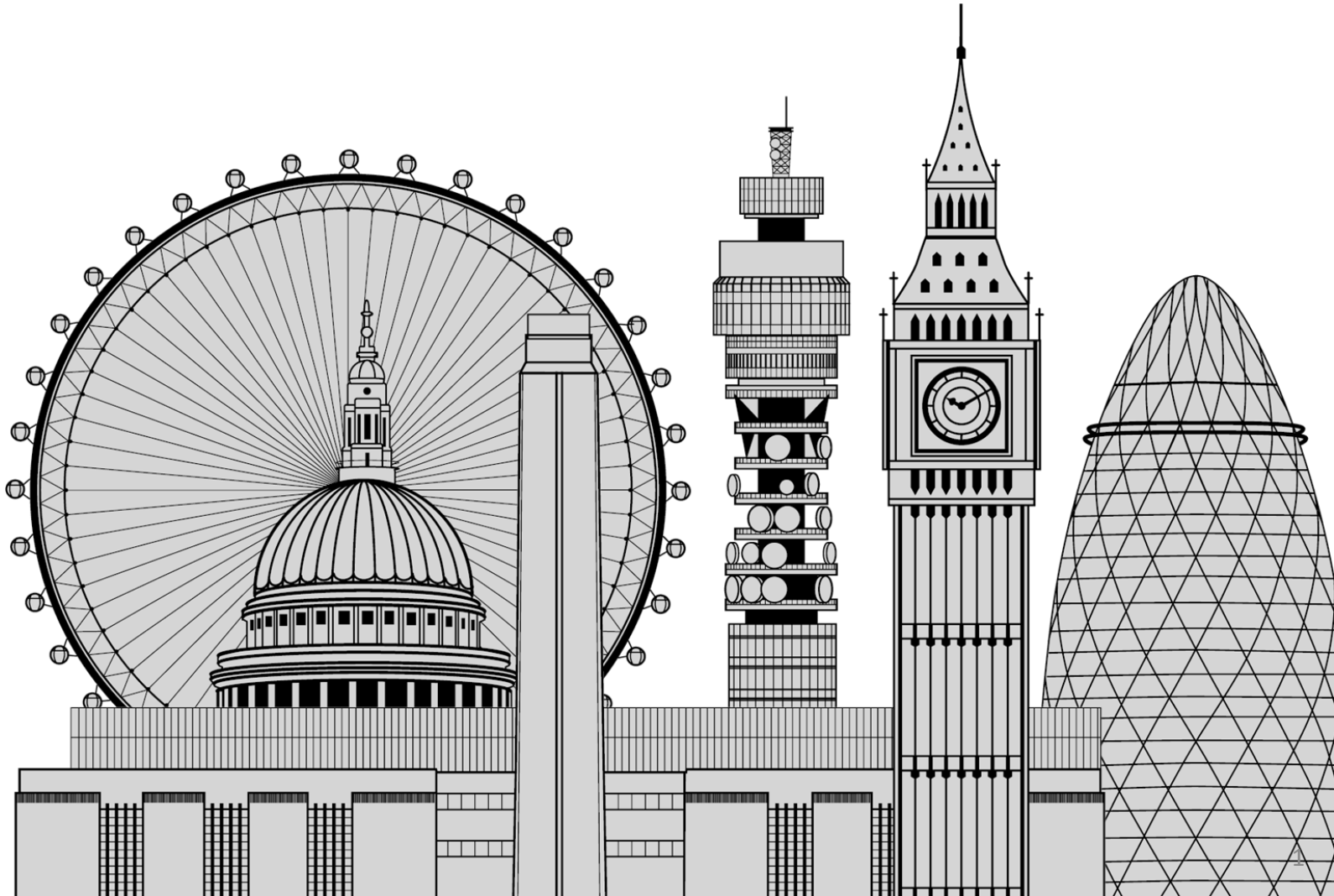
5COSC020W DATABASE SYSTEMS – LECTURE 03

Logical Database Design – Mapping to a Logical Entity-Relationship Model

UNIVERSITY OF WESTMINSTER

Dr Francois ROUBERT
F.Roubert@westminster.ac.uk

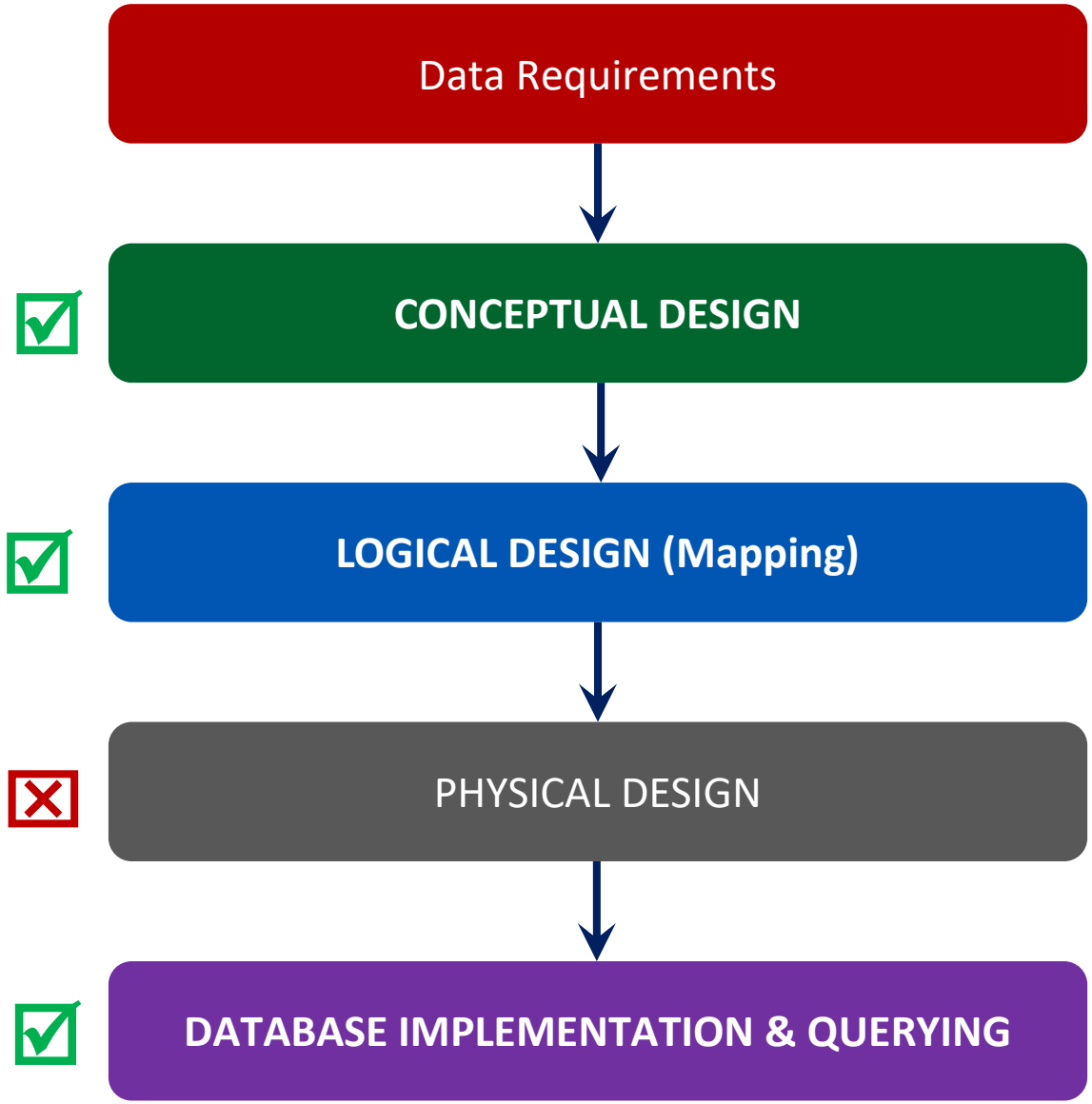


Lecture 03 – Outline

- **Relational Model**
  - Relations to represent data: tables with records and fields.
  - Relating table with relational keys: primary keys and foreign keys.
  - Entity Integrity and Referential Integrity.

- **Logical Design for Relational Databases**
  - Mapping Conceptual to Logical Entity-Relationship Model.
  - Mapping one-to-many relationships.
  - Mapping one-to-one relationships.
  - Mapping many-to-many relationships.
  - Mapping complex relationships: ternary and quaternary.
  - Mapping specialisations.

Phases and outputs of Database Design (recap)



BUSINESS USER'S DATA NEEDS
to support business processes

CONCEPTUAL DATA MODEL OR CONCEPTUAL SCHEMA
of the data required by business users

LOGICAL DATA MODEL OR LOGICAL SCHEMA
based on specific file data organisation (e.g. relational model)

PHYSICAL OR INTERNAL SCHEMA
storage structures, file organizations, indexes, access paths

DATABASE IMPLEMENTED & QUERIED IN DBMS
data structures, constraints, data values, data description

Classifying DBMSs based on Data Models (recap)

- **Relational Model or SQL Model** (covered in lectures 01-07)
  - Structured Data as collection of tables with fields, records, PKs and FKs.
  - Uses high level Structured Query Language (SQL) for CRUD operations.

- **XML Model** (covered in lectures 08-10)
  - Semistructured data as elements that can be nested to create tree structures.
  - Standard for structuring and exchanging data over the Web.

- **NoSQL Model** (covered in lectures 11)
  - Key-value data model: each data value associated to unique key for fast retrieval.
  - Graph data model: data represented as graphs i.e. labelled nodes and edges.
  - Document data model: data as collection of documents in JSON or XML.

- **Object Model** (not covered in this module)
  - Data as collection of objects (in classes) with properties & operations.

Relational Model – Starting with an example

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

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

Relational Model – Terminology

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Relational Model

Data stored as collection of relations (tables) interconnected with PKs and FKs that follow a set of Integrity Constraints.

- **Relation**
  - Table with columns and rows.
  - Only applies to logical structure of the DB, not conceptual or physical.

- **Attribute**
  - Named column of a relation that contains values in the domain of the attribute.
  - Domain: the set of allowable values for one or more attributes.
  - Degree: number of attributes in a relation.

- **Tuple**
  - Row in a relation that contains the data values of a single record.
  - Cardinality: number of tuples in a relation

Essential properties of relations

- **Relational Schema**:
  - Collection of named relations defined by a set of attributes & domain name pair.

- **Relation**
  - Relation name is distinct from all other relation names in relational schema.
  - Each cell of relation contains exactly one atomic (single) value.

- **Attribute**
  - Each attribute has a distinct name.
  - Values of an attribute are from the same domain.
  - Order of attributes has no significance.

- **Tuple**
  - Each tuple is distinct; no duplicate tuples.
  - Order of tuples has no significance, theoretically.

Relational Keys

- **Superkey**
  - Set of attributes that uniquely identifies a tuple within a relation.
- **Candidate Key**
  - Minimal set of attributes that uniquely identifies each tuple within a relation.
- **PRIMARY KEY (PK)**
  - Selected Candidate Key: unique and irreducible.
  - Selected minimal set of attributes that uniquely identifies each tuple within relation.
- **Alternate Key (AK)**
  - Not selected Candidate Key.
  - Not selected minimal set of attributes that uniquely identifies each tuple within a relation.
- **FOREIGN KEY (FK)**
  - Attribute, or set of attributes, within one relation that matches candidate key (most often the Primary Key) of another relation (possibly same).

Integrity Constraints

- **Null values**
  - Value for an attribute that is currently unknown or not applicable for tuple.
  - Absence of a value, not the same as zeros or spaces, which are actual values.
  - Deals with incomplete or exceptional data.

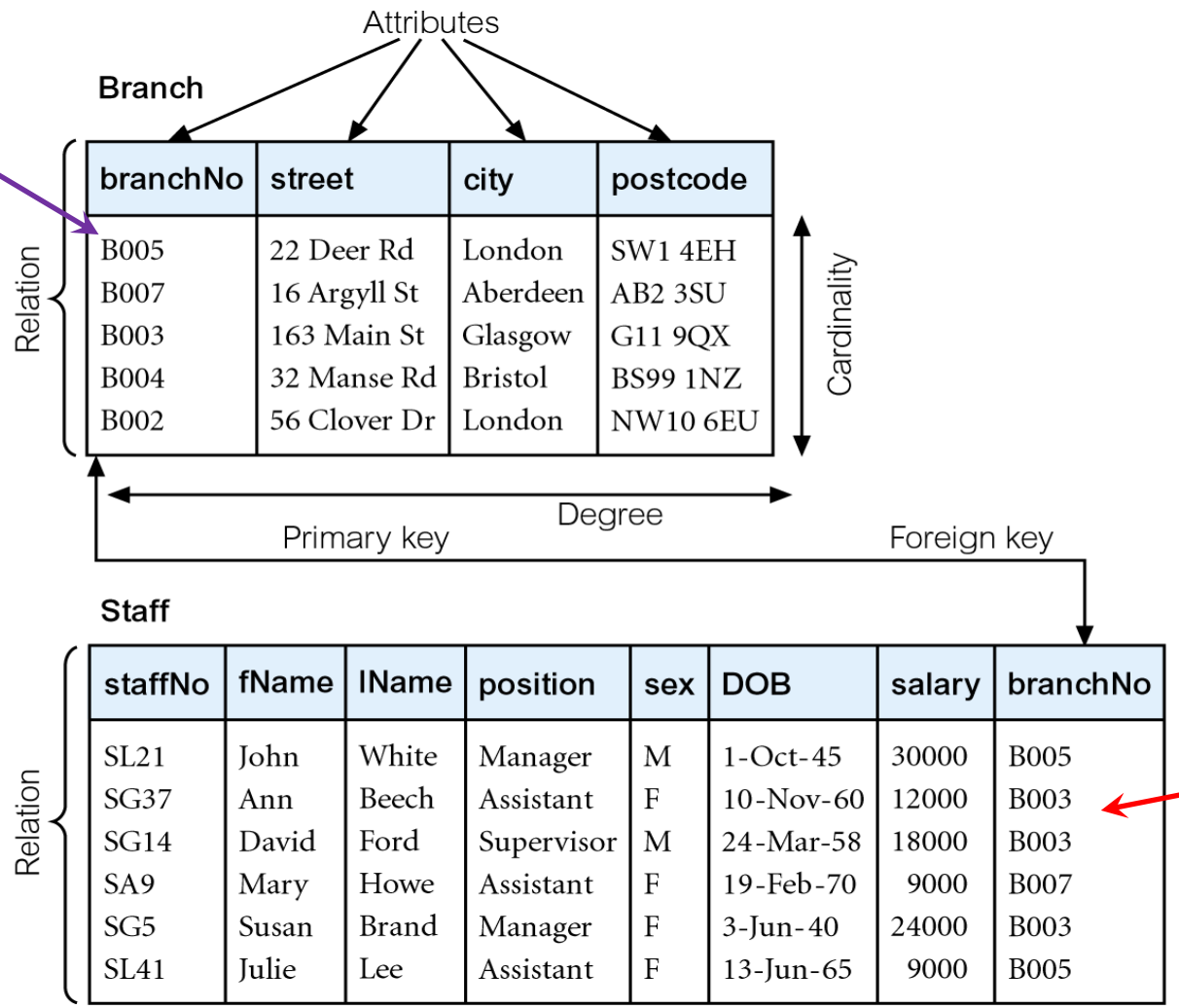
- **Entity Integrity**
  - In a relation, no attribute of a PK can be null.

- **Referential Integrity**

  The FK in a relation can either:
  - Be wholly null (if multiplicity allows);
  - Or match a candidate key value (most often the FK) of some tuple in its home relation.

Relational Model – Recap

ENTITY INTEGRITY:
values of a PK can never be null!



REFERENTIAL INTEGRITY:
values of a FK can either be null or match values of the PK they reference, nothing else!

Logical Mapping: from Conceptual to Logical

- **Logical Mapping**
  - Convert a Conceptual EERD (high-level architecture) to a Logical ERD (a relational schema directly implementable in SQL)

- **10 Mapping rules**

⚠ Only consider cardinality

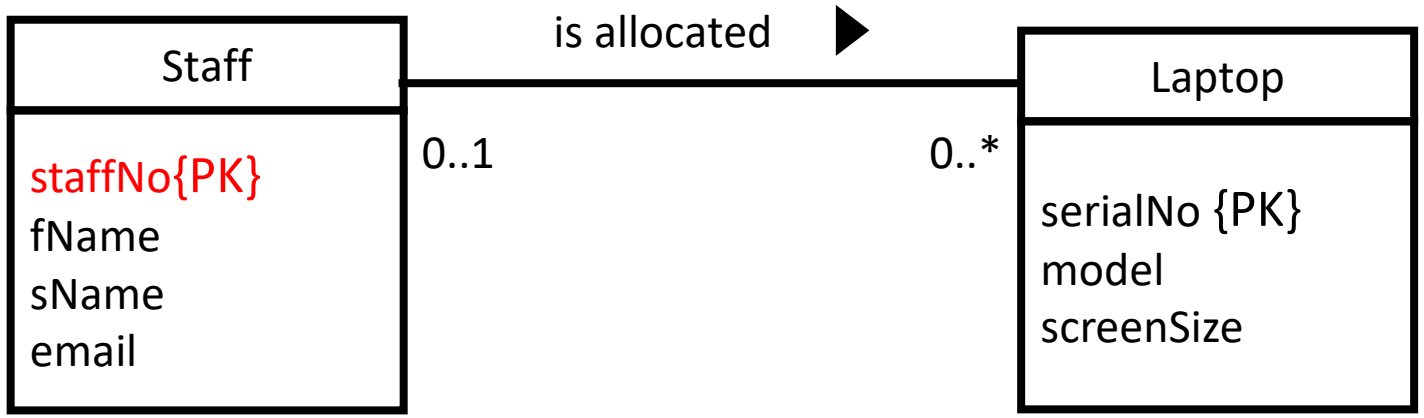
1) One-to-Many relationships
2) One-to-One relationships mandatory on both sides
3) One-to-One relationships optional on one side
4) One-to-One relationships optional on both sides
5) Many-to-Many relationships
6) Complex relationships: ternary and quaternary
7) Specialisations with {Mandatory, And}
8) Specialisations with {Optional, And}
9) Specialisations with {Mandatory, Or}
10) Specialisations with {Optional, Or}

1) Mapping One-to-Many Relationships – Rule

- Create **2 tables** based on the 2 original entities.
- Parent table on the “one” side.
- Child table on the “many” side.
- Create FK on the Child table as a copy of the PK of the Parent table.
- FK of the Child Table references the PK of Parent Table.

1) Mapping One-to-Many Relationships – Example

Conceptual



Logical

Parent **Child**

Tables

Staff (staffNo{PK}, fName, sName, email)

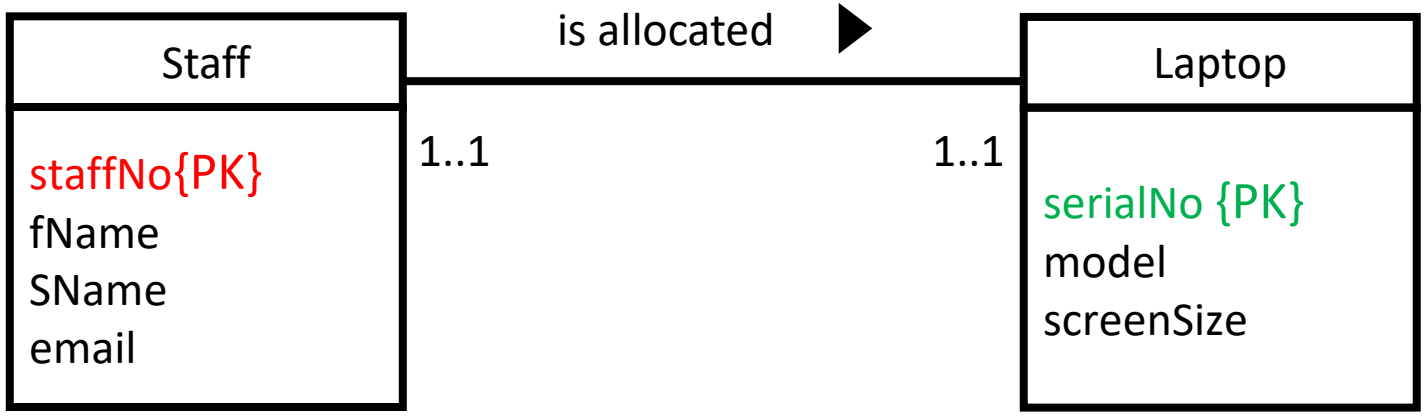
Laptop (serialNo {PK}, model, screenSize, staffNo{FK})

2) Mapping One-to-One Relationships Mandatory on Both Sides — Rule

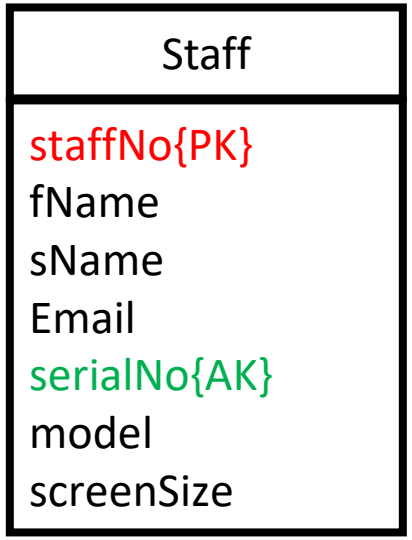
- Merge 2 entities into **one table** with all attributes under new table.
- Choose one PK from the two original PKs.
- Make the other one the AK.

2) Mapping One-to-One Relationships Mandatory On Both Sides — Example

Conceptual



Logical



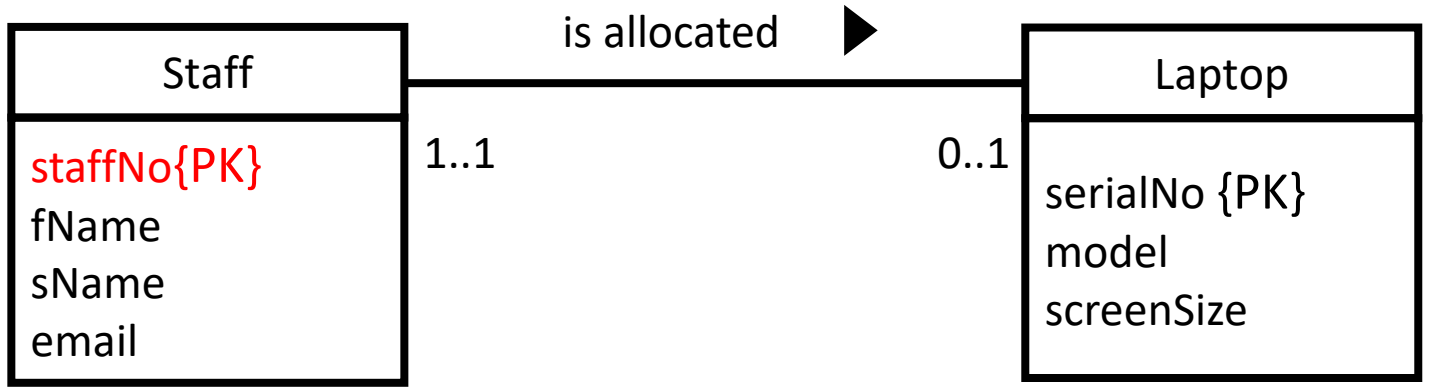
Tables Staff (staffNo{PK}, fName, sName, email, serialNo{AK}, model, screenSize)

3) Mapping One-to-One Relationships Optional on One Side — Rule

- Create **2 tables** based on the 2 original entities.
- Parent table on the “mandatory” side.
- Child table on the “optional” side.
- Create FK on the Child table as a copy of the PK of the Parent table.
- FK of the Child Table references the PK of Parent Table.

3) Mapping One-to-One Relationships Optional on One Side — Example

Conceptual



Logical

Tables

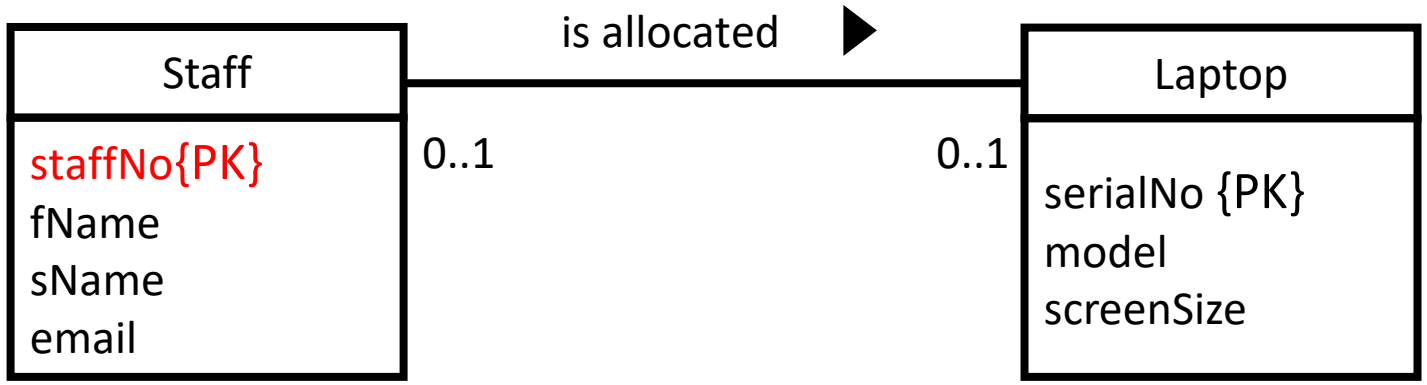
Staff (staffNo{PK}, fName, sName, email)
Laptop (serialNo{PK}, model, screenSize, staffNo{FK})

4) Mapping One-to-One Relationships Optional on Both Sides — Rule

- Create **2 tables** based on the 2 original entities.
- Parent table on the “more mandatory” side.
- Child table on the “more optional” side.
- Create FK on the Child table as a copy of the PK of the Parent table.
- FK of the Child Table references the PK of Parent table.

4) Mapping One-to-One Relationships Optional on Both Sides — Example

Conceptual



Logical

Tables

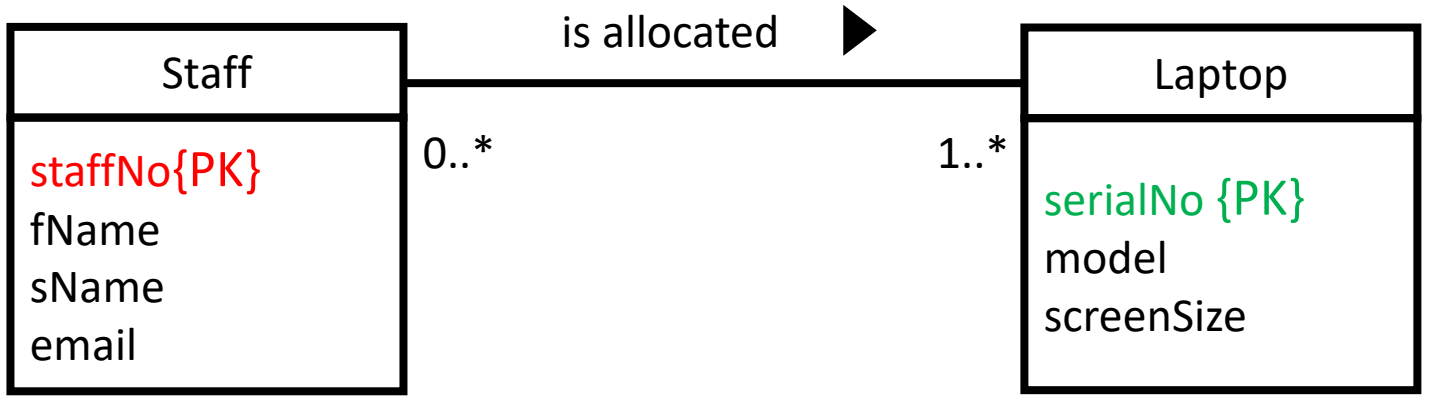
Staff (staffNo{PK}, fName, sName, email)
Laptop (serialNo{PK}, model, screenSize, staffNo{FK})

5) Mapping Many-to-Many Relationships – Rule

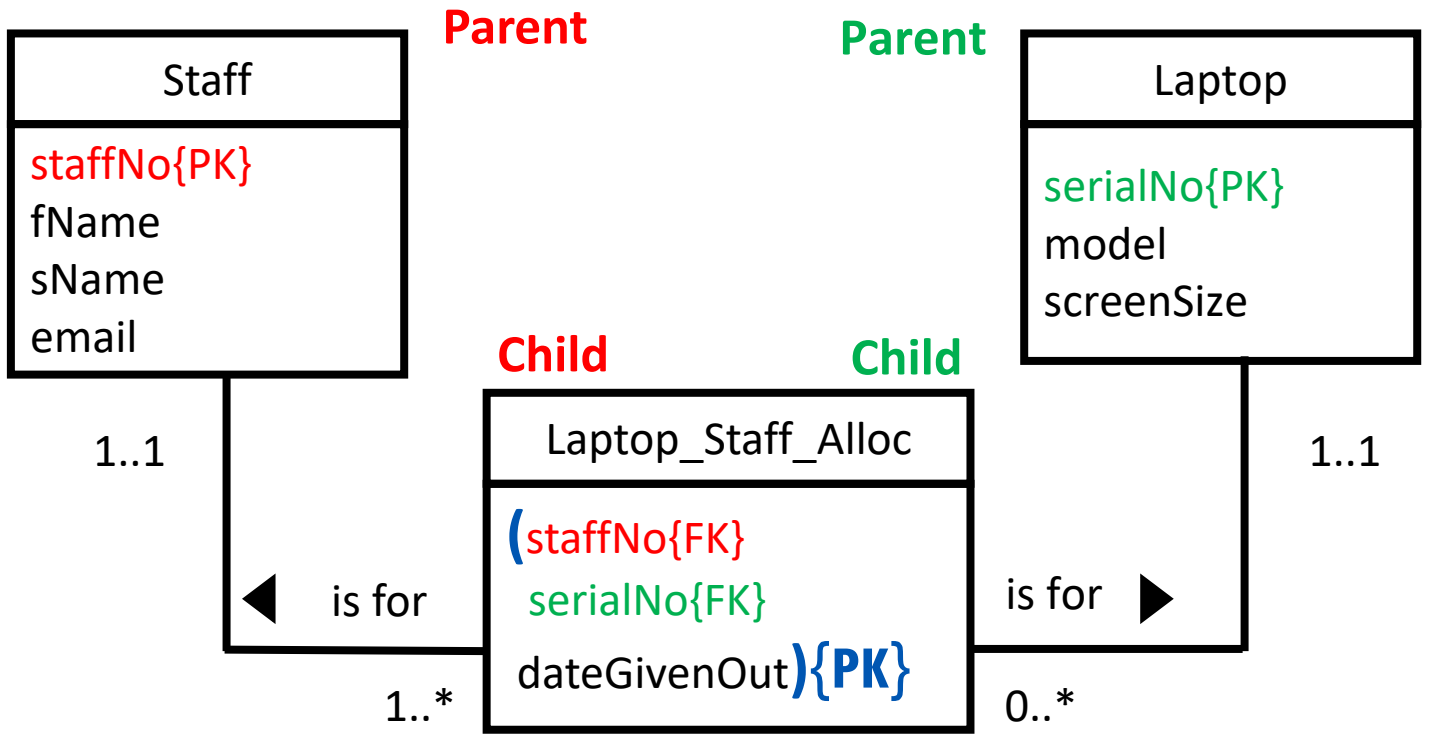
- Create **3 Tables**.
- Create 2 original Parent tables based on original entities.
- Create a Link table associated to the two Parent tables through two 1:M relationships
  - Link table is the Child table of both Parent tables.
  - FKs of Link Child table reference the PKs of the Parent tables.
- PK of Link Child table combination of the 2 PKs of the Parent Tables.
  - Compound PK: if combination of 2 PKs will never be repeated.
  - Composite PK with additional date (and possibly time): if combination of 2 PKs can be repeated.

5) Mapping Many-to-Many Relationships – Example

Conceptual



Logical



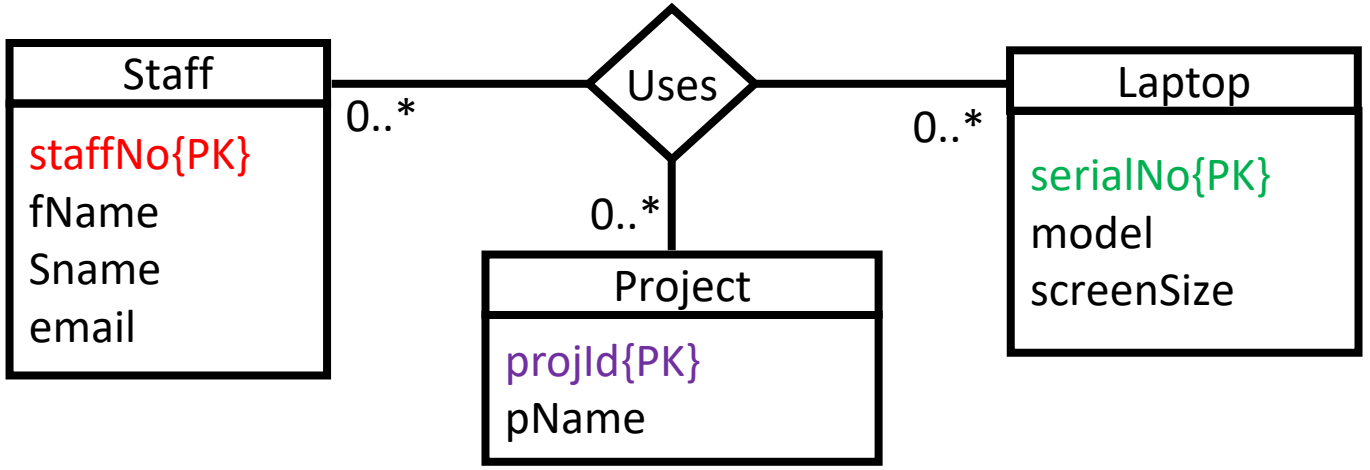
Compound or Composite PK?

6) Mapping Complex Relationships: Ternary & Quaternary Rule

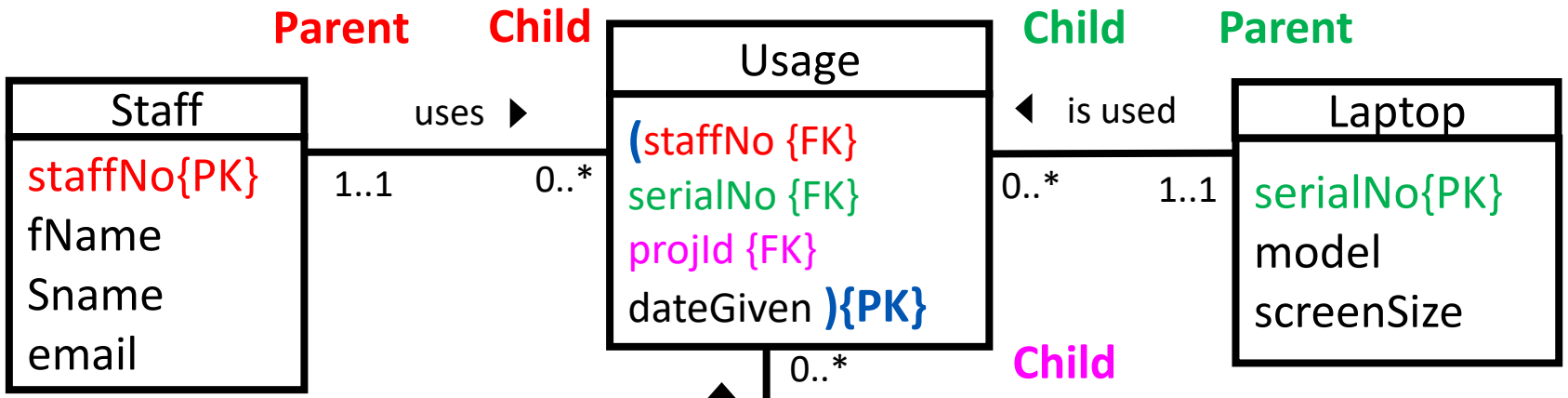
- For ternary, create **4 Tables**.
- Create 3 original Parent tables based on original entities.
- Create a Link table associated to the three Parent tables through three 1:M relationships
  - Link table is the Child table of all three Parent tables.
  - FKs of Link Child table reference the PKs of the Parent tables.
- PK of Link Child table combination of the 3 PKs of the Parent Tables.
  - Compound PK: if combination of 3 PKs will never be repeated.
  - Composite PK with additional date (and possibly time): if combination of 3 PKs can be repeated.

6) Mapping Complex Relationships: Ternary & Quaternary — Example

Conceptual



Logical



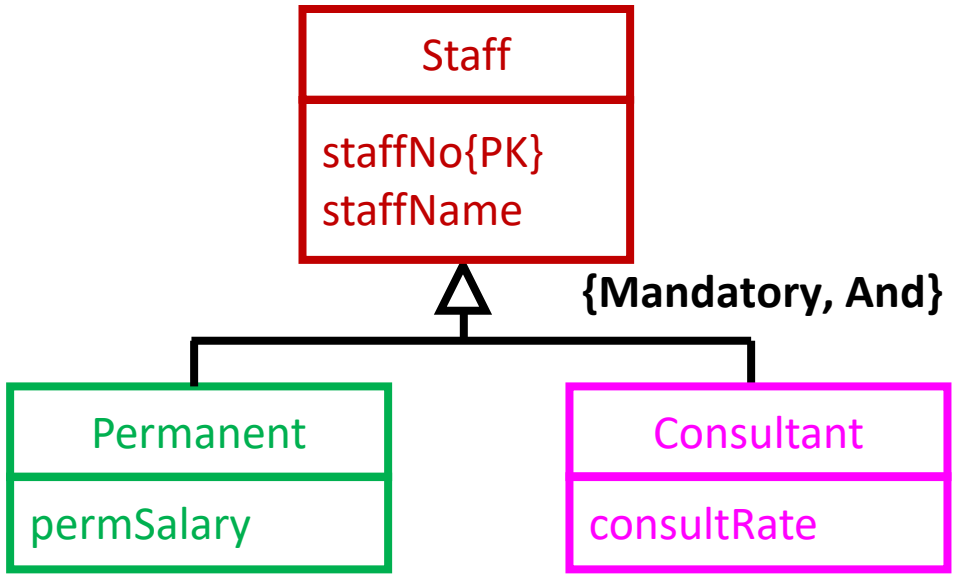
Compound or Composite PK?

7) Mapping Generalisations with {Mandatory, And} – Rule

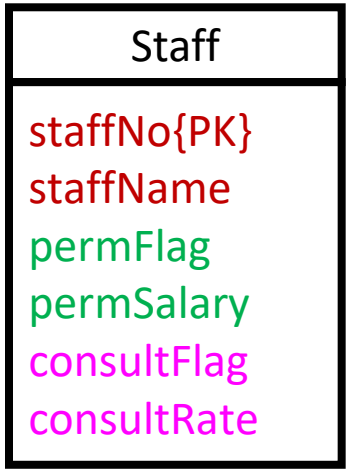
- Merge all entities into **one table** with all attributes under new table.
- Create PK of the new table as the PK of the generalised entity.
- Add flags to differentiate between records of previous specialised entities.

7) Mapping Generalisations with {Mandatory, And} – Example

Conceptual



Logical



Tables Staff (staffNo{PK}, staffName, permSalary, consultRate, permFlag, consultFlag)

8) Mapping Generalisations with {Optional, And} – Rule

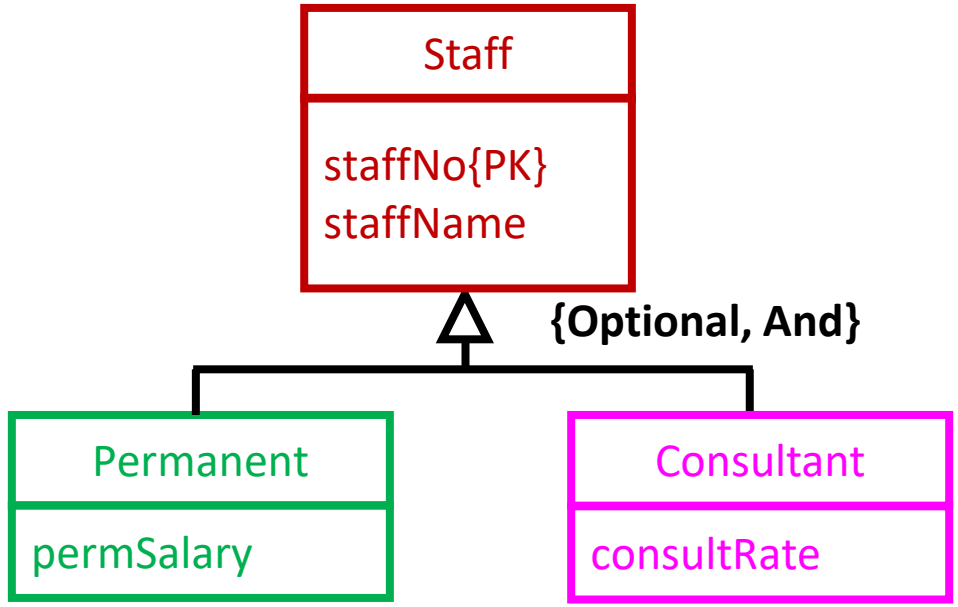
- Create **2 tables**
  - One table for general entity which becomes the Parent table.
  - One table for both specialised entities merged together which becomes the Child table.

- Create ONE one-to-one relationship optional on one side between the 2 tables
  - FK of the Child table references PK of the Parent table.
  - PK of the Child table is the same as the PK of Parent table.

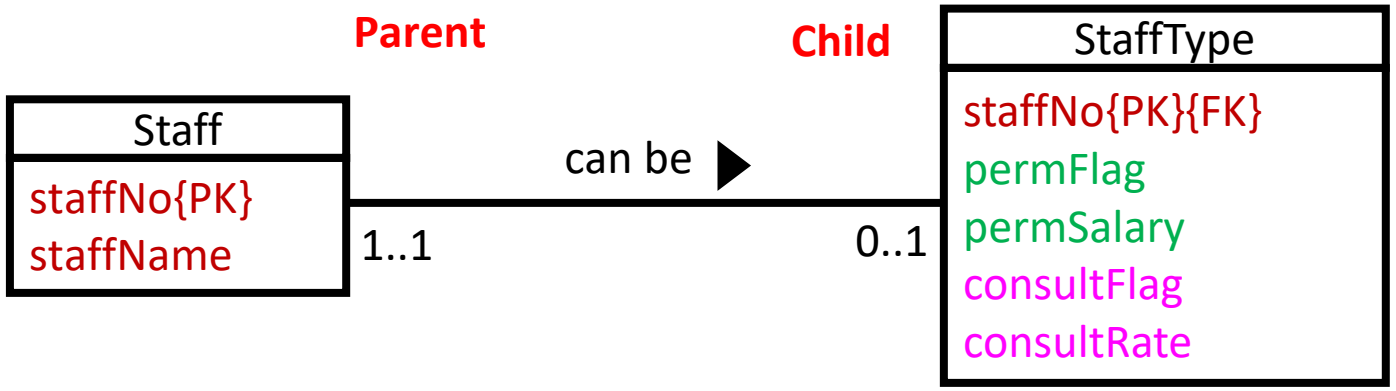
- Add flags to differentiate between records of previous specialised entities.

8) Mapping Generalisations with {Optional, And} – Example

Conceptual



Logical



Tables

Staff (staffNo{PK}, staffName)
StaffDetails (staffNo{PK}{FK}, permFlag, permSalary, consultFlag, consultRate)

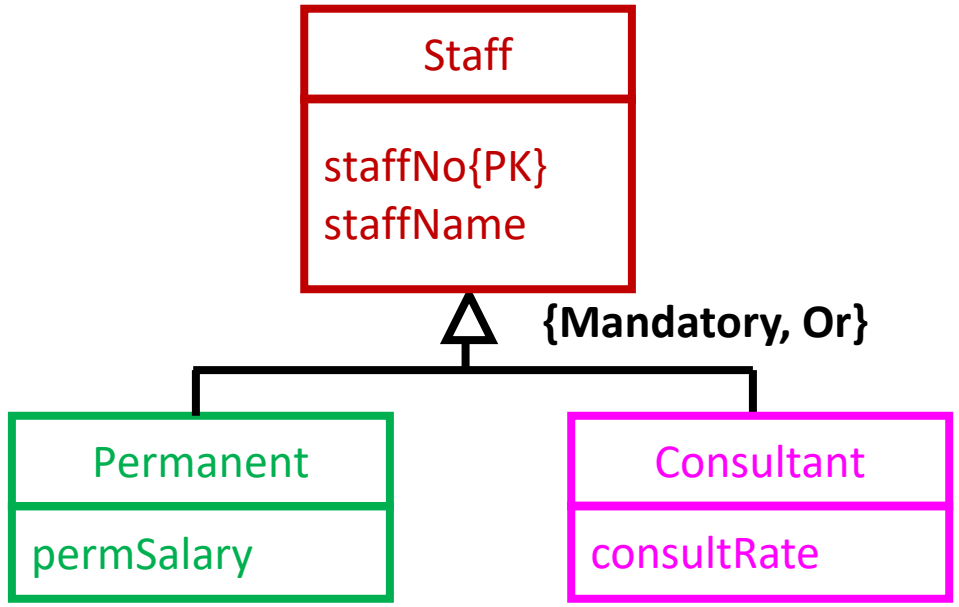
9) Mapping Generalisations with {Mandatory, Or} – Rule

- Create **2 separate tables**
  - One table for each of the specialised entities with the attributes of the general entity also added.
  - PK for both tables is the PK of original general entity.

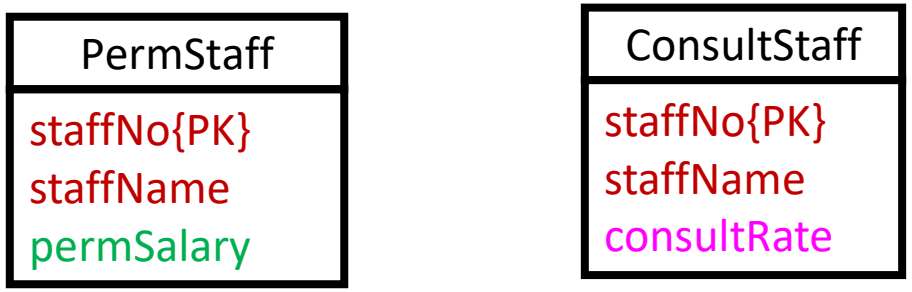
- Each table have their own relationships with the rest of the logical schema
  - The relationships that were associated to the original general entity are doubled up.
  - The relationships that were associated to each specialised entities remain the same.

9) Mapping Generalisations with {Mandatory, Or} – Example

Conceptual



Logical



Tables

PermStaff (staffNo{PK}, staffName, permSalary)
ConsutlStaff(staffNo{PK}, staffName, consultRate)

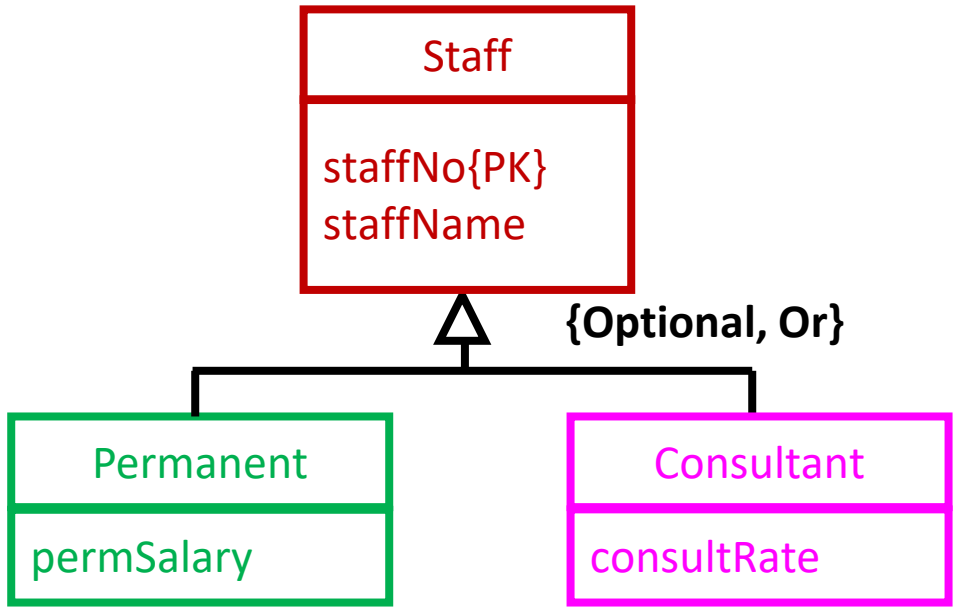
10) Mapping Generalisations with {Optional, Or} – Rule

- Create **2 tables**
  - One table for general entity which becomes the Parent table.
  - One table for each of the specialised entities, which becomes the Child table respectively.

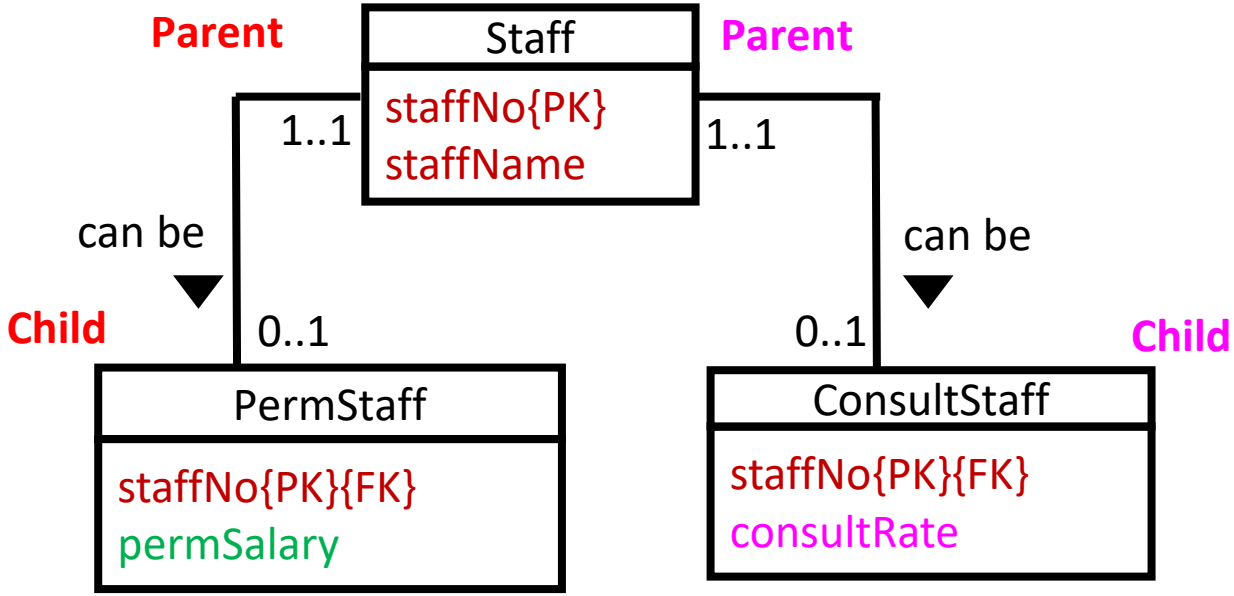
- Create TWO one-to-one relationships optional on one side between the Parent table and the 2 respective Child tables
  - FK of each Child table references PK of the Parent table.
  - PK of each Child table is the same as the PK of Parent table.

10) Mapping Generalisations with {Optional, Or} – Example

Conceptual



Logical



Step-by-step Approach to Logical Mapping

I. Map specialisations (rules 7, 8, 9 or 10)

- Consider constraint and apply appropriate rule.

II. Map one-to-one relationships mandatory on both sides (rule 2)

- Merge 2 entities into one table, select PK and AK.

III. Map complex & many-to-many relationships (rules 5 or 6)

- Reproduce original entities and make them parent tables.
- Introduce link table as a child, define new multiplicities and define FKs and PK.

IV. Map one-to-many relationships and one-to-one relationships that are optional on one side or on both sides (rules 1, 3 or 4)

- Reproduce original entities, make one the parent table, make the other the child.
- Introduce FK in the child table to reference PK of the parent table.

References and Essential Readings

- Module Reading List: https://rl.talis.com/3/westminster/lists/2CAA7D6B-DCAD-AB71-C97B-7FEFCB499C28.html

- Connolly, T. & Begg, C. E. (2015). Database systems: a practical approach to design, implementation and management. 6th Edition (Global Edition). Pearson Education. Ch. 1, 12, 13, 16.

- Elmasri, R. & Navathe, S. (2017). Fundamentals of Database Systems. 7th Edition (Global Edition). Pearson Education. Ch 9.