# 3ISY402 – DATABASE SYSTEMS

## Lecture 2 –
## The Relational Model

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

# Lecture - Objectives

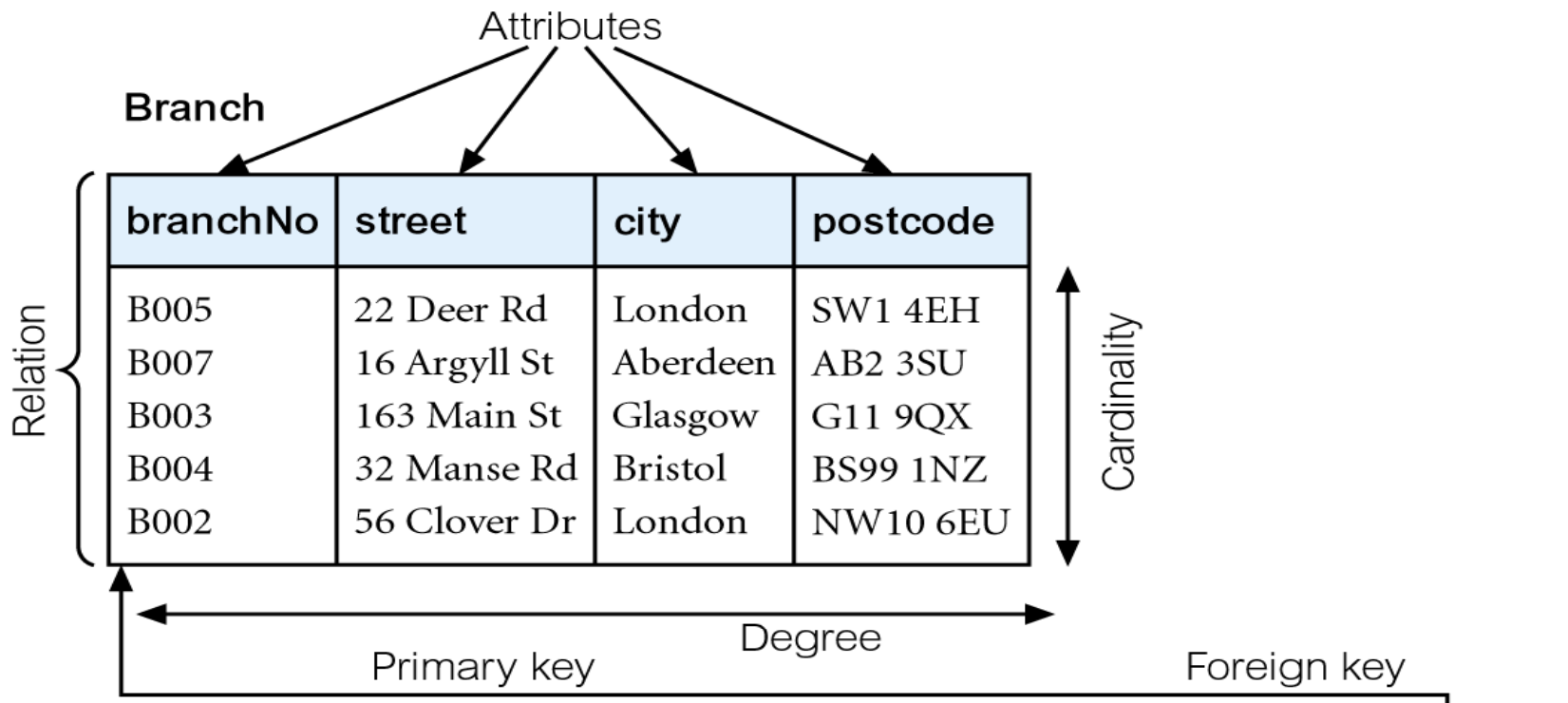
- Terminology of relational model.
- How tables are used to represent data.
- Properties of database relations.
- How to identify CK, PK, and FKs.
- Meaning of entity integrity and referential integrity.
- Purpose and advantages of views.
- Build the Logical Data Model:
  - Create relations for the logical data model to represent the entities, relationships, and attributes that have been identified.

# Relational Model Terminology

- A *relation* is a table with columns and rows.
  - Only applies to logical structure of the database, not the physical structure.
- *Attribute* is a named column of a relation.
- *Domain* is the set of allowable values for one or more attributes.
- *Tuple* is a row of a relation.
- *Degree* is the number of attributes in a relation.
- *Cardinality* is the number of tuples in a relation.
- *Relational Database* is a collection of normalized relations with distinct relation names.

# Relational Model Terminology

- Alternative terminology for Relational Model:
  - Relations (tables or files).
  - Tuples (rows or records).
  - Attributes (columns, fields or data items).
- Relational database schema:
  - Set of relation schemas, each with a distinct name.
- Relation schema:
  - Named relation defined by a set of attribute and domain name pairs.




**Staff**

Relation

| 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 |

# Examples of Attribute Domains

| Attribute | Domain Name | Meaning | Domain Definition |
|---|---|---|---|
| branchNo | BranchNumbers | The set of all possible branch numbers | character: size 4, range B001–B999 |
| street | StreetNames | The set of all street names in Britain | character: size 25 |
| city | CityNames | The set of all city names in Britain | character: size 15 |
| postcode | Postcodes | The set of all postcodes in Britain | character: size 8 |
| sex | Sex | The sex of a person | character: size 1, value M or F |
| DOB | DatesOfBirth | Possible values of staff birth dates | date, range from 1-Jan-20, format dd-mmm-yy |
| salary | Salaries | Possible values of staff salaries | monetary: 7 digits, range 6000.00–40000.00 |

# Properties of Relations

- Relation name is distinct from all other relation names in the relational schema.
- Each cell of the relation contains exactly one atomic (single) value.
- Each attribute has a distinct name.
- Values of an attribute are all from the same domain.
- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically. (But in practice, the order may affect the efficiency of accessing tuples).

# Relational Keys

- Superkey:
  - An attribute, or set of attributes, that uniquely identifies a tuple within a relation.
- Candidate Key:
  - An attribute, or set of attributes that can uniquely identify a tuple within a relation.
- Primary Key
  - Candidate key selected to identify tuples uniquely within relation.

# Relational Keys

- Composite or Compound Key:
  - A primary key comprising more than one attribute.
- Alternate Keys:
  - Candidate keys that are not selected to be the primary key.
- Foreign Key:
  - Attribute, or set of attributes, within one relation that matches the primary key of some (possibly same) relation.

# Foreign Key

- When we have a relationship between two entities, we need to associate the instances at one end of the relationship with the related instance at the other end.
- In a relational model this is achieved by including the primary key of the *MASTER* (the entity at the ONE end of the relationship) in the set of attributes of the *DETAIL* (the entity at the MANY end of the relationship).
- Foreign keys are NOT shown on an E-R Diagram (Conceptual Data Model).
- Foreign keys are added in the Logical Data Model.

# Integrity Constraints

- Relational model has a set of integrity rules which ensure that the data is accurate.
- Entity Integrity:
  - *In a base relation, no attribute of a primary key can be null.*
  - Because primary key values are used to identify the individual tuples in a relation.
- Referential Integrity:
  - *If a foreign key exists in a relation, either the foreign key value must match a primary key value of some tuple in its home relation, or the foreign key value must be wholly null.*
  - Used to specify a relationship among two tuples.

# Integrity Constraints

- Null:
  - *Represents value for an attribute that is currently unknown or not applicable for a tuple.*
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value and is not the same as zero or spaces, which are values.
- General (Enterprise) Constraints:
  - Additional rules specified by users or DBA (database administrators) that define or constrain some aspect of the enterprise.

# Views

- Base Relation
  - Named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.
- View
  - The *dynamic* result of one or more relational operations operating on the base relations to produce another relation.
  - A *virtual* relation that does not necessarily actually exist in the database but is produced upon request by a given user, at the time of request.

# Views

- A virtual or *derived relation*.
- Contents of a view are defined as a query on one or more base relations.
- Views are dynamic, meaning that changes made to base relations that affect the view attributes are immediately reflected in the view.

# Purpose of Views

- Provides powerful and flexible security mechanism by hiding parts of database from certain users.
  - E.g. can create a view containing general staff details but not their salary or other sensitive data.
- Permits users to access data in a way customized to their needs, so that the same data can be seen by different users in different ways simultaneously.
- Can simplify complex operations on the base relations.

# Updating Views

- All updates to a base relation should be immediately reflected in all views that reference that base relation.
- If a view is updated, underlying base relation should reflect change.

# Updating Views - Restrictions

- There are restrictions on types of modifications that can be made through views:
  - Updates are allowed if query involves a single base relation and contains a candidate key of the base relation.
  - Updates are *not allowed* involving multiple base relations.
  - Updates are *not allowed* involving aggregation or grouping operations.

# References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 4 – The Relational Model.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on The Relational Model.

# 3ISY402 – DATABASE SYSTEMS

## Lecture 2 – Logical Database Design for the Relational Model

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

# Building the Logical Data Model

- Build the Logical Data Model (LDM):
  - Create a set of relations for the logical data model to represent the entities, relationships, and attributes that have been identified.
- Translate the conceptual data model of the database (represented by the E-R Diagram) to the logical data model (or *relational schema)*.
- *ER-to-Relational Schema Mapping*.
- Must follow several steps to derive a set of relations from a conceptual data model.

# Derive relations for Logical Data Model – Step 1 - Map all Entities to Relations

- Map all entities to relations:
  - For each entity in the data model, create a relation.
  - Each simple attribute of that entity is mapped to an attribute in the relation.
  - Choose one candidate key of the entity type as the primary key for the relation.
  - The primary key is underlined.

# Derive relations for Logical Data Model – Step 1 - Example

- Map all entities to relations:

| Course |
| --- |
| code {PK} |
| name |
| fee |
| startDate |
| endDate |

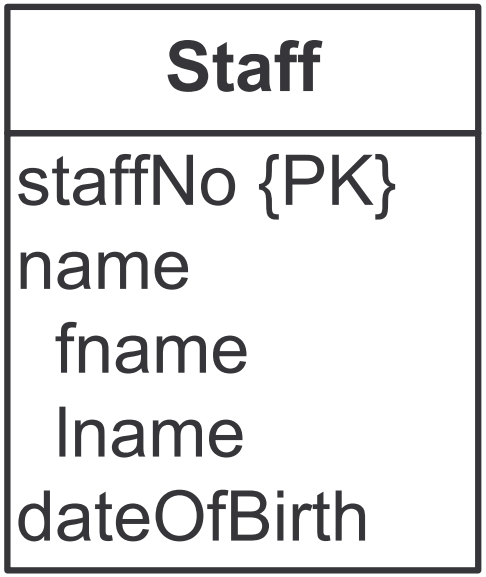
**Course (<u>code</u>, name, fee, startDate, endDate)**

# Derive relations for Logical Data Model – Step 2 – Map Composite attributes

- Map composite attributes:
  - Map each *simple component* of a composite attribute to a separate attribute in the corresponding relation.
  - Do not map the original composite attribute itself.

# Derive relations for Logical Data Model – Step 2 - Example

- Map composite attributes:




**Staff (staffNo, fname, lname, dateOfBirth)**

# Derive relations for Logical Data Model – Step 3 – Map Multi-valued Attributes

- Map multi-valued attributes:
  - Create a new relation to represent the multi-valued attribute, and include the primary key of owner entity in the new relation to act as a foreign key.
  - Do not include multi-valued attribute in the relation for the owner entitiy.
  - The foreign key is denoted by placing an asterisk (*) after it.
  - The primary key of the new relation is the combination of the primary key of the owner entity and the multi-valued attribute itself.

# Derive relations for Logical Data Model – Step 3 - Example

- Map multi-valued attributes:

| Branch |
| --- |
| branchNo {PK}<br>telNo [1..*]<br>faxNo |

**BranchTelNo (branchNo*, telNo)**

**Branch (branchNo, faxNo)**

*Note - branchNo* in BranchTelNo relation is a foreign key and part of the relation's primary key.

# Derive relations for Logical Data Model – Step 4 – Eliminate Derived Attributes

- Eliminate derived attributes:
  - A derived attribute is ***not*** included in the set of attributes for the corresponding relation.
  - Derived attributes are not shown in the relational schema.

# Derive relations for Logical Data Model – Step 4 - Example

- Eliminate derived attributes:

| Staff |
| --- |
| staffNo {PK}<br>name<br>salary<br>NIN<br>/totalStaff |

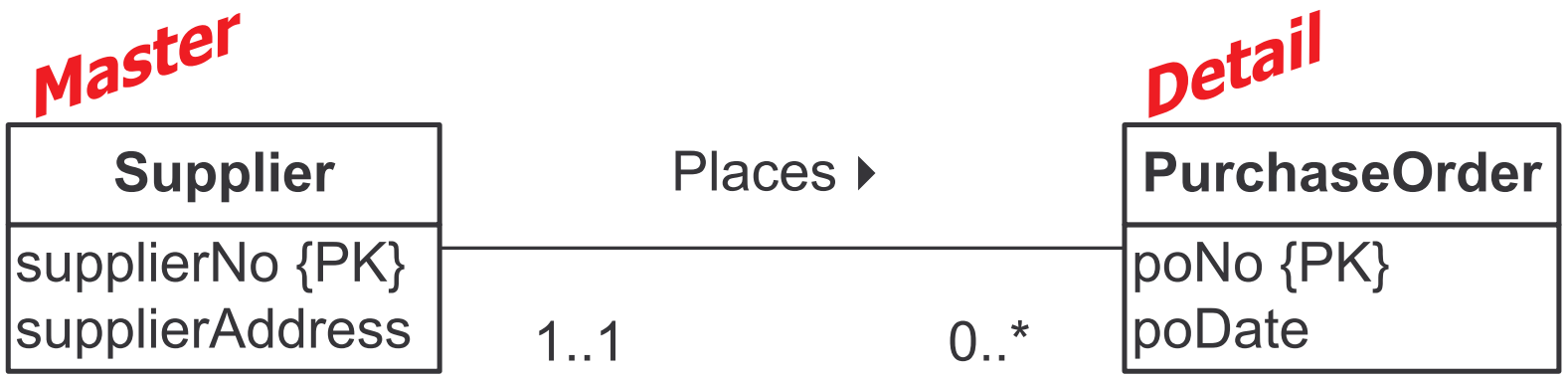
**STAFF (staffNo, name, salary, NIN)**

# Derive relations for Logical Data Model – Step 5 – Map 1:* Relationship Types

- Map one-to-many (1:*) relationship types:
  - For each 1:* relationship, the entity on the ONE side of the relationship is designated as the parent entity (*master*) and the entity on the MANY side is designated as the child entity (*detail*).
  - To represent this relationship, post a copy of the primary key attribute(s) of the parent entity into the relation representing the child entity, to act as a foreign key.
  - Basically, each 1:* relationship is mapped to a foreign key in the relation at the *many-side* referring to the relation at the *one-side*.

# Derive relations for Logical Data Model – Step 5 - Example

- Map one-to-many (1:*) relationship types:




**PurchaseOrder (poNo, poDate, supplierNo*)**

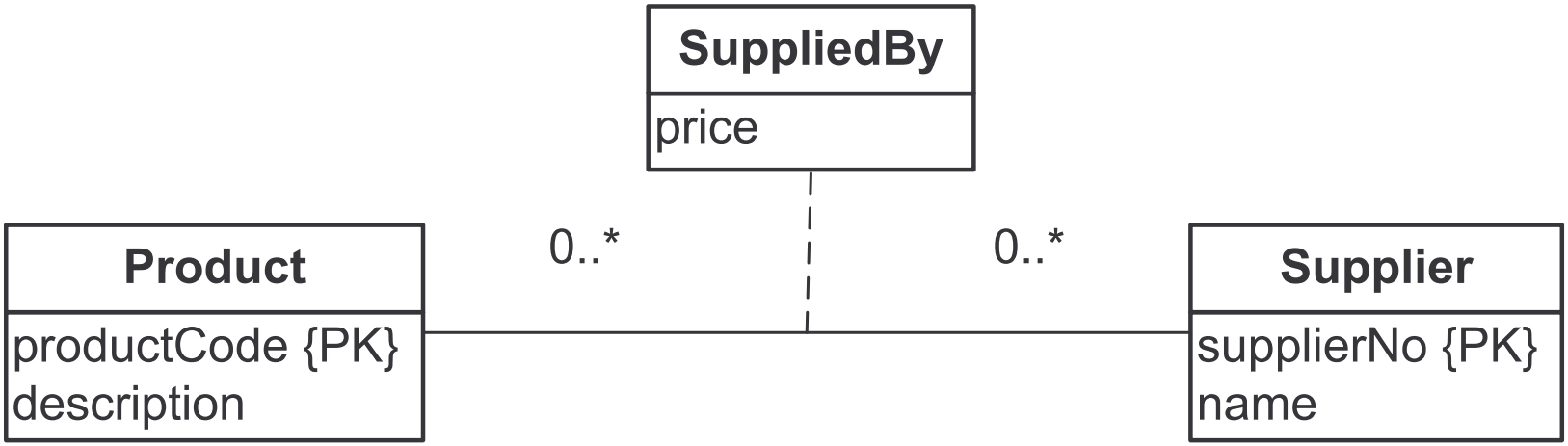
**Supplier (supplierNo, supplierAddress)**

# Derive relations for Logical Data Model – Step 6 – Map *:* Relationship Types

- Map many-to-many (*:*) relationship types:
  - *Create a new relation to represent the relationship.*
  - Include any attributes that are part of the relationship in the new relation.
  - Post a copy of the primary key attribute(s) of both of the entities that participate in the relationship into the new relation, to act as foreign keys.
  - The primary key of the new relation must also include both these foreign keys, possibly in combination with some of the attributes of the relationship (unless a surrogate/artificial key is produced).

# Derive relations for Logical Data Model – Step 6 - Example

- Map many-to-many (*:*) relationship types:




**SupplierProduct (productCode*, supplierNo*, price)**

**Supplier (supplierNo, name)**

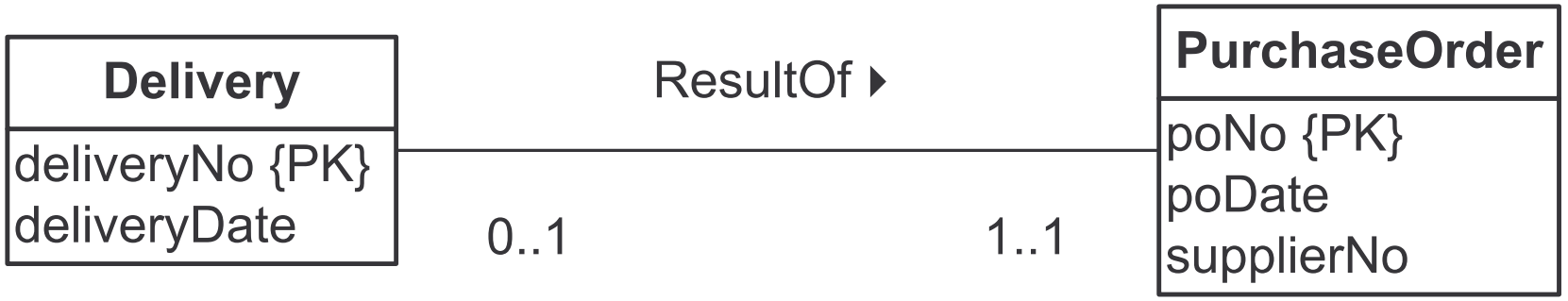
**Product (productCode, description)**

# Derive relations for Logical Data Model – Step 7 - Map 1:1 Relationship Types

- Map one-to-one (1:1) relationship types:
  - Each one-to-one (1:1) relationship type is mapped to a foreign key from one relation referring to the other relation.
  - The foreign key is placed in the relation corresponding to the entity type that has a mandatory participation in the relationship type, or the entity type that is created second.
  - Alternatively, it is possible to map both entity types to a single relation, especially if the participation of both entity types in the 1:1 relationship is mandatory.

# Derive relations for Logical Data Model – Step 7 - Example

- Map one-to-one (1:1) relationship types:




**Delivery (deliveryNo, deliveryDate, poNo*)**

**PurchaseOrder (poNo, poDate, supplierNo)**

# Lecture Case Study 1 – Relational Schema

- Lecture Case Study 1 - *Gudelf and Wellbean Health Authority Database System.*
- Tasks:
  - Given the Conceptual ER Diagram, derive the Relational Schema by *applying all the steps given for each entity in turn.*

# References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 16 – Logical Database Design for the Relational Model.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on Logical Database Design for the Relational Model.