THE UNIVERSITY OF WESTERN AUSTRALIA

CITS1402 Relational Database Management Systems

Week 7—Logical Database Design

Sem1/2024

Dr Mehwish Nasim/ CSSE
cits1402-sem1-pmc@uwa.edu.au

CITS1402 Week7

THE UNIVERSITY OF WESTERN AUSTRALIA

Contents

How to derive a set of relations from a conceptual data model.

How to validate a logical data model

How to merge local logical data models

How to ensure that the final logical data model is a true and accurate representation of the data requirements of the enterprise.

Ref: Chapter 17

Summary of how to map entities and relationships to relations

Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key of entity on 'one' side to act as foreign key in relation representing entity on 'many' side. Any attributes of relationship are also posted to 'many' side.
1:1 binary relationship: (a) Mandatory participation on both sides (b) Mandatory participation on one side	Combine entities into one relation. Post primary key of entity on 'optional' side to act as foreign key in relation representing entity on 'mandatory' side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 16.1.
: binary relationship, complex relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.

Part 2

Week 8….

Relations for the Staff user views of *DreamHome*

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo) **Primary Key** staffNo **Foreign Key** supervisorStaffNo **references** Staff(staffNo)	**PrivateOwner** (ownerNo, fName, lName, address, telNo) **Primary Key** ownerNo
BusinessOwner (ownerNo, bName, bType, contactName, address, telNo) **Primary Key** ownerNo **Alternate Key** bName **Alternate Key** telNo	**Client** (clientNo, fName, lName, telNo, prefType, maxRent, staffNo) **Primary Key** clientNo **Foreign Key** staffNo **references** Staff(staffNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo) **Primary Key** propertyNo **Foreign Key** ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo) **Foreign Key** staffNo **references** Staff(staffNo)	**Viewing** (clientNo, propertyNo, dateView, comment) **Primary Key** clientNo, propertyNo **Foreign Key** clientNo **references** Cl ent(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo)
Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo) **Primary Key** leaseNo **Alternate Key** propertyNo, rentStart **Alternate Key** clientNo, rentStart **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Derived** deposit (PropertyForRent.rent*2) **Derived** duration (rentFinish – rentStart)	

Chapter 17 - Objectives

How to derive a set of relations from a conceptual data model.

How to validate a logical data model

How to merge local logical data models

How to ensure that the final logical data model is a true and accurate representation of the data requirements of the enterprise.

Overview Database Design Methodology

Logical database design for the relational model

Step 2 Build and validate logical data model

- Step 2.1 Derive relations for logical data model
- **Step 2.2 Validate relations using normalization**
- **Step 2.3 Validate relations against user transactions**
- Step 2.4 Define integrity constraints
- Step 2.5 Review logical data model with user
- Step 2.6 Merge logical data models into global model
- Step 2.7 Check for future growth

Step 2.2 Validate relations using normalization

To validate the relations in the logical data model using normalization.

- **ensures that the set of relations has a minimal set of attributes to meet enterprise requirements**
- **3NF to avoid redundancy issues (update issues)**
- **Relations derived from ER Model should be in 3NF**
- **Conversion from conceptual to logical could break normalisation**

Step 2.3 Validate relations against user transactions

To ensure that the relations in the logical data model support the required transactions.

Overview Database Design Methodology

Logical database design for the relational model

Step 2 Build and validate logical data model

- Step 2.1 Derive relations for logical data model
- Step 2.2 Validate relations using normalization
- Step 2.3 Validate relations against user transactions
- **Step 2.4 Define integrity constraints**
- Step 2.5 Review logical data model with user
- Step 2.6 Merge logical data models into global model
- Step 2.7 Check for future growth

Step 2.4 Check integrity constraints

To check integrity constraints are represented in the logical data model. This includes identifying:

- Required data
- Attribute domain constraints
- Multiplicity
- Entity integrity
- Referential integrity
  - Mandatory [1..*] => nulls NOT allowed
  - Optional [0..*] => nulls allowed
- General constraints

Step 2.4 Check existence constraints

Referential integrity issues when candidate and foreign keys

- **Inserted, updated, or deleted**
- **Case 1: Insert tuple into child relation**
- **Case 2: Delete from child relation**
- **Case 3: Update foreign key of child**
- **Case 4: Insert into parent relation**
- **Case 5: Delete from parent relation**
- **Case 6: Update primary key of parent**

Step 2.4 Check existence constraints

Referential integrity issues when candidate and foreign keys

- **Inserted, updated, or deleted**
- **Case 1: Insert tuple into child relation**
- **Case 2: Delete from child relation**
- **Case 3: Update foreign key of child**
- **Case 4: Insert into parent relation**
- **Case 5: Delete from parent relation**
- **Case 6: Update primary key of parent**

Step 2.4 Check existence constraints

Case 5: Delete from parent relation

- **NO ACTION: Prevent deletion**
- **CASCADE: Automatically delete all child tuples**
- **SET NULL: Foreign keys set to null**
- **SET DEFAULT: Foreign keys set to a default value**
- **NO CHECK: do nothing to ensure integrity!**

Referential integrity constraints for relations in Staff user views of *DreamHome*

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)
Primary Key staffNo
Foreign Key supervisorStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Client (clientNo, fName, lName, telNo, prefType, maxRent, staffNo)
Primary Key clientNo
Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)
Primary Key propertyNo
Foreign Key ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

Viewing (clientNo, propertyNo, dateView, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key propertyNo **references** PropertyForRent(propertyNo)
ON UPDATE CASCADE ON DELETE CASCADE

Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)
Primary Key leaseNo
Alternate Key propertyNo, rentStart
Alternate Key clientNo, rentStart
Foreign Key clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key propertyNo **references** PropertyForRent(propertyNo)
ON UPDATE CASCADE ON DELETE NO ACTION

Overview Database Design Methodology

Logical database design for the relational model

Step 2 Build and validate logical data model

- Step 2.1 Derive relations for logical data model
- Step 2.2 Validate relations using normalization
- Step 2.3 Validate relations against user transactions
- Step 2.4 Define integrity constraints
- **Step 2.5 Review logical data model with user**
- Step 2.6 Merge logical data models into global model
- Step 2.7 Check for future growth

Chapter 17 - Objectives

How to derive a set of relations from a conceptual data model.

How to validate a logical data model

How to merge local logical data models

How to ensure that the final logical data model is a true and accurate representation of the data requirements of the enterprise.

Overview Database Design Methodology

Logical database design for the relational model

Step 2 Build and validate logical data model

- Step 2.1 Derive relations for logical data model
- Step 2.2 Validate relations using normalization
- Step 2.3 Validate relations against user transactions
- Step 2.4 Define integrity constraints
- Step 2.5 Review logical data model with user
- **Step 2.6 Merge logical data models into global model**
- Step 2.7 Check for future growth

Step 2.6 Merge local logical data models into global model

To merge local logical data model into a single global logical data model.

This activities in this step include:

- **Step 2.6.1 Merge local logical data models into global model**
- **Step 2.6.2 Validate global logical data model**
- **Step 2.6.3 Review global logical data model with users.**

Step 2.6 Merge local logical data models into global model

Tasks typically includes:

(1) Review the names and contents of entities/relations and their candidate keys.
(2) Review the names and contents of relationships/foreign keys.
(3) Merge entities/relations from the local data models
(4) Include (without merging) entities/relations unique to each local data model
(5) Merge relationships/foreign keys from the local data models.
(6) Include (without merging) relationships/foreign keys unique to each local data model.
(7) Check for missing entities/relations and relationships/foreign keys.
(8) Check foreign keys.
(9) Check Integrity Constraints.
(10) Draw the global ER/relation diagram
(11) Update the documentation.

Step 2.6.2 Validate global logical data model

To validate the relations created from the global logical data model using the technique of normalization and to ensure they support the required transactions, if necessary.

Step 2.6.3 Review global logical data model with users

To review the global logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of an enterprise.

Relations for the Staff user views of *DreamHome*

Staff (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo) **Primary Key** staffNo **Foreign Key** supervisorStaffNo **references** Staff(staffNo)	**PrivateOwner** (ownerNo, fName, lName, address, telNo) **Primary Key** ownerNo
BusinessOwner (ownerNo, bName, bType, contactName, address, telNo) **Primary Key** ownerNo **Alternate Key** bName **Alternate Key** telNo	**Client** (clientNo, fName, lName, telNo, prefType, maxRent, staffNo) **Primary Key** clientNo **Foreign Key** staffNo **references** Staff(staffNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo) **Primary Key** propertyNo **Foreign Key** ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo) **Foreign Key** staffNo **references** Staff(staffNo)	**Viewing** (clientNo, propertyNo, dateView, comment) **Primary Key** clientNo, propertyNo **Foreign Key** clientNo **references** Cl ent(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo)
Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo) **Primary Key** leaseNo **Alternate Key** propertyNo, rentStart **Alternate Key** clientNo, rentStart **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Derived** deposit (PropertyForRent.rent*2) **Derived** duration (rentFinish – rentStart)	

Relations for the Branch user views of *DreamHome*

Branch (branchNo, street, c ty, postcode, mgrStaffNo) **Primary Key** branchNo **Alternate Key** postcode **Foreign Key** mgrStaffNo **references** Manager(staffNo)	**Telephone** (telNo, branchNo) **Primary Key** telNo **Foreign Key** branchNo **references** Branch(branchNo)
Staff (staffNo, name, position, salary, supervisorStaffNo, branchNo) **Primary Key** staffNo **Foreign Key** supervisorStaffNo **references** Staff(staffNo) **Foreign Key** branchNo **references** Branch(branchNo)	**Manager** (staffNo, mgrStartDate, bonus) **Primary Key** staffNo **Foreign Key** staffNo **references** Staff(staffNo)
PrivateOwner (ownerNo, name, address, telNo) **Primary Key** ownerNo	**BusinessOwner** (bName, bType, contactName, address, telNo) **Primary Key** bName **Alternate Key** telNo
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, bName, branchNo) **Primary Key** propertyNo **Foreign Key** ownerNo **references** PrivateOwner(ownerNo) **Foreign Key** bName **references** BusinessOwner(bName) **Foreign Key** staffNo **references** Staff(staffNo) **Foreign Key** branchNo **references** Branch(branchNo)	**Client** (clientNo, name, telNo, prefType, maxRent) **Primary Key** clientNo
Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo) **Primary Key** leaseNo **Alternate Key** propertyNo, rentStart **Alternate Key** clientNo, rentStart **Foreign Key** clientNo **references** Client(clientNo) **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Derived** deposit (PropertyForRent.rent*2) **Derived** duration (rentFinish – rentStart)	**Registration** (clientNo, branchNo, staffNo, dateJoined) **Primary Key** clientNo **Foreign Key** cl entNo **references** C ient(clientNo) **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** staffNo **references** Staff(staffNo)
Advert (propertyNo, newspaperName, dateAdvert, cost) **Primary Key** propertyNo, newspaperName, dateAdvert **Foreign Key** propertyNo **references** PropertyForRent(propertyNo) **Foreign Key** newspaperName **references** Newspaper(newspaperName)	**Newspaper** (newspaperName, address, telNo, contactName) **Primary Key** newspaperName **Alternate Key** telNo

Relations that represent the global logical data model for *DreamHome*

Branch (branchNo, street, city, postcode, mgrStaffNo)
Primary Key branchNo
Alternate Key postcode
Foreign Key mgrStaffNo **references** Manager(staffNo)

Telephone (telNo, branchNo)
Primary Key telNo
Foreign Key branchNo **references** Branch(branchNo)

Staff (staffNo, fName, lName, position, sex, DOB, salary, supervisorStaffNo, branchNo)
Primary Key staffNo
Foreign Key supervisorStaffNo **references** Staff(staffNo)
Foreign Key branchNo **references** Branch(branchNo)

Manager (staffNo, mgrStartDate, bonus)
Primary Key staffNo
Foreign Key staffNo **references** Staff(staffNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)
Primary Key ownerNo
Alternate Key bName
Alternate Key telNo

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Primary Key propertyNo
Foreign Key ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
Foreign Key staffNo **references** Staff(staffNo)
Foreign Key branchNo **references** Branch(branchNo)

Viewing (c ientNo, propertyNo, dateView, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo)
Foreign Key propertyNo **references** PropertyForRent(propertyNo)

Client (clientNo, fName, lName, telNo, prefType, maxRent)
Primary Key clientNo

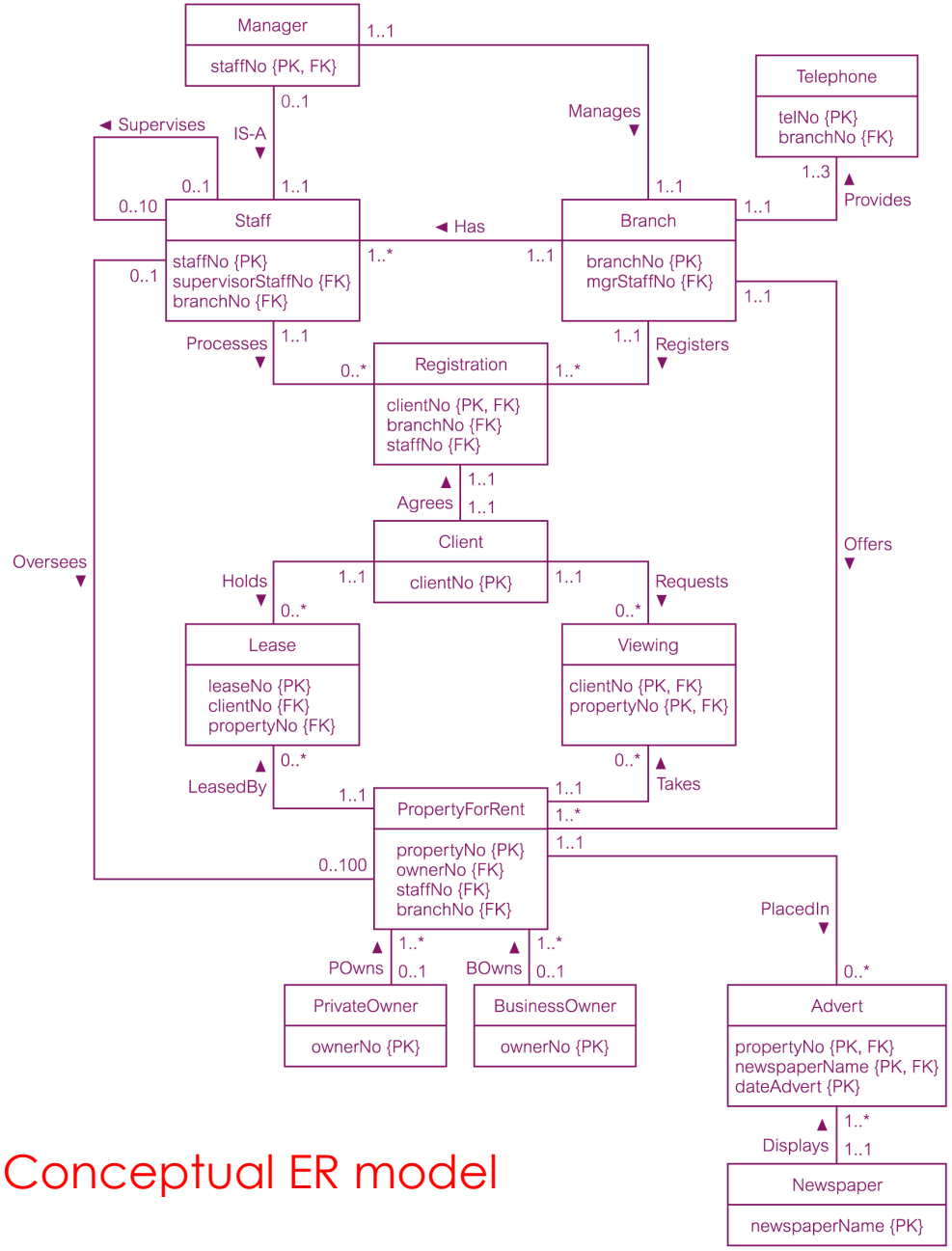
Registration (clientNo, branchNo, staffNo, dateJoined)
Primary Key clientNo
Foreign Key clientNo **references** Client(clientNo)
Foreign Key branchNo **references** Branch(branchNo)
Foreign Key staffNo **references** Staff(staffNo)

Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, cl entNo, propertyNo)
Primary Key easeNo
Alternate Key propertyNo, rentStart
Alternate Key c ientNo, rentStart
Foreign Key clientNo **references** Client(clientNo)
Foreign Key propertyNo **references** PropertyForRent(propertyNo)
Derived depos t (PropertyForRent.rent*2)
Derived duration (rentFinish – rentStart)

Newspaper (newspaperName, address, telNo, contactName)
Primary Key newspaperName
Alternate Key telNo

Advert (propertyNo, newspaperName, dateAdvert, cost)
Primary Key propertyNo, newspaperName, dateAdvert
Foreign Key propertyNo **references** PropertyForRent(propertyNo)
Foreign Key newspaperName **references** Newspaper(newspaperName)

Global relation/logical diagram for *DreamHome*



note: This is not a Conceptual ER model

Overview Database Design Methodology

Logical database design for the relational model

Step 2 Build and validate logical data model

- Step 2.1 Derive relations for logical data model
- Step 2.2 Validate relations using normalization
- Step 2.3 Validate relations against user transactions
- Step 2.4 Define integrity constraints
- Step 2.5 Review logical data model with user
- Step 2.6 Merge logical data models into global model
- **Step 2.7 Check for future growth**