

DATABASE FOUNDATIONS

ORACLE ACADEMY



Juan Carlos Herrera H.

6 DE MAYO DE 2025

<https://academy.oracle.com/>

[HTTPS://GITHUB.COM/ISC-UPA/2025-2-ISC05-DB](https://github.com/ISC-UPA/2025-2-ISC05-DB)

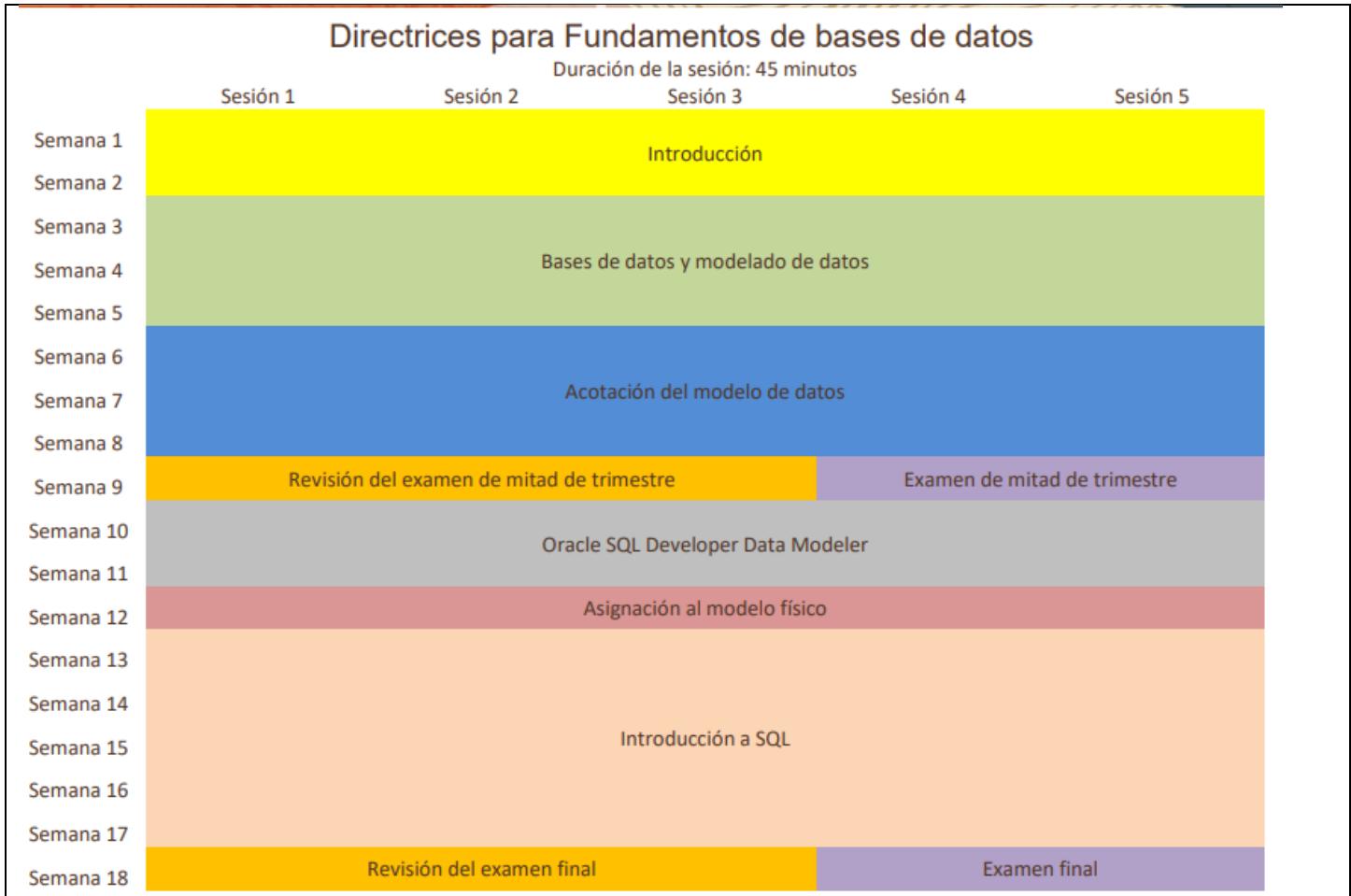
Contenido

1.	Introduction	3
1.1.	Introduction	4
1.2.	Introduction to Databases	5
1.3.	Types of Database Models	7
1.4.	Business Requirements	8
2.	Databases and Data Modeling	9
2.1.	Relational Databases	9
2.2.	Conceptual and Physical Data Models	11
2.3.	Entities and Attributes.....	12
2.4.	Unique Identifiers.....	13
2.5.	Relationships	13
2.6.	Entity Relationship Modeling (ERDs)	15
3.	Refining the Data Model	18
3.1.	More with Relationships.....	18
3.1.1.	Identifying (Barred) Relationships	18
3.1.2.	M:M Relationships	18
3.1.3.	Non-Transferable Relationships ◇.....	19
3.1.4.	Supertype and Subtype Entities	19
3.1.5.	Modeling Hierarchical Data.....	20
3.1.6.	Recursive Relationships.....	20
3.1.7.	Arc Relationship ◎.....	20
3.2.	Tracking Data Changes	21
3.3.	Normalization and Business Rules.....	22
	1NF	22
	2NF	22
	3NF	23
	Business Rules	23
3.4.	Data Modeling Terminology and Mapping.....	24
3.4.1.	Mapping Arcs.....	24
3.4.2.	Mapping Supertype/Subtypes	25
4.	Oracle SQL Developer Data Modeler	27
4.1.	Oracle SQL Developer Data Modeler.....	27
	Set Primary and Secondary UIDs.....	27
	Creating the Supertype Entity.....	28

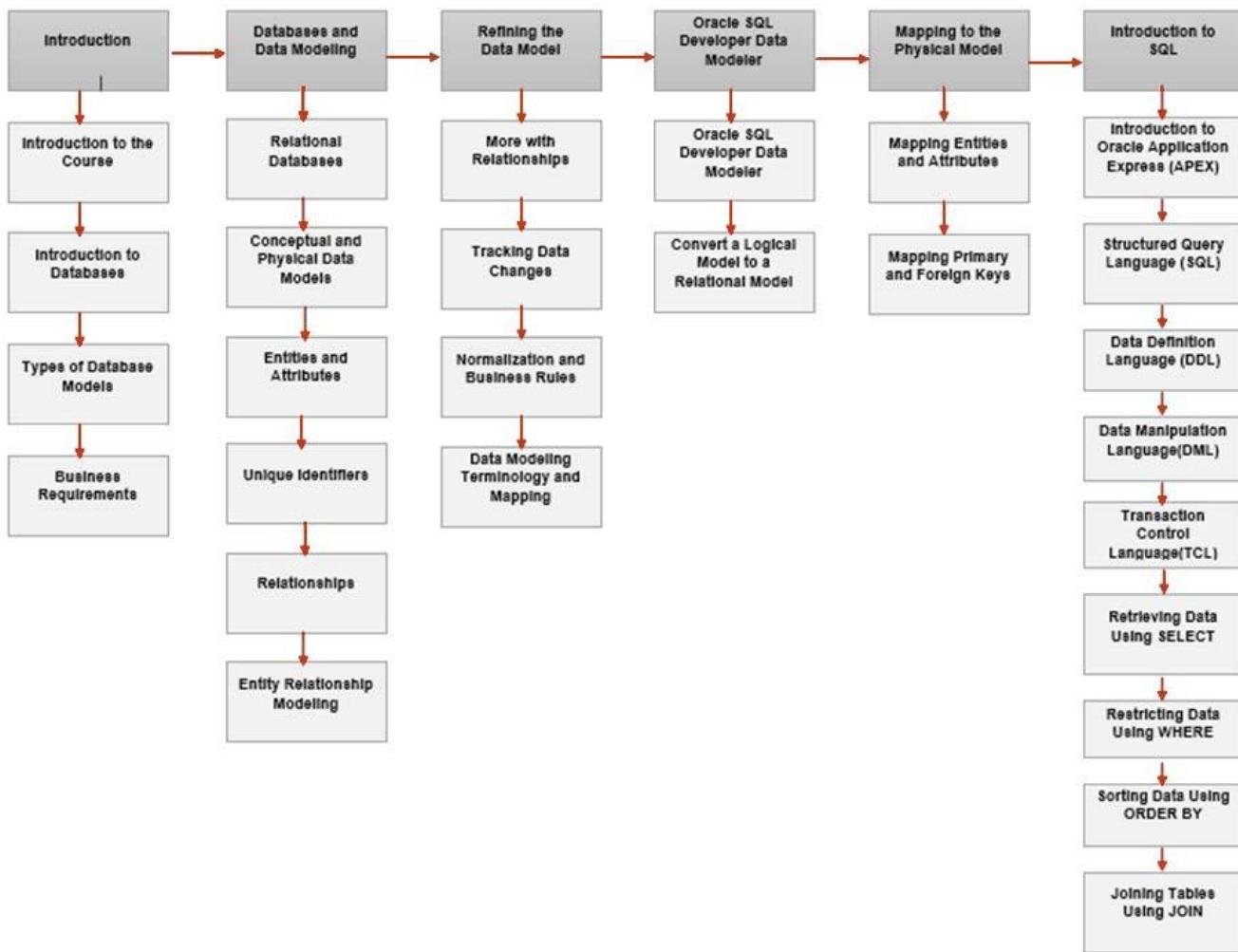
Creating the Arc Relationship	28
Creating the Barred Relationship.....	28
Creating the Hierarchical Relationship	28
Creating the Recursive Relationship	28
4.2. Convert a Logical Model to a Relational Model	30
Logical Model to a Relational Model:	30
Reverse engineering from a Relational to a Logical Model	30
5. Mapping to the Physical Model	31
5.1. Mapping Entities and Attributes	31
Create a Glossary from a Logical Model	31
Adding the Glossary as the Naming Standard	32
Applying the Naming Standard	32
5.2. Mapping Primary and Foreign Keys.....	33
Create Name Abbreviations.....	33
Defining Naming Templates.....	34
Applying Naming Abbreviations.....	34
Mapping Subtypes to Tables.....	35
6. Introduction to SQL.....	36
6.1. Introduction to Oracle Application Express.....	36
6.2. Structured Query Language (SQL)	36
6.3. Data Definition Language (DDL)	37
Data Types.....	38
Date Data Types	39
Data Integrity Constraints	39
Defining Constraints.....	40
FOREIGN KEY Constraint	40
FOREIGN KEY Constraint: Keywords	40
CHECK Constraint	41
ALTER TABLE Statement.....	41
SET UNUSED Option (Exclusive Oracle).....	41
Read-Only Tables	42
Dropping a Table	42
6.4. Data Manipulation Language (DML).....	43
6.5. Transaction Control Language (TCL)	44
6.6. Retrieving Data Using SELECT	44

6.7.	Restricting Data Using WHERE	44
6.8.	Sorting Data Using ORDER BY	44
6.9.	Joining Tables Using JOIN	44

1. Introduction




1.1. Introduction



Technological Requirements:

Oracle SQL Developer or Oracle APEX application
Oracle Data Modeler

→

1.2. Introduction to Databases

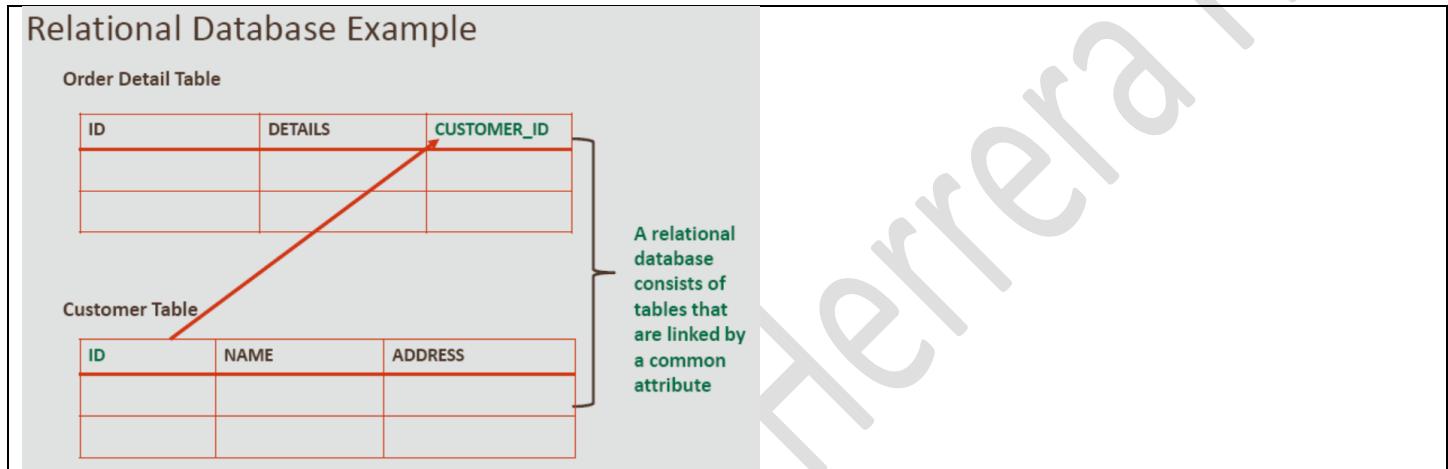
Data vs Information.

Data: Collected facts about a topic or item

Information: The result of combining, comparing, and performing calculations on data.

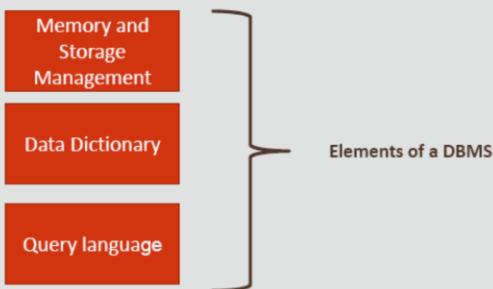
Introduction to Relational Databases

- A relational database stores information in tables with rows and columns
- A table is a collection of records
- A row is called a record (or instance)
- A record is a collection of fields
- A column is referred to as a field (or attribute)



Database Management System

- A DBMS is software that controls the storage, organization, and retrieval of data



Key Computing Terms

Hardware: physical parts of a computer

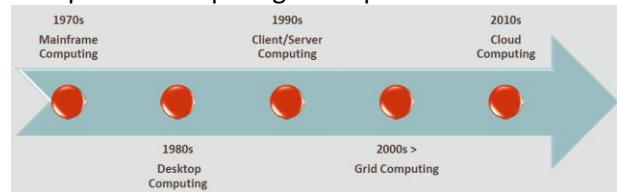
Software: instructions to tell hardware what to do

Operating system : software that directly controls the hardware

Application: performs specific task

Client : workstation used by end users

Server : accepts work requiring more power from clients

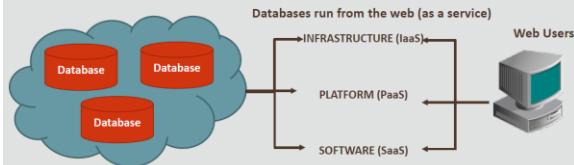


2000s: Grid Computing (Shared Processing)

- In the grid-computing model, all of an organization's computers in different locations can be utilized just like a pool of computing resources
- Grid computing builds a software infrastructure that can run on a large number of networked servers
- A user makes a request for information or computation from his or her workstation and that request is processed somewhere in the grid as efficiently as possible



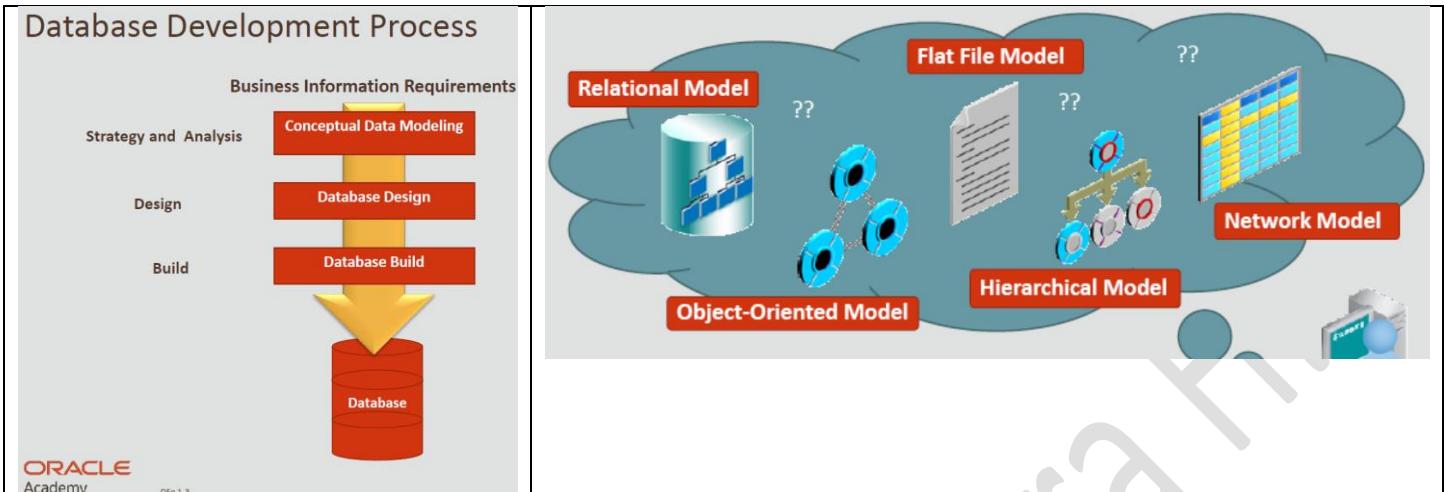
2010s: Cloud Computing (Internet Based Processing)



- Cloud computing allows the delivery of computing services over the Internet
- The three main categories of cloud services are:
 - IaaS – Allows you to rent cloud based servers, storage, operating systems etc
 - PaaS – Gives access to an online environment for developing and testing software without any setup or management costs
 - SaaS – Delivers software direct from the Internet. Users normally access it through a web browser

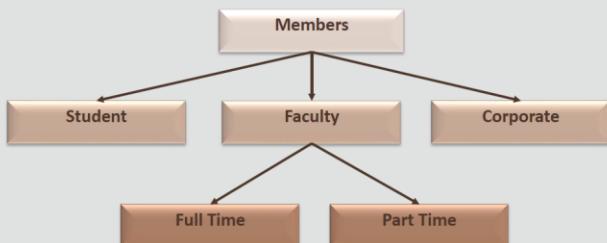
→

1.3. Types of Database Models

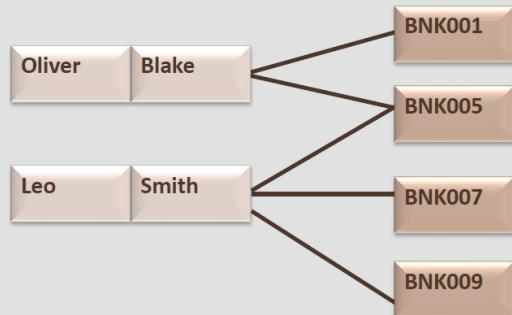


Example of a Hierarchical Model

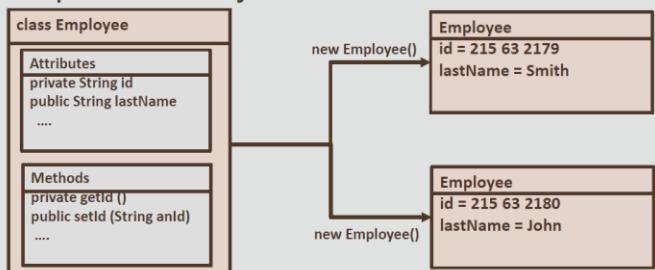
- Data is organized in a tree-like structure and stored as records that are connected to one another through links



Example of a Network Model



Example of an Object-Oriented Model



Los nombres de clases son en Singular

Los nombres de las tablas son en Plural

Example of a Relational Model

EMPLOYEE

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
200	Jennifer	Whalen	10
205	Shelley	Higgins	110

DEPARTMENT

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
50	Shipping

Foreign Key

In this example a relationship is created between the two tables using the common field of DEPARTMENT_ID



1.4. Business Requirements

Case Scenario: Need a Database Solution						Case Scenario: Possible Database Solution																																																											
<table border="1"> <thead> <tr> <th></th><th>STUDENT_ID</th><th>SPORT_1</th><th>PRICE_1</th><th>SPORT_2</th><th>PRICE_2</th></tr> </thead> <tbody> <tr> <td>Record 1</td><td>ST0001</td><td>Tennis</td><td>\$100</td><td>Badminton</td><td>\$150</td></tr> <tr> <td>Record 2</td><td>ST0002</td><td>Soccer</td><td>\$175</td><td>Tennis</td><td>\$100</td></tr> <tr> <td>Record 3</td><td>ST0003</td><td>Cycling</td><td>\$200</td><td>Badminton</td><td>\$150</td></tr> <tr> <td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td></tr> </tbody> </table>							STUDENT_ID	SPORT_1	PRICE_1	SPORT_2	PRICE_2	Record 1	ST0001	Tennis	\$100	Badminton	\$150	Record 2	ST0002	Soccer	\$175	Tennis	\$100	Record 3	ST0003	Cycling	\$200	Badminton	\$150	<table border="1"> <thead> <tr> <th colspan="3">Student Details Table</th> </tr> <tr> <th>ID</th><th>FIRST_NAME</th><th>LAST_NAME</th></tr> </thead> <tbody> <tr> <td>ST0001</td><td>Sean</td><td>Smith</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3">Sport Details Table</th> </tr> <tr> <th>ID</th><th>NAME</th><th>PRICE</th></tr> </thead> <tbody> <tr> <td>TN001</td><td>Tennis</td><td>\$100</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="3">Participant Details Table</th> </tr> <tr> <th>STUDENT_ID</th><th>SPORT_ID</th><th>SEMESTER_DETAILS</th></tr> </thead> <tbody> <tr> <td>ST0001</td><td>TN001</td><td>Fall2017</td></tr> </tbody> </table>			Student Details Table			ID	FIRST_NAME	LAST_NAME	ST0001	Sean	Smith	Sport Details Table			ID	NAME	PRICE	TN001	Tennis	\$100	Participant Details Table			STUDENT_ID	SPORT_ID	SEMESTER_DETAILS	ST0001	TN001	Fall2017
	STUDENT_ID	SPORT_1	PRICE_1	SPORT_2	PRICE_2																																																												
Record 1	ST0001	Tennis	\$100	Badminton	\$150																																																												
Record 2	ST0002	Soccer	\$175	Tennis	\$100																																																												
Record 3	ST0003	Cycling	\$200	Badminton	\$150																																																												
.....																																																												
Student Details Table																																																																	
ID	FIRST_NAME	LAST_NAME																																																															
ST0001	Sean	Smith																																																															
Sport Details Table																																																																	
ID	NAME	PRICE																																																															
TN001	Tennis	\$100																																																															
Participant Details Table																																																																	
STUDENT_ID	SPORT_ID	SEMESTER_DETAILS																																																															
ST0001	TN001	Fall2017																																																															
						<p>Flat file was split into three tables eliminating issues related to:</p> <ul style="list-style-type: none"> • Redundancy • Data entry anomalies • Inconsistency 																																																											

Importance of Business Rules

It is important to identify and document business rules when designing a database

Business rules:

- Allow the developer/architect to understand the relationship and constraints of the participating entities
- Help you understand the standardization procedure that an organization follows when handling huge data
- Should be simple and easy to understand
- Must be kept up-to-date

Note: Not all business rules can be modeled in a database, but must be documented

Case Scenario: Identifying Key Business Rules, Problems, and Assumptions

- Business rule: Used to understand business processes and the nature, role, and scope of the data
- Assumption: Can be defined as a fact or a statement that has been taken for granted
- Problem: Can be defined as a situation or scenario that requires attention and a possible solution to alleviate the situation

Example:

Note	Business Rule	Assumption	Problem
To ensure that new book arrivals happen on the 21 st of every month.			
Librarian cannot easily identify DVDs that are seriously overdue (more than two weeks late).			
Our current system probably uses Oracle Database 10g and is on UNIX.			

Identify the statements as a business rule, a problem, or an assumption.



2. Databases and Data Modeling

2.1. Relational Databases

Relational Database: Example

STUDENTS

ID	LAST_NAME	DATE_OF_BIRTH	ADDRESS	COURSE_ID

Primary Key

Foreign Key

Relationship

Each table is assigned a PRIMARY_KEY column which uniquely identifies the entity instance

A PRIMARY_KEY column in one table is designated as a FOREIGN_KEY column in a related table to form a relationship between the tables

COURSES

ID	NAME	DURATION

This relationship between the STUDENTS table and the COURSES table lets you store the data and query it to determine the specific courses that a student is attending (or has attended)

Relational Tables

- A table is a simple structure where data is organized and stored

Table: EMPLOYEES

columns

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	DEPARTMENT_ID	PAYROLL_ID	NICKNAME
100	SMITH	DANA	10	21215	Dana
310	ADAMS	TYLER	15	59877	Ty
210	CHEN	LAWRENCE	10	1101	Larry
405	GOMEZ	CARLOS	10	52	Chaz
378	LOUNGANI	NEIL	22	90386	Neil

Primary Key
Column (PK)

Foreign Key
Column (FK)

Unique Key
Column (UK)

Rules for Relational Database Tables

- Each table has a distinct name
- Each table may contain multiple rows
- Each table has a value to uniquely identify the rows
- Each column in a table has a unique name
- Entries in columns are single values
- Entries in columns are of the same kind
- Order of rows and columns is insignificant

Key Terms

Table –A basic storage structure

Column –attribute that describes the information in the table

Primary Key –the unique identifier for each row

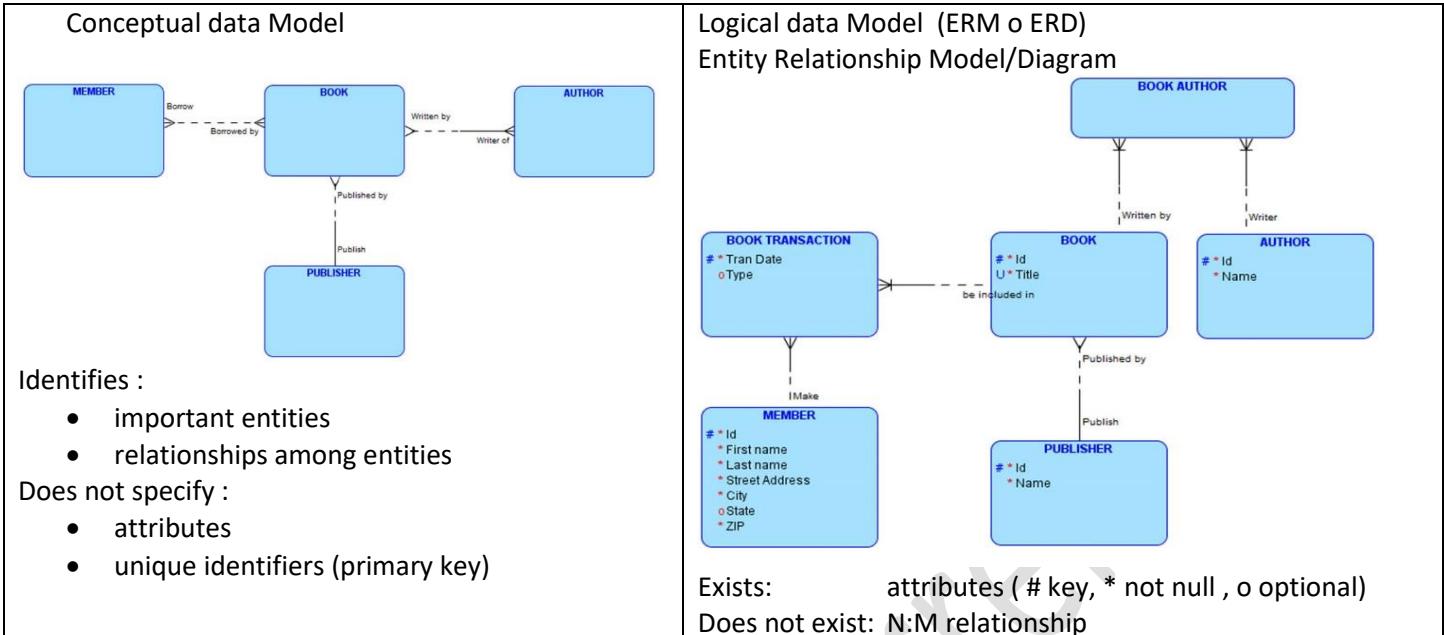
Foreign Key –a column that refers to a primary key column in another table

Row –data for one table instance

Field –the one value found at the intersection of a row and column

→

2.2. Conceptual and Physical Data Models

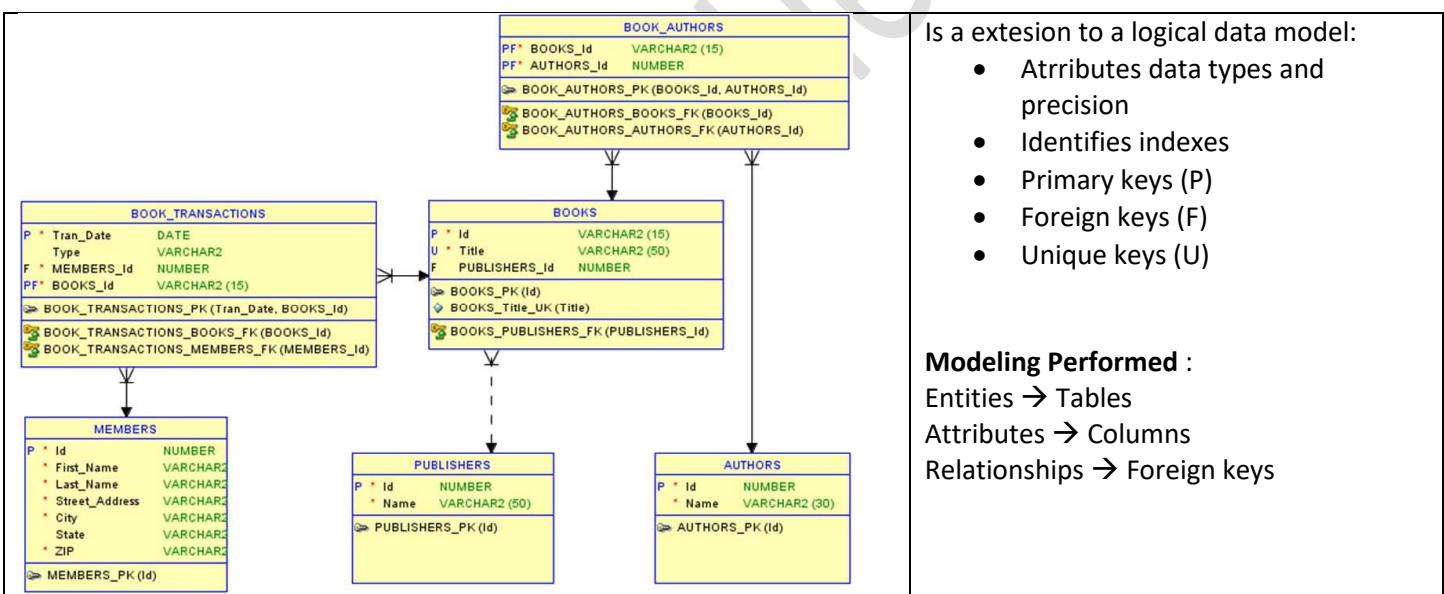


Physical data Model: Entities ->

Relationships ->

Attributes ->

Constrains



→

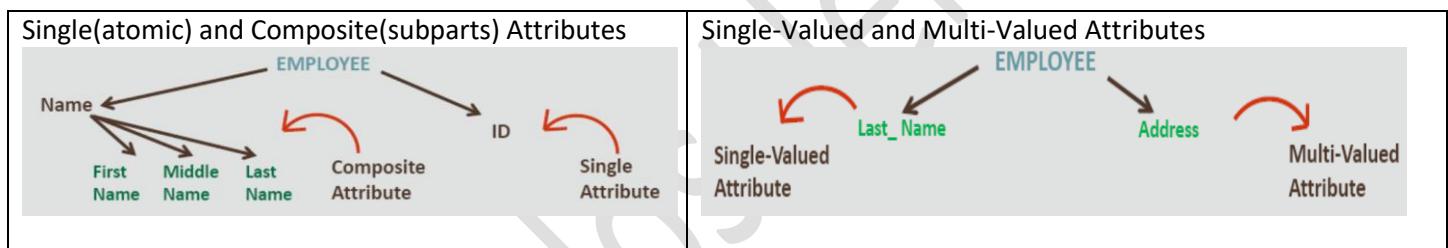
2.3. Entities and Attributes

Identify UID(#), mandatory(*), optional(o), volatile or derivate(age), and nonvolatile(birthDate) attributes

Entity Types

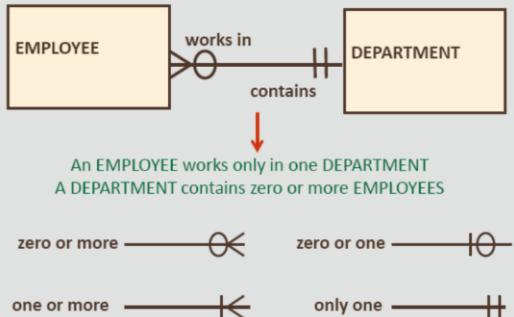
An entity can be classified as one of the following types:

Name	Description	Example	Tipos de Entidad: Principal Característica Intersección
Prime	Exists independently	CUSTOMER, INSTRUCTOR	
Characteristic	Exists because of another (prime) entity	ORDER, CLASS OFFERING	
Intersection	Exists because of two or more entities	ORDER ITEM, CLASS ENROLLMENT	Entidades: Fuertes Débiles



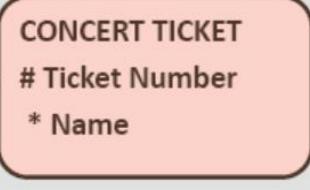
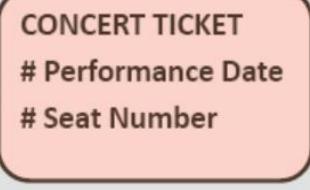
Data Model Notations: Examples

Barker Notation		Continua: obligatorio Discontinua: opcional
Bachman Notation		Black: debe White: puede Arrow: many
Information Engineering Notation		Círculo: opcional Pipe: obligatorio

Information Engineering Notation		Data Model Notations			
		Notation (Read left to right)	Barker Notation	Bachman Notation	Information Engineering
		Zero or one	- - - - -	0	0
An EMPLOYEE works only in one DEPARTMENT A DEPARTMENT contains zero or more EMPLOYEES		Only one	- - - - -	•	•
zero or more	zero or one	Zero or more	- - - - -	0	0
one or more	only one	One or more	- - - - -	•	•
		Primary Key/Unique key	#	P	

Note: Barker notation is used for this course

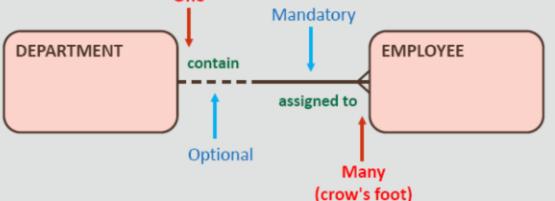
2.4. Unique Identifiers

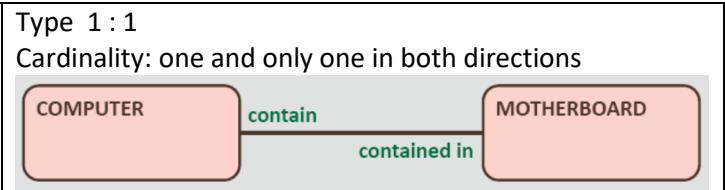
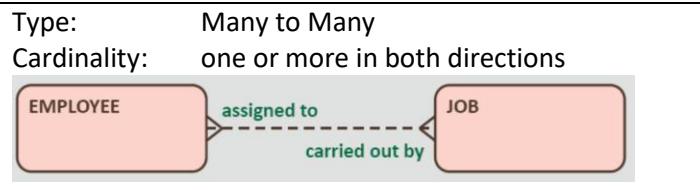
Simple UID		SHOE Color Size Style	
Composite UID			
Each entity must have a unique identifier Otherwise, it is not an entity			Candidate unique identifiers or secondary UIDs, so define primary key
It has a non-Null It has a value that never changes			

Artificial UIDs do not occur in the natural world but are created for identification purposes in a system

Example Composite UID: Bank_No and Account_No.

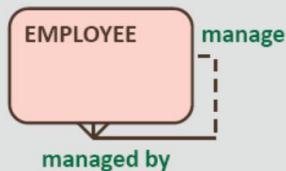
2.5. Relationships

	Relationship Types or Cardinality
Type: 1 : M Cardinality: one and only one in one direction and one or more in other direction Optional: Use "may be" or "may." Mandatory: Use "must be" or "must." Line: Use "one and only one." Crow's feet: Use "one or more."	Many-to-one (M : 1) or one-to-many (1 : M) Many-to-many (M : M) One-to-one (1 : 1)



Recursive Relationships

- A recursive relationship is a relationship with an entity and itself



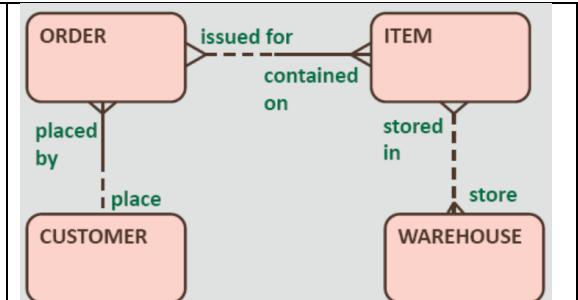
- Business rules:
 - Each EMPLOYEE may manage one or more EMPLOYEE
 - Each EMPLOYEE must be managed by one and only one EMPLOYEE

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	101	Kochhar	100
2	201	Hartstein	100
3	124	Mourgos	100
4	149	Zlotkev	100
5	102	De Haan	100
6	200	Whalen	101
7	205	Higgins	101
8	103	Hunold	102
9	104	Ernst	103
10	107	Lorentz	103
11	142	Davies	124
12	144	Vargas	124
13	143	Matos	124
14	141	Rais	124
15	176	Taylor	149
16	174	Abel	149
17	178	Grant	149
18	202	Fav	201
19	206	Gietz	205
20	100	King	(null)

Relationship Matrix: Mapping the Contents

	CUSTOMER	ITEM	ORDER	WAREHOUSE
CUSTOMER			place	
ITEM			contained on	stored in
ORDER	placed by	issued for		
WAREHOUSE		store		

A relationship matrix can be used to collect initial information about the relationships among a set of entities

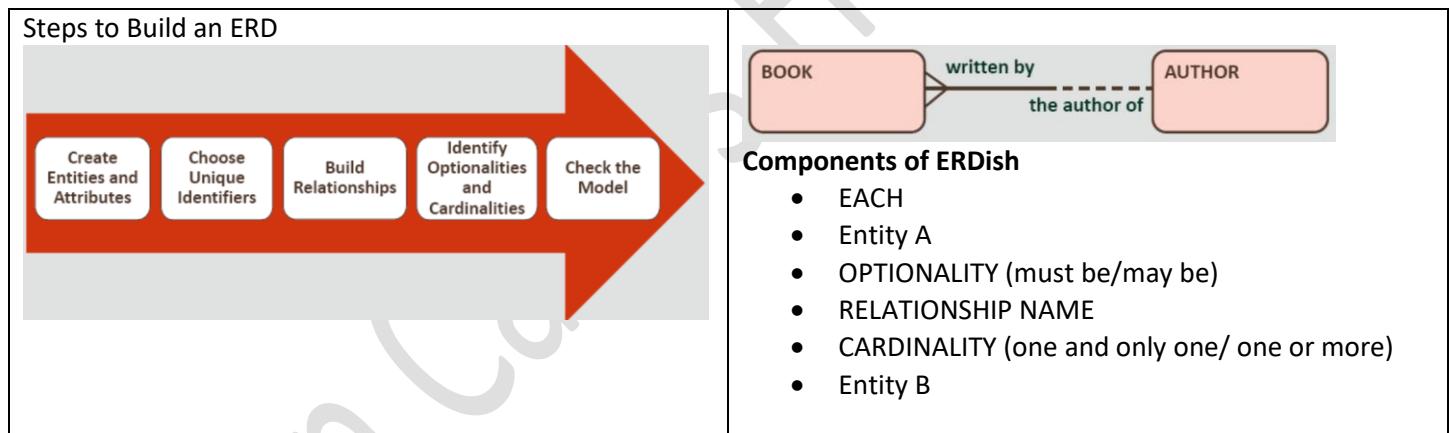
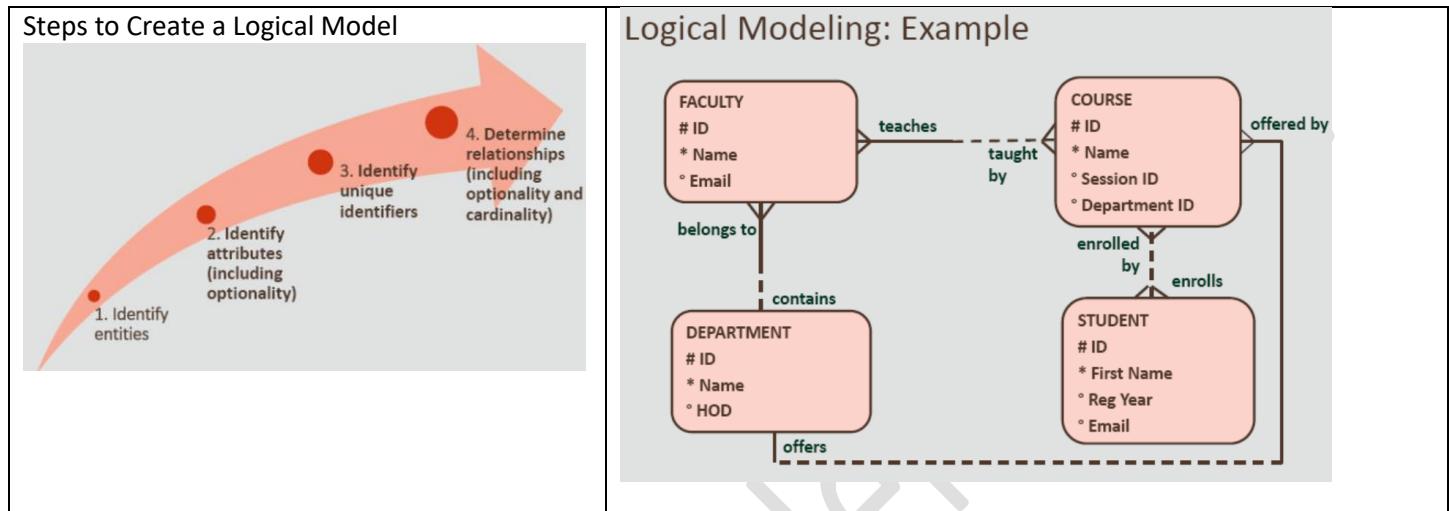


2.6. Entity Relationship Modeling (ERDs)

DB roles: designers, database administrators, and application developers

Logical Modeling:

Includes all entities, attributes, UIDs and relationships as well as optionality and cardinality of these items



ERDish Example

Because a relationship has two sides, first read one side from left to right.



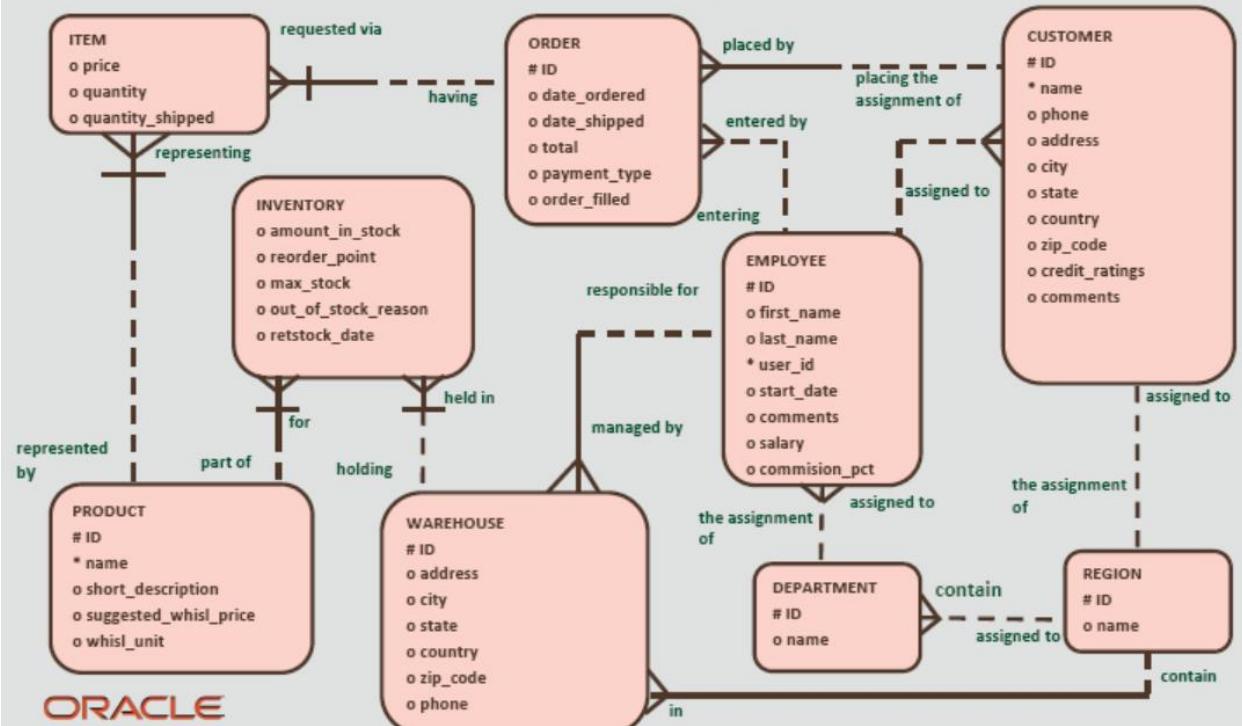
1. EACH
2. BOOK (entity A)
3. MUST BE (optionality, solid line)
4. WRITTEN BY (relationship name)
5. ONE (AND ONLY ONE) (cardinality, single toe)
6. AUTHOR (entity B)



1. EACH
2. AUTHOR (entity B)
3. MAY BE (optionality, dotted line)
4. THE AUTHOR OF (relationship name)
5. ONE OR MORE (cardinality, crow's foot)
6. BOOK (entity A)

Next, read the relationship from right to left.

Sample Solution for Sporting Goods ERD



ORACLE
Academy

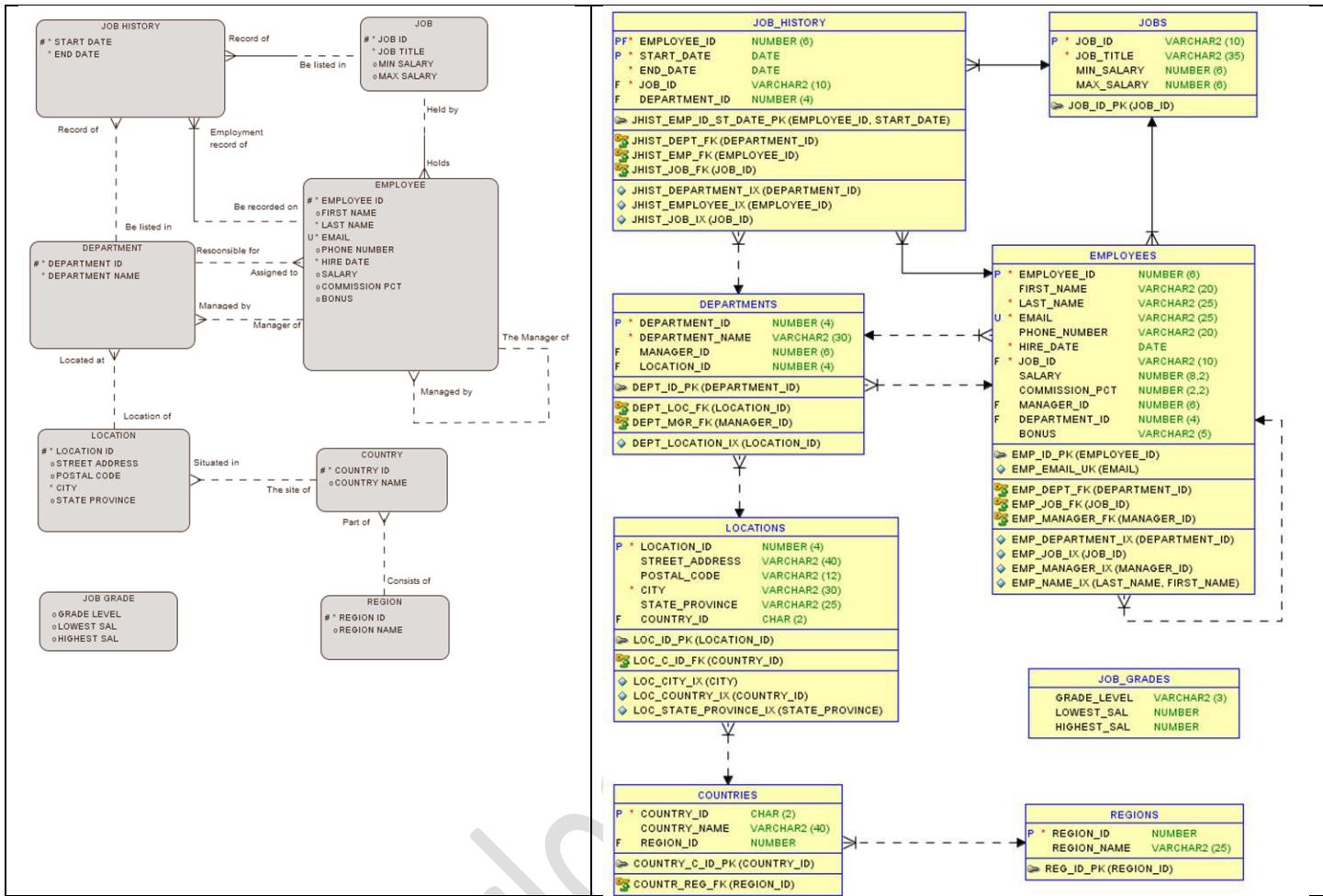
DFo 2-6
Entity Relationship Modeling (ERDs)

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

25

Logical Data Model

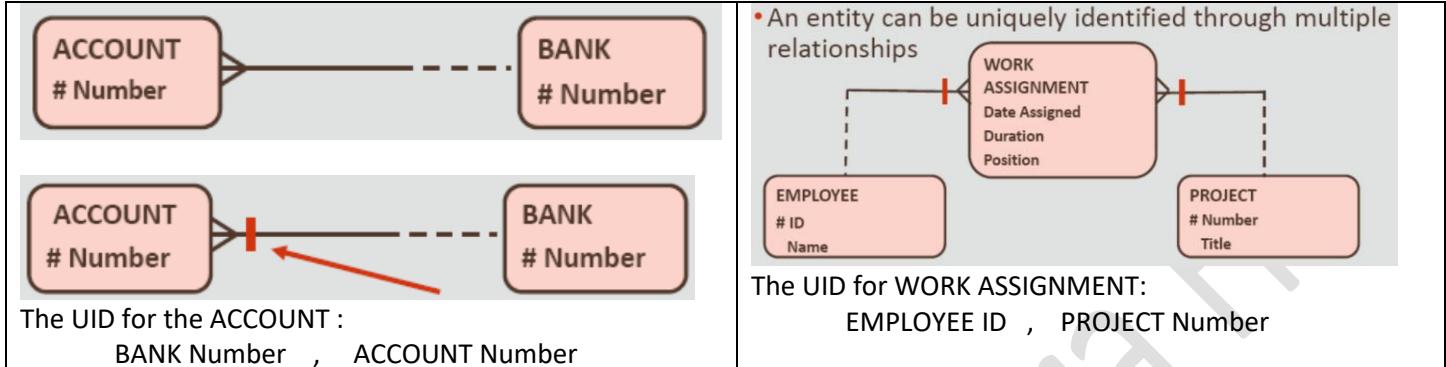
Physical data Model



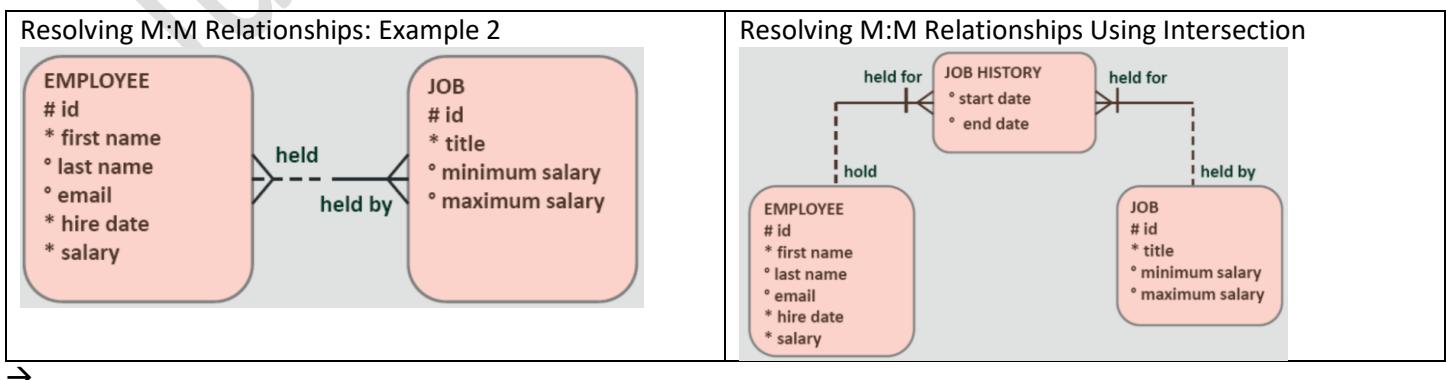
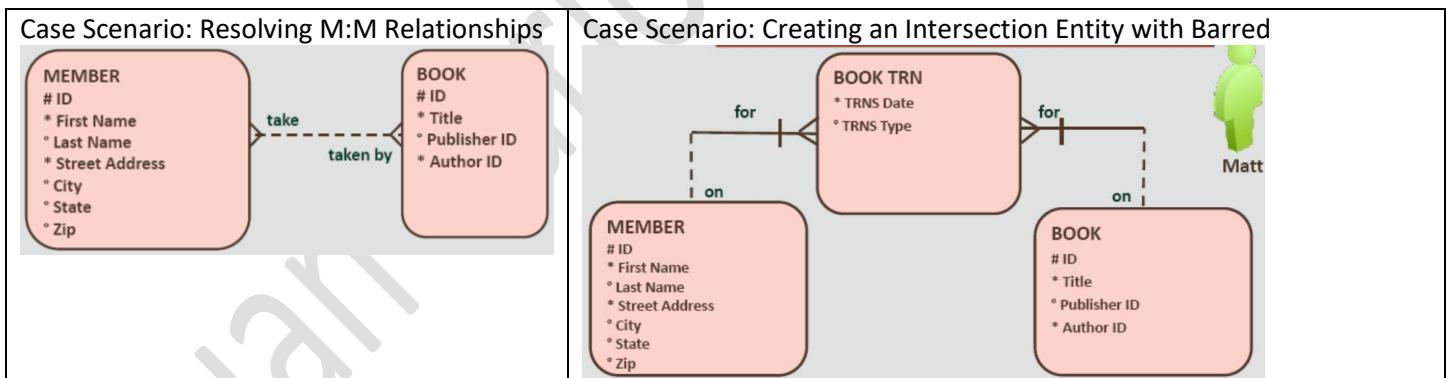
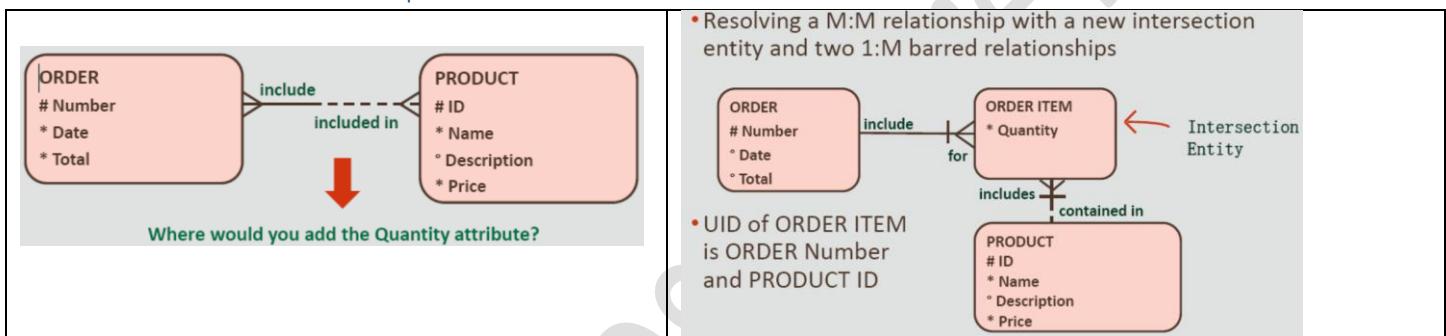
3. Refining the Data Model

3.1. More with Relationships

3.1.1. Identifying (| Barred) Relationships



3.1.2. M:M Relationships

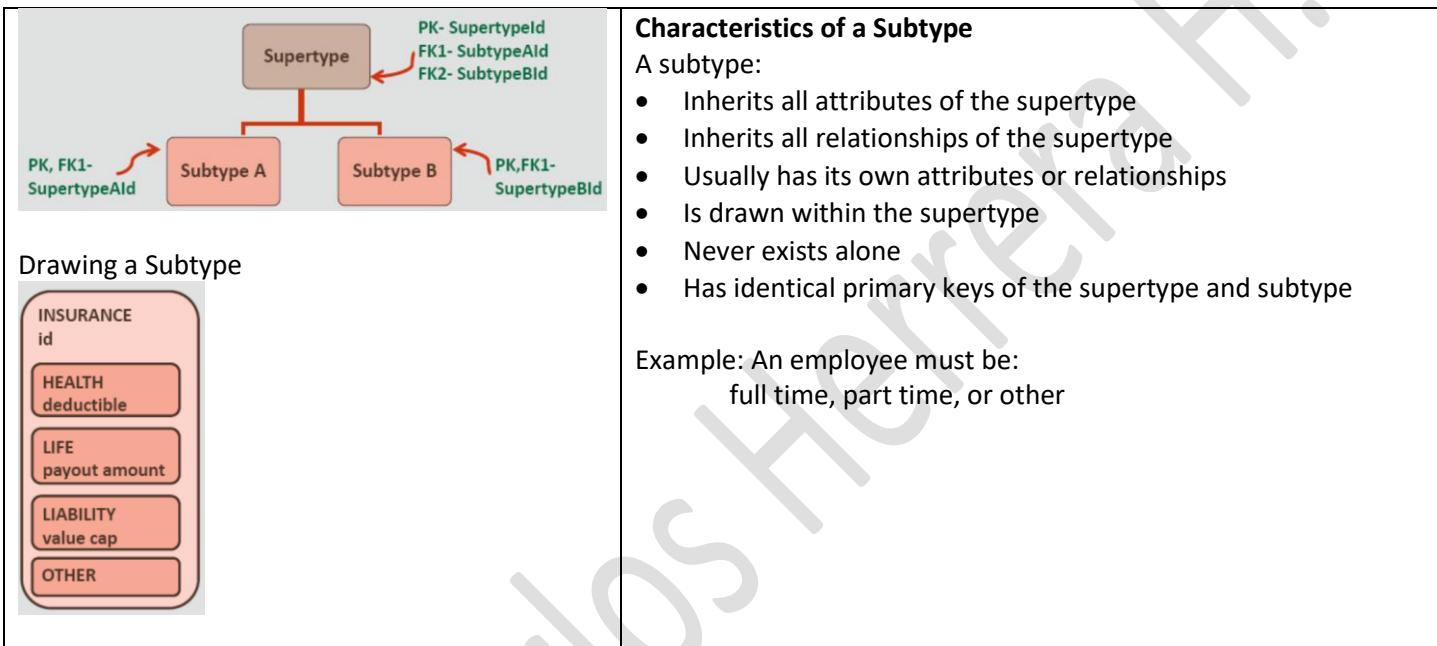


3.1.3. Non-Transferable Relationships ◇



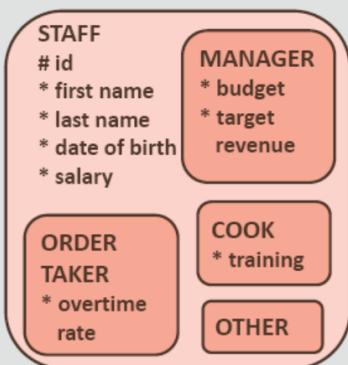
3.1.4. Supertype and Subtype Entities

- Supertype has a parent-child relationship with one or more subtypes
- Subtype is a subgrouping of the entity in an entity type which has attributes that are distinct from those in other subgroupings



Identifying Subtypes Correctly

- Is this subtype a kind of supertype?
- Have I covered all possible cases? (exhaustive)
- Does each instance fit into one and only one subtype? (mutually exclusive)

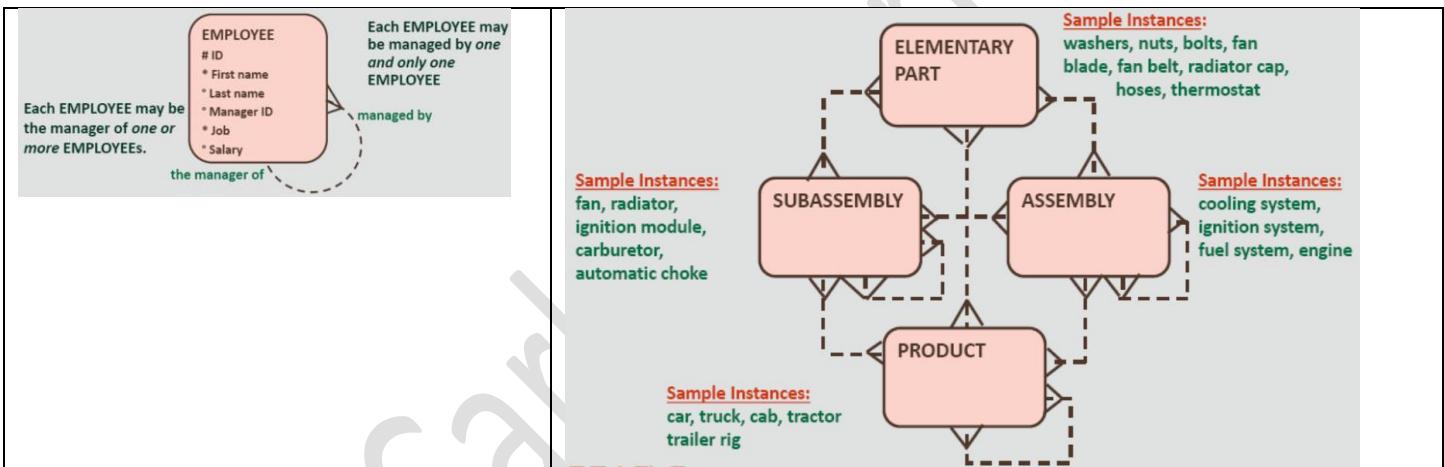


3.1.5. Modeling Hierarchical Data

<pre> graph TD COMPANY[COMPANY # ID] -- "made up of" --> DIVISION[DIVISION # Div ID] DIVISION -- "made up of" --> DEPARTMENT[DEPARTMENT # Dept ID] DEPARTMENT -- "made up of" --> TEAM[TEAM # Team ID] style COMPANY fill:#f0e6d2,stroke:#333,stroke-width:1px style DIVISION fill:#f0e6d2,stroke:#333,stroke-width:1px style DEPARTMENT fill:#f0e6d2,stroke:#333,stroke-width:1px style TEAM fill:#f0e6d2,stroke:#333,stroke-width:1px </pre>	<p>Resolver: La jerarquía del Hotel</p> <p>En un hotel se tienen varias recamaras en cada departamento o suite(1,2 o 3), cuenta con 2 edificios, uno de 15 pisos y otro de 20 pisos.</p>
---	---

3.1.6. Recursive Relationships

- A recursive relationship is always modeled with a loop.
- A recursive relationship is one where an entity instance is related to another instance in the same entity

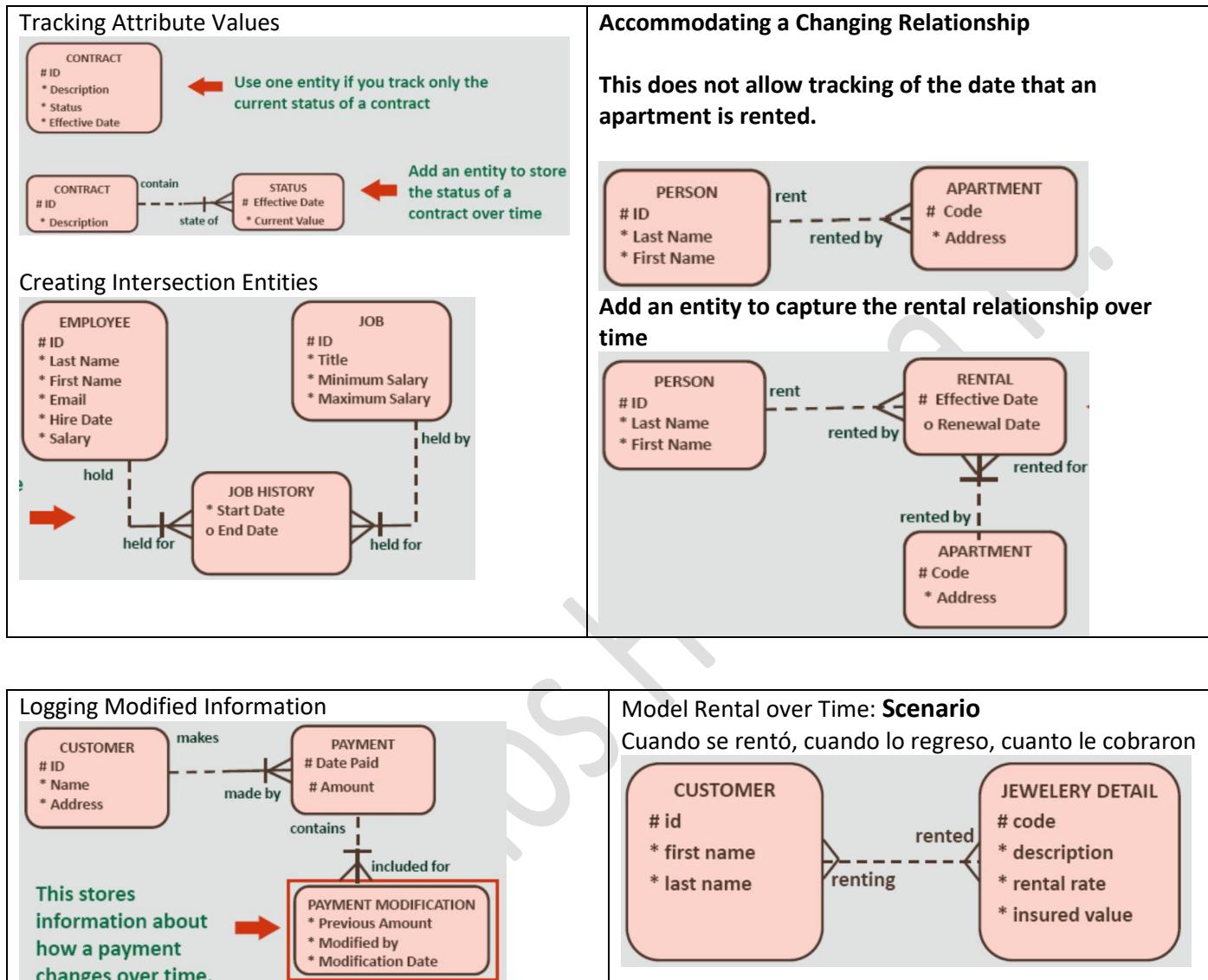


3.1.7. Arc Relationship ↗

<pre> graph TD EMPLOYEE[EMPLOYEE] -- "be" --> CURRENTEMPLOYEE[CURRENT EMPLOYEE] EMPLOYEE -- "be" --> EXEMPLOYEE[EX-EMPLOYEE] style EMPLOYEE fill:#f0e6d2,stroke:#333,stroke-width:1px style CURRENTEMPLOYEE fill:#f0e6d2,stroke:#333,stroke-width:1px style EXEMPLOYEE fill:#f0e6d2,stroke:#333,stroke-width:1px </pre>	<p>An arc is an exclusive relationship group, which is defined such that only one of the relationships can exist for any instance of an entity</p> <p>Exor</p> <p>A supertype entity and its subtypes can be modeled as an arc relationship</p>
---	---

3.2. Tracking Data Changes

Objetive: Keep track of data that changes over time



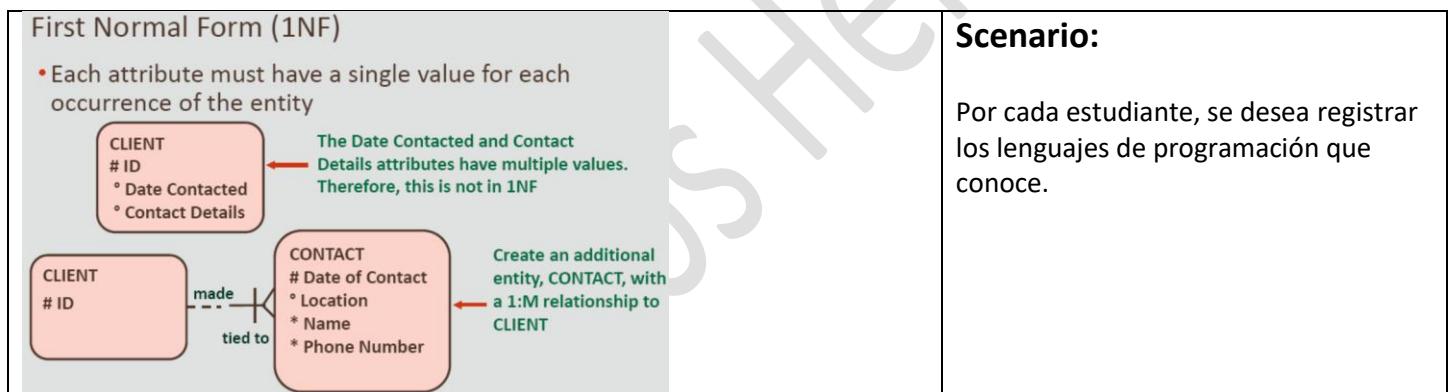
3.3. Normalization and Business Rules

Normalization: Is the process of organizing the attributes and tables of a relational database to minimize redundancy.

No duplicate content, Increase the integrity of data.

Rule	Description
First Normal Form (1NF)	All attributes must be single-valued. (no multi-valued = atomic)
Second Normal Form (2NF)	An attribute must be dependent on its entity's entire UID. No existen dependencias parciales
Third Normal Form (3NF)	No non-UID attributes can be dependent on another non-UID attribute. Ningún atributo no UID puede depender de otro atributo no UID. No existen dependencias transitivas.

1NF : If an attribute is multi-valued, create an additional entity and relate it to the original entity with a 1:M relationship



2NF : If an attribute is not dependent on the entire UID, create an additional entity with the partial UID



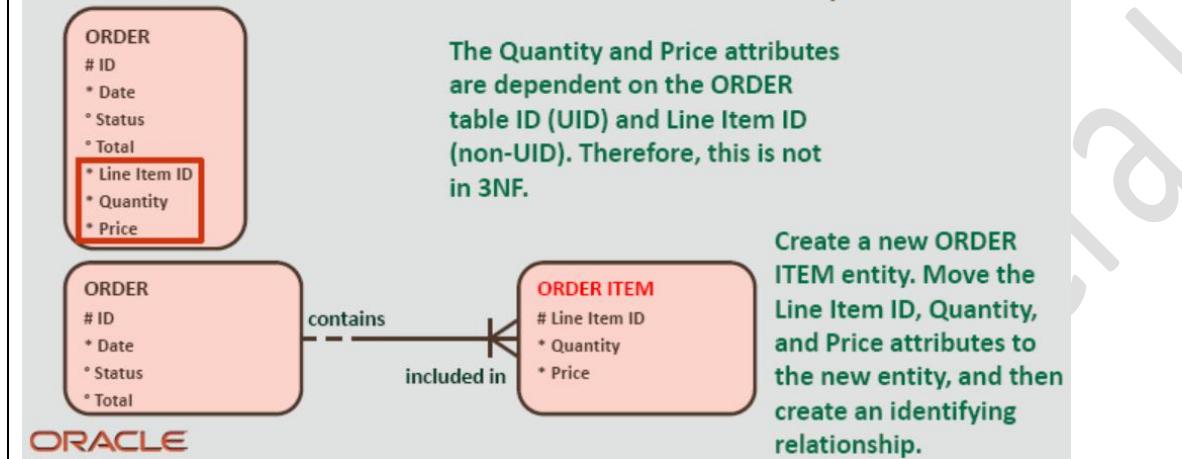
The Bank Location attribute is dependent on BANK, not on ACCOUNT. Therefore, this is not in 2NF.
Move the attribute to the BANK entity

3NF : You need to move any non-UID attribute that is dependent on another non-UID attribute into a new entity

A transitive dependency exists when any attribute in an entity is dependent on any other non-UID attribute in that entity

Third Normal Form (3NF)

- Each attribute depends only on the UID of its entity
- Move any non-UID attribute that is dependent on another non-UID attribute into a new entity



Business Rules

A business rule is a statement that defines or constrains some aspect of the business

There are two types of business rules:

- Structural: These rules can always be diagrammed in the ERD
- Procedural: Some procedural business rules cannot be diagrammed. But must still be documented.

Example: Event A must happen before event B

La aprobación debe estar firmada por el director.

La tienda no acepta entregar pedido al día siguiente, si se recibe después de las 3pm.

El libro debe entregarse antes de las 4 días.

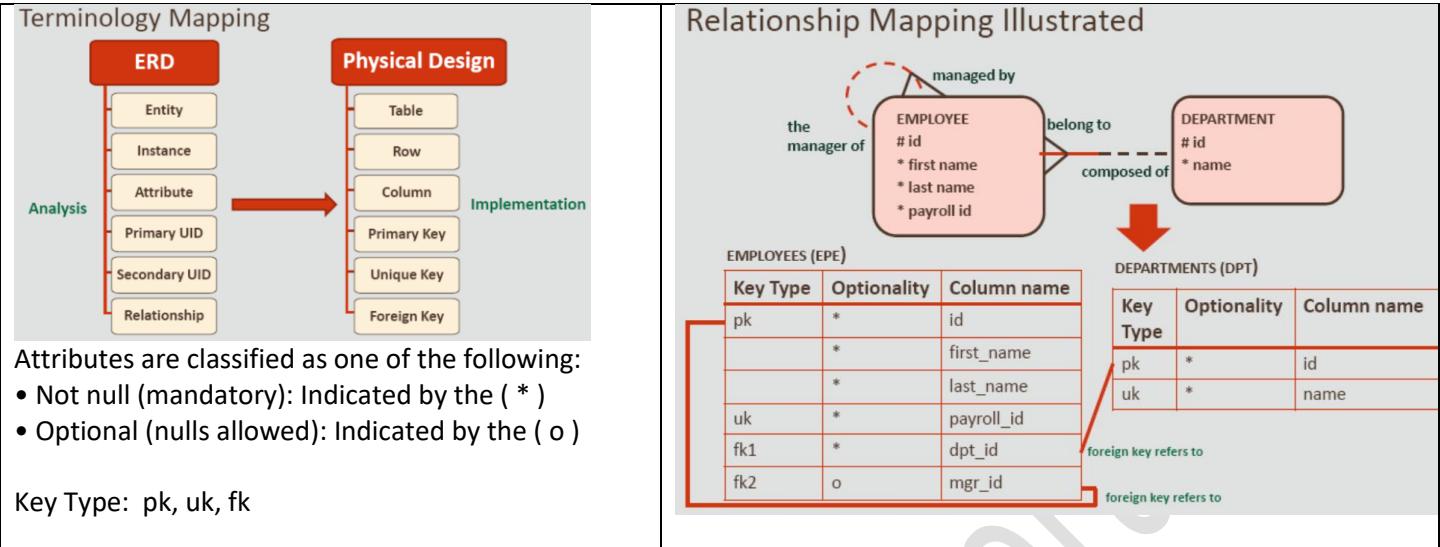
No se permite prestar otro libro, si debe libros con fecha expirada.

Las horas extras deberán pagarse al 1.5 veces la tarifa.

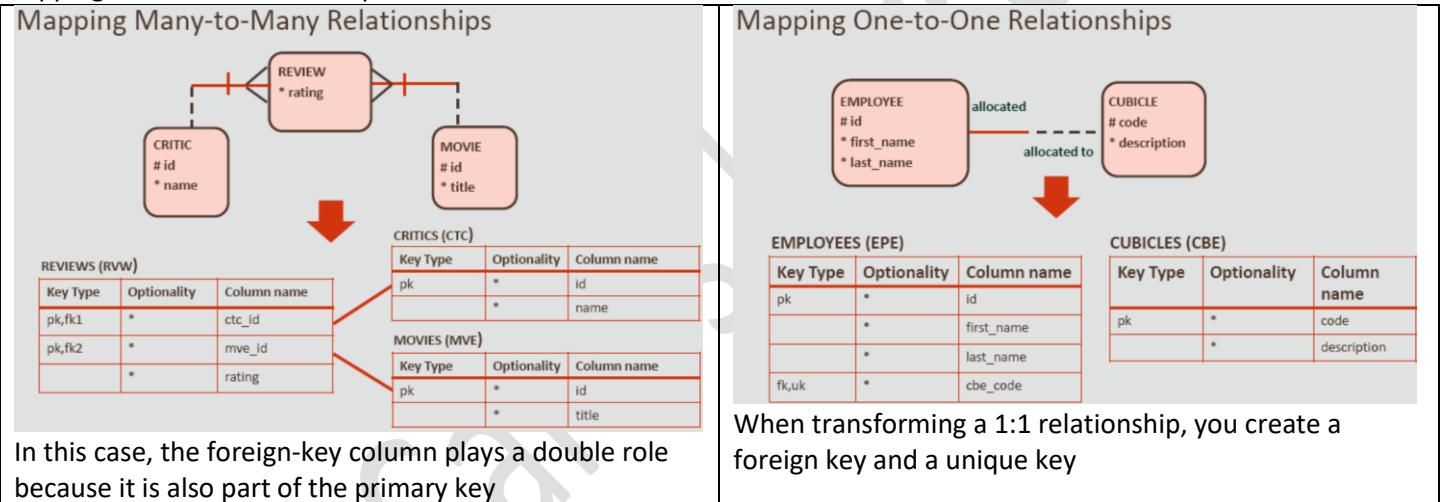
Written documentation

- Procedures
- Standards
- Operations manuals

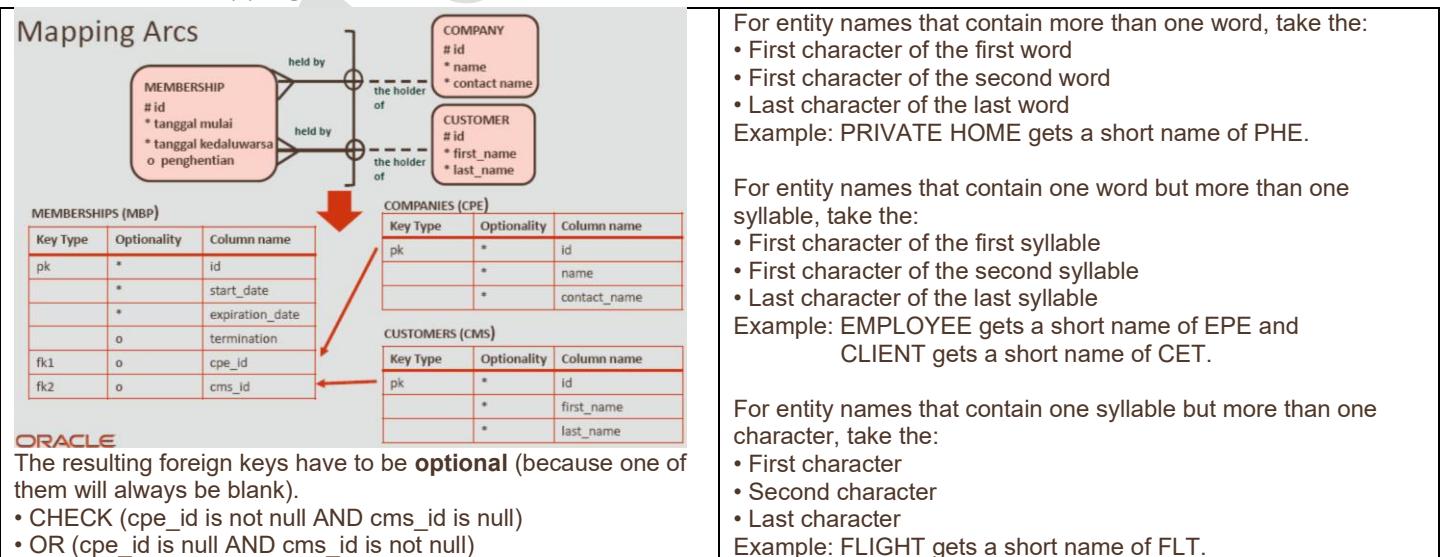
3.4. Data Modeling Terminology and Mapping



Mapping of Barred Relationships



3.4.1. Mapping Arcs

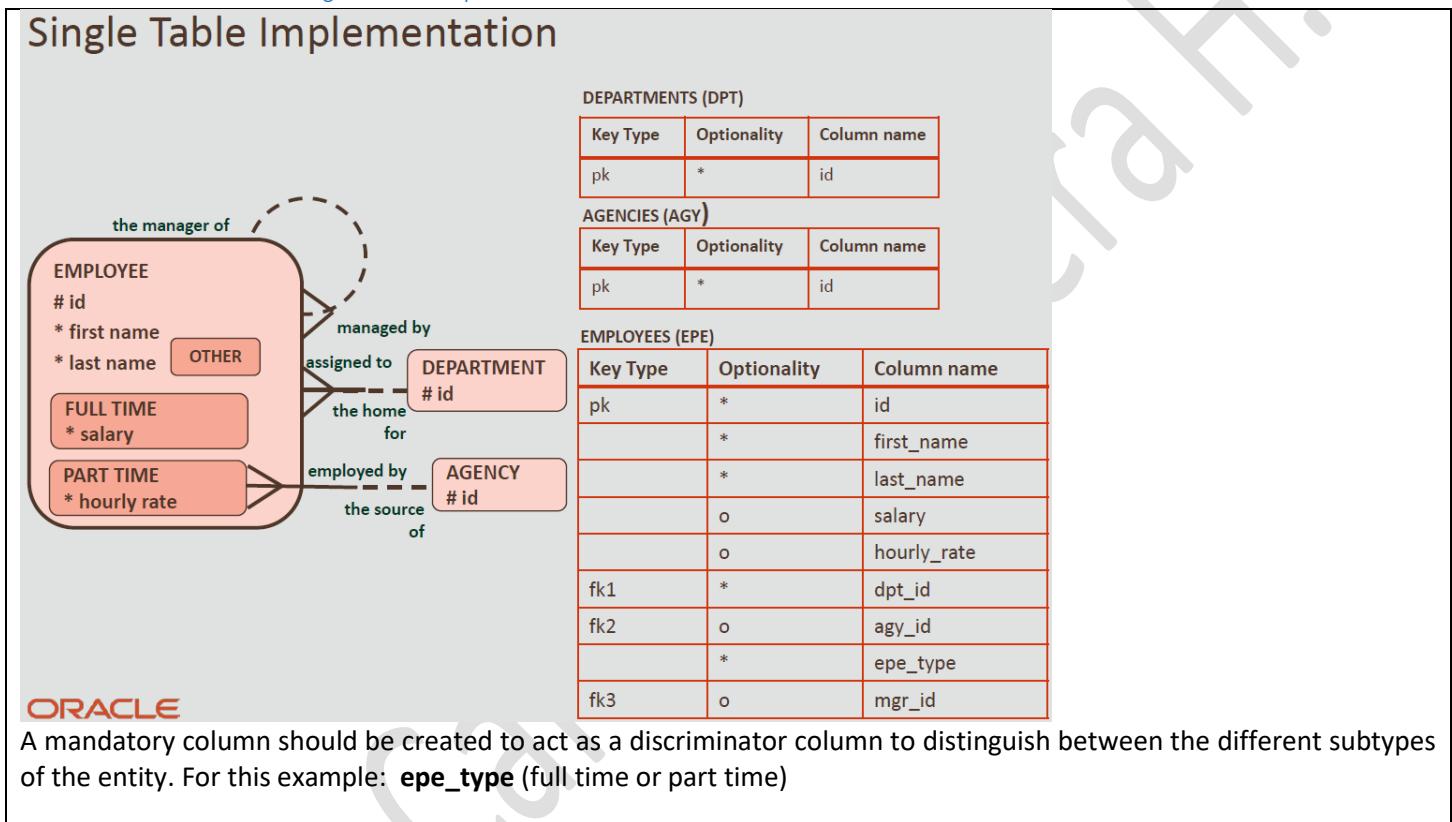


3.4.2. Mapping Supertype/Subtypes

Supertype/subtype entities can be mapped in multiple ways:

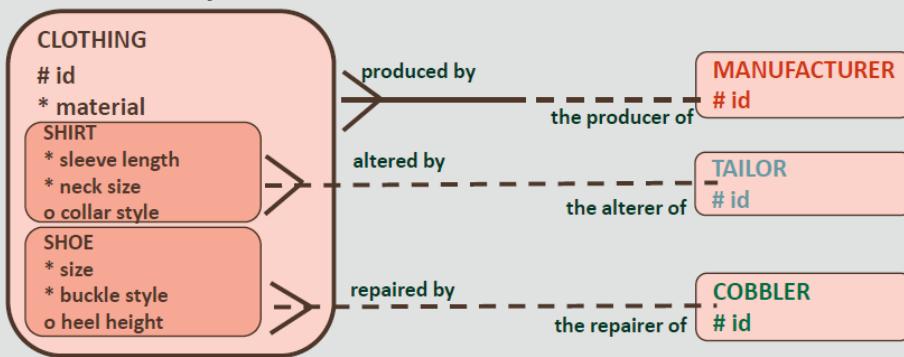
- Single table implementation : one table is created regardless of number of subtypes, used when most of the attributes and relationships are shared and therefore at the supertype level
- Two table implementation : a table is created for each of the subtypes (so there can be more than two tables), used when subtypes have little in common and few shared attributes and relationships

3.4.2.1. Single Table Implementation



3.4.2.2. Two Table Implementation

Two Table Implementation



SHIRTS (SHT)

Key Type	Optionality	Column name
pk	*	id
	*	material
	*	sleeve_length
	*	neck_size
	o	collar_style
fk1	o	tlr_id
fk2	*	mnr_id

SHOES (SHE)

Key Type	Optionality	Column name
pk	*	id
	*	material
	*	size
	*	buckle_style
	o	heel_height
fk1	o	clr_id
fk2	*	mnr_id

refers to tailors

refers to manufacturers

refers to cobblers

ORACLE

Academy

DFo 3-4
Data Modeling Terminology and Mapping

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

29

For relationships at the subtype levels, the foreign key is implemented in the table it is mapped to, original optionality is retained. (tailors, cobblers)

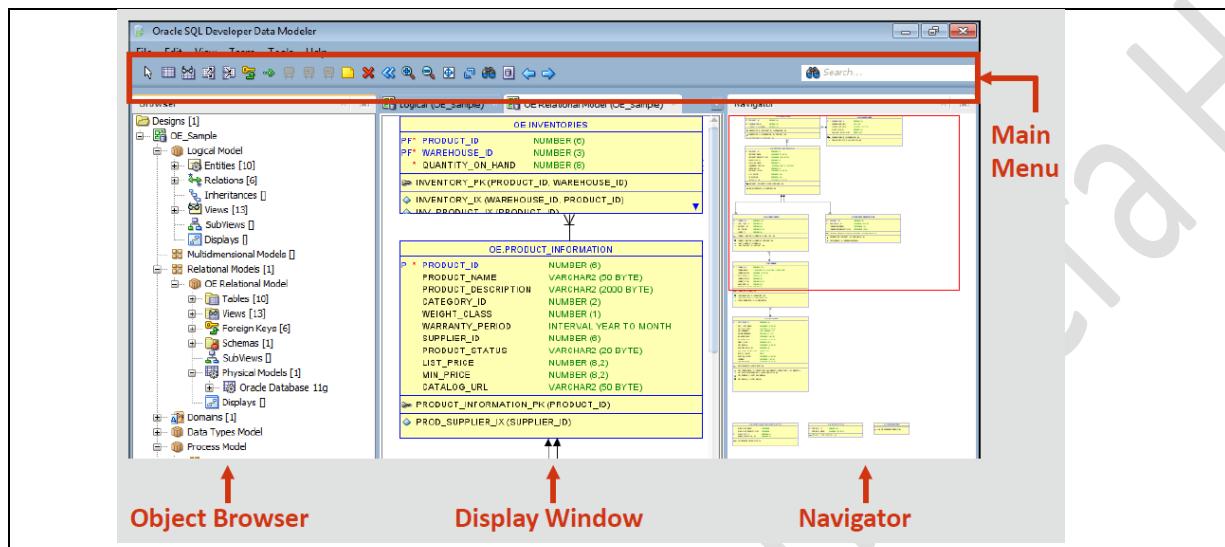
→

4. Oracle SQL Developer Data Modeler

4.1. Oracle SQL Developer Data Modeler

Use Oracle SQL Developer Data Modeler to create:

- Entities, attributes, and UIDs with correct optionality and cardinality
- Supertype and subtype entities
- Arc, hierarchical, barred, and recursive relationships



Building an ERD

a. Create entities
b. Add attributes and UIDs
c. Define relationship between entities
d. Set the source and target values for the relationship

Tools -> Preferences -> Data Modeler -> Model Boolean, Date, NUMERIC, VARCHAR

The preferences dialog for the Data Modeler shows settings for RDBMS type (Oracle Database 11g), columns & attributes defaults (Allow Similar Data Types for Foreign Keys), and preferred domains and logical types. Preferred domains include Boolean, Date, NUMERIC, and VARCHAR.

Entity -> Attributes -> Attributes Details -> Preferred

Set Primary and Secondary UIDs

The Entity Properties dialog for the 'STUDENT' entity shows the 'Attributes' tab. An attribute named 'ID' is selected, with its data type set to 'NUMBER'. The 'Primary UID' checkbox is checked, indicating it is a primary key.

• Define the relationships between the entities

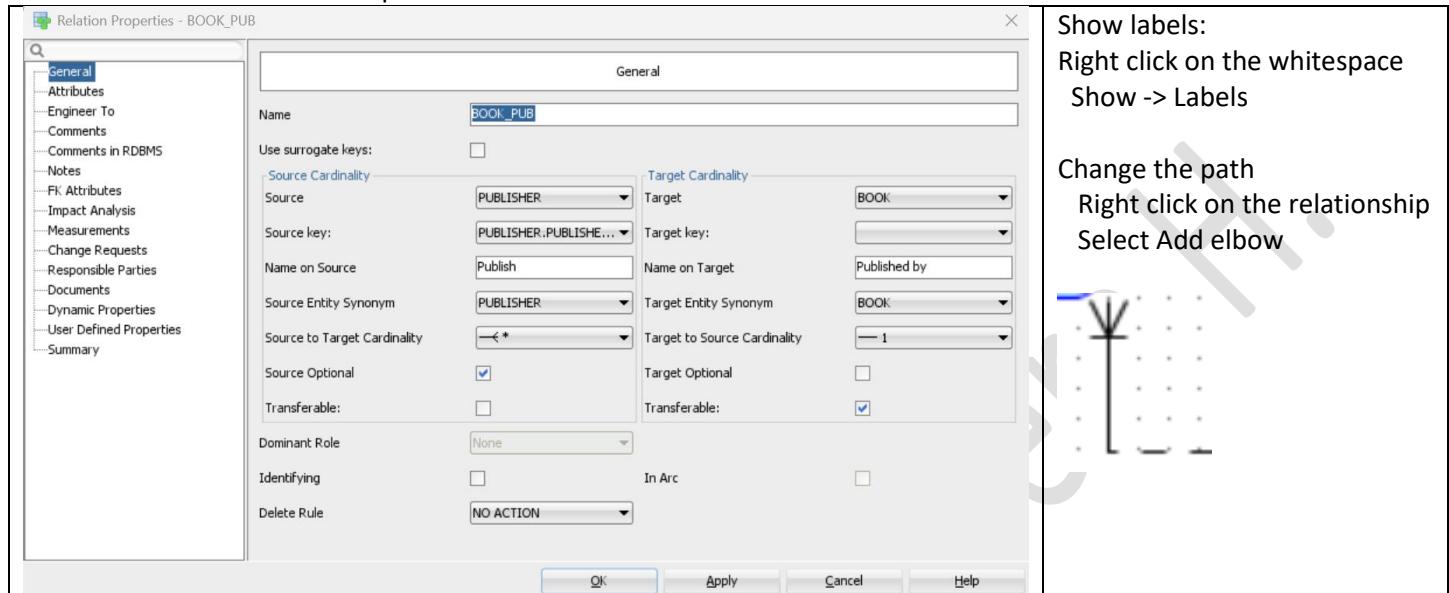
• The relationships available in Oracle SQL Developer are:

- 1:1 (one-to-one)
- 1:N (one-to-many)
- 1:N Identifying Relationship (one-to-many barred relationship)
- M:N (many-to-many)

To define the relationships between entities in Oracle SQL Developer, perform the following steps:

- Click a relationship type on the toolbar
- Click the source entity(PK) and then click the target entity(FK) . The relationship is created

Double-Click on the relationship established



Creating the Supertype Entity

Select the table that is subtype (Full Time)
General properties -> Super Type (select Employee)

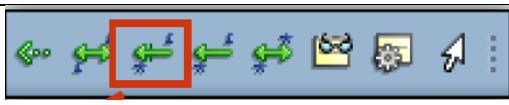
Creating the Arc Relationship

Perform the following steps:
a. Hold the ctrl key and select the intersecting entity and both relationships on which you want to create the Arc relationship
b. Click the New Arc icon in the toolbar. The exclusive relationship is created with the arc



Creating the Barred Relationship

To add a barred relationship select Identifying Relationship from the toolbar, and click the source and target entities to add the relationship between the entities



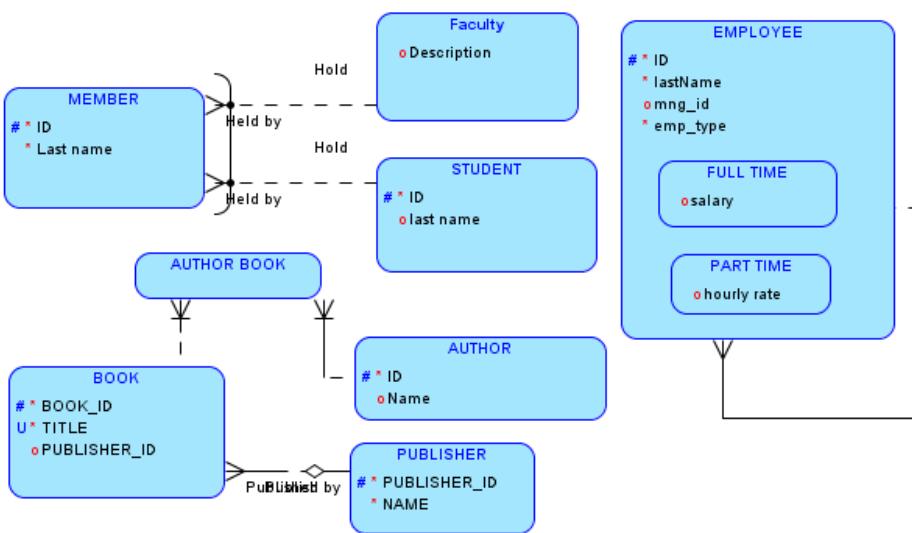
Creating the Hierarchical Relationship

The UIDs for a set of hierarchical entities can be propagated through multiple relationships by making the relationships Identifying

Creating the Recursive Relationship

To add a Recursive Relationship, select the required relationship from the toolbar as normal, then click on the entity to make it the source, and click on the same entity a second time to make it the target.

Practices with the following example:



→

4.2. Convert a Logical Model to a Relational Model

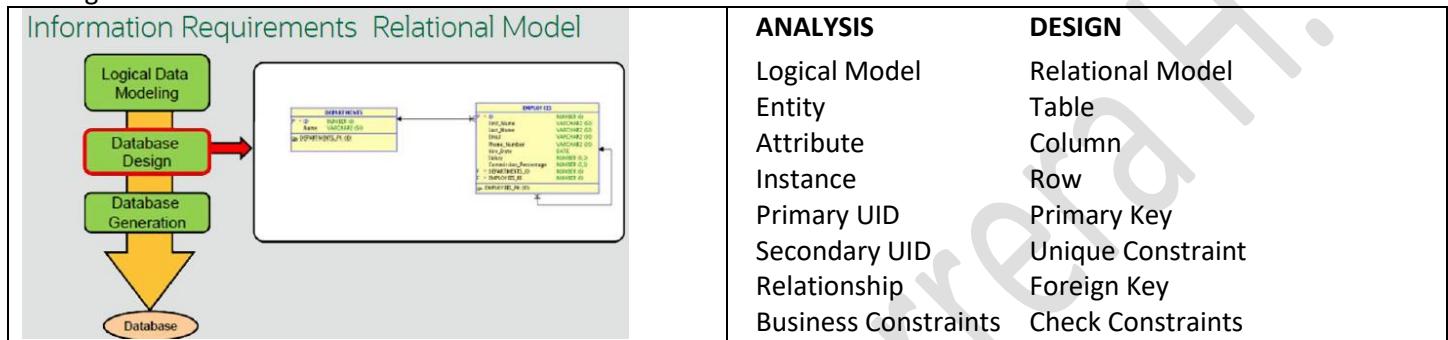
Objectives:

- Describe how to convert a logical model to a relational model
- List the steps to convert a logical model to a relational model
- List the steps to convert a relational model to a logical model

In Oracle SQL Developer Data Modeler a physical model is represented by a Relational Model.

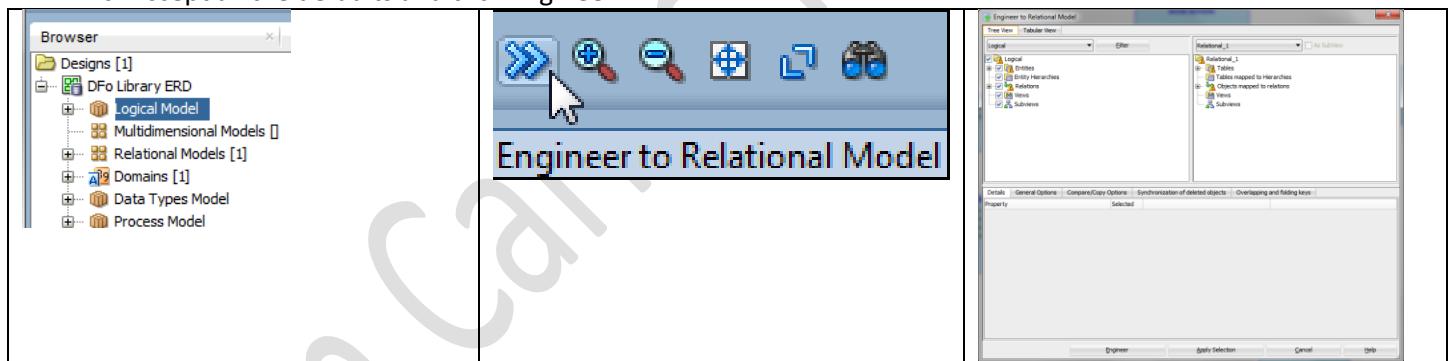
Forward engineering is the process of transforming a logical data model to a relational model.

Reverse engineering is the process of creating a conceptual or logical model by extracting the information from an existing data source.



Logical Model to a Relational Model:

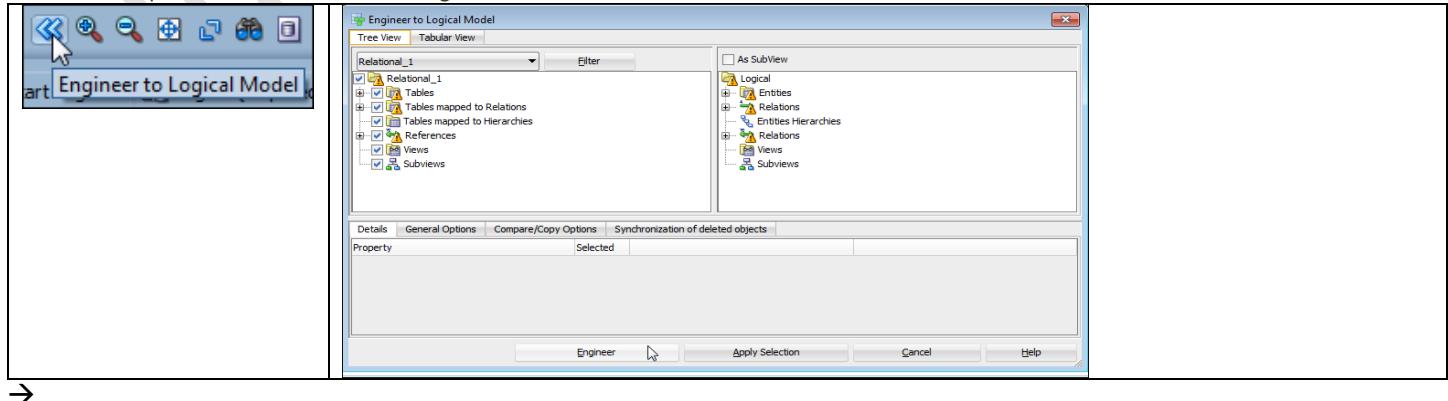
- Select the logical model.
- Click the Engineer to Relational Model icon
- Accept all the defaults and click Engineer



Reverse engineering from a Relational to a Logical Model

Allows an ERD to be created from an existing Physical design

- Click the Engineer to Logical Model icon
- Accept all the defaults and click Engineer



5. Mapping to the Physical Model

5.1. Mapping Entities and Attributes

In Oracle SQL Developer Data Modeler a Physical model is known as a Relational Model.

An Entity Relationship Model (ERM) does not highlight the physical and database constraints. It is essential to transform the ERM into a relational model that can serve as the basis for defining the physical implementation of the database.

Table short names (abbreviations)

A unique abbreviation for every table is a very useful element for the naming of foreign key columns or foreign key constraints.

Applying Naming Standards from Logical to a Relational Model

Logical Model (ERD) Physical model (Relational Model)

Entity Table

EMPLOYEE to EMPLOYEES

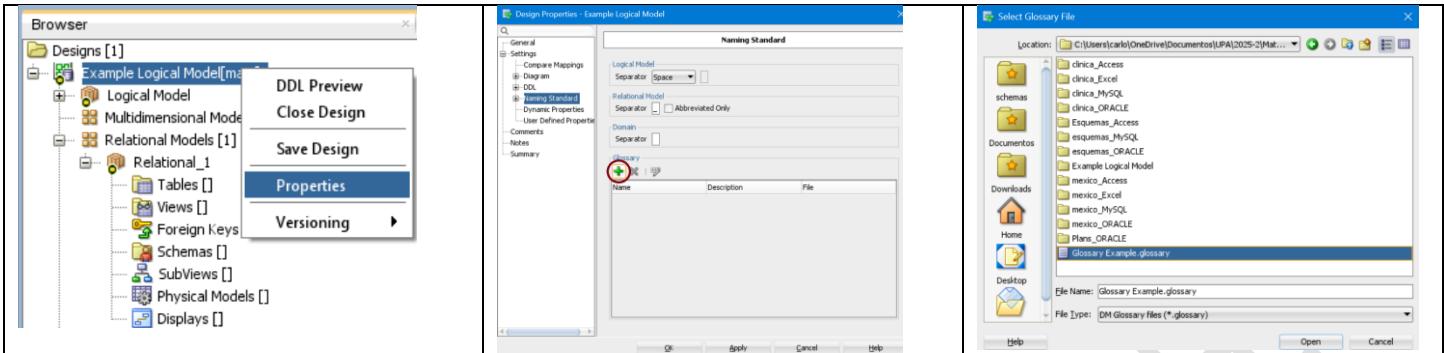
Last name to Last_name

Create a Glossary from a Logical Model

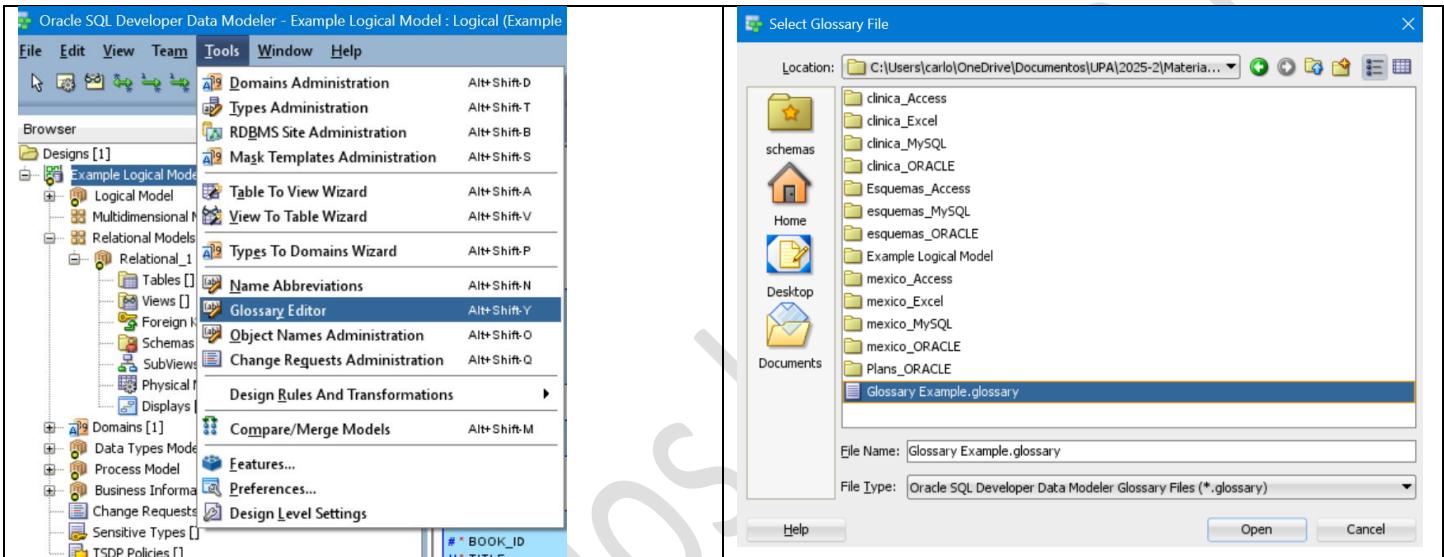
The screenshot shows the Oracle SQL Developer Data Modeler interface. On the left, the 'Browser' panel displays a tree structure of designs, with 'Example Logical Model[main]' selected. A context menu is open over this selection, with 'Create Glossary from Logical model' highlighted in blue. To the right, the 'Glossary Editor' window is open. It contains a 'Glossary properties:' section with 'Name' set to 'Example Logical Model' and 'Description' set to 'generated from logical model of design Example Logical Model'. Below this are 'Options' with checkboxes for 'Incomplete Modifiers' (checked), 'Case Sensitive' (unchecked), and 'Unique Abbreviations' (unchecked). The 'Words' section lists various entities and their abbreviations:

Name	Plural	Abbreviation
AUTHOR		
BOOK	BOOKS	BOO
BOOK_ID		
Description		DSC
emp_type		
EMPLOYEE	EMPLOYEES	EMP
Entity_7		
Faculty		
FULL		FUL
hourly		
ID		
Last		LST
lastName		
MEMBER		MBR
mng_id		
Name		
PART		PAT

Adding the Glossary as the Naming Standard

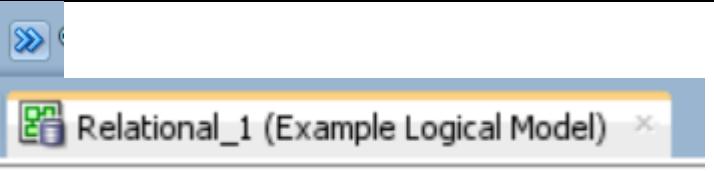


Or



Applying the Naming Standard

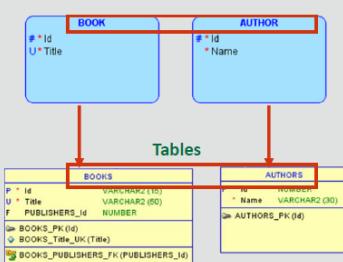
- To Apply the Glossary as the naming standard,
- Engineer the Logical model once more
 - Click the relational tab to view the results



Mapping Entities to Table names

- Entities

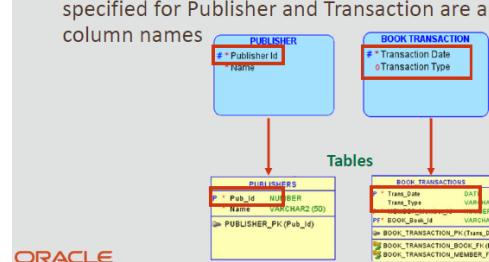
By applying the naming standards contained in the glossary, singular entity names in the logical model are mapped to plural table names in the relational model



Mapping Attributes to Column names

- Entities

If, for example, we had included the terms Publisher and Transaction in our Entity attribute names, by applying the naming standards contained in the glossary, the abbreviations specified for Publisher and Transaction are applied in the column names



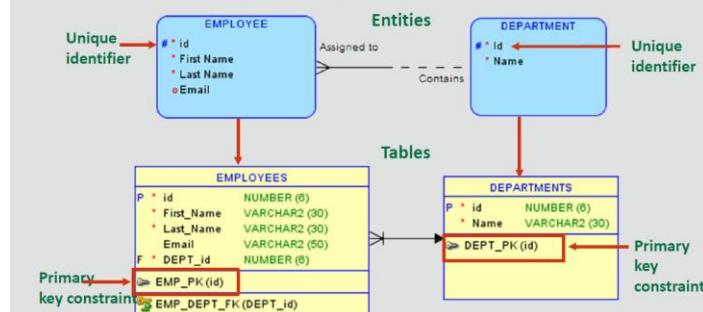
5.2. Mapping Primary and Foreign Keys

Decide on naming conventions for:

- Primary key constraint names
- Foreign key constraint names
- Foreign key column names
- Map subtypes to tables

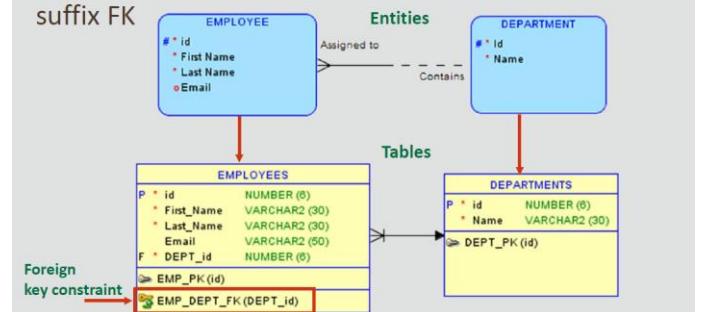
Naming Conventions for Primary Key Constraints

- Primary key constraints are named using the short table name, an underscore and the suffix PK



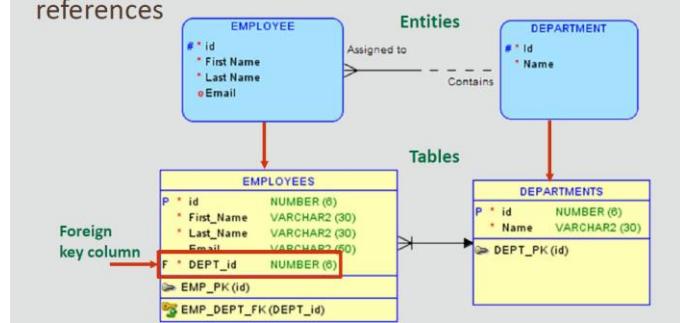
Naming Conventions for Foreign Key Constraints

- Foreign key constraints are named using the short table names of both tables, an underscore and the suffix FK



Naming Conventions for Foreign Key Columns

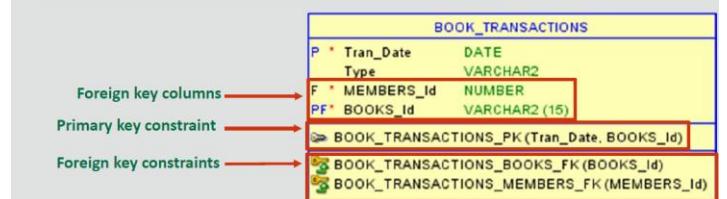
- Foreign key columns are named using the short table name and column name of the table the foreign key references



Sugerencia Personal: En ambas tablas
IdEmployee = ID_EMPLOYEE = employee_id

Apply naming standards in Oracle SQL Developer Data Modeler

- By default, constraint names are created using the full table name in Oracle SQL Developer Data Modeler. This can lead to constraint names that are very long, difficult to manage, and may exceed the maximum permitted number of characters in SQL



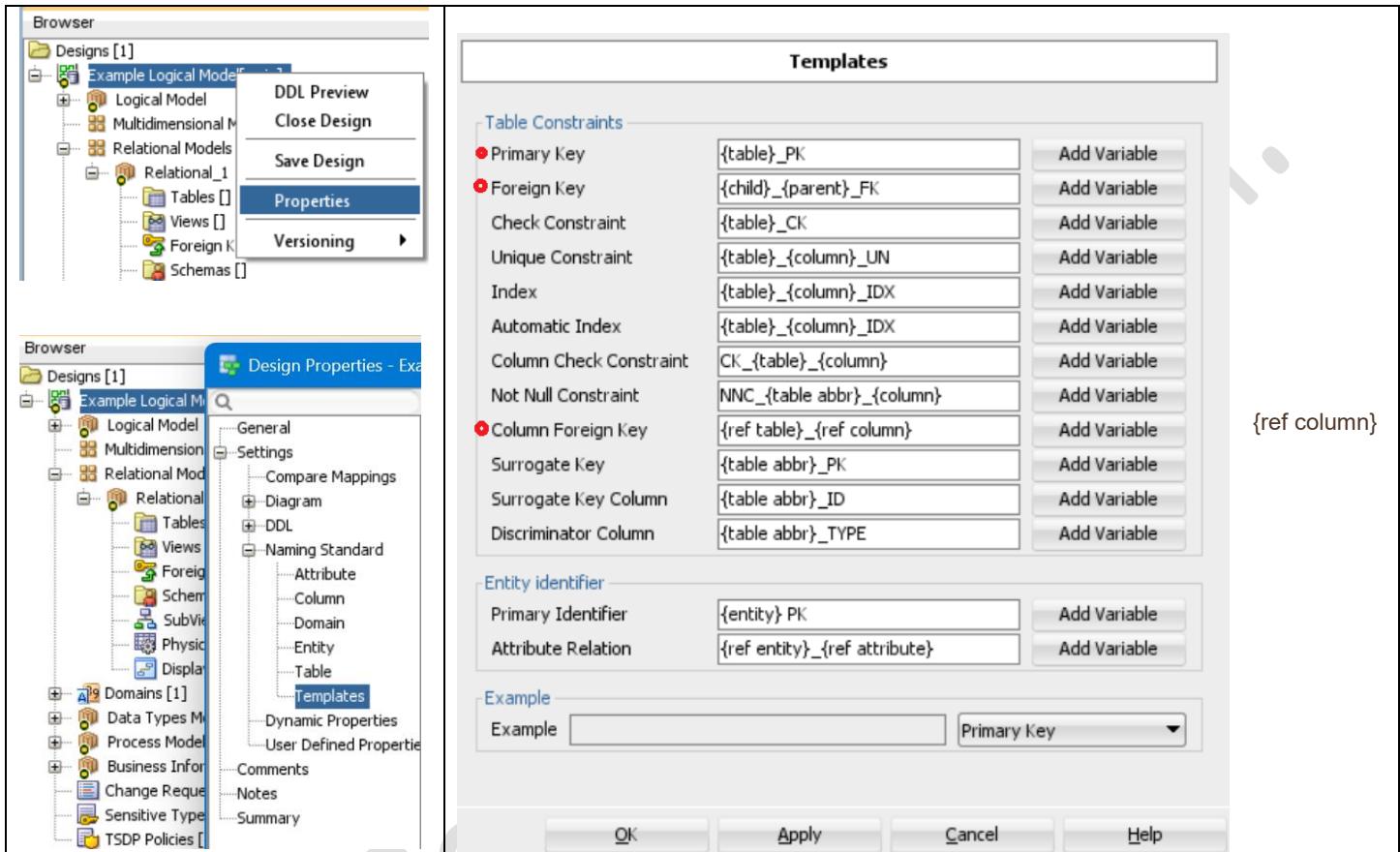
Create Name Abbreviations

- To apply naming standards to constraint names, firstly create a .csv file in a spreadsheet application
 - In the first column specify the full (plural) table names, and in the second column the abbreviation to use.
- Save as .csv file and note the location, example: constrain_names.csv

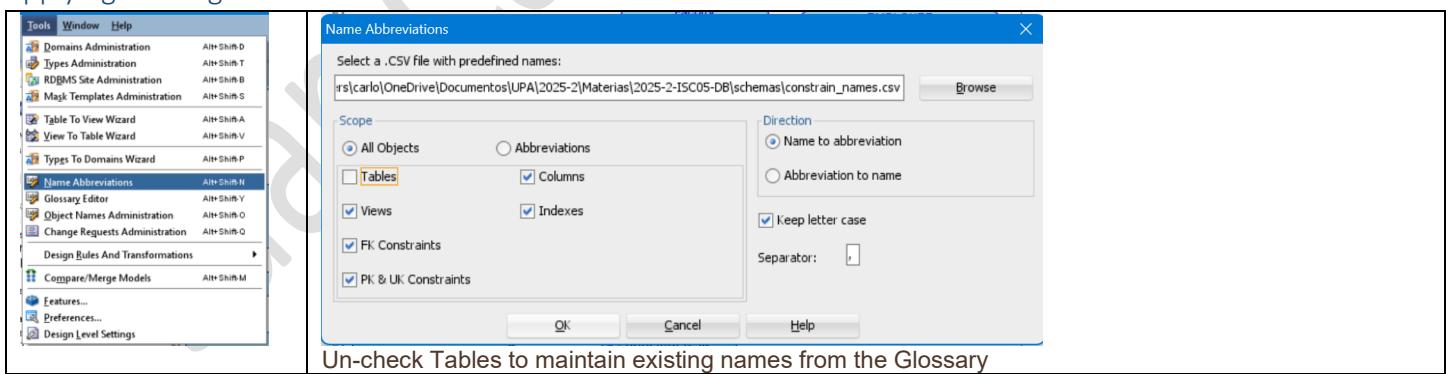
A	B
1 PUBLISHERS	PUB
2 BOOKS	BK
3 AUTHORS	ATHR
4 MEMBERS	MEM
5 TRANSACTIONS	TRN

Defining Naming Templates

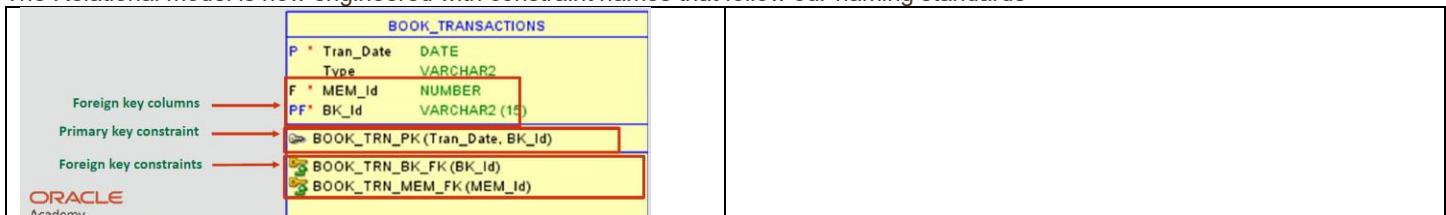
- To apply our abbreviations to our model, we need to set the pattern in the Naming template
- To access the Templates page:
 - a. Right-click the name of the Design in the Object Browser
 - b. Select Properties
 - c. expand Settings and Naming Standard
 - d. Select Templates



Applying Naming Abbreviations



The Relational model is now engineered with constraint names that follow our naming standards



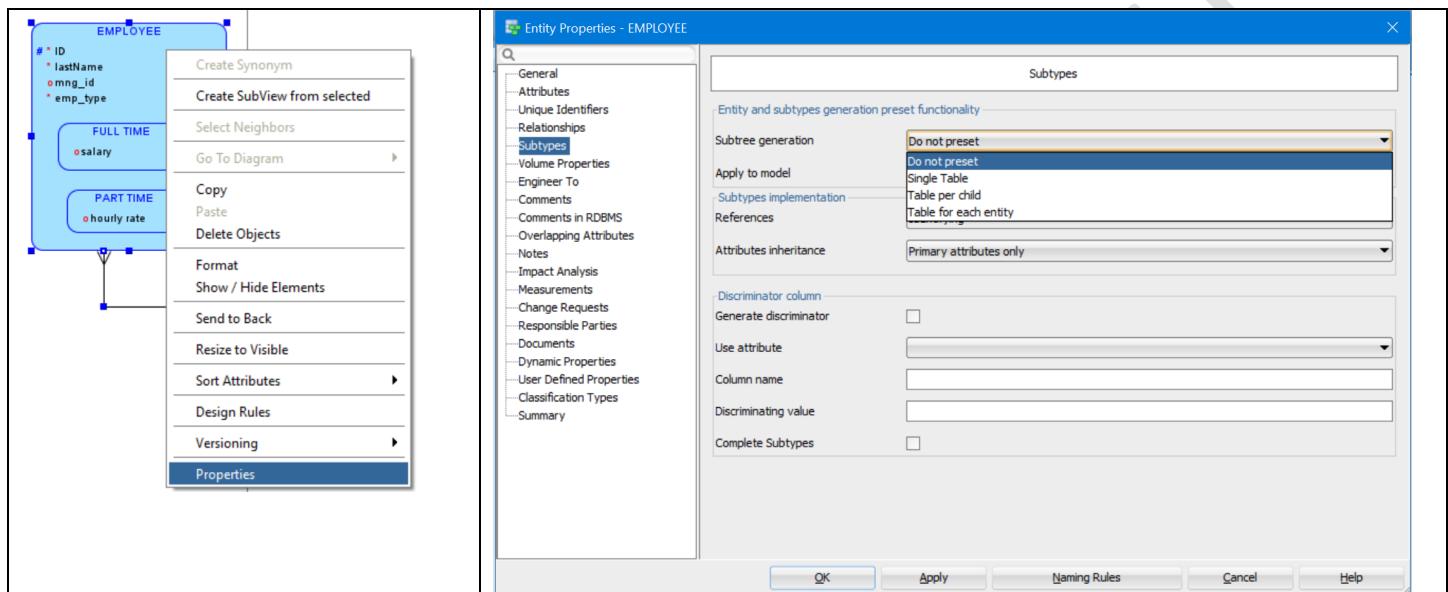
Mapping Subtypes to Tables

subtypes can be mapped to tables in a number of ways:

- One table (single table implementation)
- Two table (table for each child (subtype) implementation)

To select how subtypes are engineered in Oracle SQL Developer Data Modeler:

- a) Double click the Super type entity to edit properties
- b) Select Subtypes
- c) Select required method for generating subtypes from the Subtree generation dropdown box options



→

6. Introduction to SQL

6.1. Introduction to Oracle Application Express

Oracle APEX has these components:

- SQL Workshop: To learn SQL
- Application Builder: To design an application
- Object Browser

Upload and run a script in APEX

File zip -> 2 – Course -> Code Files -> DFo_Section_6_1 -> obl Sports.zip (obl Sports.ddl)

Create table...

Alter table...

SQL workshop

(Zip - obl Sports Script) and must first be extracted: chose the obl Sports.ddl

run the script

DESCRIBE customers;

6.2. Structured Query Language (SQL)

Types of SQL Commands

- DDL(Data Definition Language) –defines database structures (CREATE, ALTER, drop, rename)
- DML(Data Manipulation Language) –manipulates data (INSERT, UPDATE, DELETE)
- DQL(Data Query Language) –SELECTs data
- DCL(Data Control Language) –controls user access (Grant, revoke)
- TCL (Transactional Control Language) –manages database transactions (begin, commit, rollback)

Three tools are:

- SQL*Plus (consola)
- SQLDeveloper
- APEX (Oracle Application Express)

→

6.3. Data Definition Language (DDL)

Object	Description
Table	Is the basic unit of storage; consists of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives an alternative name to an object

CREATE TABLE [schema.]table (column datatype [DEFAULT expr][, ...]);	CREATE TABLE dept(deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(13), create_date DATE DEFAULT SYSDATE);
---	---

Describe dept

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPTNO	NUMBER	-	2	0	-	-	-	-
	DNAME	VARCHAR2	14	-	-	-	-	-	-
	LOC	VARCHAR2	13	-	-	-	-	-	-
	CREATE_DATE	DATE	7	-	-	-	-	SYSDATE	-

Data Types

Data Type	Description
VARCHAR2(size)	Variable-length character data (A maximum size must be specified; minimum size is 1.) Maximum size: 32767 bytes
CHAR(size)	Fixed-length character data of length (size) bytes. (Default and minimum size is 1; maximum size is 2,000)
NUMBER(p,s)	Variable-length numeric data. Precision is p, and scale is s. (Precision is the total number of decimal digits, and scale is the number of digits to the right of the decimal point; precision can range from 1 to 38, and scale can range from -84 to 127.)
DATE	Date and time values to the nearest second between January 1, 4712 B.C, and December 31, 9999 A.D.
LONG	Variable-length character data (up to 2 GB)
CLOB	A character large object (CLOB) containing single-byte or multibyte characters. Maximum size is (4 GB -1) * (DB_BLOCK_SIZE); stores national character set data.
NCLOB	A CLOB containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 GB -1) * (database block size); stores national character set data.
RAW (Size)	Raw binary data of length size bytes. You must specify size for a RAW value. Maximum size: 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY
LONG RAW	Raw binary data of variable length up to 2 GB.
BLOB	A binary large object. Maximum size is (4 GB -1) * (DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)).
BFILE	Binary data stored in an external file (up to 4 GB).
ROWID	Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn.

Date Data Types

Data Type	Description
TIMESTAMP	Enables storage of time as a date with fractional seconds. It stores the year, month, day, hour, minute, the second value of the DATE data types, and the fractional seconds value. There are several variations of this data type, such as WITH TIMEZONE and WITH LOCALTIMEZONE.
INTERVAL YEAR TO MONTH	Enables storage of time as an interval of years and months. Used to represent the difference between two datetime values in which the only significant portions are the year and month.
INTERVAL DAY TO SECOND	Enables storage of time as an interval of days, hours, minutes, and seconds; used to represent the precise difference between two datetime values.
TIMESTAMP WITH TIME ZONE	Variant of TIMESTAMP that includes a time zone region name or time zone offset in its value.

<pre>CREATE TABLE table_ts(c_id NUMBER(6), c_ts TIMESTAMP);</pre>	<pre>INSERT INTO table_ts VALUES(1, '01-JAN-2003 2:00:00');</pre>
---	---

<pre>CREATE TABLE time_table(start_time TIMESTAMP, duration_1 INTERVAL DAY (6) TO SECOND (5), duration_2 INTERVAL YEAR TO MONTH);</pre>	
---	--

Data Integrity Constraints

Constraints	Description
NOT NULL	The column cannot contain a null value (constraint only at the column level)
UNIQUE	The values for a column or a combination of columns must be unique for all rows in the table
PRIMARY KEY	The column (or a combination of columns) must contain the unique AND IS NOT NULL value for all rows
FOREIGN KEY REFERENCES	The column (or a combination of columns) must establish and enforce a reference to a column or a combination of columns in another (or the same) table
CHECK	A condition must be true

Name a constraint (otherwise, the Oracle server generates a name in the SYS_Cn format). Example SYS_C0014370

Defining Constraints

```
CREATE TABLE [schema.]table  
(column datatype [DEFAULT expr]  
[column_constraint],  
...  
[table_constraint][,...]);
```

- Column-level constraint syntax:
column datatype [CONSTRAINT constraint_name] constraint_type,
- Table-level constraint syntax:
Column datatype, ...
[CONSTRAINT constraint_name] constraint_type (column, ...),

• Column-level constraint:

```
CREATE TABLE employees (  
    employee_id NUMBER(6) CONSTRAINT emp_emp_id_pk  
        PRIMARY KEY,  
    first_name VARCHAR2(20),  
    ...  
) ;
```

• Table-level constraint:

```
CREATE TABLE employees (  
    employee_id NUMBER(6),  
    first_name VARCHAR2(20),  
    ...  
    job_id VARCHAR2(10),  
    CONSTRAINT emp_emp_id_pk PRIMARY KEY (employee_id)  
) ;
```

FOREIGN KEY Constraint

• Defined at the table level:

```
CREATE TABLE employees (  
    employee_id NUMBER(6),  
    last_name VARCHAR2(25),  
    email VARCHAR2(25),  
    salary NUMBER(8,2),  
    commission_pct NUMBER(2,2),  
    hire_date DATE,  
    ...  
    department_id NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id)
```

• Defined at the column level:

```
CREATE TABLE employees (  
    employee_id NUMBER(6),  
    last_name VARCHAR2(25),  
    email VARCHAR2(25),  
    salary NUMBER(8,2),  
    commission_pct NUMBER(2,2),  
    hire_date DATE,  
    ...  
    department_id NUMBER(4) CONSTRAINT emp_dept_fk  
        REFERENCES departments(department_id)
```

FOREIGN KEY Constraint: Keywords

- FOREIGN KEY: Defines the column in the child table at the table-constraint level
- REFERENCES: Identifies the table and column in the parent table
- ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted
- ON DELETE SET NULL: Converts dependent foreign key values to null

CHECK Constraint

```
CREATE TABLE employees(
employee_id NUMBER(6),
last_name VARCHAR2(25),
salary NUMBER(8,2) CONSTRAINT emp_salary_min CHECK (salary > 0),
commission_pct NUMBER(2,2),
hire_date DATE DEFAULT SYSDATE,
email VARCHAR2(25),
CONSTRAINT hire_date_min CHECK (hire_date >'01-JAN-2018')
);
```

ALTER TABLE Statement

Use the ALTER TABLE statement to change the table structure:

- Add a column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change a table to read-only status

```
ALTER TABLE table
ADD (column data type [DEFAULT expr]
[, column data type]...);
```

```
ALTER TABLE table
MODIFY (column data type [DEFAULT expr]
[, column data type]...);
```

```
ALTER TABLE table
DROP (column [, column] ...);
```

```
ALTER TABLE employees
ADD termination_date DATE;
```

```
ALTER TABLE employees
MODIFY first_name VARCHAR2(30);
```

```
ALTER TABLE employees
DROP (termination_date);
```

SET UNUSED Option (Exclusive Oracle)

```
ALTER TABLE <table_name>
SET UNUSED(<column_name> [ , <column_name>]);
OR
ALTER TABLE <table_name>
SET UNUSED COLUMN <column_name> [ , <column_name>];
```

```
ALTER TABLE dept
SET UNUSED (dname);
```

```
ALTER TABLE <table_name>
DROP UNUSED COLUMNS;
```

```
ALTER TABLE dept
DROP UNUSED COLUMNS;
```

Read-Only Tables

- Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
- Put the table back into read/write mode

```
ALTER TABLE dept READ ONLY;  
--perform table maintenance and then  
--return table back to read/write mode
```

```
ALTER TABLE dept READ WRITE;
```

Dropping a Table

```
DROP TABLE dept; -- Observar el Directorio PURGE, la tabla todavia existe
```

→

6.4. Data Manipulation Language (DML)

A DML statement is executed when you:

- Add new rows to a table (INSERT)
- Modify existing rows in a table (UPDATE)
- Remove existing rows from a table (DELETE)

A **transaction** consists of a collection of DML statements that form a logical unit of work

INSERT Statement Syntax

<pre>INSERT INTO table [(column [, columna ...])] VALUES (value [, value...]); -- only one row is inserted at a time</pre> <pre>CREATE TABLE copy_departments AS (SELECT * FROM departments); -- does not copy all constraints (only NOT NULL)</pre>	<pre>INSERT INTO copy_departments VALUES (40, 'Advertising', 201, 1800, SYSDATE);</pre> <pre>INSERT INTO copy_departments VALUES (100, 'Finance', NULL, NULL, TO_DATE('Dec 7, 2002', 'MON DD, YYYY'));</pre> <pre>INSERT INTO copy_departments (department_id, department_name) VALUES(30,'Purchasing');</pre>
---	---

Default: DD-MON-RR

Tools -> preferences -> DB -> NLS

alter session set nls_date_format = 'dd-mm-yyyy'

<pre>-- MySQL: Dia de la semana (1-7, 1=Dom) select dayofweek(sysdate()) from dual;</pre> <pre>(columna, position, [length]) select substr('UNIVERSIDAD', 7,4) FROM DUAL; select substr('UNIVERSIDAD', 7) FROM DUAL;</pre>	<pre>-- Oracle: Dia de la semana (1-7, 1=Dom) select to_char(sysdate, 'D') from dual;</pre>
---	---

UPDATE Statement Syntax

<pre>UPDATE table SET column = value [, column = value, ...] [WHERE condition]; -- Si se omite afecta todos los registros</pre>	<pre>UPDATE copy_employees SET department_id = 50, Salary = 12000, COMMISSION_PCT = NULL WHERE employee_id = 113;</pre>
---	---

DELETE Statement

<pre>DELETE [FROM] table [WHERE condition]; -- Si se omite afecta todos los registros</pre>	<pre>DELETE FROM copy_departments WHERE department_name = 'Purchasing';</pre>
---	---

TRUNCATE Statement

Removes all rows from a table, leaving the table empty and the table structure intact

<pre>TRUNCATE TABLE table_name;</pre>	<pre>TRUNCATE TABLE copy_employees;</pre>
---------------------------------------	---

→

6.5. Transaction Control Language (TCL) \$\$

6.6. Retrieving Data Using SELECT

6.7. Restricting Data Using WHERE

6.8. Sorting Data Using ORDER BY

6.9. Joining Tables Using JOIN

→

Juan Carlos Herrera H.