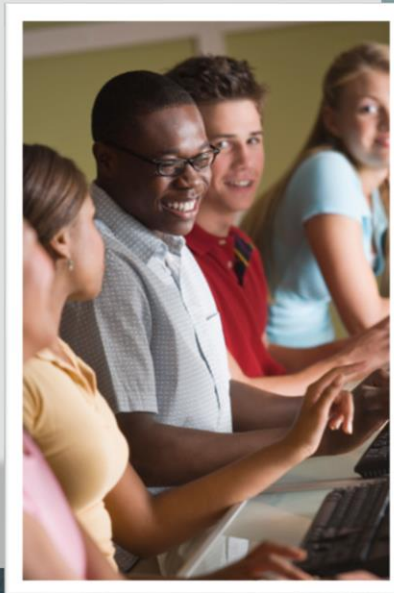
ORACLE

Academy

Database Design

9-4

Subtype Mapping



ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - State and apply the table, column, identifiers, relationship, and integrity constraint rules for mapping:
    - supertype implementations
    - subtype implementations
    - supertype and subtype arc implementations

ORACLE Academy

There are three ways to implement subtypes in the database. All three will be discussed in this lesson.

Purpose

- A carpenter who is building your dream house may know that you will use different types of light bulbs all around the house
- However, if you do not provide information on where certain types of light bulbs should be installed, you could end up with an overly bright bedroom and a dimly lit kitchen!
- Mapping supertypes and subtypes makes sure that the right information gets stored with each type

Supertype Implementation: Single Table

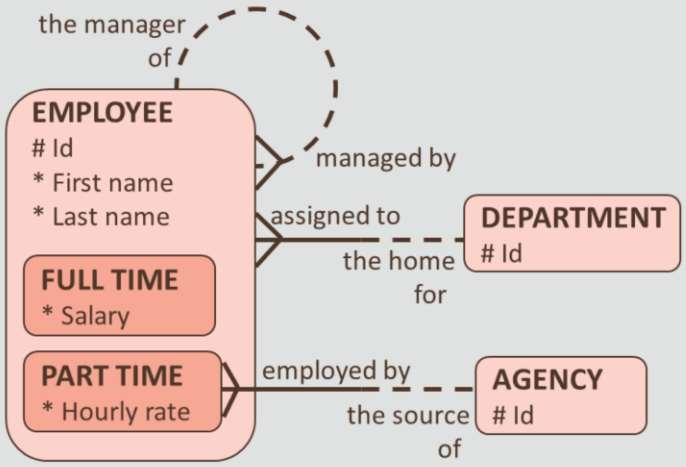
- This choice produces a single table for the implementation of the supertype entity and its subtypes
- This is also called "single-table (or one-table) implementation"
- Rules:
  - Tables: Only one table is created, regardless of the number of subtypes
  - Columns: The single table gets one column for each attribute of the supertype, along with the original optionality of the attribute

Supertype: A means of classifying an entity that has subtypes.

Supertype Implementation: Single Table

- Rules (cont.):
  - The table also gets a column for each attribute belonging to the subtype, but the columns all become optional
  - Additionally, a mandatory column should be created to act as a discriminator column to distinguish between the different subtypes of the entity
  - The value it can take is from the set of all the subtype short names (FTE, PTE, OTR in the example)
  - This discriminator column is usually called <table_short_name>_type, which would be epe_type in the example

Supertype Implementation: Single Table



DEPARTMENTS (DPT)		
pk	*	id

AGENCIES (AGY)		
pk	*	id

EMPLOYEES (EPE)		
Key Type	**Optionality**	**Column Name**
pk	*	id
	*	first_name
	*	last_name
	o	salary
	o	hourly_rate
fk1	*	dpt_id
fk2	o	agy_id
	*	epe_type
fk3	o	mgr_id

ORACLE Academy

DDS9L4
Subtype Mapping

7

Salary and hourly_rate are both optional, even if the ERD has them as mandatory. Dpt_id can remain mandatory, because it comes from the relationship to the supertype EMPLOYEE. Agy_id comes from a relationship to the subtype and has to become optional.

Epe_type is the discriminator column. It can have values of FTE or PTE.

Notice that mgr_id represents the recursive relationship for EMPLOYEE. It is a foreign key column and would normally be named epe_id, after the parent table. However, it makes sense to rename it at this stage, to make it easier to understand.

Supertype Implementation: Single Table

- Rules:
  - Identifiers: Unique identifiers transform into primary and unique keys
  - Relationships: Relationships at the supertype level transform as usual, relationships at the subtype level are implemented as optional foreign-key columns
  - Integrity constraints: A check constraint is needed to ensure that for each particular subtype, all columns that come from mandatory attributes are not null

Supertype Implementation: Single Table

- In the conceptual model, salary is mandatory for full-time employees and hourly rate is mandatory for part-time employees
- When the EMPLOYEE supertype is implemented as a single table in the physical model, these attributes become optional
- A check constraint is needed to enforce the business rules modeled in the ERD

ORACLE Academy

DDS9L4
Subtype Mapping

A note about the OTHER subtype:
An OTHER subtype is recommended in the conceptual model to ensure that the subtypes are exhaustive. However, by the time we start the design phase, we should have done extensive analysis to determine if another subtype is truly needed. If so, then this subtype must be named, and its attributes specified. If not, then the OTHER subtype is not part of the transformation process.

Supertype Implementation: Single Table

- In the example, the code for the check constraint would look like this:
  - CHECK (epe_type = 'FTE' and salary is not null and hourly_rate is null and agy_id is null)
  - OR (epe_type = 'PTE' and salary is null and hourly_rate is not null and agy_id is not null)

Supertype Implementation: Single Table

- The code checks that if it is a full-time employee (epe_type = ‘FTE’), then a value must exist in the salary column and the hourly_rate and agy_id columns must be empty
- Conversely, if it is a part-time employee (epe_type = ‘PTE’), then a value must exist in hourly_rate and agy_id, but salary must be left blank

Supertype Implementation: Single Table

- Sample Data for **EMPLOYEES**

id	first_name	last_name	salary	hourly_ rate	dpt_id	agy_id	epe_type	mgr_id
2000	Joan	Merrick	50000		10		FTE	111
111	Sylvia	Patakis	90000		10		FTE	
2101	Marcus	Rivera		65,00	10	17	PTE	111
2102	Hector	Chen		75,00	25	17	PTE	45
45	Rajesh	Vishwan	90000		25		FTE	

When Do You Choose the Single Table/Supertype Implementation?

- The single-table implementation is a common and flexible implementation
- It is the one you are likely to consider first and is especially appropriate where:
  - Most of the attributes are at the supertype level
  - Most of the relationships are at the supertype level
  - Business rules are globally the same for the subtypes

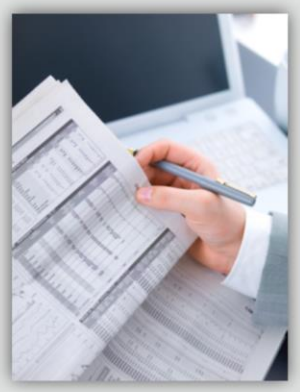
ORACLE Academy

It is elegant and appropriate for most situations and should be considered as the first choice. Usually you would create a view for every subtype, showing only the columns that belong to that particular subtype. The correct rows are selected using a condition based on the discriminator column.

Views will be discussed later in the course.

Subtype Implementation: Two Table

- This is also called "two-table implementation"
- You create a table for each of the subtypes
- So, in reality, you could have more than two tables, if you had more than two subtypes



Subtype: Something an entity may be split into based on common attributes and/or relationships.

Subtype Implementation: Two Table

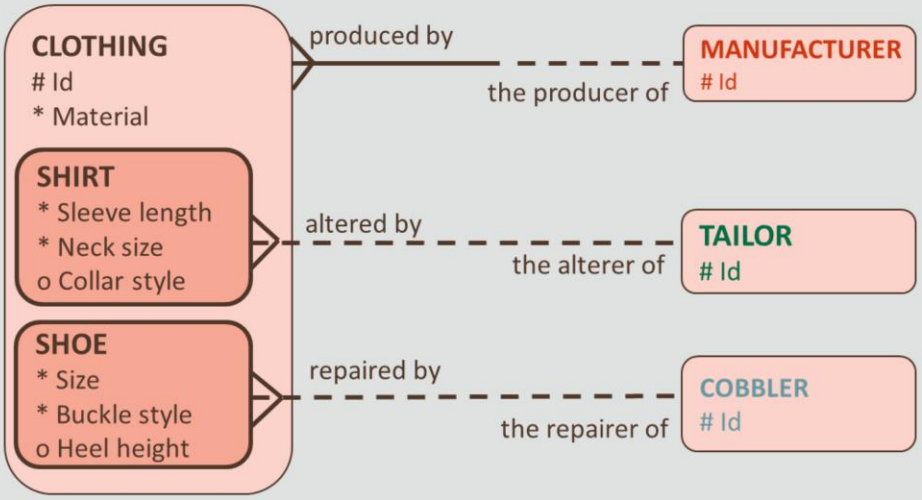
- Rules:
  - Tables: One table per first-level subtype
  - Columns: Each table gets one column for each attribute of the supertype along with its original optionality
  - Each table also gets one column for each attribute belonging to the subtype along with its original optionality

Subtype Implementation: Two Table

- Rules (cont.):
  - Identifiers:
    - The primary UID at the supertype level creates a primary key for each table
    - Secondary UIDs of the supertype become unique keys in each table
  - Relationships:
    - All tables get a foreign key for a relationship at the supertype level, with the original optionality
    - For relationships at the subtype levels, the foreign key is implemented in the table it is mapped to
    - Original optionality is retained

The original optionality of the attributes and relationships is carried over to the columns and foreign keys, so there is no need for a check constraint as there is in the one-table implementation.
If there were a need to have unique values of the primary key (ID) across both tables, then additional programming would be necessary.

Subtype Implementation: Two Table



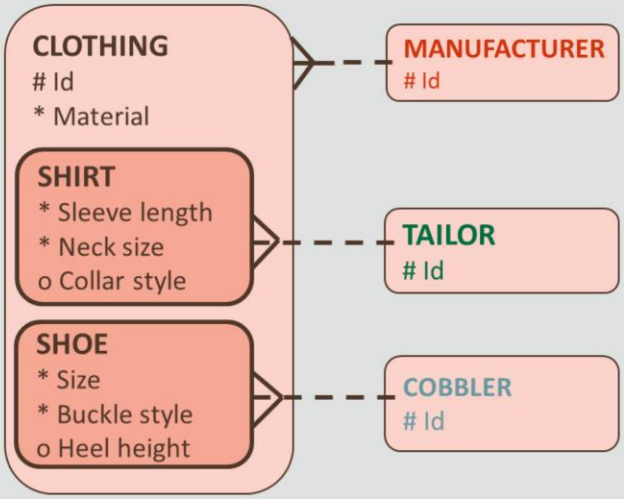
ORACLE Academy

DDS9L4
Subtype Mapping

17

Id, material, and mnr_id come from attributes in OR relationships to the supertype. Therefore, they appear in both tables.

Subtype Implementation: Two Table



SHIRTS (SHT)		
pk	*	id
	*	material
	*	sleeve_length
	*	neck_size
	o	collar_style
fk1	o	tlr_id
fk2	*	mnr_id

SHOES (SHE)		
pk	*	id
	*	material
	*	size
	*	buckle_style
	o	heel_height
fk1	o	clr_id
fk2	*	mnr_id

ORACLE Academy

DDS9L4
Subtype Mapping

18

Id, material, and mnr_id come from attributes in OR relationships to the supertype. Therefore, they appear in both tables.

Subtype Implementation: Two Table

- In the example, a separate table would be created for SHIRTS and SHOES

Sample Data for SHIRTS

id	material	sleeve_length	neck_size	collar_style	mnr_id	tlr_id
10	linen	33	16	kancing bawah	65	14
11	wol	32	15,5	nehru	65	22
14	cotton	33	15,5		60	22

Sample Data for SHOES

id	material	size	buckle_style	heel_height	mnr_id	clr_id
3	leather	7,5	monkstrap	1,5	75	44
7	canvas	8	velcro	1	70	44

ORACLE Academy

DDS9L4
Subtype Mapping

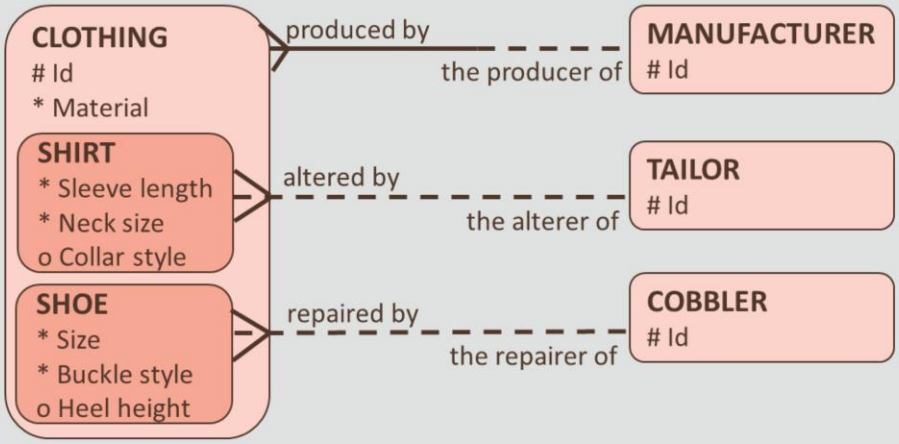
Usually you would create an additional view that represents the supertype, showing all columns of the supertype and the various subtypes

When to Consider Subtype Implementation

- Subtype implementation may be appropriate when:
  - Subtypes have very little in common, there are few attributes at the supertype level and several at the subtype level
  - Most of the relationships are at the subtype level
  - Business rules and functionality are quite different between subtypes
  - How tables are used is different -- for example, one table is being queried while the other is being updated

Modeling the Supertype as an Arc

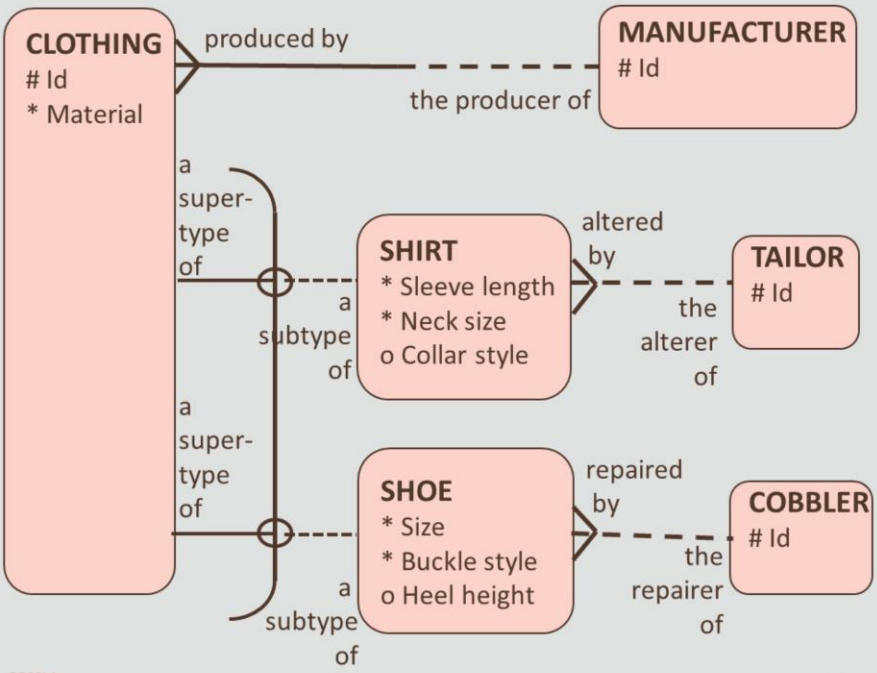
- A supertype entity and its subtypes can be modeled as an arc relationship
- Here again is the original ERD with the supertype and subtypes



ORACLE Academy

Model An Arc Illustrated

- In this ERD, we have redrawn the CLOTHING supertype and its subtypes of SHIRT and SHOE as standalone entities…



ORACLE Academy

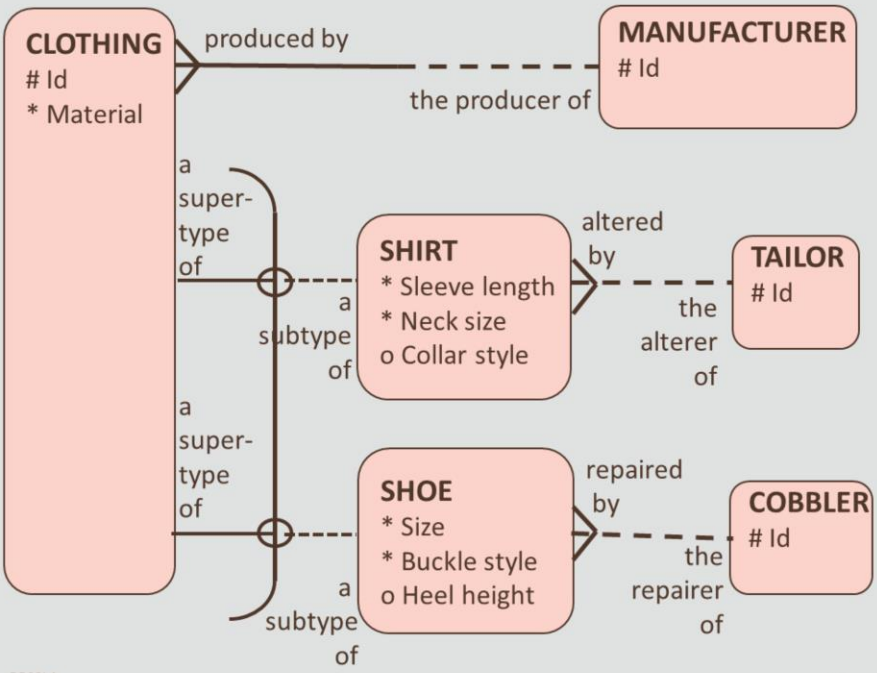
DDS9L4
Subtype Mapping

22

The relationships are 1:1 mandatory and exclusive:
Each SHIRT must be one piece of CLOTHING. Each SHOE must be one piece of CLOTHING.
Conversely, each piece of CLOTHING must be either a SHOE or a SHIRT but not both.

Model An Arc Illustrated

- …with each one having mandatory 1:1 relationships with the supertype
- The relationships are in an arc



ORACLE Academy

DDS9L4
Subtype Mapping

23

The relationships are 1:1 mandatory and exclusive:
Each SHIRT must be one piece of CLOTHING. Each SHOE must be one piece of CLOTHING.
Conversely, each piece of CLOTHING must be either a SHOE or a SHIRT but not both.

Supertype and Subtype (Arc) Implementation

- This choice produces one table for every entity
- The supertype table has a foreign key for each subtype table
- These foreign keys represent exclusive relationships
- They are optional because only one of them can have a value for each row in the table

Supertype and Subtype (Arc) Implementation

- Rules:
  - Tables: As many tables are created as there are subtypes, as well as one for the supertype
  - Columns: Each table gets a column for all attributes of the entity it is based on, with the original optionality
- Identifiers:
  - The primary UID of the supertype level creates a primary key for each of the tables
  - All other unique identifiers become unique keys in their corresponding tables

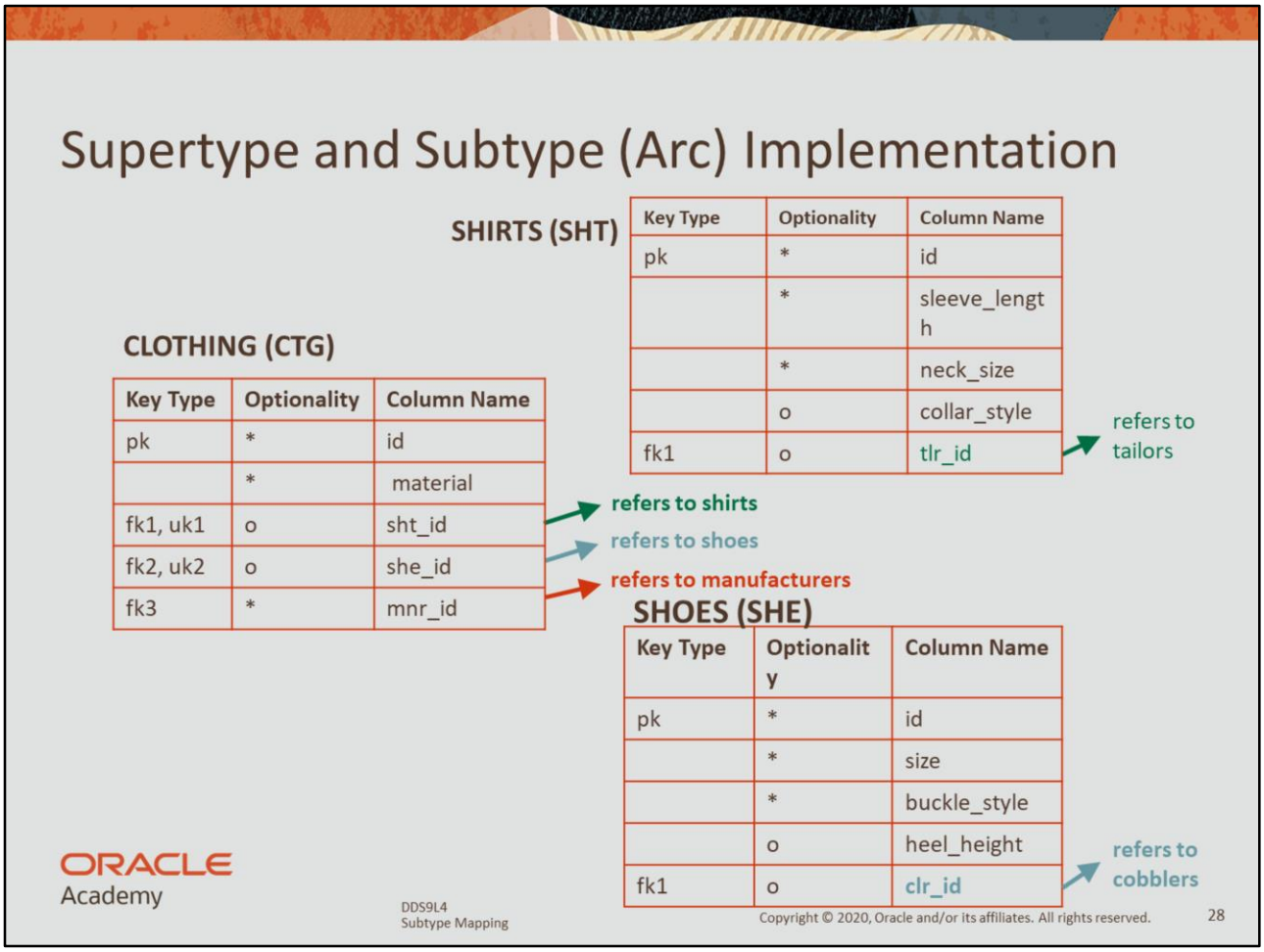
Supertype and Subtype (Arc) Implementation

- Relationships:
  - All tables get a foreign key for a relevant relationship at the entity level, with the original optionality
  - Integrity constraints: Two additional columns are created in the table based on the supertype
  - They are foreign-key columns referring to the tables that implement the subtypes

Supertype and Subtype (Arc) Implementation

- The columns are optional because the foreign keys are in an arc
- An additional check constraint is needed to implement the arc
- The foreign-key columns are also unique keys because they implement a mandatory 1:1 relationship

Supertype and Subtype (Arc) Implementation



In this case, it is highly unlikely that an arc implementation would be chosen, and is shown only to demonstrate that it is an option to be considered.
You would need a check constraint on CLOTHING to enforce that (sht_id is not null and she_id is null) OR (sht_id is null and she_id is not null).

When to Consider Both a Supertype and Subtype (Arc) Implementation

- This implementation is rarely used, but it could be appropriate when:
  - Subtypes have very little in common and each table represents information that can be used independently
  - For example, when the CLOTHING table gives all global information, and both SHOES and SHIRTS give specific information, and the combination of global and specific information is hardly ever needed
  - Business rules and functionality are quite different between all types
  - How tables are used is different

Terminology

- Key terms used in this lesson included:
  - Arc implementations
  - Subtype implementations
  - Supertype implementations

Summary

- In this lesson, you should have learned how to:
  - State and apply the table, column, identifiers, relationship, and integrity constraint rules for mapping:
    - supertype implementations
    - subtype implementations
    - supertype and subtype arc implementations


ORACLE
Academy