ORACLE ACADEMY

Release Date: August, 2015

Updates:

Database Design

9-3
Relationship Mapping

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Apply the rule of relationship mapping to correctly transform 1:M and barred relationships
- Apply the rule of relationship mapping to correctly transform M:M relationships
- Transform 1:1 relationships
- Apply the rule of relationship mapping to correctly transform relationships in an arc

Purpose

- Suppose that you are building a house for someone.
- You have all of the materials – wood, paint, doors, windows, nails, screws, etc. – and the skills, but you do not have a design.
- As you start, you don't know how many rooms should be included, where the windows should be placed, how the doors should be oriented, or what color each room should be painted.

Purpose

- You could build a house in such a manner and make these decisions as you go, but if you do not start with a blueprint of the structural design, the final product may not be the house that the customer has in mind.

Purpose

- Relationships are mapped between primary keys and foreign keys to allow one table to reference another.
- If we don't map relationships, we just have a lot of standalone tables containing information that does not connect to anything else in the database.
- Mapping relationships between entities serves as a critical "first-step" to facilitate discussion between the customer, designer, developer, and administrator of the database product.

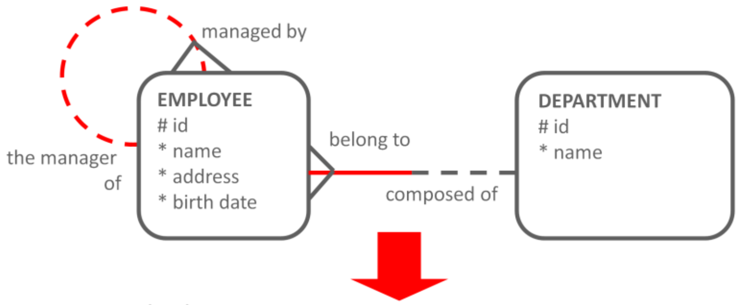
Rules for Relationships

- A relationship creates one or more foreign-key columns in the table on the many side of the relationship.
- We use the short name of the table to name the foreign-key column.
- In the example on the next page, the foreign-key column in the EMPLOYEES table is dpt_id for the relationship with DEPARTMENT, and mgr_id for the recursive relationship with itself.

Rules for Relationships

- The foreign-key column may be either mandatory or optional, depending on the needs of the business.
- In the example, dpt_id is mandatory and mgr_id is optional.

Rules for Relationships Illustrated



EMPLOYEES (EPE)

Key Type	Optionality	Column Name
pk	*	id
	*	name
	*	address
	*	birth_date
fk1	*	dpt_id
fk2	o	mgr_id

DEPARTMENTS (DPT)

Key Type	Optionality	Column Name
pk	*	id
uk	*	name

foreign key refers to

foreign key refers to

ORACLE ACADEMY

DDS9L3
Relationship Mapping

Mgr_id is optional to reflect that “each EMPLOYEE may be managed by…”

The ER diagram captures the relationships between entities, expressed in business terms. When the conceptual model is transformed, the relationships become foreign-key columns, but the relationship name itself is not carried over. The database design will be the basis for the system, but starting with a conceptual model ensures that the tables, columns, and constraints created in the database are relevant to the business and fulfill its requirements.

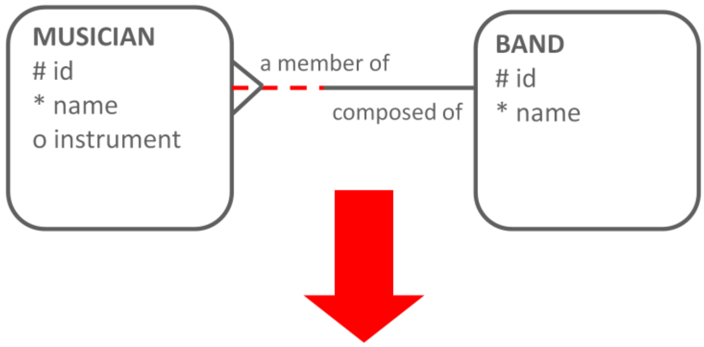
Mapping of Mandatory Relationship at the One Side

- Relationships that are mandatory on the one side, or mandatory on both sides, are mapped exactly the same way as a relationship that is optional on the one side.
- The conceptual model is rich enough to capture optionality at both ends of the relationship.
- However, the physical model is limited in that a foreign-key constraint can enforce a mandatory relationship only at the many end.

Mapping of Mandatory Relationship at the One Side

- In the following example, the physical model cannot enforce that a BAND must be composed of at least one MUSICIAN.
- The optionality at the one end will have to be implemented through additional programming.

Enforcing Optionality



MUSICIANS (MSN)

Key type	Optionality	Column name
pk	*	id
	*	name
	o	instrument
fk	o	bad_id

foreign key refers to

BANDS (BAD)

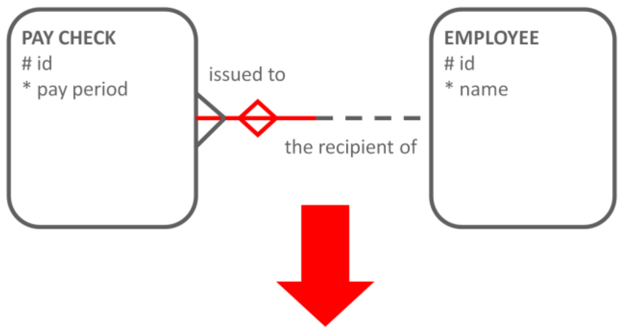
Key type	Optionality	Column name
pk	*	id
	*	name

Mapping of Nontransferable Relationships

- A nontransferable relationship in the conceptual model means that the foreign-key column in the database table cannot be updated.
- The foreign-key constraint by itself cannot enforce this in the database.
- Additional programming will be needed to make sure that the database follows this business rule.
- It is important to document rules like this so that the team remembers to write the appropriate code and enforce this business rule.

Nontransferable relationship: information that cannot be updated.

Enforcing Nontransferable Relationships



PAYCHECKS (PCK)

Key Type	Optionality	Column Name
pk	*	id
	*	pay_period
fk	*	epe_id

the value in this foreign-key column cannot be changed

ORACLE ACADEMY

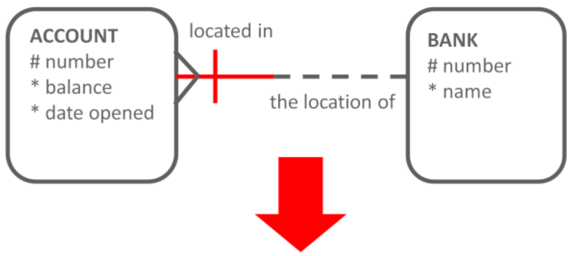
DDS9L3
Relationship Mapping

In the example, a paycheck may not be transferred to another employee. This means that the epe_id, which is the foreign-key column in the table PAYCHECKS, cannot be updated. This will additional programming to enforce.

Mapping of Barred Relationships

- A barred relationship is mapped to a foreign-key column on the many side, just like any other 1:M relationship.
- In this case, the foreign-key column plays a double role because it is also part of the primary key.
- In the example, bak_number is a foreign-key column in ACCOUNTS that refers to the primary key of BANKS.
- It is also part of the primary key of ACCOUNTS.

Mapping of Barred Relationships



ACCOUNTS (ACT)

Key Type	Optionality	Column Name
pk	*	act_nbr
	*	balance
	*	date_opened
pk,fk	*	bak_nbr

refers to

BANKS (BAK)

Key Type	Optionality	Column Name
pk	*	bank_number
	*	name

The act_nbr alone would not be unique within the table, but the combination of act_nbr and bak_nbr would be.

Cascade Barred Relationships

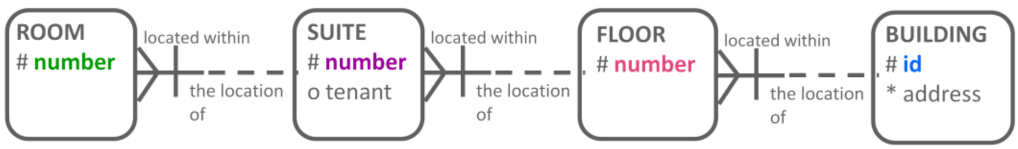
- Hierarchies can lead to cascade barred relationships, where the UID of the entity at the top of the hierarchy is carried all the way down to the UID of the entity at the bottom of the hierarchy.
- In the example, the UID of ROOM is composed of the ROOM number, SUITE number, FLOOR number, and BUILDING id.
- This is represented by the barred relationships.

Cascade barred relationships: A series of relationships implying that the unique identifier of each entity in the chain is carried down to the entity on the next level.

Cascade Barred Relationships

- When this is mapped to a physical model, the result can be a very long foreign-key column name because it uses the short names of the originating tables as a prefix.
- The suggested convention is to never use more than two table prefixes. In the following example, the foreign-key column in ROOMS that comes all the way from BUILDINGS is named sue_bdg_id, instead of sue_flr_bdg_id.

Cascade Barred Relationships



ROOMS (ROM)

pk	*	rom_nbr
pk, fk	*	sue_nbr
pk, fk	*	sue_flr_nbr
pk, fk	*	sue_bdg_id

SUITES (SUE)

pk	*	sue_nbr
pk, fk	*	flr_nbr
pk, fk	*	flr_bdg_id
	o	tenant

FLOORS (FLR)

pk	*	flr_nbr
pk, fk	*	bdg_id

BUILDINGS (BDG)

pk	*	id
	*	address

The UIDs and resulting primary keys are highlighted in different colors to help you trace a UID through the hierarchy.

The primary key of FLOORS now becomes a composite of flr_nbr and bldg_id. The primary key of SUITES is a composite of sue_nbr, flr_nbr, and flr_bldg_id. Point out that the composite is one primary key (a table is only allowed to have one PK).

In this case, it is also one foreign key for each table, even if that key is a composite of multiple columns. The foreign key in SUITES is the combination of flr_nbr and flr_bldg_id. The foreign key in ROOMs is a combination of sue_nbr, sue_flr_nbr, and sue_bdg_id.

Cascade Barred Relationship Illustrated

- Sample data for each table illustrates the cascade barred relationships.

SUITES

sue_nbr	flr_nbr	flr_bdg
15	2	100
25	2	100
5E	1	201
7B	2	201

BUILDINGS

id	address
100	40 Potters Lane
201	57G Maricopa Way

FLOORS

flr_nbr	bdg_id
1	100
2	100
1	201
2	201

ROOMS

rom_nbr	sue_nbr	sue_flr_nbr	sue_bdg_id
1	15	2	100
2	15	2	100
1	7B	2	201

What is the primary key of ROOMS?

Answer: The combination of rom_nbr, sue_nbr, sue_flr_nbr, and sue_bdg_id.

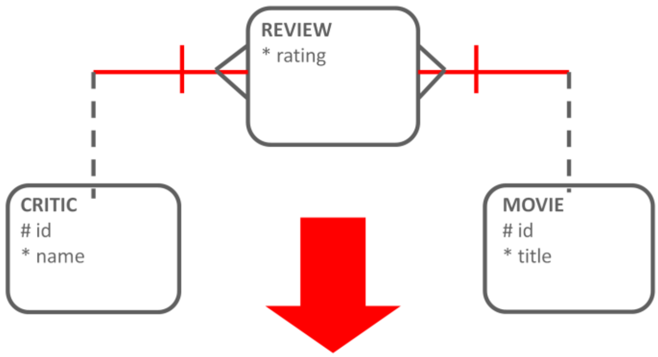
Verify that the combination of all four is what makes a row unique.

Mapping Many-to-Many Relationships

- A M:M relationship is resolved with an intersection entity, which maps to an intersection table.
- This intersection table will contain foreign-key columns that refer to the originating tables.
- In the example, REVIEWS contains all the combinations that exist between a CRITIC and a MOVIE.

Intersection entity: The product of the resolution of a many to many relationship.

Mapping Many-to-Many Relationships



CRITICS (CTC)

pk	*	id
	*	name

MOVIES (MVE)

pk	*	id
	*	title

REVIEWS (RVW)

Key Type	Optionality	Column Name
pk, fk1	*	ctc_id
pk, fk2	*	mve_id
	*	rating

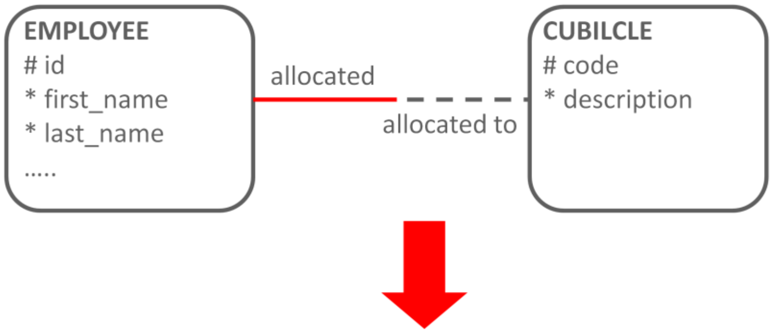
Mapping One-to-One Relationships

- When transforming a 1:1 relationship, you create a foreign key and a unique key.
- All columns of this foreign key are also part of the unique key.
- If the relationship is mandatory on one side, the foreign key is created in the corresponding table.
- In the example, cbe_code is the foreign-key column in EMPLOYEES that refers to the primary key of CUBICLES.
- Cbe_code would also be unique within the EMPLOYEES table.

Each EMPLOYEE must be allocated one and only one CUBICLE.

Each CUBICLE may be allocated to one and only one EMPLOYEE.

Mapping One-to-One Relationships



EMPLOYEES (EPE)

pk	*	id
	*	name
fk, uk	*	cbe_code

CUBICLES (CBE)

pk	*	code
	*	description

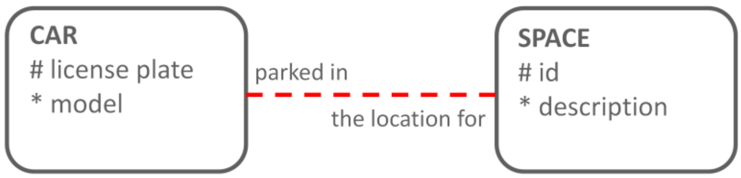
Optional One-to-One

- If the relationship is optional on both sides, you can choose which table gets the foreign key.
- There are no absolute rules, but here are some guidelines:
  - Implement the foreign key in the table with fewer rows to save space.
  - Implement the foreign key where it makes more sense for the business.

Optional One-to-One

- In the example, a car-rental agency would be more concerned about cars than spaces, so it makes sense to put the foreign key in CARS.
- However, in a parking-lot business, the main object is the parking space.
- Therefore, it would make sense to put the foreign key in SPACES.

Business Rules for Optional One-to-One



Car-Rental Business

CARS (CAR)

pk	*	lic_plate
	*	model
fk, uk	o	spe_id

SPACES (SPE)

pk	*	id
	*	description

Parking-Lot Business

CARS (CAR)

pk	*	lic_plate
	*	model

SPACES (SPE)

pk	*	id
	*	description
fk, uk	o	car_lic_plate

Enforcing One-to-Many

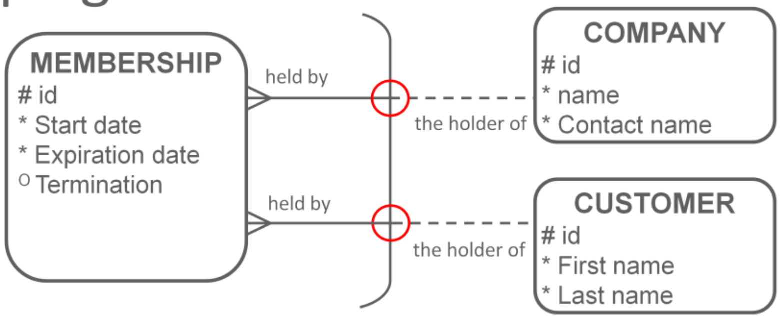
- If the relationship is mandatory at both ends, you have the same limitation in the database as a 1:M relationship that is mandatory at the one end.
- Therefore, you would need to write additional code to enforce it.

Mandatory 1:1 relationships are rare. In most cases, this will be modeled as a single table, without a need for a 1:1 relationship.

Mapping Arcs

- The entity that has the arc will map to a table that contains foreign keys from the tables on the “one” end of the relationships.
- Note that even if the relationships in the arc are mandatory on the many side, the resulting foreign keys have to be optional (because one of them will always be blank).

Mapping Arcs



COMPANIES (CPE)

pk	*	id
	*	name
	*	contact _name

MEMBERSHIPS (MBP)

pk	*	id
	*	start_date
	*	expiration_date
	o	termination
fk1	o	cpe_id
fk2	o	cms_id

CUSTOMERS (CMS)

pk	*	id
	*	first_name
	*	last_name

Mapping Arcs

- Since the arc represents exclusive relationships, additional code is needed to enforce that only one of the foreign keys has a value for every row in the table.
- A check constraint stored in the database can easily do this.
- In the example, the code for the check constraint would look like this:
  – CHECK (pse_id is not null AND phe_id is null)
  – OR (pse_id is null AND phe_id is not null)
- If the relationships were fully optional, you would add:
  – OR (pse_id is null AND phe_id is null)

A check constraint is programming code that can be stored in the database. It can enforce simple rules that apply to a single row in the table (such as comparing values or ensuring that they are null or not null). This is the case with the arc.

In the case of a mandatory one end of a 1:M or 1:1 relationship, we have to check that if a row is entered in one table (the master), a row must also be entered in another table (the child, or detail). A check constraint cannot span two tables or different rows in the same table. It cannot prevent insert, update, or delete operations. This is why additional programming (instead of a check constraint) is necessary.

Terminology

Key terms used in this lesson included:

- Cascade barred relationship
- Intersection entity
- Nontransferable relationship

Summary

In this lesson, you should have learned how to:

- Apply the rule of relationship mapping to correctly transform 1:M and barred relationships
- Apply the rule of relationship mapping to correctly transform M:M relationships
- Transform 1:1 relationships
- Apply the rule of relationship mapping to correctly transform relationships in an arc

ORACLE® ACADEMY