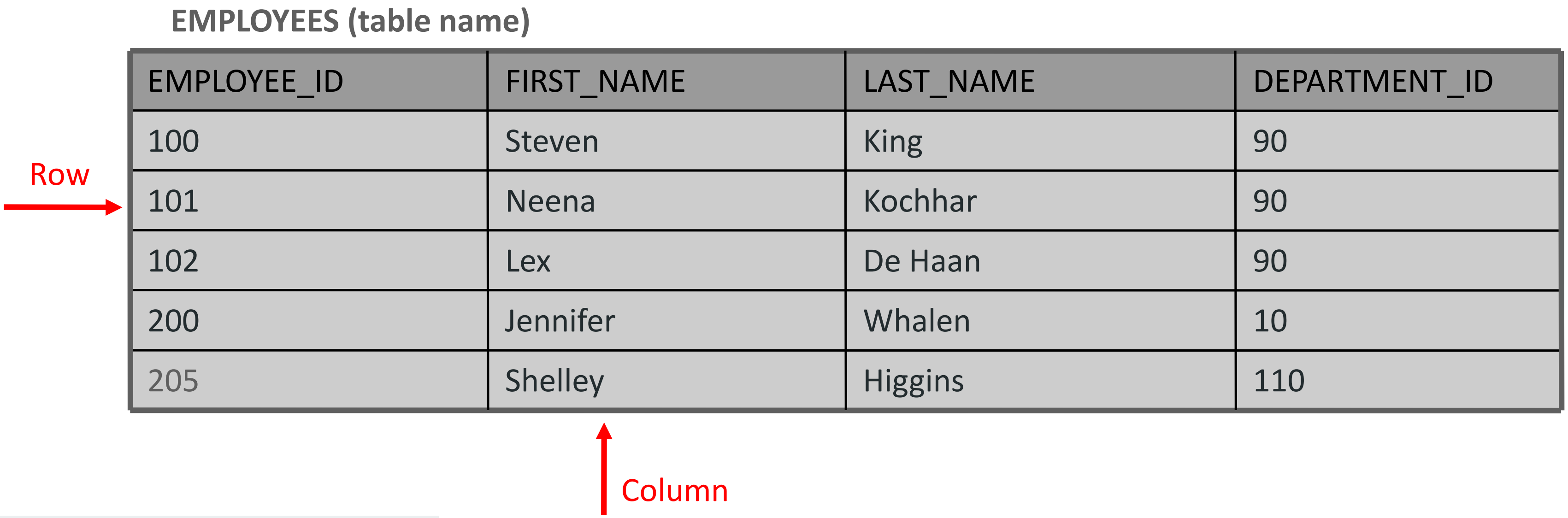
Database Design

9-1
Introduction to Relational Database Concepts

ORACLE ACADEMY

Relational Database Illustrated

- A relational database is a database that is seen by the user as a collection of two-dimensional tables, each containing rows and columns.
- The table below contains employee data.



EMPLOYEES (table name)

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

Primary Key

- A primary key (PK) is a column or set of columns that uniquely identifies each row in a table.

ACCOUNTS

BANK_NO	ACCT_NO	BALANCE	DATE_OPENED
104	75760	12,0050.00	21-OCT-89
104	77956	100.10	
105	89570	55,775.00	15-JAN-85
103	55890	15,001.85	10-MAR-91
105	75760	5.00	22-SEP-03

Multiple Column Primary Key

EMPLOYEES

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

Single Column Primary Key

Primary Key

- Each table should have a primary key, and a primary key must be unique.

ACCOUNTS

BANK_NO	ACCT_NO	BALANCE	DATE_OPENED
104	75760	12,0050.00	21-OCT-89
104	77956	100.10	
105	89570	55,775.00	15-JAN-85
103	55890	15,001.85	10-MAR-91
105	75760	5.00	22-SEP-03

Multiple Column Primary Key

EMPLOYEES

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

Single Column Primary Key

Primary Key

- No part of the primary key can be null.

ACCOUNTS

BANK_NO	ACCT_NO	BALANCE	DATE_OPENED
104	75760	12,0050.00	21-OCT-89
104	77956	100.10	
105	89570	55,775.00	15-JAN-85
103	55890	15,001.85	10-MAR-91
105	75760	5.00	22-SEP-03

Multiple Column Primary Key

EMPLOYEES

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

Single Column Primary Key

Primary Key Candidates

- A table can have more than one column, or combinations of columns, that could serve as the table’s primary key.
- Each column, or combination of columns, is called a "candidate" key because it could be selected for use as the primary key.

MEMBERS

MEMBER_ID	LAST_NAME	FIRST_NAME	PAYROLL_ID
100	SMITH	DANA	21215
310	ADAMS	TYLER	59877
210	CHEN	LAWRENCE	1101
405	GOMEZ	CARLOS	52
378	LOUNGANI	NEIL	90386

↑ Candidate Key

Candidate Key ↑

Choose a Candidate Key

- Select one candidate key to be the primary key for the table.
- The other candidates become alternate keys (or unique keys).

MEMBERS

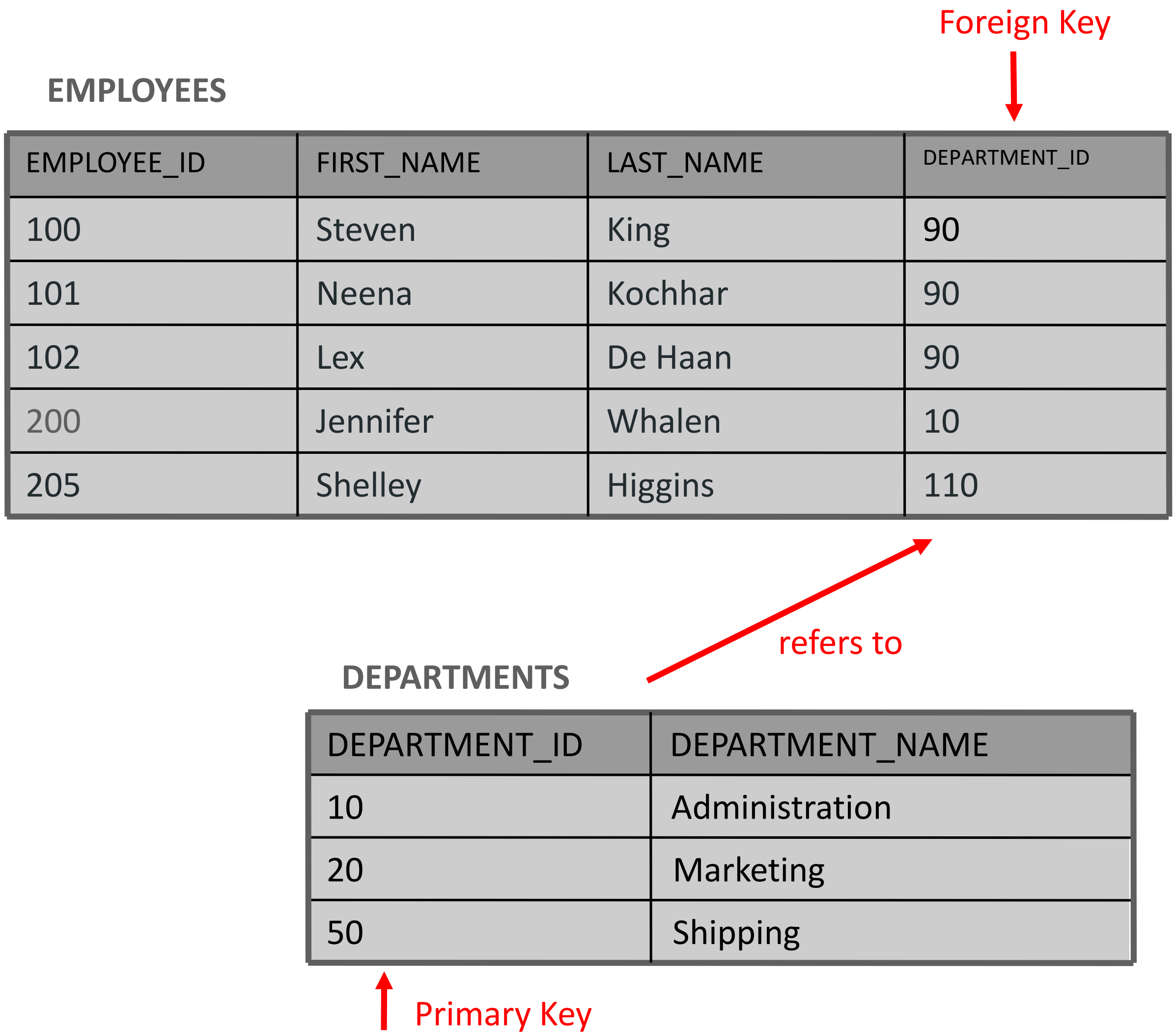
MEMBER_ID	LAST_NAME	FIRST_NAME	PAYROLL_ID
100	SMITH	DANA	21215
310	ADAMS	TYLER	59877
210	CHEN	LAWRENCE	1101
405	GOMEZ	CARLOS	52
378	LOUNGANI	NEIL	90386

Primary Key

Alternate or
Unique Key (UK)

Foreign Key

- A foreign key (FK) is a column, or combination of columns, in one table that contains values that match the primary key value in another table.



Foreign Key

EMPLOYEES

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

refers to

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
50	Shipping

Primary Key

Summary of Data-Integrity Rules

Constraint Type	Explanation	Example
Entity Integrity	A primary key must be unique, and no part of the primary key can be null	The column emp_no in the EMPLOYEES table cannot be null
Referential Integrity	A foreign key must match an existing primary key value (or else be null if nulls are allowed)	The value in the dept_no column of the EMPLOYEES table must match a value in the dept_no column in the DEPARTMENTS table
Column Integrity	A column must contain only values consistent with the defined data format of the column	The value in the balance column of the ACCOUNTS table must be numeric
User-Defined Integrity	The data stored in a database must comply with the rules of the business	If the value in the balance column of the ACCOUNTS table is below 1.00, we must send a letter to the account owner (this will need additional programming to enforce)

Database Design

9-2
Basic Mapping: The Transformation Process

ORACLE ACADEMY

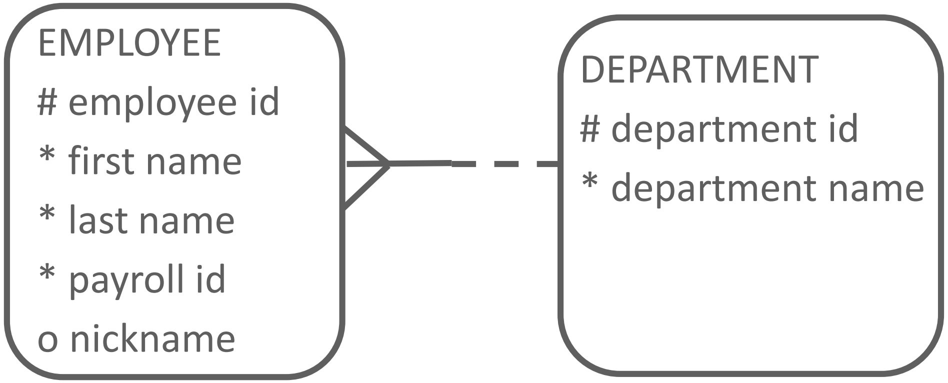
Transforming Conceptual To Physical

- The conceptual model (ER diagram) is transformed into a physical model.
- The physical implementation will be a relational database.

Transforming Conceptual To Physical

Conceptual Model (ERD)

Transformation process



Physical Implementation: Relational Database

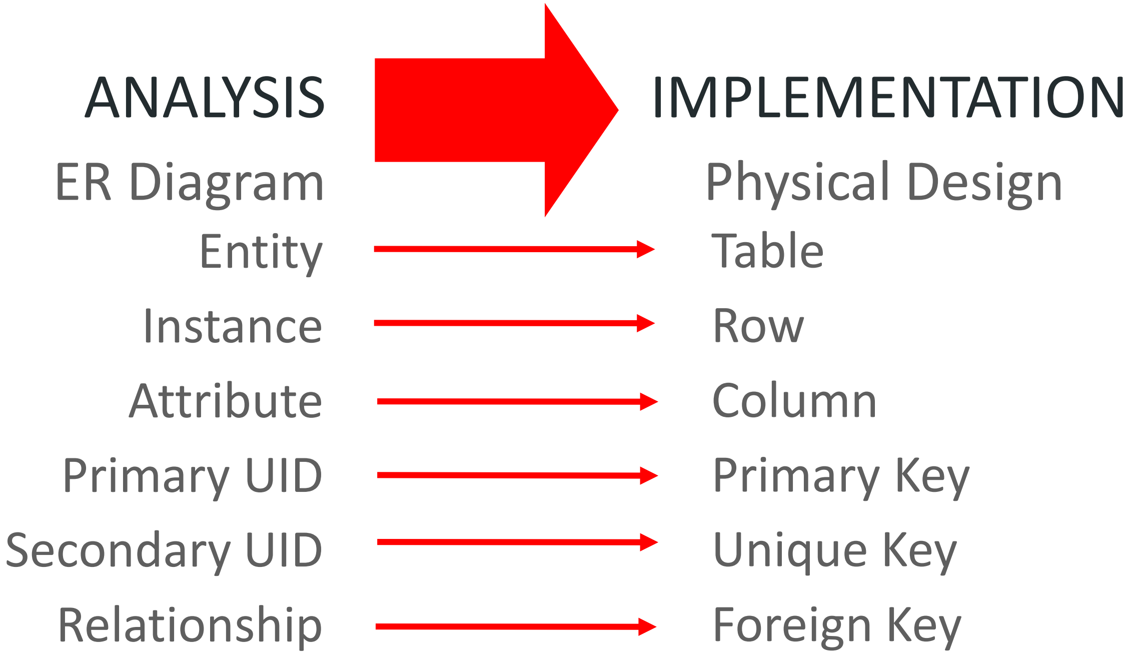
EMPLOYEES (EPE)

Key type	Optionality	Column name
pk	*	employee_id
uk	*	payroll_id
	*	last_name
	*	first_name
	o	nickname
fk	*	department_id

DEPARTMENTS (DPT)

Key type	Optionality	Column name
pk	*	department_id
	*	department_name

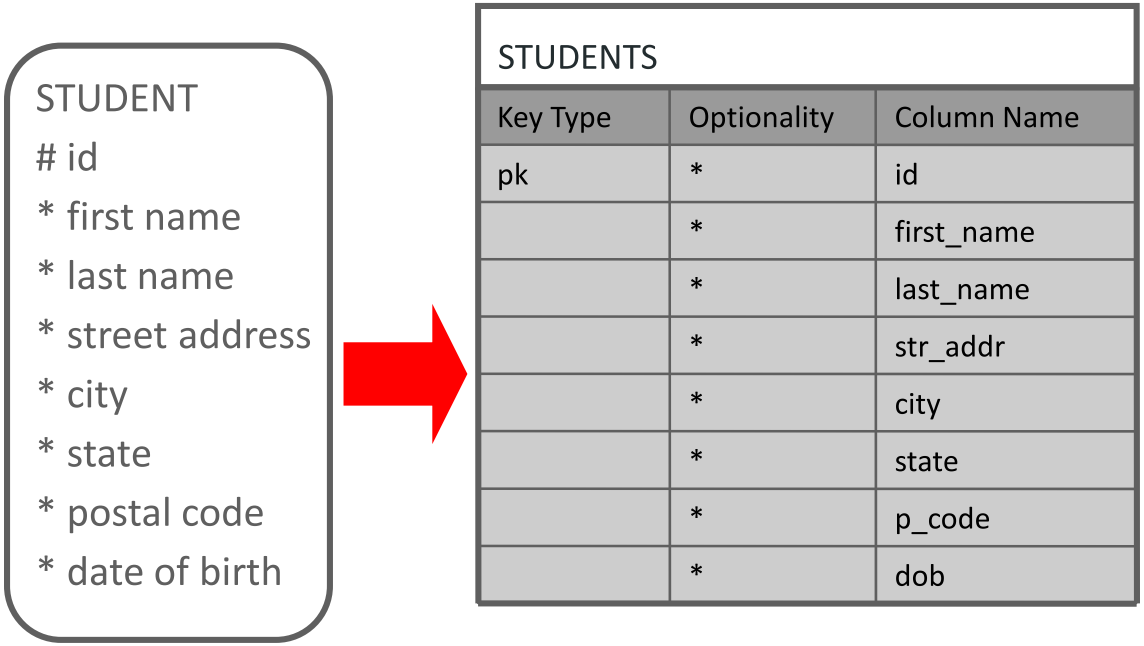
Terminology Mapping



ANALYSIS	IMPLEMENTATION
ER Diagram	Physical Design
Entity	Table
Instance	Row
Attribute	Column
Primary UID	Primary Key
Secondary UID	Unique Key
Relationship	Foreign Key

Naming Conventions for Tables and Columns

- The table name is the plural of the entity name.
- Example: STUDENT becomes STUDENTS



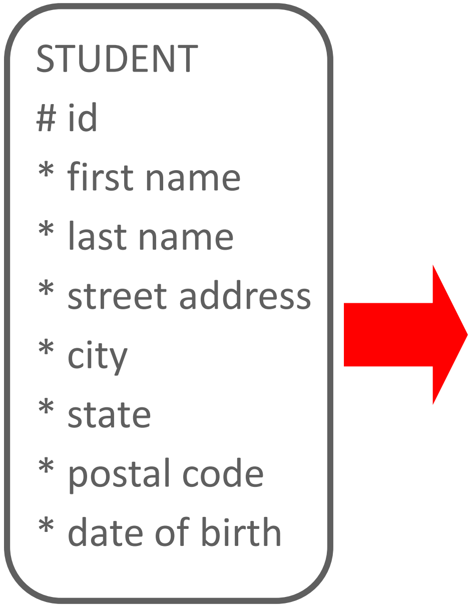
STUDENT
# id
* first name
* last name
* street address
* city
* state
* postal code
* date of birth

STUDENTS

Key Type	Optionality	Column Name
pk	*	id
	*	first_name
	*	last_name
	*	str_addr
	*	city
	*	state
	*	p_code
	*	dob

Naming Conventions for Tables and Columns

- Column names are identical to the attribute names except that special characters and spaces are replaced with underscores.



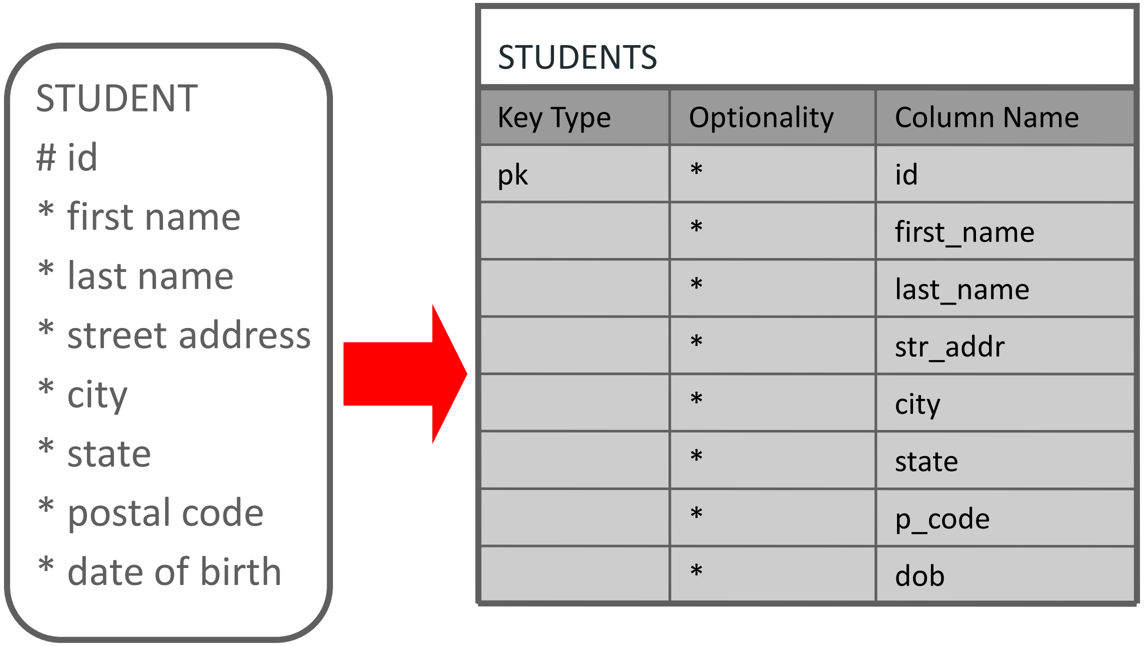
STUDENT
# id
* first name
* last name
* street address
* city
* state
* postal code
* date of birth

STUDENTS

Key Type	Optionality	Column Name
pk	*	id
	*	first_name
	*	last_name
	*	str_addr
	*	city
	*	state
	*	p_code
	*	dob

Naming Conventions for Tables and Columns

- Column names often use more abbreviations than attribute names. Example: first name becomes first_name, or fname



STUDENT
# id
* first name
* last name
* street address
* city
* state
* postal code
* date of birth

STUDENTS

Key Type	Optionality	Column Name
pk	*	id
	*	first_name
	*	last_name
	*	str_addr
	*	city
	*	state
	*	p_code
	*	dob

Table Short Names

- A unique short name for every table is useful in the naming of foreign-key columns.
- One possible way to make these short names is based on the following rules:
- For entity names of more than one word, take the:
  - First character of the first word
  - First character of the second word
  - Last character of the last word
- Example: JOB ASSIGNMENT gets a short name of JAT

Naming Restrictions with Oracle

Table and column names:

- Must start with a letter
- Can contain up to 30 alphanumeric characters
- Cannot contain spaces or special characters such as “!,” but “$,” “#,” and “_” are permitted.
- Table names must be unique within one user account in the Oracle database.
- Column names must be unique within a table.

Naming Restrictions with Oracle

- Some words have a special meaning in the Oracle database and in the SQL programming language.
- These are called “reserved” words.
- It is best to avoid using these as names for your tables and columns.

Naming Restrictions with Oracle

- Some common examples of Oracle reserved words are:
  - TABLE
  - NUMBER
  - SEQUENCE
  - ORDER
  - VALUES
  - LEVEL
  - TYPE
- A complete list can be found on otn.oracle.com.

Database Design

9-3
Relationship Mapping

ORACLE ACADEMY

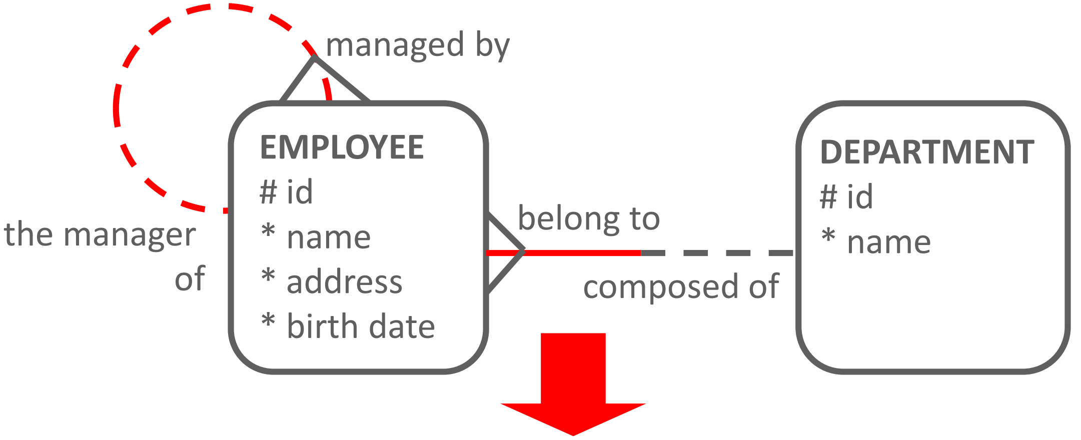
Rules for Relationships

- 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 ahead, 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



managed by

the manager of

EMPLOYEE
# id
* name
* address
* birth date

belong to

composed of

DEPARTMENT
# id
* name

EMPLOYEES (EPE)

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

DEPARTMENTS (DPT)

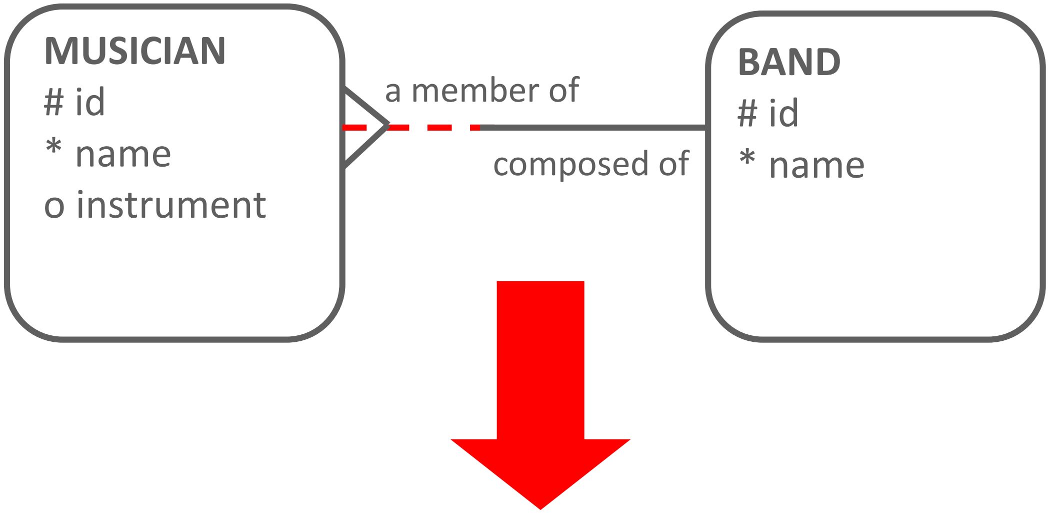
Key Type	Optionality	Column Name
pk	*	id
uk	*	name

foreign key refers to

foreign key refers to

ORACLE ACADEMY

Enforcing Optionality



MUSICIAN
# id
* name
o instrument

a member of

composed of

BAND
# id
* name

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 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.

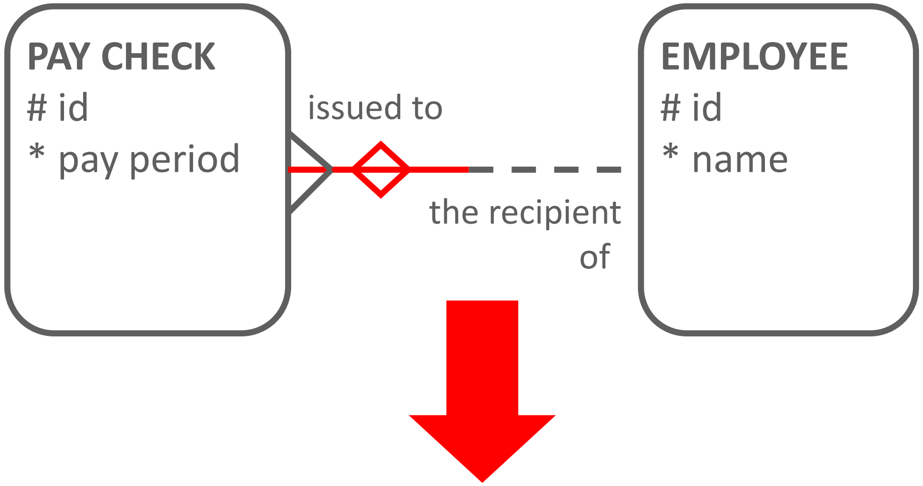
Enforcing One-to-Many

- 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.

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.

Enforcing Nontransferable Relationships



PAY CHECK
# id
* pay period

issued to

the recipient of

EMPLOYEE
# id
* name

PAYCHECKS (PCK)

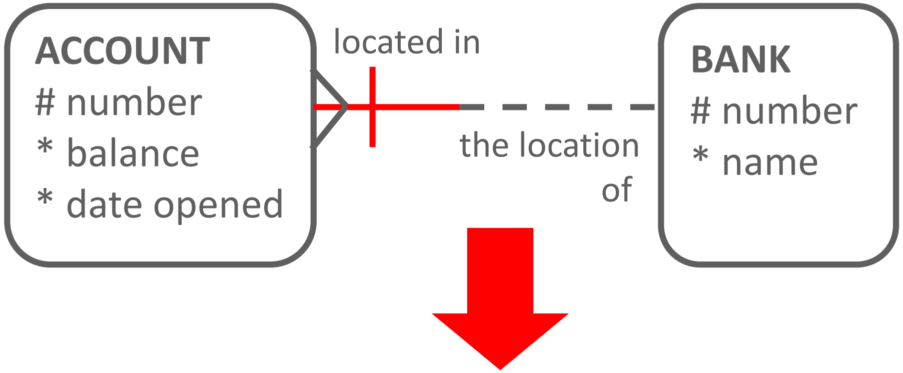
Key Type	Optionality	Column Name
pk	*	id
	*	pay_period
fk	*	epe_id

the value in this foreign-key column cannot be changed

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

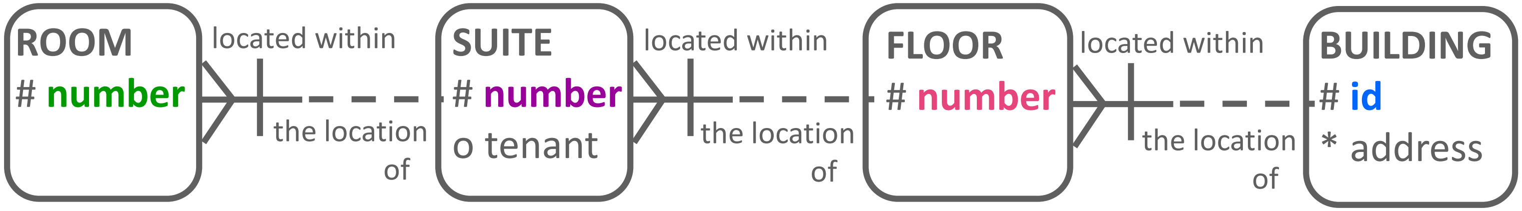
Cascade Barred Relationships

- 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

- 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)

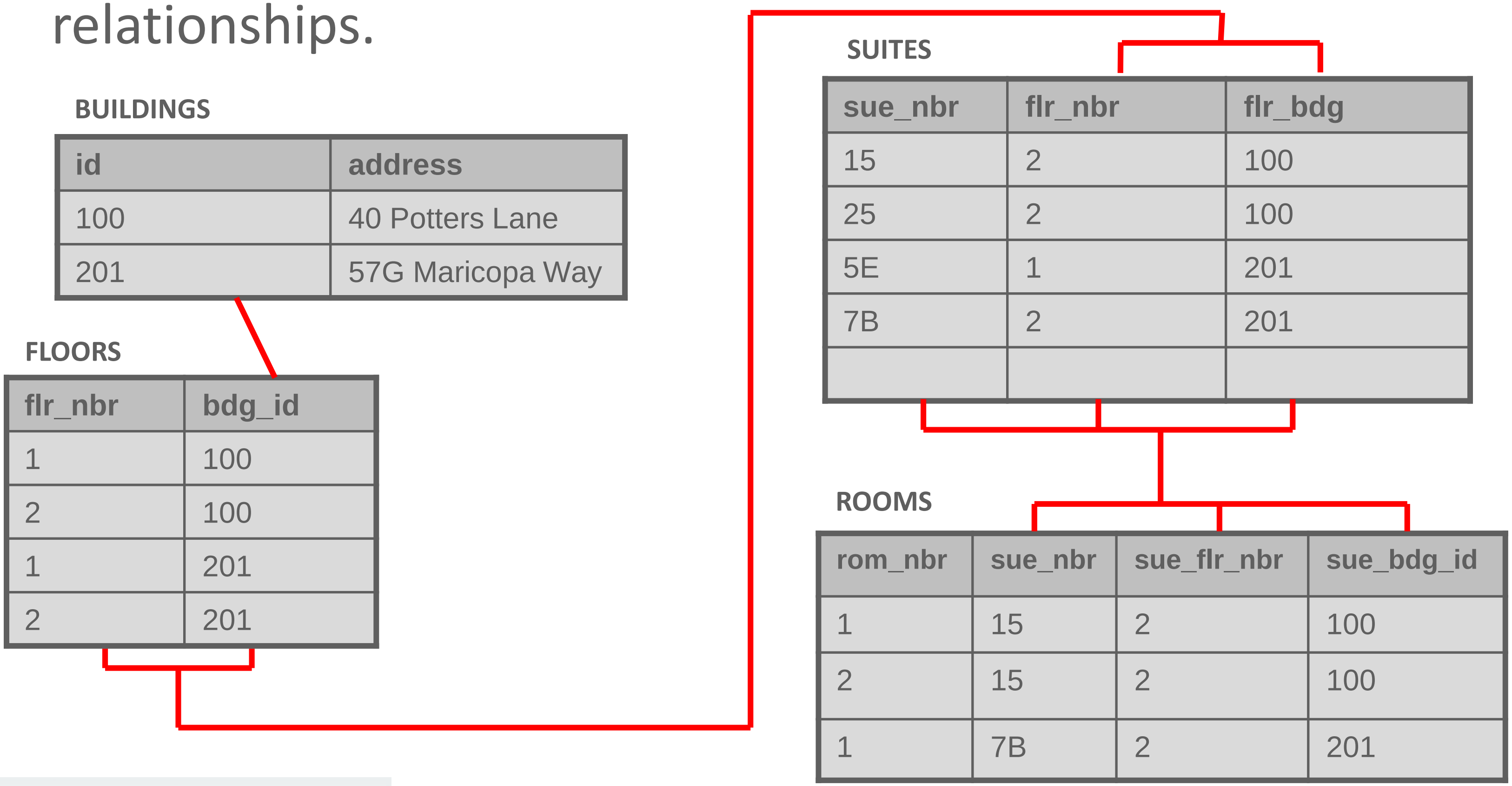
pk	*	flr_nbr
pk, fk	*	bdg_id

BUILDINGS (BDG)

pk	*	id
	*	address

Cascade Barred Relationship Illustrated

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



BUILDINGS

id	address
100	40 Potters Lane
201	57G Maricopa Way

FLOORS

flr_nbr	bdg_id
1	100
2	100
1	201
2	201

SUITES

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

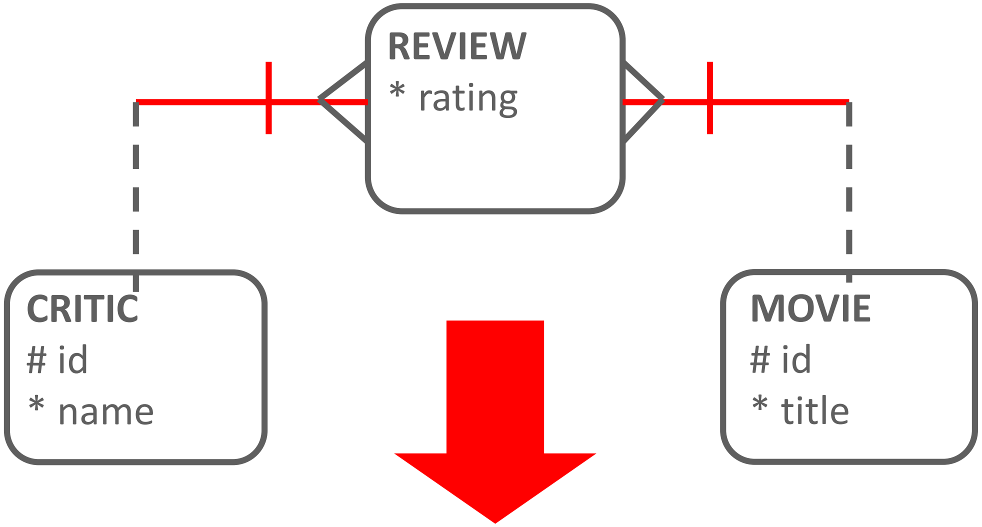
ROOMS

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

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.

Mapping Many-to-Many Relationships



CRITICS (CTC)

pk	*	id
	*	name

MOVIES (MVE)

pk	*	id
	*	title

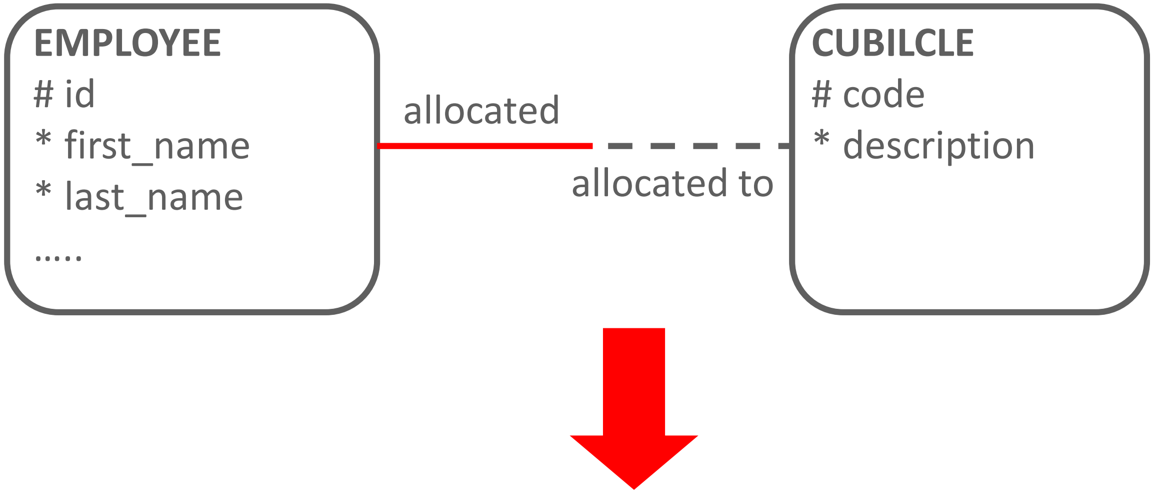
REVIEWS (RVW)

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.

Mapping One-to-One Relationships



EMPLOYEES (EPE)

pk	*	id
	*	name
fk, uk	*	cbe_code

CUBICLES (CBE)

pk	*	code
	*	description

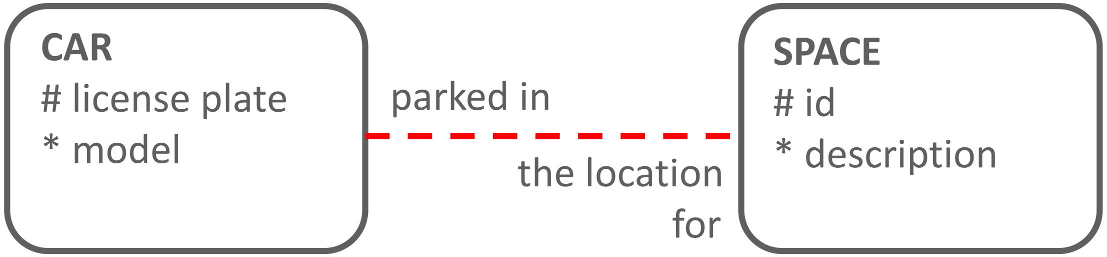
Optional One-to-One

- 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
# license plate
* model

parked in

the location for

SPACE
# id
* description

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

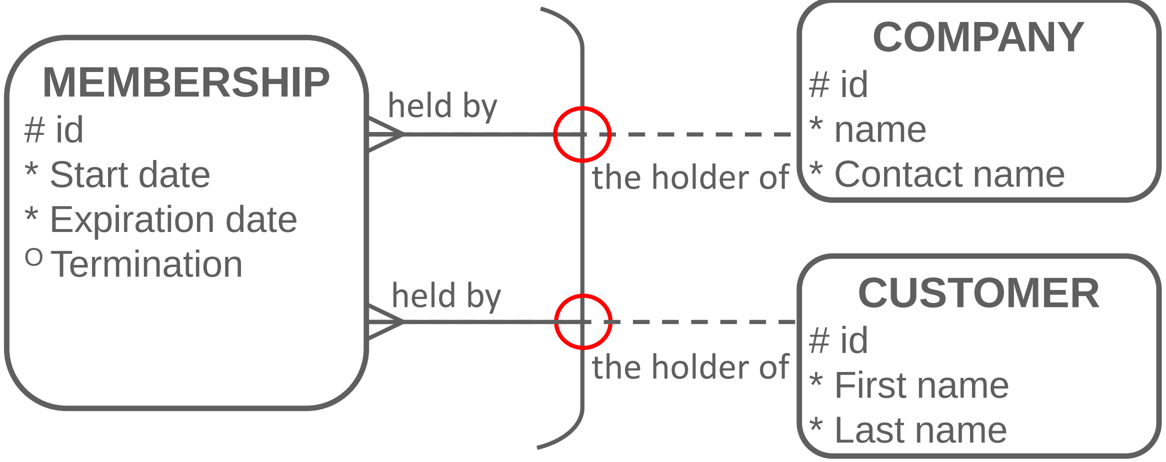
SPACES (SPE)

pk	*	id
	*	description
fk, uk	o	car_lic_plate

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



MEMBERSHIP
# id
* Start date
* Expiration date
o Termination

held by

the holder of

held by

the holder of

COMPANY
# id
* name
* Contact name

CUSTOMER
# id
* First name
* Last name

MEMBERSHIPS (MBP)

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

COMPANIES (CPE)

pk	*	id
	*	name
	*	contact _name

CUSTOMERS (CMS)

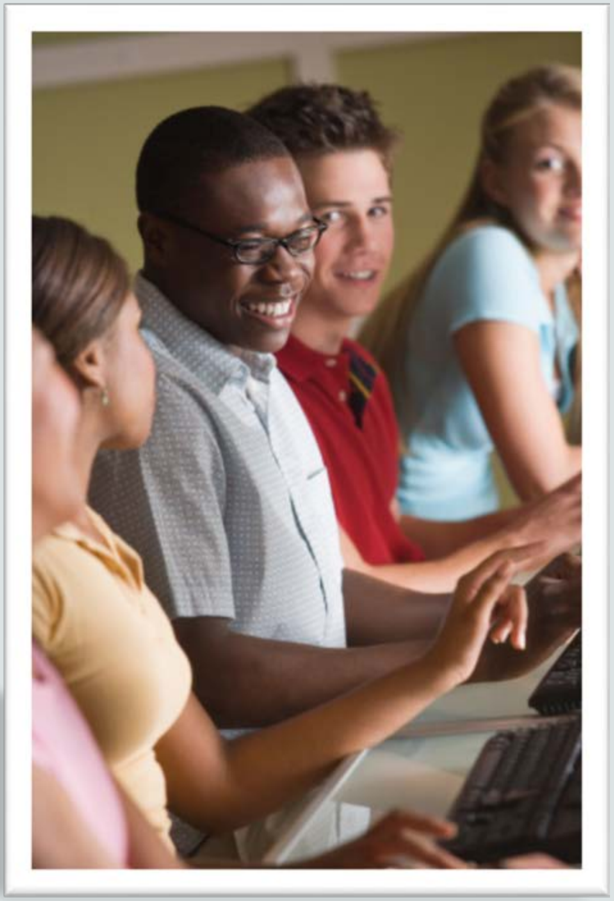
pk	*	id
	*	first_name
	*	last_name

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)

Database Design

9-4
Subtype Mapping



ORACLE ACADEMY

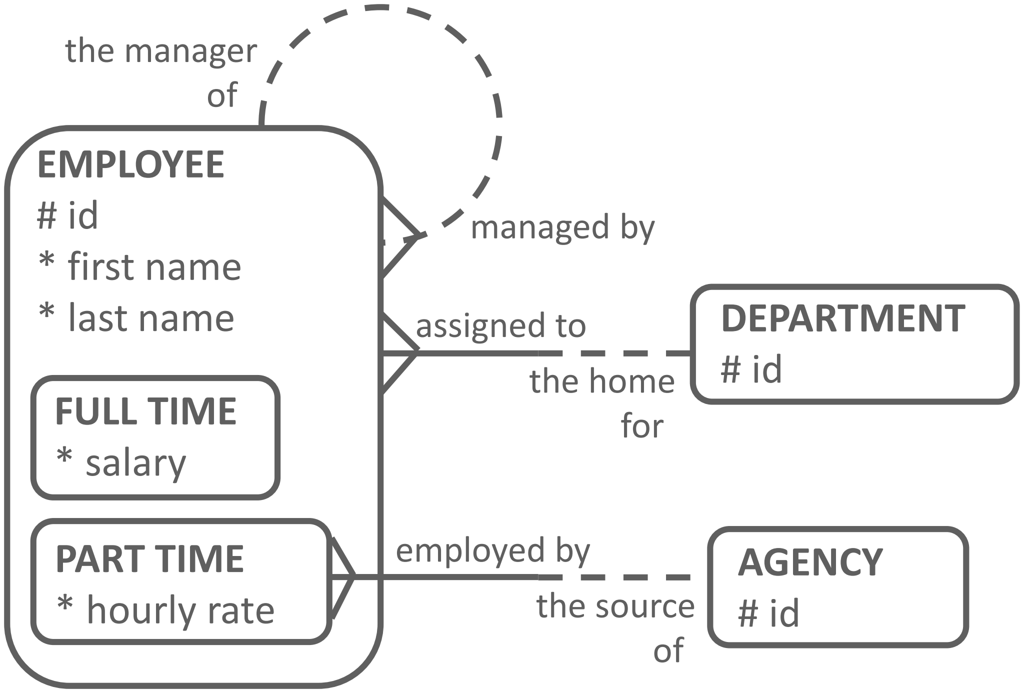
Supertype Implementation: Single Table

- 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 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

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.

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	epe_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.

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.

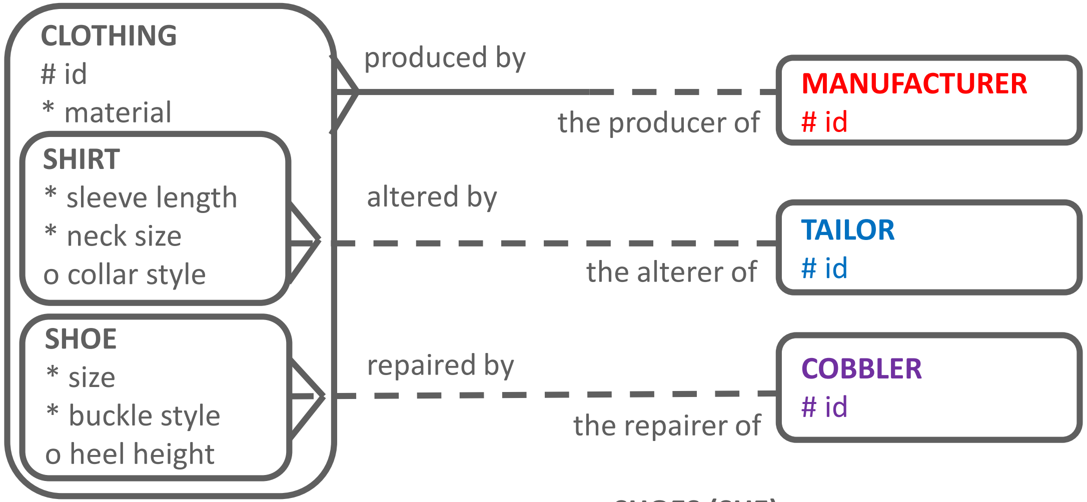
Subtype Implementation: Two Table

- 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.

Subtype Implementation: Two Table



SHIRTS (SHT)

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

refers to manufacturers

refers to tailors

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 manufacturers

refers to cobblers

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	button down	65	14
11	wool	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

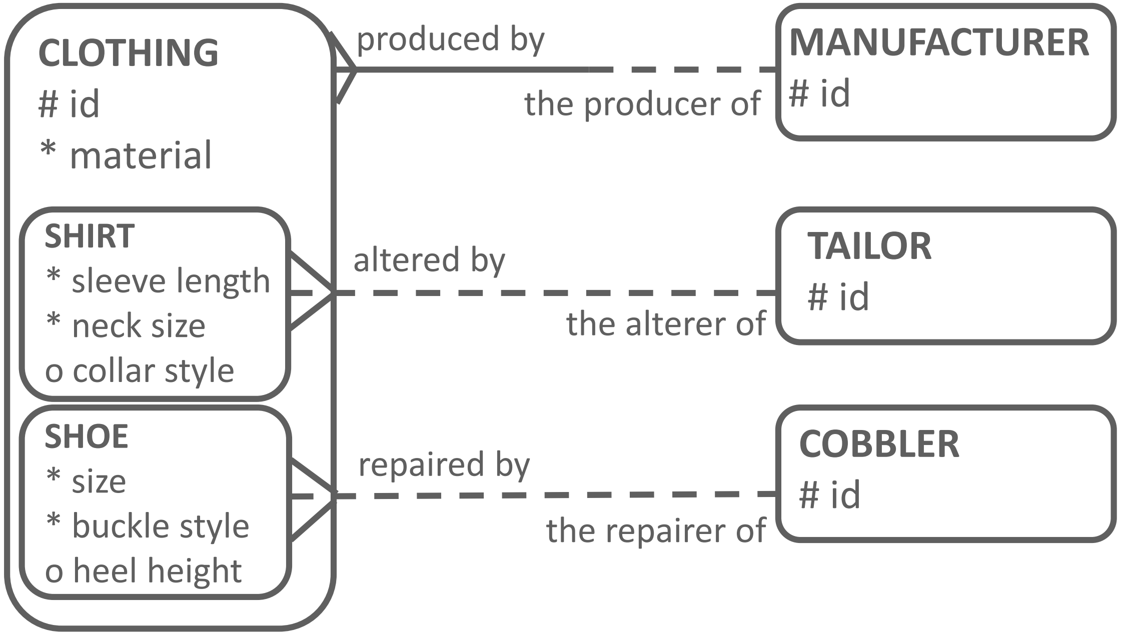
When to Consider Subtype Implementation

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.

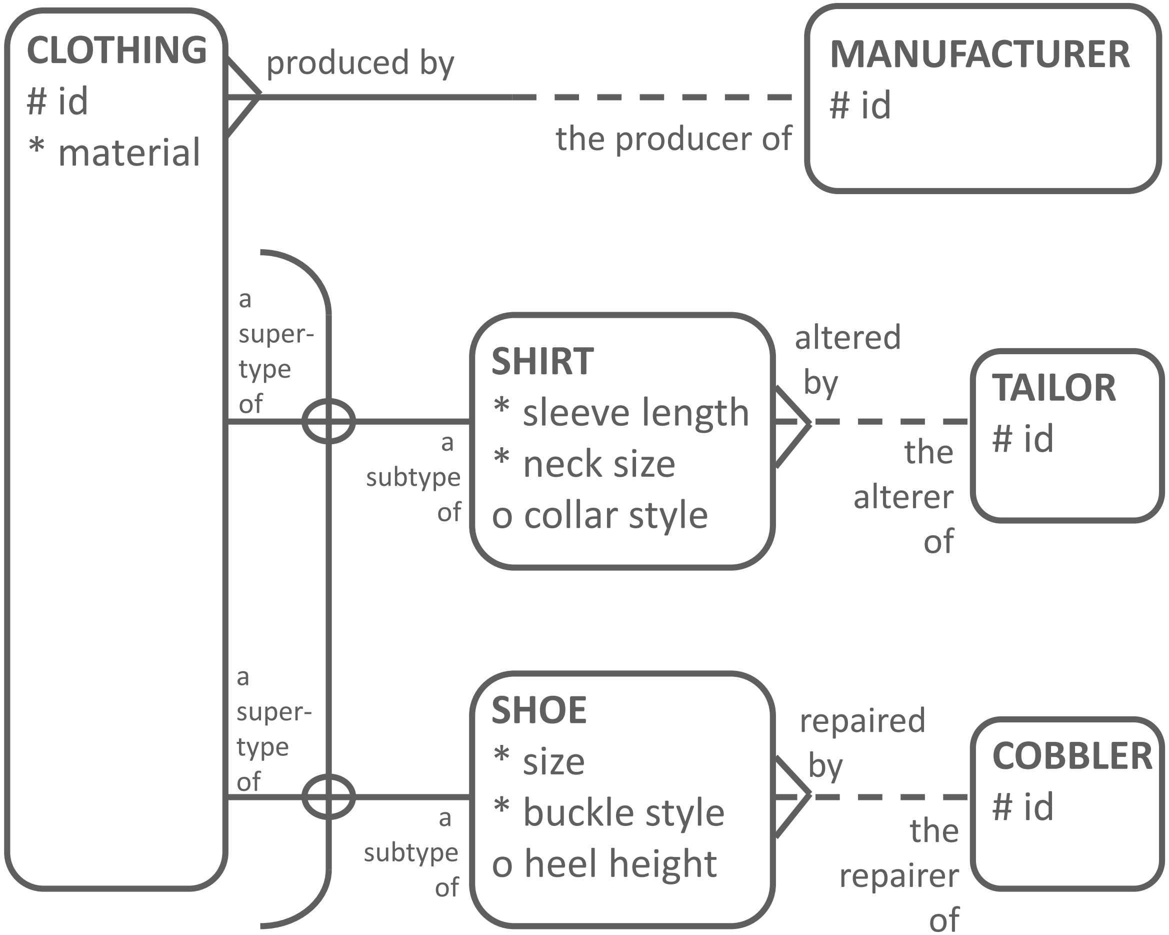
Modeling the Supertype as an Arc

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



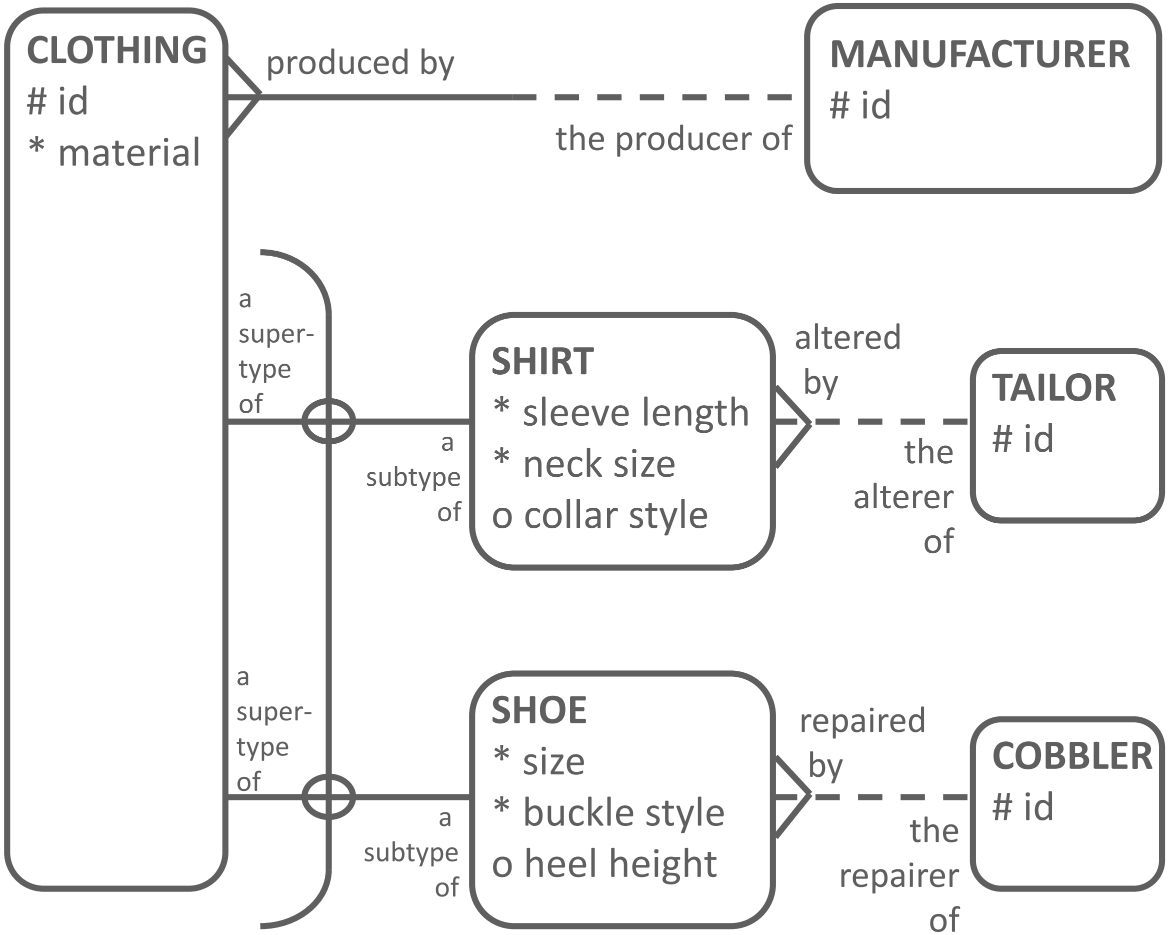
Model An Arc Illustrated

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



Model An Arc Illustrated

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



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

SHIRTS (SHT)

Key Type	Optionality	Column Name
pk	*	id
	*	sleeve_length
	*	neck_size
	o	collar_style
fk1	o	tlr_id

→ refers to tailors

CLOTHING (CTG)

Key Type	Optionality	Column Name
pk	*	id
	*	material
fk1, uk1	o	sht_id
fk2, uk2	o	she_id
fk3	*	mnr_id

→ refers to shirts

→ refers to shoes

→ refers to manufacturers

SHOES (SHE)

Key Type	Optionality	Column Name
pk	*	id
	*	size
	*	buckle_style
	o	heel_height
fk1	o	clr_id

→ refers to cobblers

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.