Database Technologies

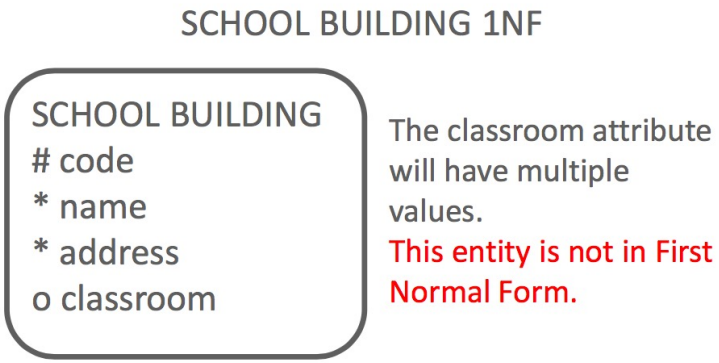
MS-218
Semester 1
Session 2021-22

Normalization(Purpose)

- Normalization is a process that is used to eliminate the problem of redundancy.
- Goal: "store information in one place and in the best possible place“.
- If you follow the rules of normalization, you will achieve this goal.

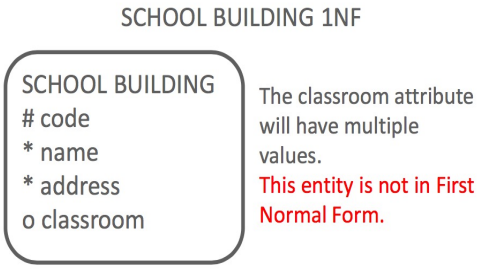
First Normal Form (1NF)

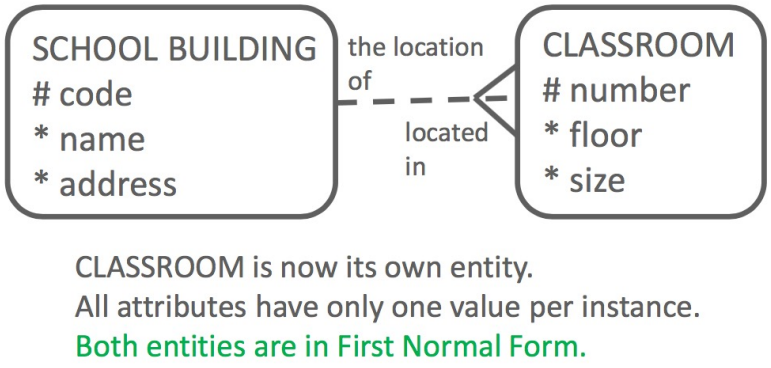
- First Normal Form requires that no multi-valued attributes exist.
- To check for 1NF, validate that each attribute has a single value for each instance of the entity.
- One code, one name, and one address exist for the school building, but not one classroom.



First Normal Form (1NF)

- Since many classrooms exist in a school building, classroom is multi-valued and violates 1NF.
- If an attribute is multi- valued, create an additional entity and relate it to the original entity with a 1:M relationship.





Second Normal Form

- Examine the entity PRODUCT SUPPLIER. Any Issue?

Second Normal Form

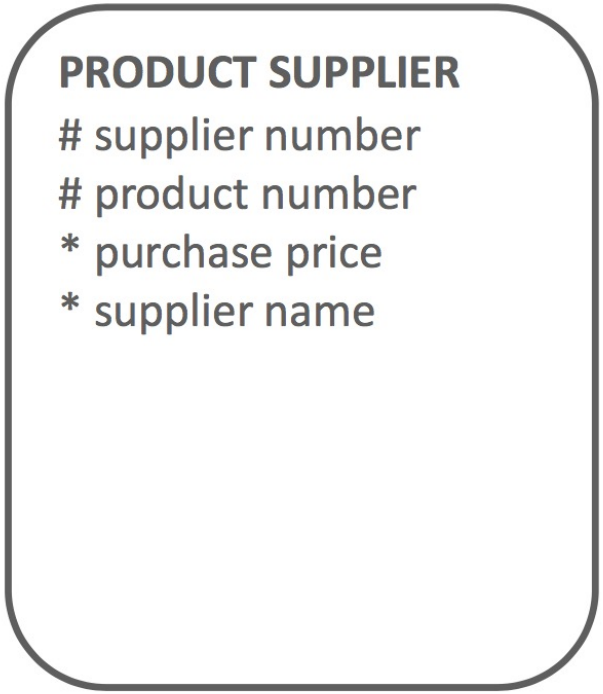
- Examine the entity PRODUCT SUPPLIER.
- The UID is a composite UID consisting of the supplier number and the product number.
- If one supplier supplies 5 different products, then 5 different instances are created.
- What happens if the supplier name changes?

Second Normal Form

- The supplier name would then need to be changed in 5 different instances.
- What if some of them were changed, but not others?
- How would users know which name is the correct name?

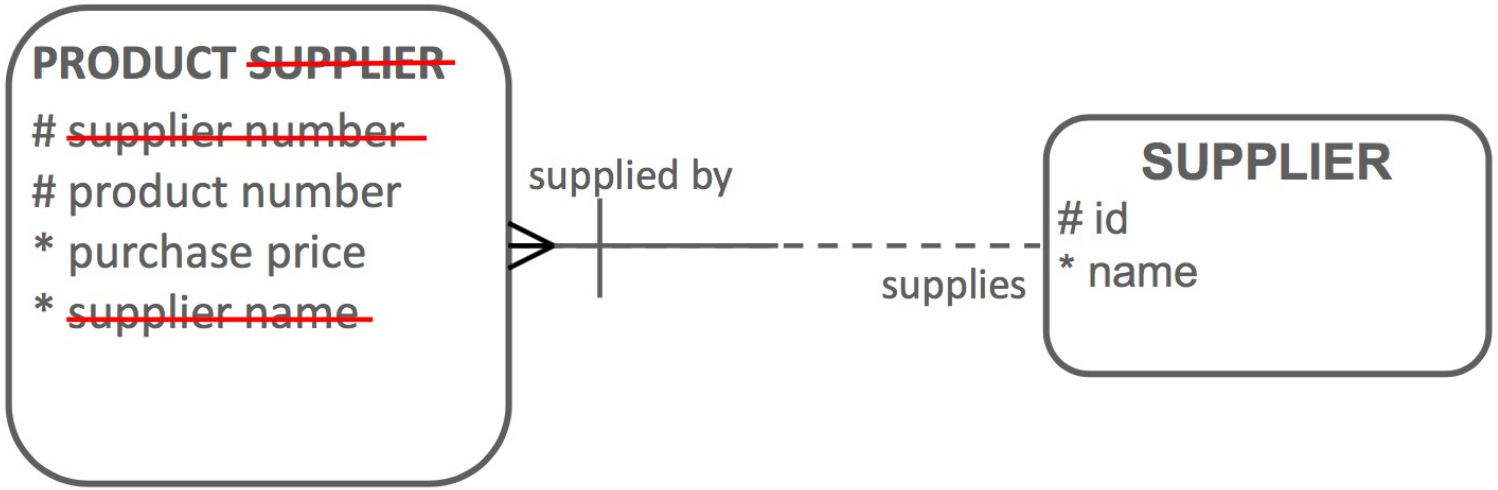
Second Normal Form

- Second Normal Form (2NF) requires that any non-UID attribute be dependent on (be a property of, or a characteristic of) the entire UID.
- Is purchase price a property of supplier number, product number, or both?



Second Normal Form

- Is supplier name a property of supplier number, product number, or both?
- 2NF requires a “both” answer to each question.



Third Normal Form Rule

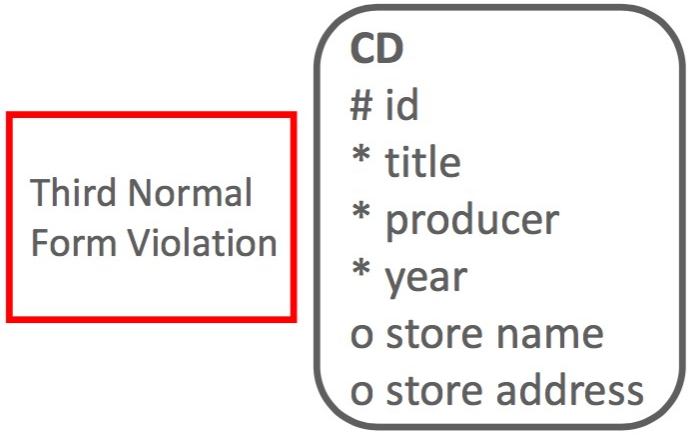
- The rule of Third Normal Form (3NF) states that no non-UID attribute can be dependent on another non-UID attribute.
- Third Normal Form prohibits transitive dependencies.
- A transitive dependency exists when any attribute in an entity is dependent on any other non-UID attribute in that entity.

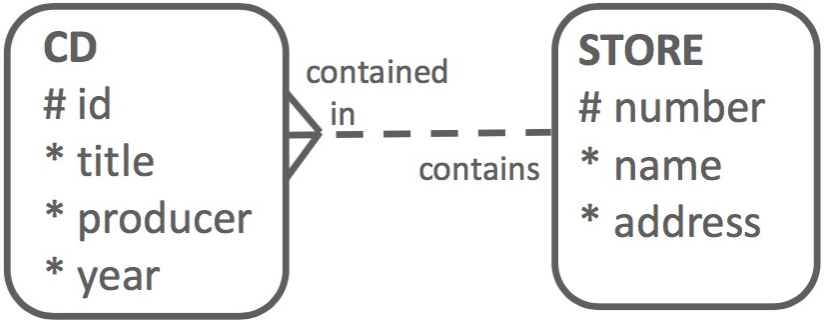
Third Normal Form Rule

- Think of the kind of information you'd like to store about your CD collection.
- Does information about the store where you bought the CD belong in the same entity?
- If the store address changed, you would have to change the information on all the CDs that were bought at that store.

Third Normal Form Rule

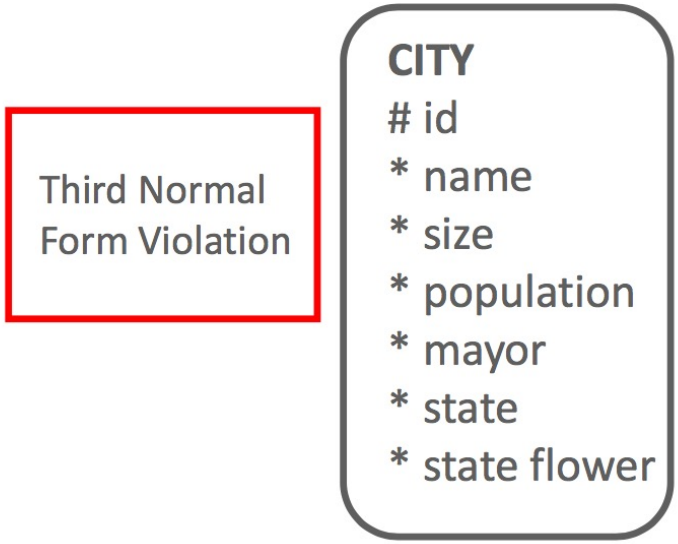
- The store address is dependent on the CD number, which is the UID of the CD entity. So this entity is in 1NF and 2NF.
- But store address is also dependent on store name, which is a non-UID attribute.
- This is an example of a transitive dependency and a violation of Third Normal Form.





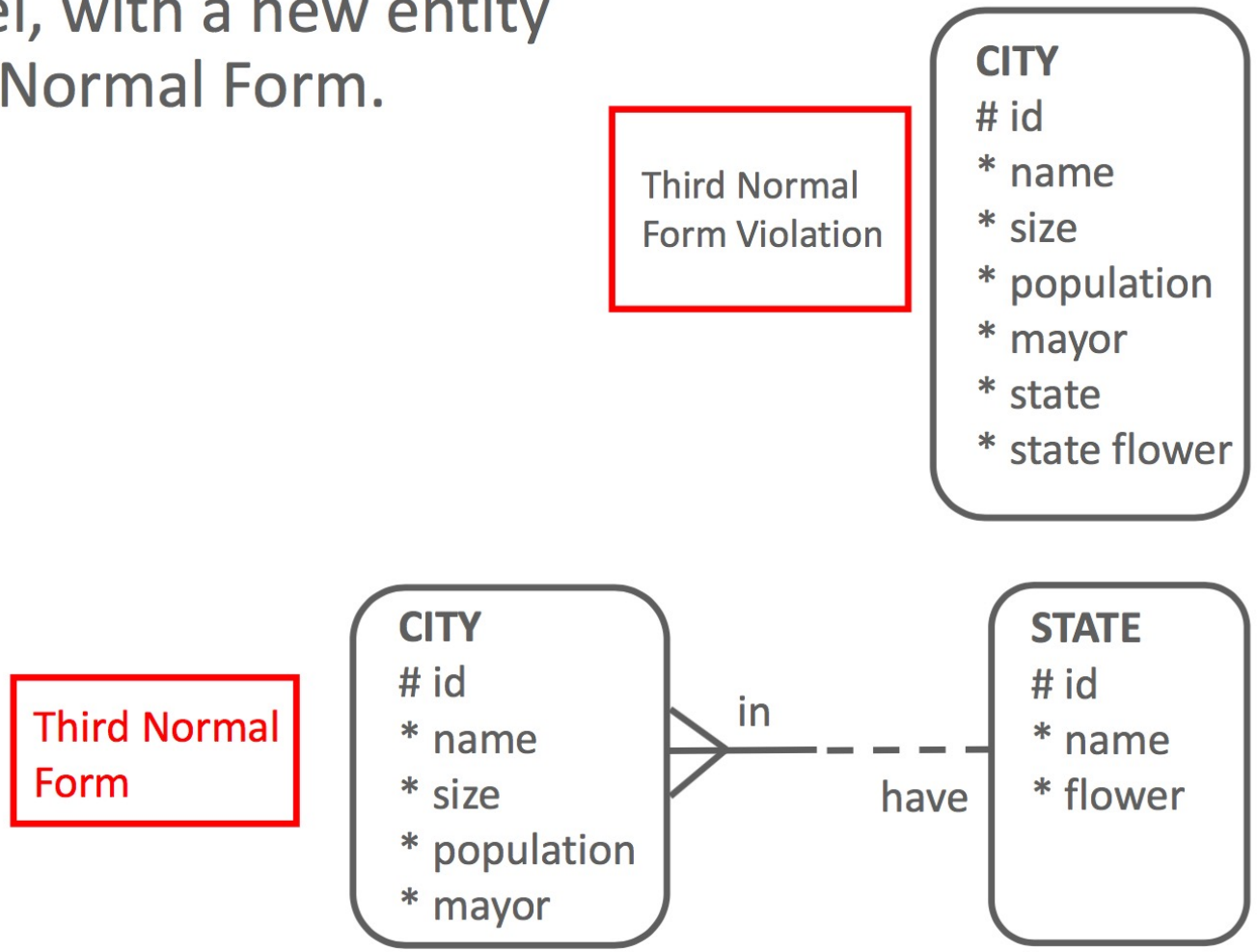
Third Normal Form Example

- Consider a system that tracks information about cities - size, population, mayor, and so on.
- The first model shows an entity that includes state information.
- Although state is an attribute of city, state flower is really an attribute of state.



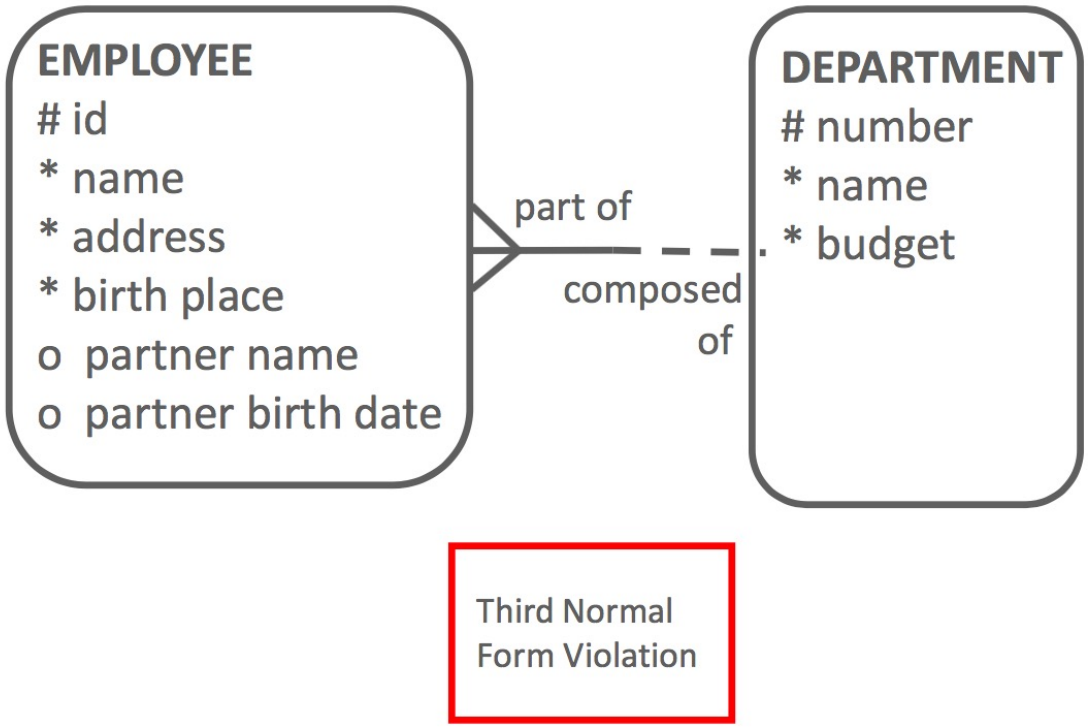
Third Normal Form

- The second model, with a new entity STATE, is in Third Normal Form.



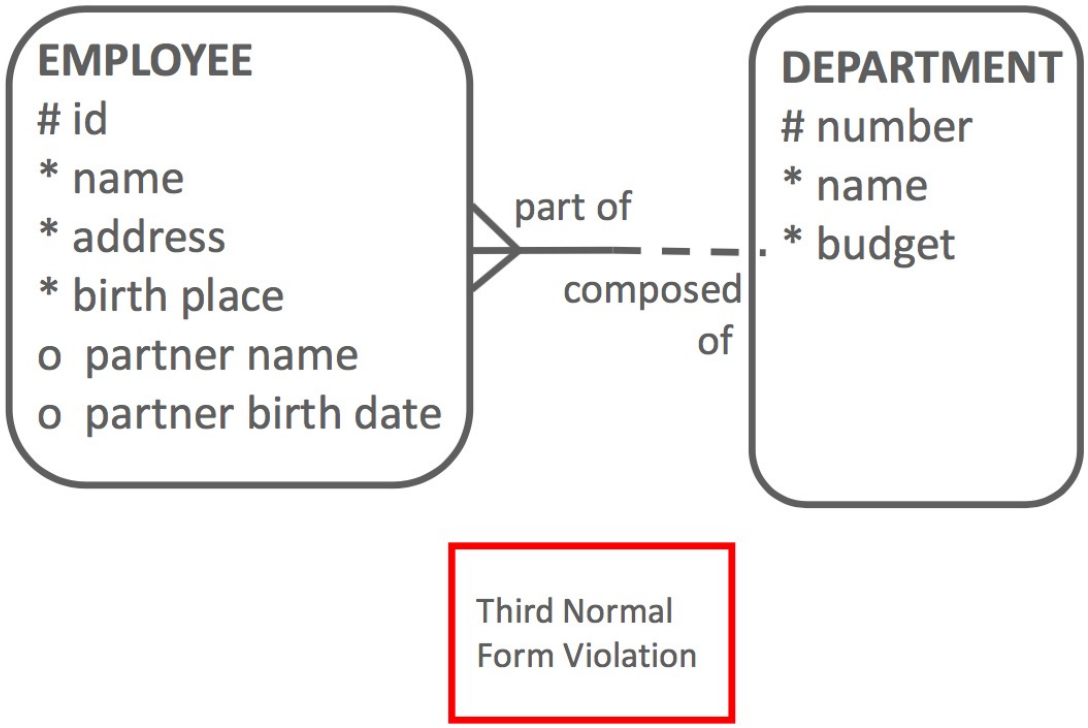
Third Normal Form Second Example

- In this example, assume the following business rule: each employee can have one partner.
- This model violates Third Normal Form because partner birth date is an attribute of partner, not of EMPLOYEE.



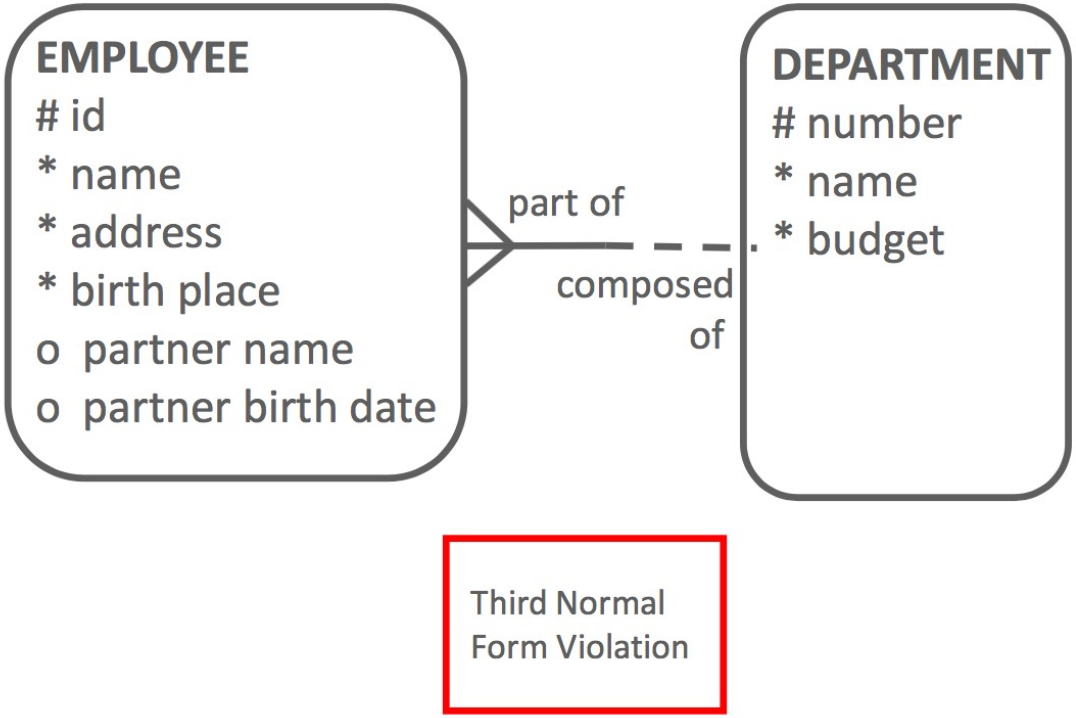
Third Normal Form Second Example

- In this example, assume the following business rule: each employee can have one partner.
- This model violates Third Normal Form because partner birth date is an attribute of partner, not of EMPLOYEE.



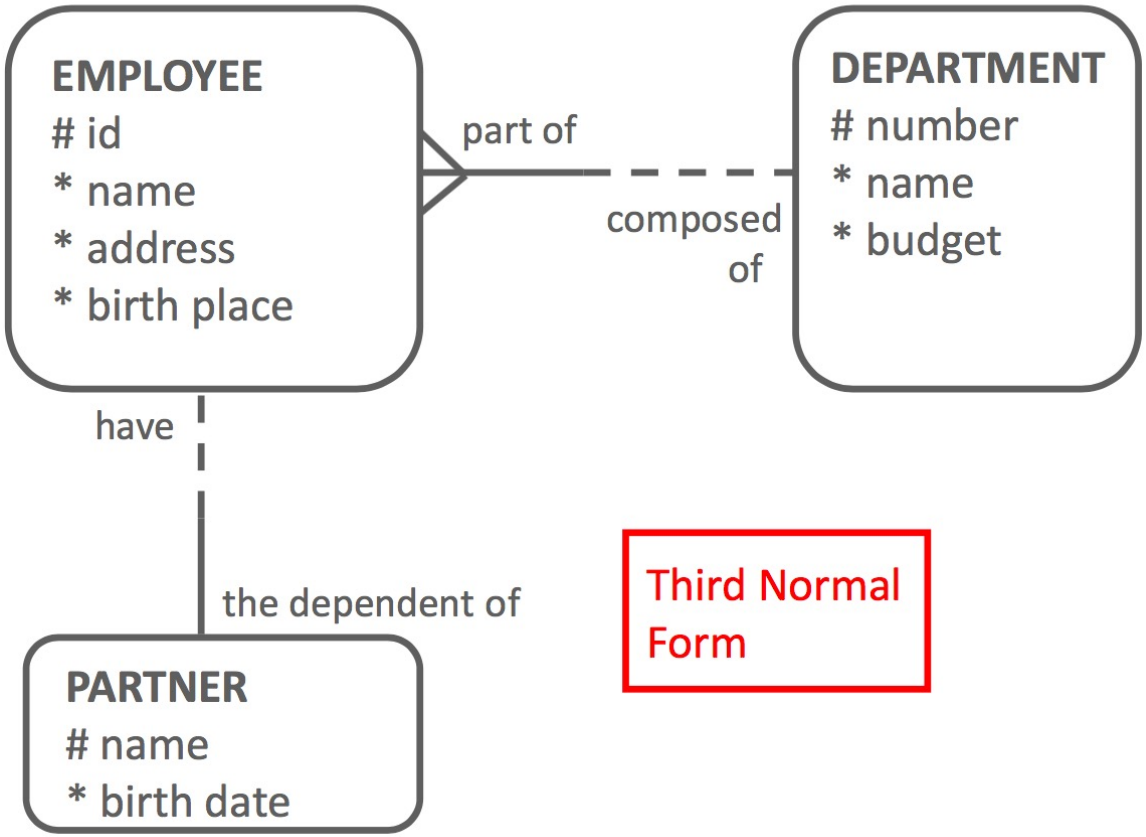
Third Normal Form Second Example

- Another way of stating Third Normal Form: non-UID attributes can't have attributes of their own.



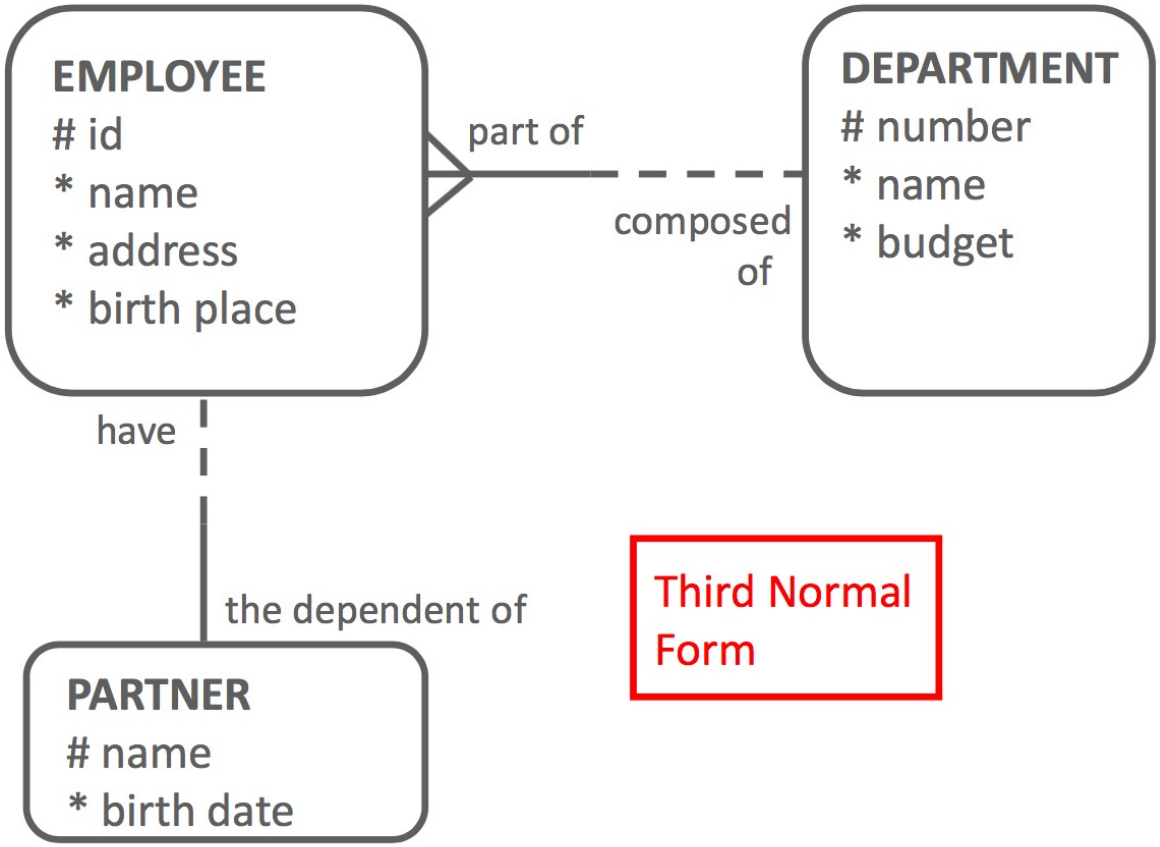
Third Normal Form Second Example

- This model supports Third Normal Form because partner birth date is an attribute of partner, not of EMPLOYEE.



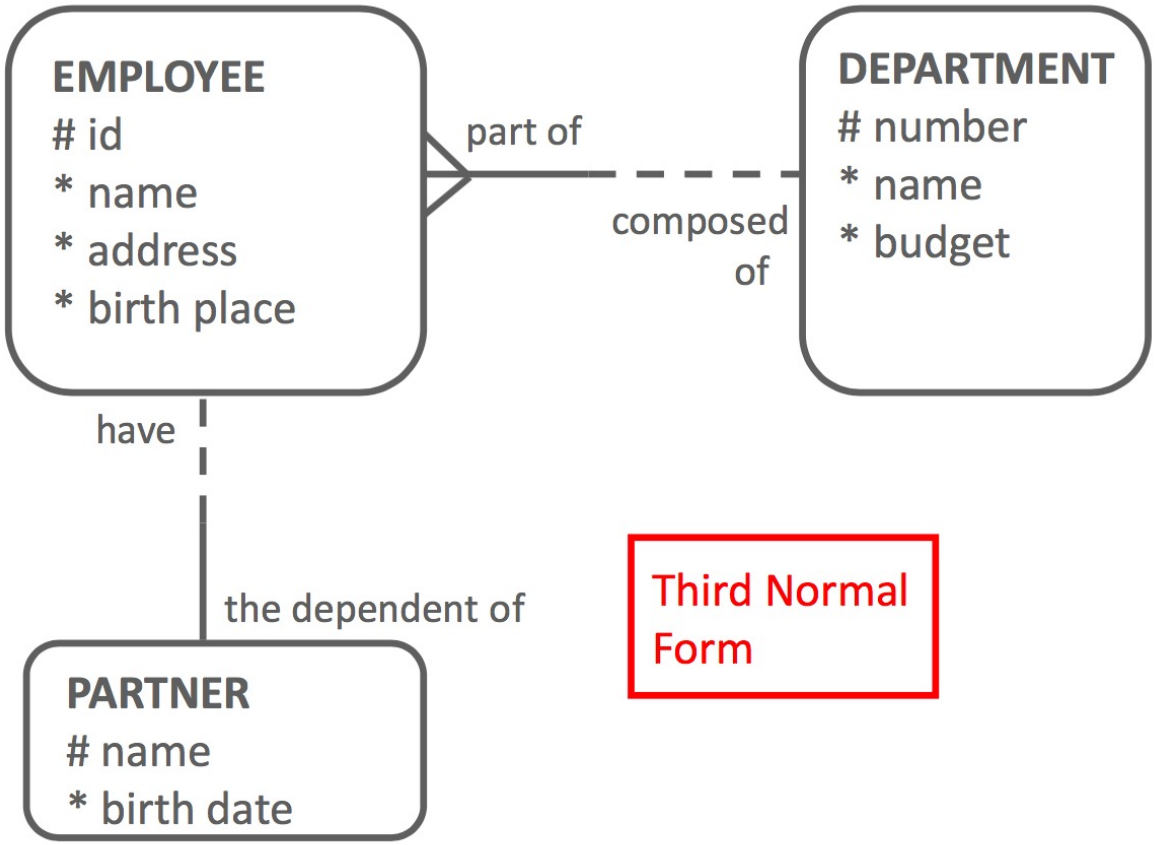
Third Normal Form Second Example

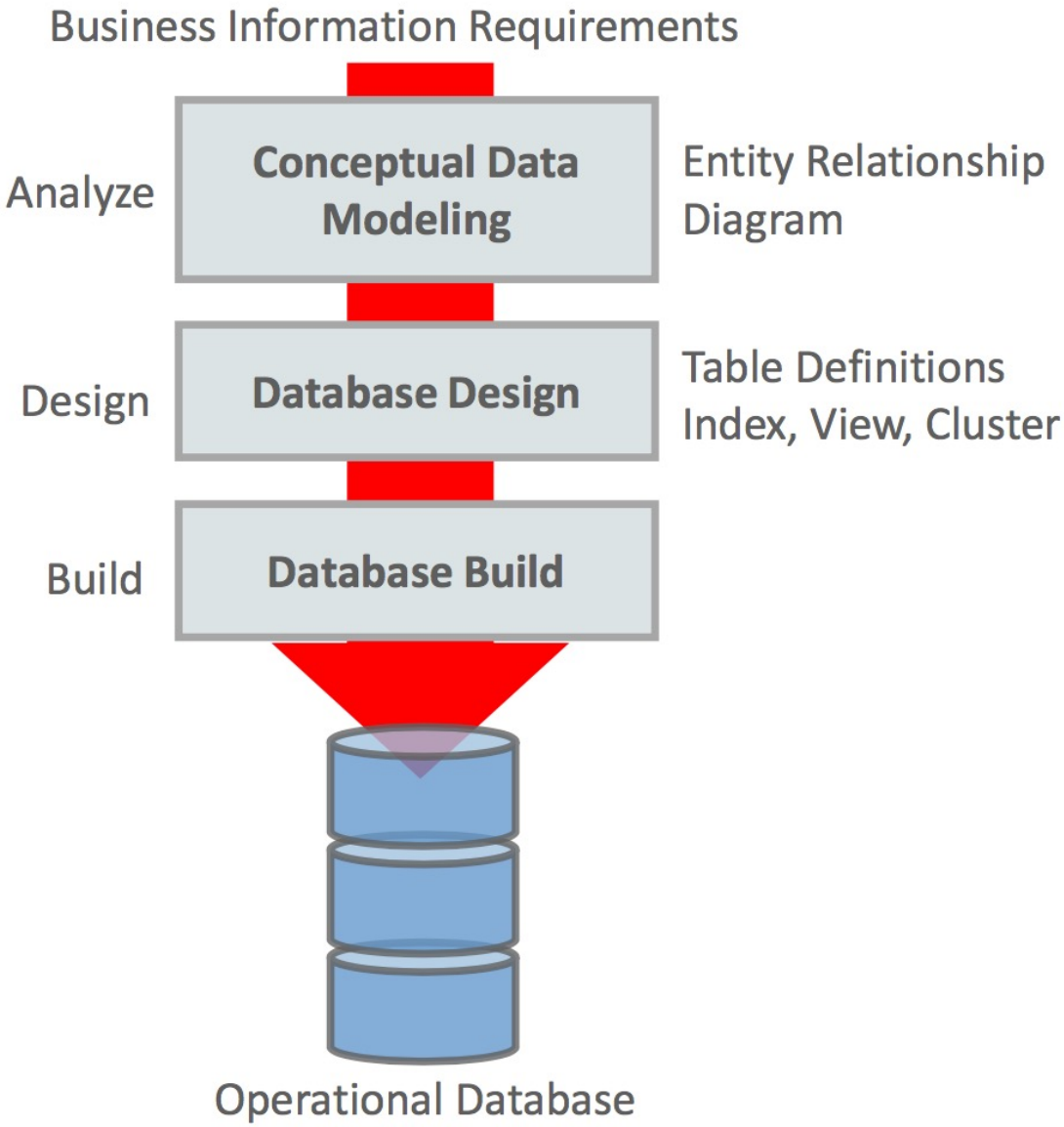
- The 1:1 relationship is optional on the EMPLOYEE end because some employees will not have partners.



Third Normal Form Second Example

- It is mandatory on the PARTNER end because information about a partner is tracked only if that person is a partner of one and only one EMPLOYEE.




Business Information Requirements
Analyze
Conceptual Data Modeling
Entity Relationship Diagram
Design
Database Design
Table Definitions
Index, View, Cluster
Build
Database Build
Operational Database

Purpose

- Conceptual data model →relational database design.
- This means that our entities, attributes, relationships, and unique identifiers will be translated into objects in a relational database.

Purpose

- Compare this to a clothing designer who is taking his design from paper and implementing it with fabric.
- The designer needs to understand how to sew the designs just like you will need to understand the structure of relational database objects.

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

Row

Column

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

Column Integrity

- A column must contain only values that are consistent with the defined data format of the column.

ACCOUNTS

BANK_NO	ACCT_NO	BALANCE	DATE_OPENED
104	75760	12,0050.00	21-OCT-1989
104	77956	100.10	
105	89570	55,775.00	15-JAN-1985
103	55890	15,001.85	10-MAR-1991
105	75760	5.00	22-SEP-2003

ACCOUNTS Table Definition

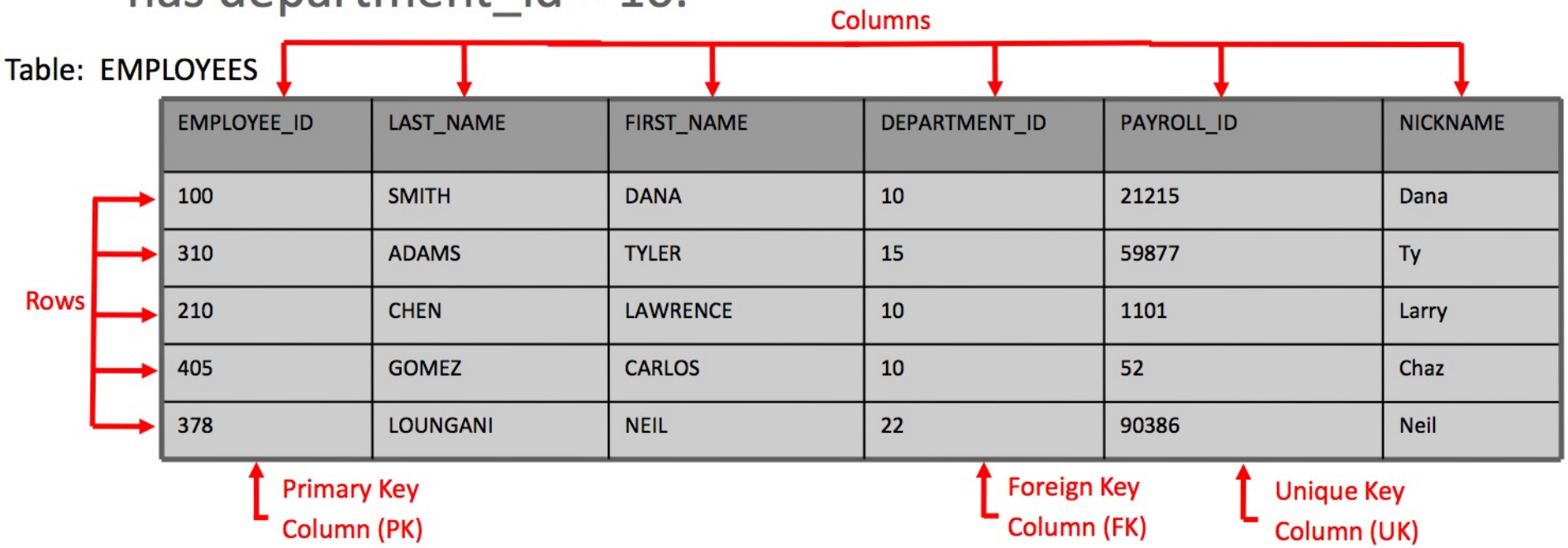
Column Name	Data Type	Optionality
BANK_NO	Number (5)	Not null
ACCT_NO	Number (8)	Not null
BALANCE	Number (12,2)	Not null
DATE_OPENED	Date	

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)

Relational Table

- We know that Dana Smith works in department 10.
- If we wanted to know more about Dana Smith's department, we would look for the row in the DEPARTMENTS table that has department_id = 10.



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

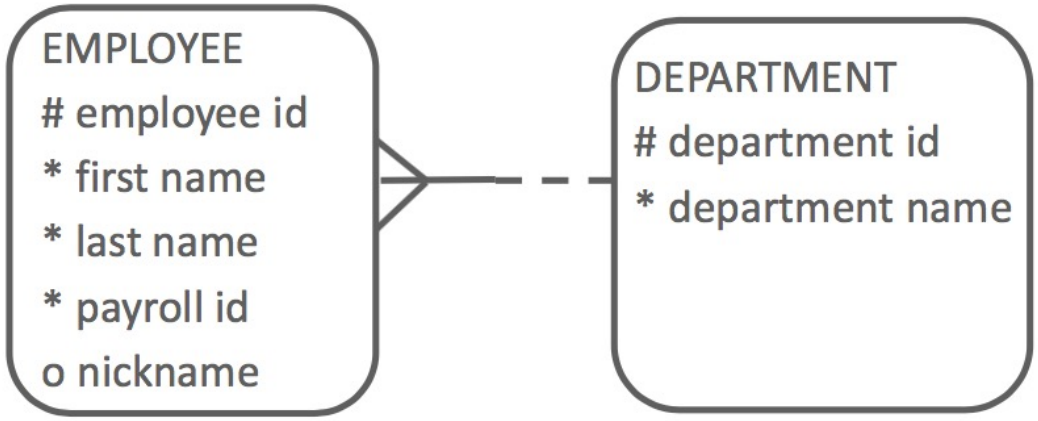
Transforming Conceptual To Physical

- 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

Changing from analysis (conceptual model) to implementation (physical model) also means changing terminology:

- An entity becomes a table.
- An instance becomes a row.
- An attribute becomes a column.
- A primary unique identifier becomes a primary key.
- A secondary unique identifier becomes a unique key.
- A relationship is transformed into a foreign-key column and a foreign key constraint.

Terminology Mapping

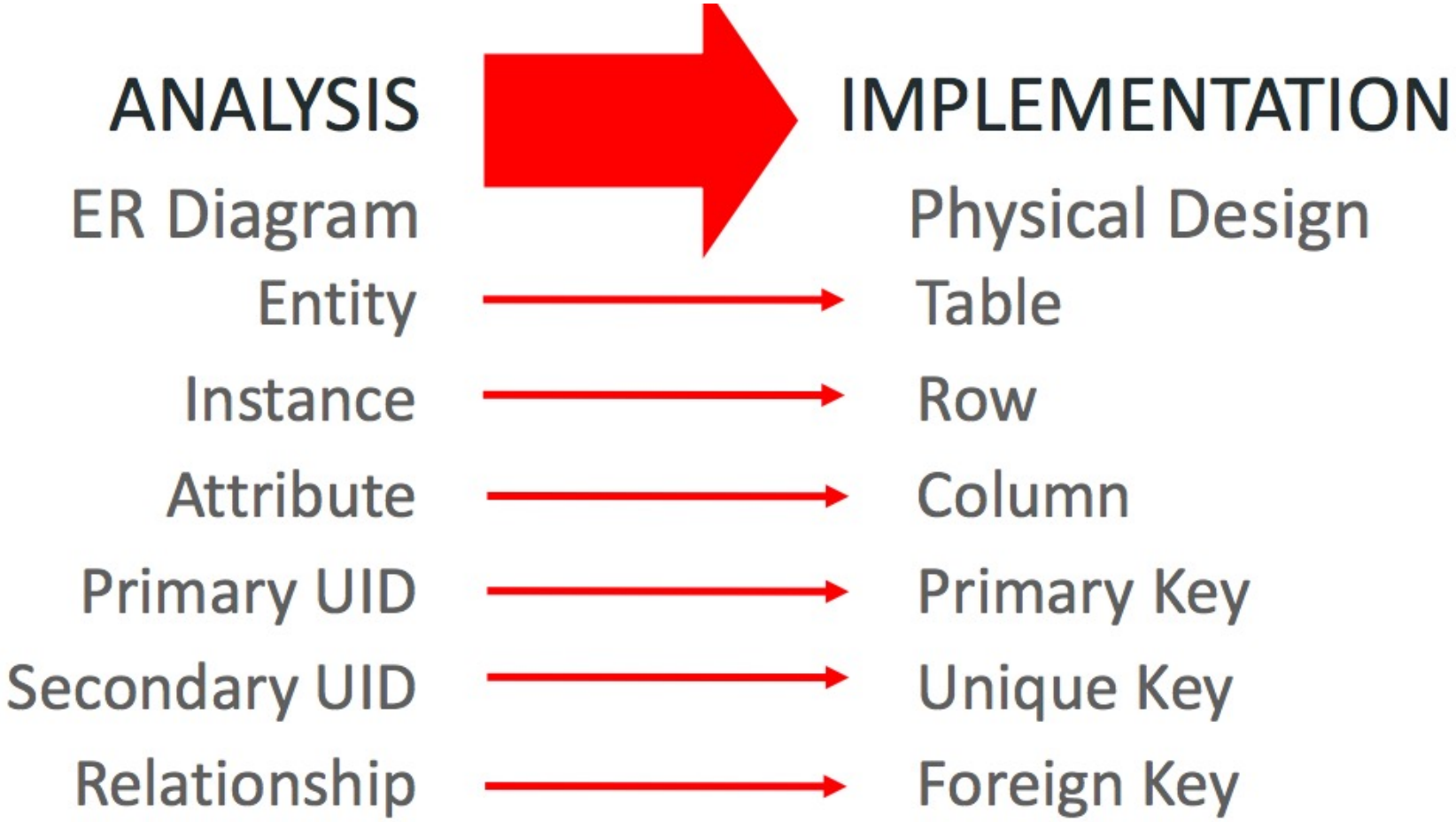


Table Diagram Notations

- The first row of the table diagram contains the table name and the short name.
- The Key Type column should contain values of “pk” for the primary key, “uk” for the unique key, and “fk” for the foreign-key column.

TABLE NAME (short name)		
Key Type (pk, uk, fk)	Optionality (“*”, “o”)	Column Name

Table Diagram Notations

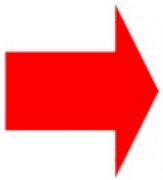
- It will be blank if the column is not a part of any key.
- The Optionality column must contain “*” if the column is mandatory and “o” if it is optional. This is similar to the entity diagram. The third column is for the column name.

TABLE NAME (short name)		
Key Type (pk, uk, fk)	Optionality (“*”, “o”)	Column Name

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



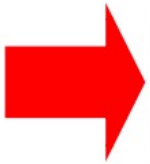
STUDENTS

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

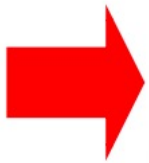


STUDENTS		
Key Type	Optionality	Column Name
pk	*	id
	*	first_name
	*	last_name
	*	str_addr
	*	city
	*	state
	*	p_code
	*	dob

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

Table Short Names

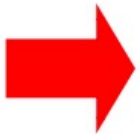
PRIVATE HOMES (PHE)		
Key Type	Optionality	Column Name
pk	*	id
	*	address
	o	comments

Table Short Names

- For entity names of one word but more than one syllable, take the:
  - First character of the first syllable
  - First character of the second syllable
  - Last character of the last syllable
- Example: EMPLOYEE gets a short name of EPE and CLIENT gets a short name of CET

Table Short Names

CLIENT
number
* first name
* last name
* phone number
o email address



CLIENTS (CET)		
Key Type	Optionality	Column Name
pk	*	client_num
	*	first_name
	*	last_name
	*	phone_num
	o	email_addr

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.

Relationship Mappings

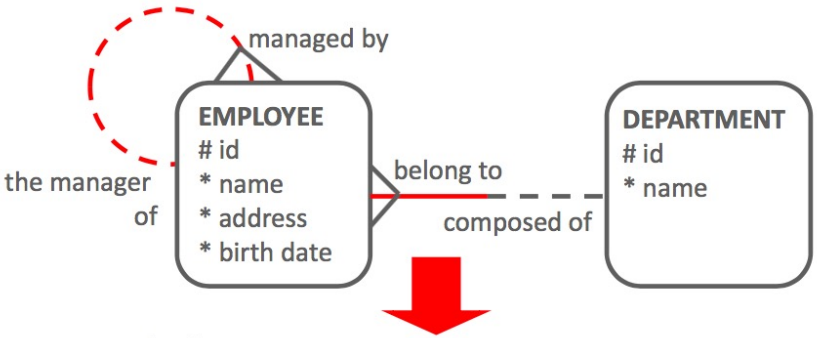
- Relationships are mapped between primary keys and foreign keys to allow one table to reference another.
- If we don’t map relationships, we just have a lot of standalone tables containing information that does not connect to anything else in the database.
- Mapping relationships between entities serves 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.



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

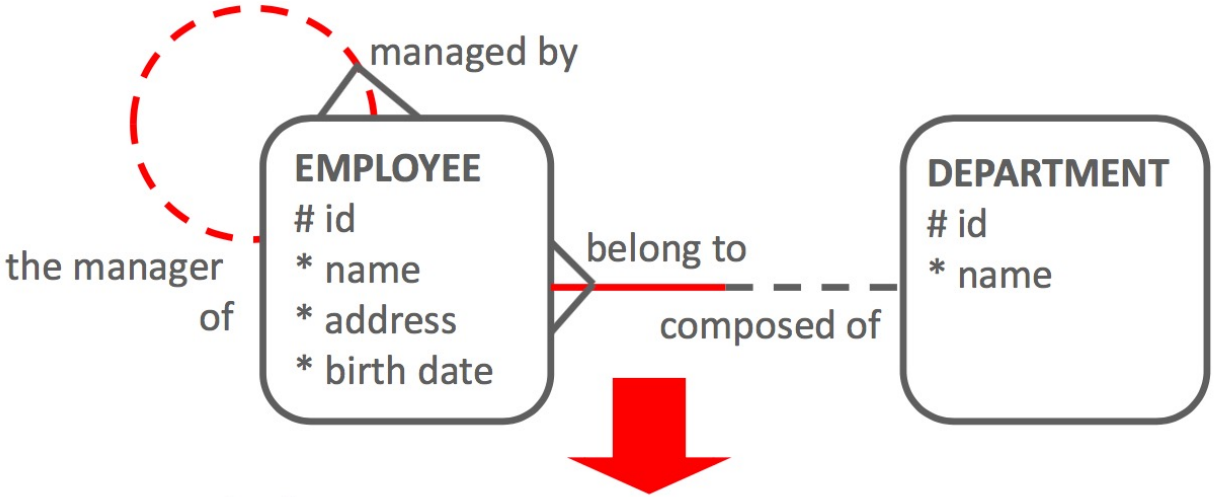
foreign key refers to

foreign key refers to

1:M Relationships

- Primary key of 1 side would be foreign key at M side
- The foreign-key column may be either mandatory or optional, depending on the needs of the business.

1:M Relationships



EMPLOYEES (EPE)

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

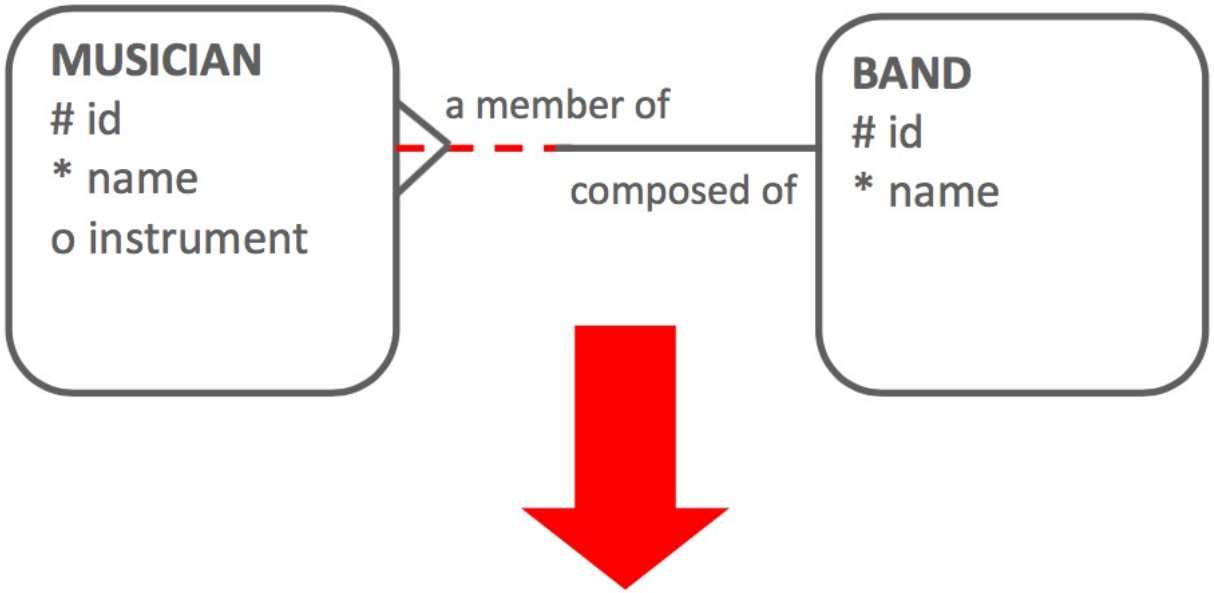
DEPARTMENTS (DPT)

Key Type	Optionality	Column Name
pk	*	id
uk	*	name

foreign key refers to

foreign key refers to

1:M Relationships



MUSICIANS (MSN)

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

foreign key refers to

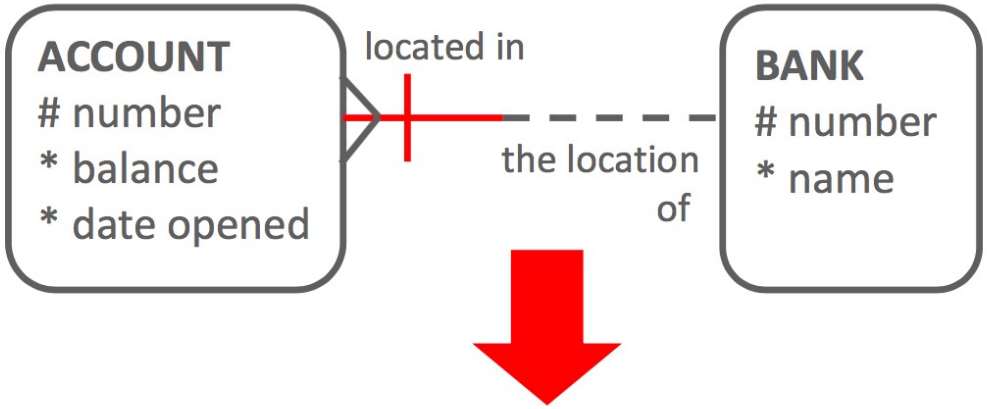
BANDS (BAD)

Key type	Optionality	Column name
pk	*	id
	*	name

Mapping of 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

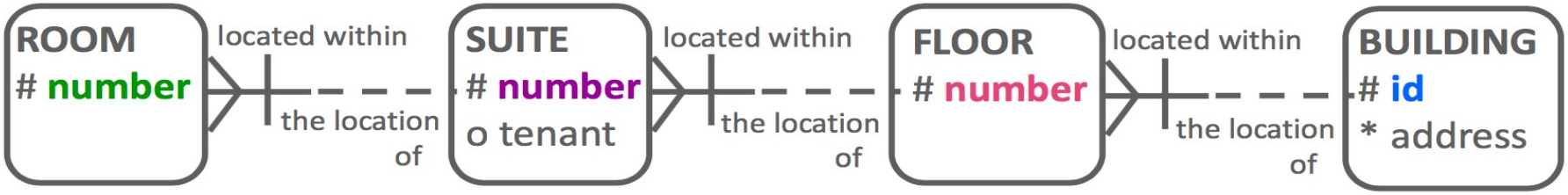
Cascade Barred Relationships

- 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.

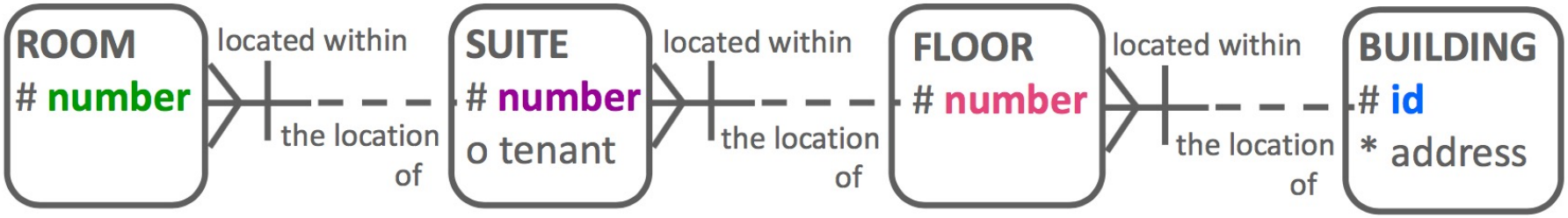
Cascade Barred Relationships

- 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



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)

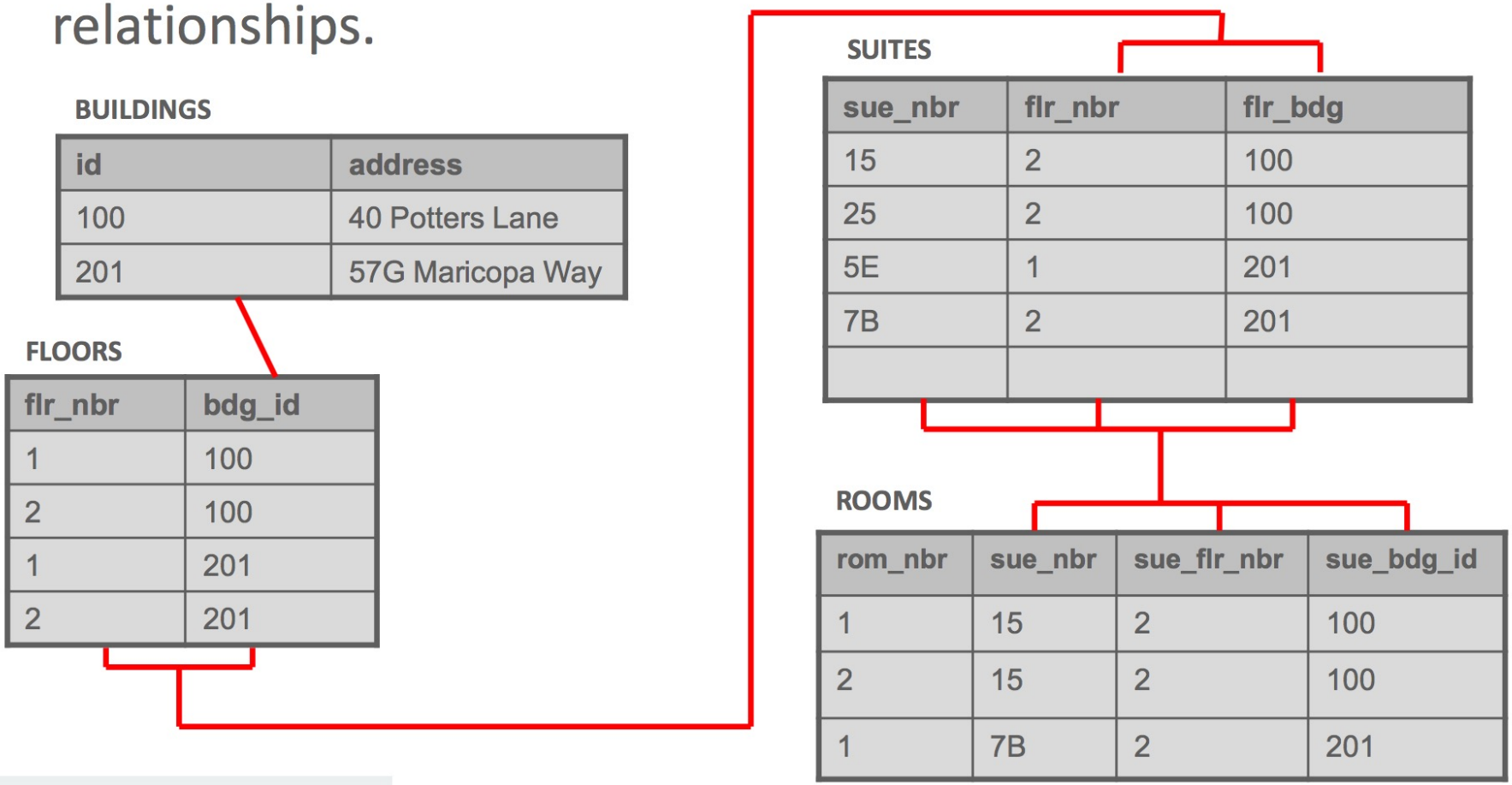
pk	*	flr_nbr
pk, fk	*	bdg_id

BUILDINGS (BDG)

pk	*	id
	*	address

Cascade Barred Relationships

- 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

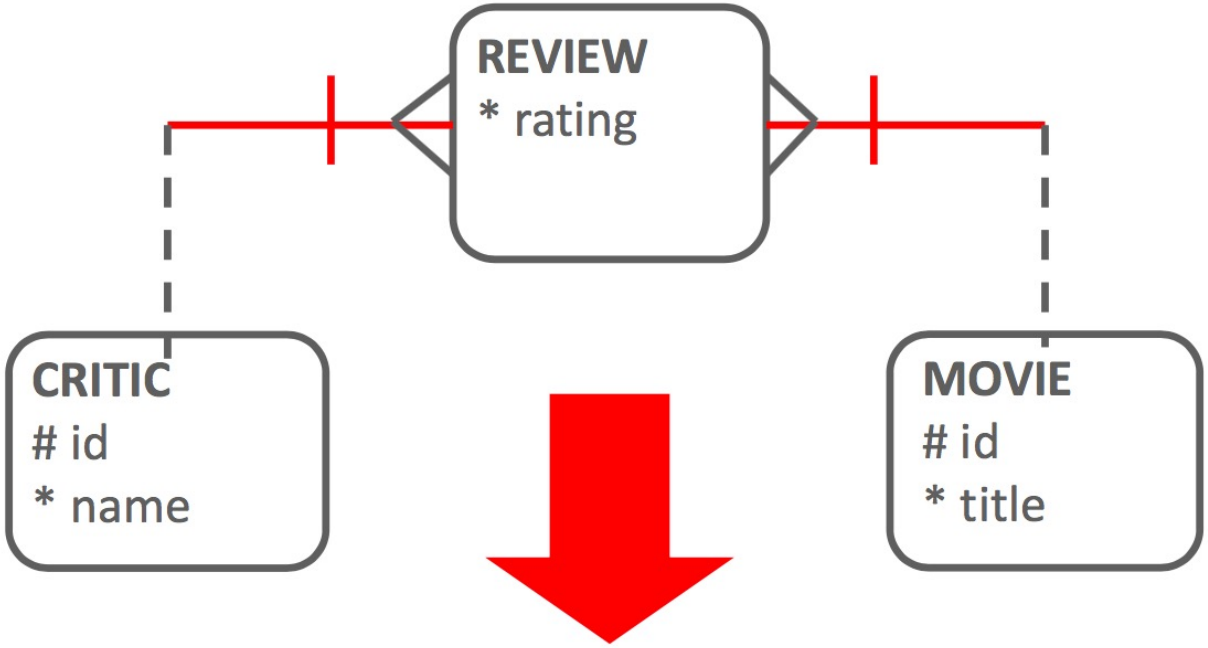
ROOMS

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

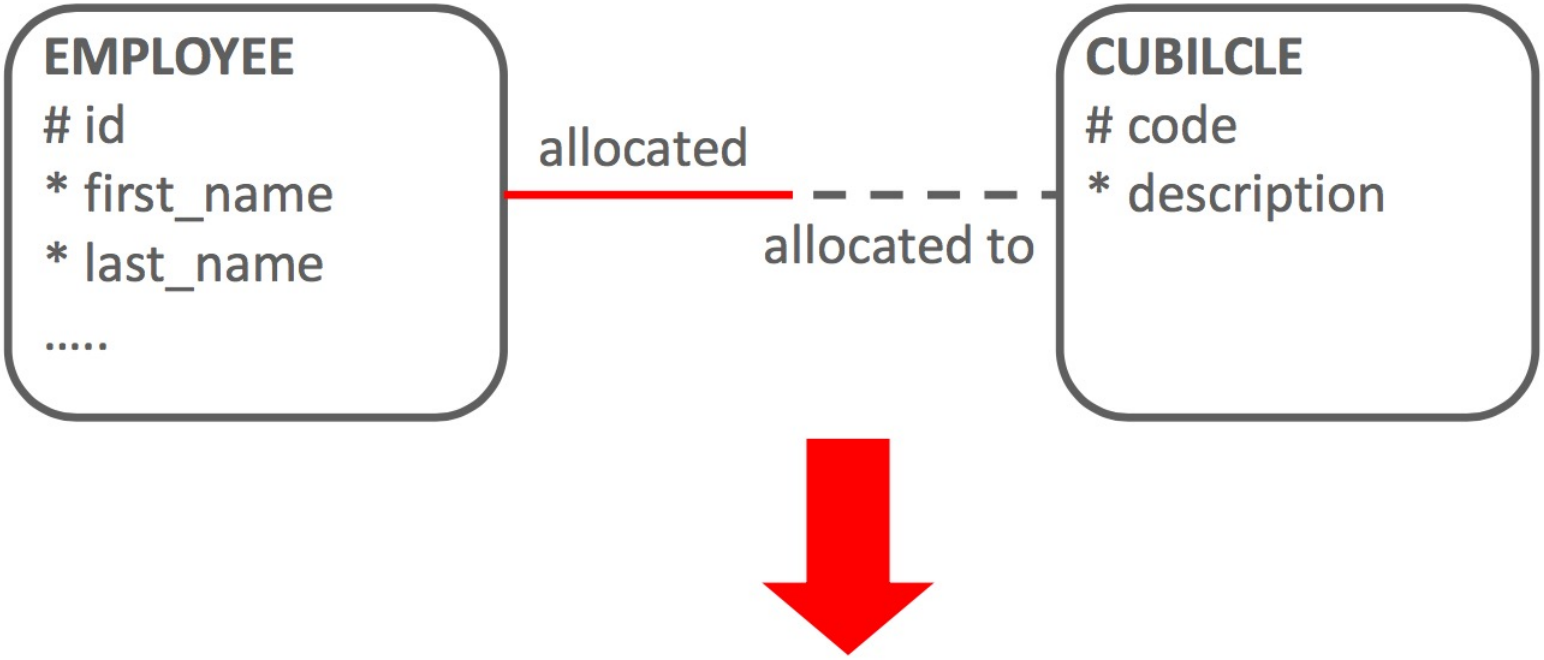
REVIEWS (RVW)

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

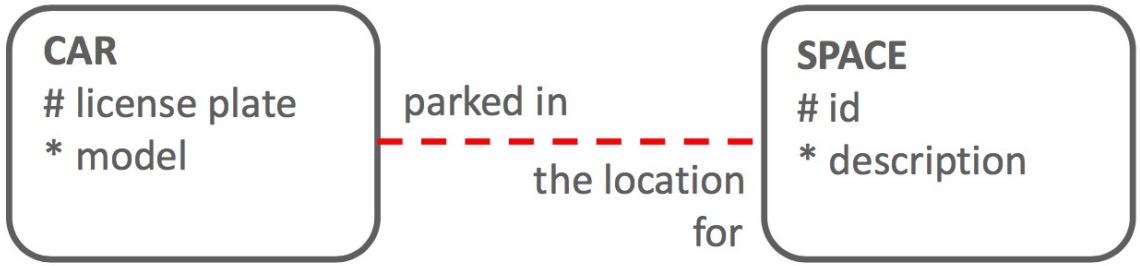
Optional One-to-One

- 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.

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