



THE UNIVERSITY OF  
MELBOURNE

# Logical & Physical Modelling


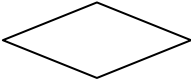

Database Systems & Information Modelling  
INFO90002.

Week 2 – Data Modelling  
Dr Tanya Linden  
Dr Renata Borovica-Gajic  
David Eccles



# So far: Intro to Modelling

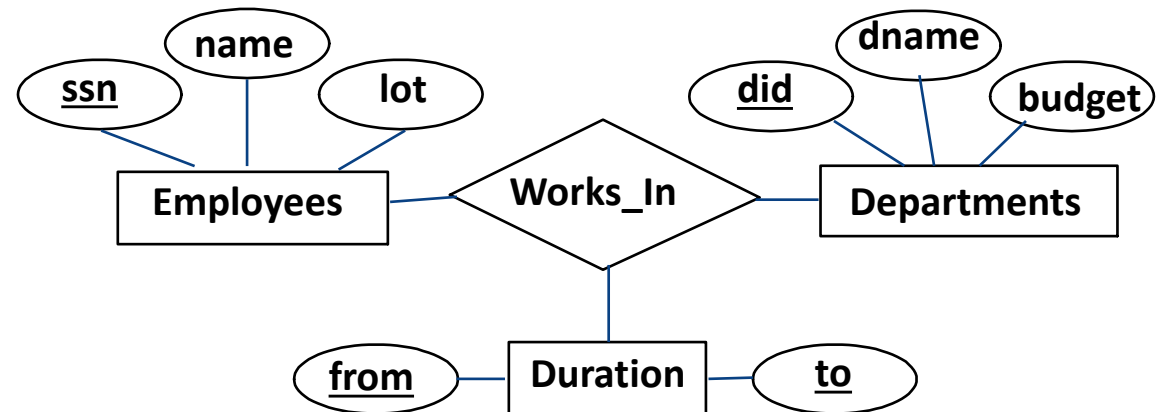
## Basic ER modeling concepts

- Entities, 
- Relationships 
- Attributes (Key Attributes) 

## Constraints

- Key Constraints M:M, 1:M, 1:1
- Participation Constraints
  - Total
  - Partial

## Conceptual Design





# Concepts

Relational Model

Keys and Integrity Constraints

Translating ER to Logical and Physical Model

*Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems*

# Relational Data Model

**Data Model** allows us to translate real world things into structures that a computer can store

Many models: Relational, ER, O-O, etc.

## Relational Model:

- Rows (or Tuples) and Columns (Attributes/fields)
- Primary Keys and Foreign Keys to link Relations

### Customer

Customer ID	Surname	First name	Address	Postcode	Credit card
C2045	Smith	Fred	1 John St, Hawthorn	3122	1234234534564567
C2048	Nguyen	Vincent	2/7 Oak Ave, Altona	3018	4554123423457899
C2146	Davis	Liz	32 Lyle St, Toorak	3142	4564564578970022

### Order

Order No	Customer ID	Date	Time	Deliv
3224	C2045	1/09/2021	17:35	The b
3228	C2045	3/09/2021	18:42	The b
3236	C2048	3/09/2021	21:05	
3248	C2045	4/09/2021	15:20	The b





# Relational Database: Definitions

**Relational database**: a set of *relations*.

**Relation**: made up of 2 parts:

- **Schema** : specifies name of relation, plus name and type of each column (attribute).

Example:

Pizza(*plD*: string, *pizzaName*: string, *price*: real)

- **Instance** : a **table**, with rows and columns.

Number of rows = *cardinality*

Number of columns/fields = *degree (or arity)*

You can think of a relation as a *set of rows or tuples*.

- all rows are *distinct*
- *no order* among rows

# Example Instance of Pizza Relation

## Pizza

Code	Pizza name	Price
P1	Mario's Supreme Pizza	6.95
P2	Vegetarian Pizza	6.95
P3	Hawaiian Pizza	6.95
P4	Hot 'n' spicy Pizza	6.95
B1	Garlic Bread	4.95
B2	Herb Bread	4.95
D1	2 Litre Cola	2.50
D2	2 Litre Lemonade	2.50

Cardinality = 8, degree (arity) = 3, all rows distinct

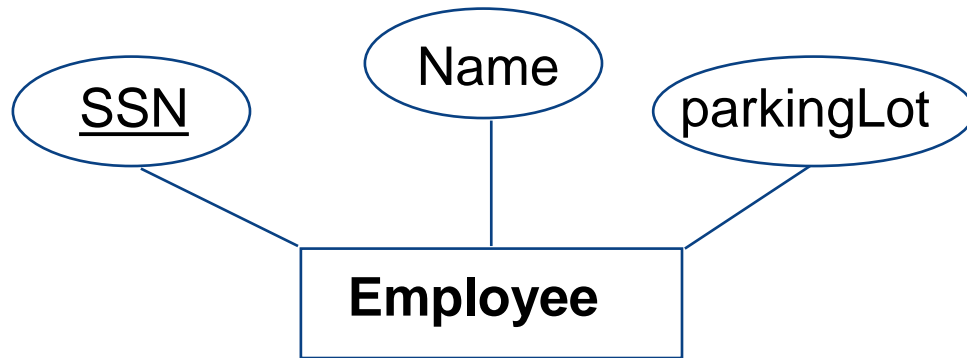
# Logical Design: ER to Relational Model

In logical design **entity** set becomes a **relation**. Attributes become attributes of the relation.

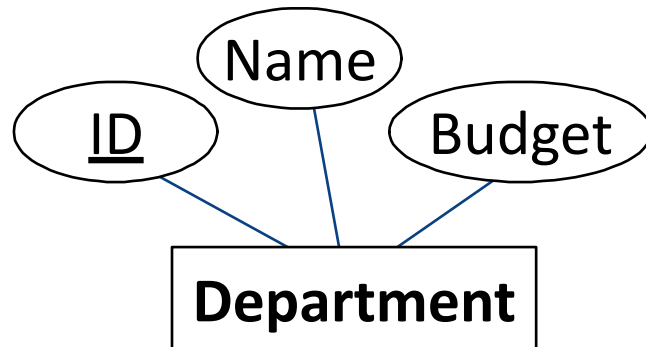
Conceptual Design:



Logical Design:



Employees = (SSN, name, parkingLot)

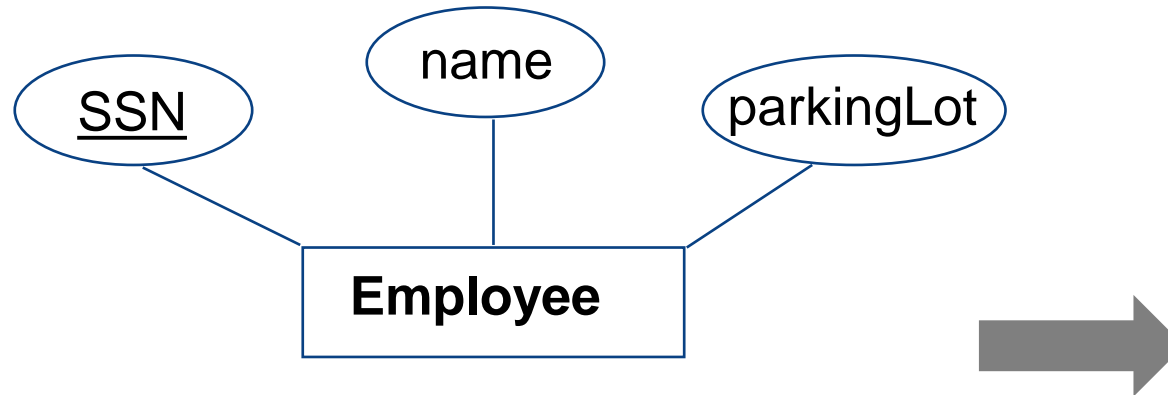


Department = (ID, Name, Budget)

# ER to Logical to Physical

In physical design we specify **data types**

## 1. Conceptual Design:



## 2. Logical Design:

Employee (SSN, name, parkingLot)

## 3. Physical Design:

Employee  
(SSN CHAR(11),  
name CHAR(20),  
parkingLot INTEGER)



# The Entire Cycle

**1. Conceptual Design**



**2. Logical Design**



**3. Physical Design**



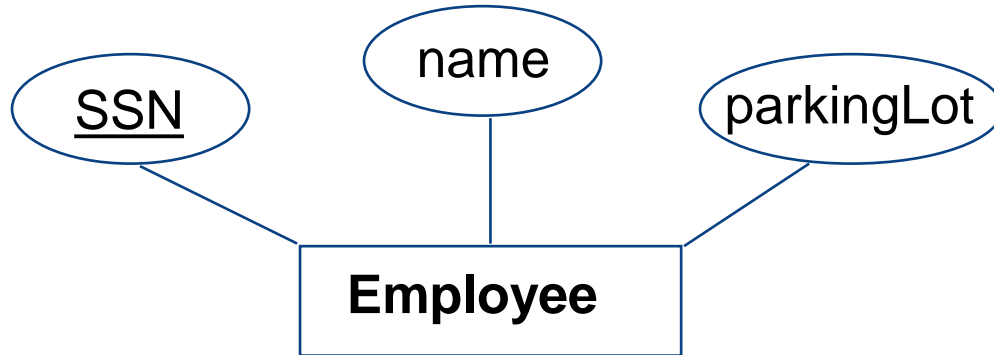
**4. Implementation**



**5. Create Instance**

# The Entire Cycle

## 1. Conceptual Design:



## 2. Logical Design:

Employee (SSN, name, lot)

## 3. Physical Design:

Employee  
(ssn CHAR(11),  
name CHAR(20),  
parkingLot INTEGER)

## 4. Implementation:

```
CREATE TABLE Employee  
  (ssn CHAR(11),  
   name CHAR(20),  
   parkingLot INTEGER,  
   PRIMARY KEY (ssn))
```

## 5. Instance:

EMPLOYEES

<u>ssn</u>	name	parkingLot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20



# Relational Model

Relational Model

**Keys and Integrity Constraints**

Translating ER to Logical and Physical Model

*Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems*

# Keys

Keys are a way to associate tuples in different relations

Keys are one form of **integrity constraint (IC)**

**Example:** *Only customers can place orders.*

*Each Foreign Key value in the Order table references a Primary Key value in the Customer table*

## Customer

Customer ID	Surname	First name	Address	Postcode	Credit card
C2045	Smith	Fred	1 John St, Hawthorn	3122	1234234534564567
C2048	Nguyen	Vincent	2/7 Oak Ave, Altona	3018	4554123423457899
C2146	Davis	Liz	32 Lyle St, Toorak	3142	4564564578970022

## Order

Order No	Customer ID	Date	Time	Delivery
3224	C2045	1/09/2021	17:35	The be
3228	C2045	3/09/2021	18:42	The be
3236	C2048	3/09/2021	21:05	
3248	C2045	4/09/2021	15:20	The be

**FOREIGN Key**

**PRIMARY Key**

# Primary Keys

A set of fields is a **superkey** if no two distinct tuples can have same values in all key fields

A set of fields is a **key** for a relation if it is a superkey and no subset of the fields is a superkey (minimal subset)

Out of all keys *one* is chosen to be the **primary key** of the relation. Other keys are called **candidate** keys.

Each *relation* has a *primary key*.

## Your turn:

1. Is *sid* a key for Student?
2. What about *name*?
3. Is the set  $\{sid, gpa\}$  a superkey? Is the set  $\{sid, gpa\}$  a key?
4. Find a primary key from this set  $\{sid, login\}$

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@cs	18	3.2
53650	Smith	smith@math	19	3.8

# Selecting the Primary Key

Superkeys



Keys



Candidate Keys



Primary Key

- Superkey – a set of fields that contains the key
- Keys are columns that in combination or alone can uniquely identify the tuple (row)
- Candidate keys are all the possible key combinations that could be the Primary Key
- Of all candidate keys the database designer *identifies* the primary key. The primary key is the *fewest number of columns* that can uniquely identify a key
- N.B.\* Not all relations will have a key. In cases when there is no key, the database designer will add a *surrogate key*

\*N.B. means Nota Bena, latin for "Note well" or in simple terms "This information is important!")

# Surrogate Keys

A surrogate key is

- A key that has no real world / business meaning
- Is usually numeric
- Is often a sequential number supplied by the RDBMS (e.g. the AutoNumber option)

Many databases around the world have been created with all their tables using surrogate keys.

When modelling business requirements

- You **speak** to **clients**
- You use terms **applicable** to **their** business
- You **do not invent** terms / fields that **do not match** their business
- You use **Natural Keys**



# Surrogate vs Natural Keys

## Example:

If a small college teaches 20 subjects, that business **may not** have/use subject codes. When identifying a subject, they simply use **natural keys** such as Subject Name.

While Subject Name may seem obviously **inadequate** for a large database, it may be sufficient for Modelling requirements.

**Forcing** a term such as Subject Code into the conversation may **confuse** clients.



# Modelling with Natural Keys

**Modelling typically uses Natural Keys**

**Modelling** is sometimes the **end of the process**.

- The **ER Model** that is produced may be the **final product**
  - There is **no database**
  - Both the **client** and the **modeler learn** about the business and its requirements via the modelling process.

If a Database is required:

- Database **implementers** can choose to **add surrogate keys**.
  - When adding a surrogate key – **you do not lose any data**.
  - The **natural key** data is **not removed**. It's just not the primary key.

# Primary and Candidate Keys in SQL

There are possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*. Keys must be chosen carefully.

## Example:

*For a given student and course, there is a single grade.*

```
CREATE TABLE Enrolled  
(sid CHAR(20)  
cid CHAR(20),  
grade CHAR(2),  
PRIMARY KEY (sid,cid))
```

VS.

```
CREATE TABLE Enrolled  
(sid CHAR(20)  
cid CHAR(20),  
grade CHAR(2),  
PRIMARY KEY (sid),  
UNIQUE (cid, grade))
```

*“Students can take only one course,  
and no two students in a course  
receive the same grade.”*



# Foreign Keys and Referential Integrity

***Foreign key*** : A set of fields in one relation that is used to ‘refer’ to a tuple in another relation. The foreign key must correspond to the primary key of the other relation.

If all foreign key constraints are enforced in a DBMS, we say a ***referential integrity*** is achieved.

# Foreign Keys in SQL

**Example:** *Only customers listed in the Customer relation should be allowed to place Orders.*

*customerID* is a foreign key referring to Customer table

```
CREATE TABLE Order
(orderNo CHAR(5),
customerID CHAR(8),
```

...,

**PRIMARY KEY** (orderNo, customerID),

**FOREIGN KEY** (customerID) **REFERENCES** Customer (customerID))

## Customer

Customer ID	Surname	First name	Address	Postcode	Credit card
C2045	Smith	Fred	1 John St, Hawthorn	3122	1234234534564567
C2048	Nguyen	Vincent	2/7 Oak Ave, Altona	3018	4554123423457899
C2146	Davis	Liz	32 Lyle St, Toorak	3142	4564564578970022

## Order

Order No	Customer ID	Date	Time	Delivery
3224	C2045	1/09/2021	17:35	The be
3228	C2045	3/09/2021	18:42	The be
3236	C2048	3/09/2021	21:05	
3248	C2045	4/09/2021	15:20	The be



# Enforcing Referential Integrity

Consider *Customer* and *Order*:

- *customerNo* in *Order* is a foreign key that references *Customer*.

What should be done if an *Order* tuple with a non-existent *customerNo* is inserted? (*Reject it!*)

What should be done if a *Customer* tuple is deleted?

- Also delete all *Order* tuples that refer to it?
- Disallow deletion of a *Customer* tuple that is referred to?
- Set *customerNo* in *Order* tuples that refer to it to a *default customerNo*?
- (In SQL, also: Set *customerNo* in *Order* tuples that refer to it to a special value *null*, denoting '*unknown*' or '*inapplicable*')

Note: Similar issues arise if primary key of *Customer* tuple is updated.



# Integrity Constraints (ICs)

**IC:** condition that must be true for *any* instance of the database; e.g., *domain constraints*.

- ICs are specified when schema is defined.
- ICs are checked when relations are modified.

A ***legal*** instance of a relation is one that satisfies all specified ICs.

- DBMS should not allow illegal instances.

This is also known as **Schema on Write**

- The schema table structure is known in advance of the data to be inserted into it



# Relational Model

Relational Model

Keys & Integrity Constraints

**Translating ER to Logical and Physical Model**

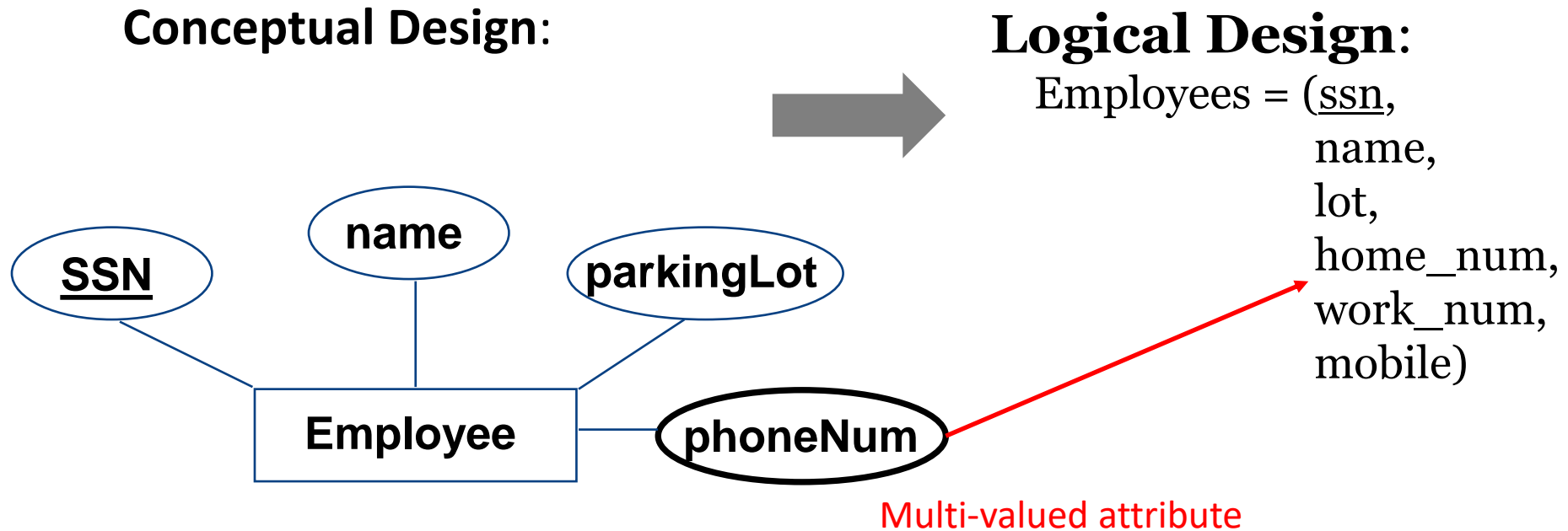
*Readings: Chapter 3, Ramakrishnan & Gehrke, Database Systems*

# Multi-valued attributes in logical design

Multi-valued attributes need to be unpacked (flattened) when converting to logical design.

## Example:

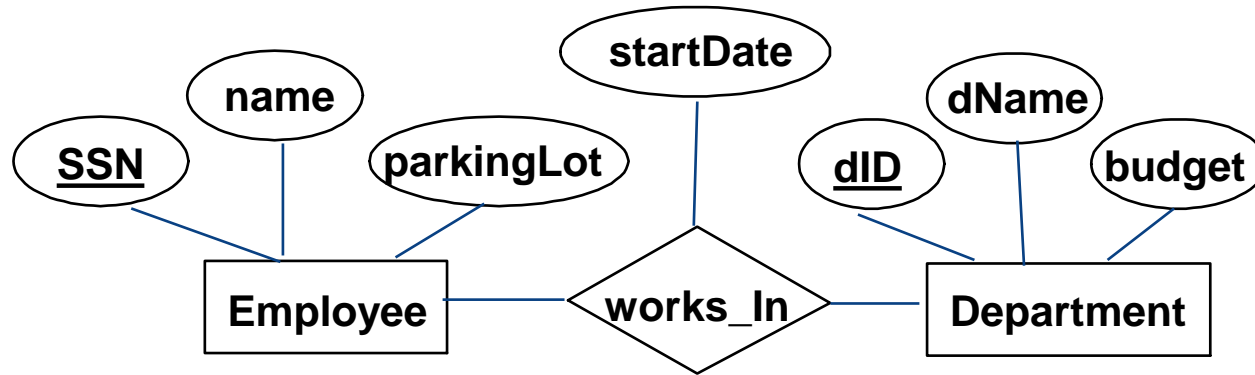
*For employees we need to capture home phone number, work phone number and mobile.*





# ER to Logical Design

## Conceptual Design:



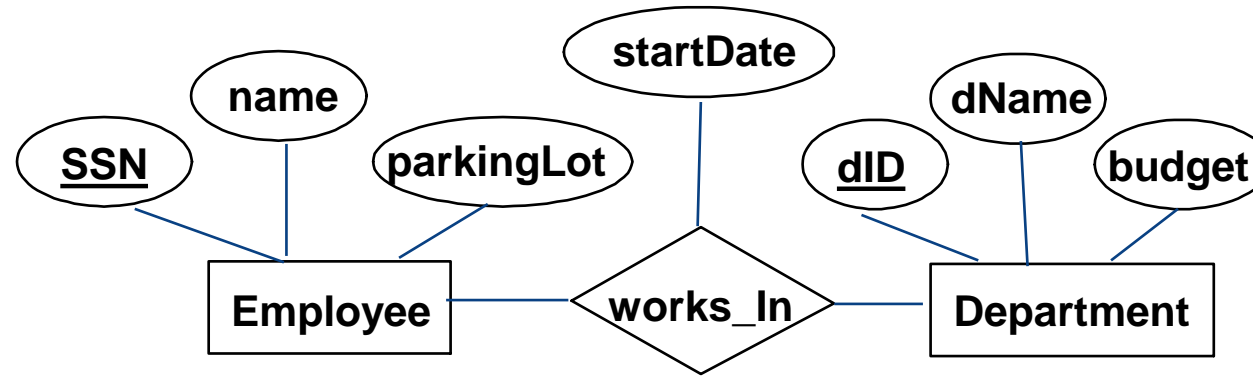
## Logical Design:

In translating a **many-to-many** relationship set to a relation, attributes of a *new* relation must include:

1. Keys for each participating entity set (as foreign keys). This set of attributes forms a ***superkey*** of the relation.
2. All descriptive attributes.

# ER to Logical Design

Conceptual Design:



Logical Design:

Employee = (ssn,  
name,  
parkingLot)

Department = (dID,  
dName,  
budget)

works\_In = (ssn,  
*dID*,  
startDate)

Keys from connecting  
entities become PFK

Note:  
Underline = PK,  
Italic = FK,  
Underline + Italic = PFK

# Logical to Physical Design

## Logical Design:

Employee = (ssn, name, parkingLot)

Department = (dID, dName, budget)

Works\_In = (ssn, *dID*, startDate)

Note:

Underline = PK,

Italic = FK,

Underline + Italic = PFK

## Physical Design:

Employee  
(ssn CHAR(11),  
name CHAR(20),  
parkingLot INTEGER)

Department  
(dID INTEGER,  
dName CHAR(20),  
budget FLOAT)

Works\_In(  
ssn CHAR(11),  
*dID* INTEGER,  
startDate DATE)

# Implementation (CREATE TABLE)

## Logical Design:

Employee = (ssn, name, parkingLot)

Department = (dID, dName, budget)

Works\_In = (ssn, *dID*, startDate)

Note:

Underline = PK,

Italic = FK,

Underline + Italic = PFK

## Implementation:

```
CREATE TABLE Employee
(ssn CHAR(11),
name CHAR(20),
parkingLot INTEGER,
PRIMARY KEY (ssn))
```

```
CREATE TABLE Department
(dID INTEGER,
dName CHAR(20),
budget FLOAT,
PRIMARY KEY (dID))
```

```
CREATE TABLE Works_In(
ssn CHAR(11),
dID INTEGER,
startDate DATE,
PRIMARY KEY (ssn, dID),
FOREIGN KEY (ssn) REFERENCES Employee(ssn),
FOREIGN KEY (dID) REFERENCES Department(dID))
```

# Sample Instances

Employee

<u>ssn</u>	name	parkingLot
0983763423	John	10
9384392483	Jane	10
3743923483	Jill	20

Department

<u>dID</u>	dName	budget
101	Sales	100K
105	Purchasing	200K
108	IT	1000K

Works\_In

<u>ssn</u>	<u>dID</u>	startDate
0983763423	101	1 Jan 2003
0983763423	108	2 Jan 2003
9384392483	108	1 Jun 2002

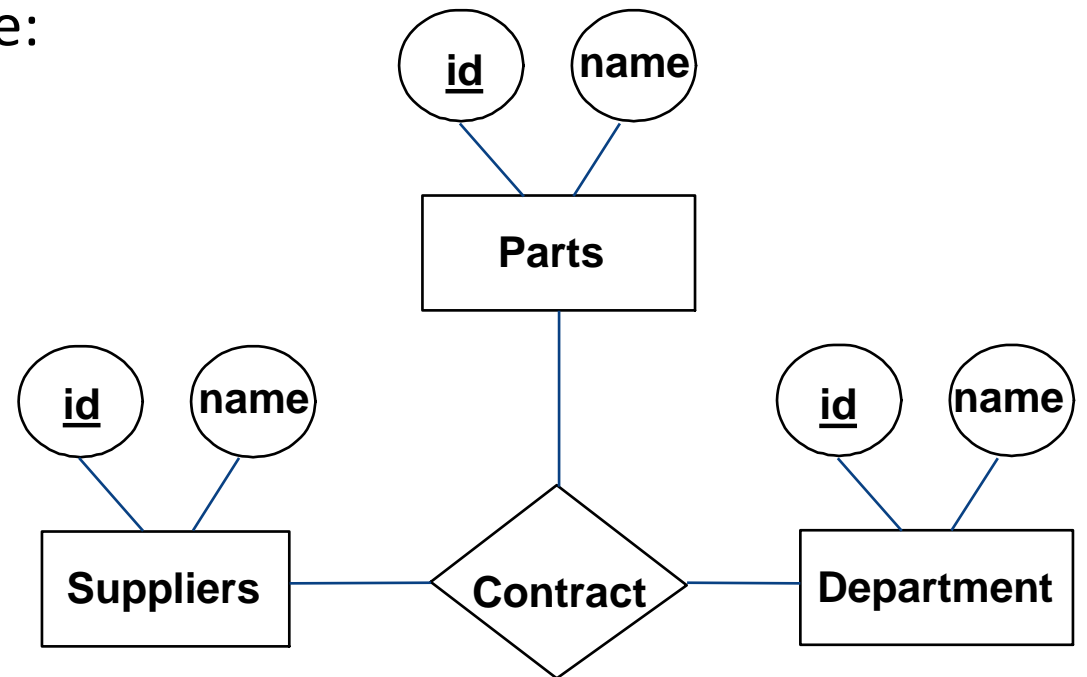
## ER to Logical Design Example 2

In translating a **many-to-many** relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys). This set of attributes forms a **superkey** for the relation.
- All descriptive attributes.

### Logical Design:

Contract (  
    supplier id,  
    part id,  
    department id)



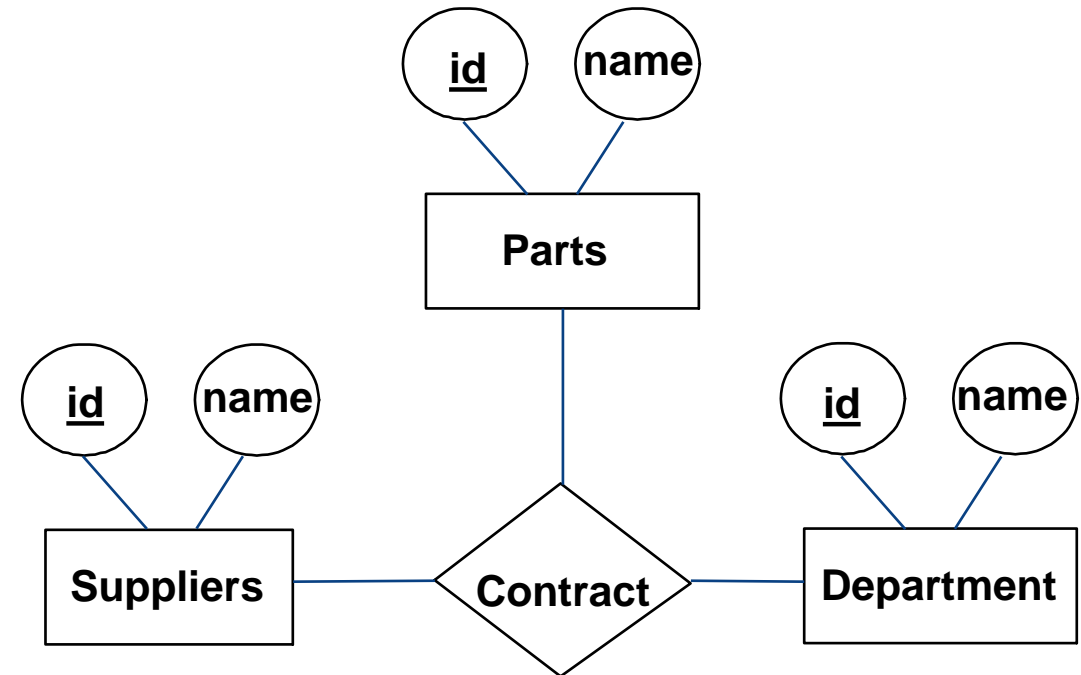
## ER to Logical to Implementation Example 2

### Logical Design:

Contract (  
    supplier id,  
    part id,  
    department id)

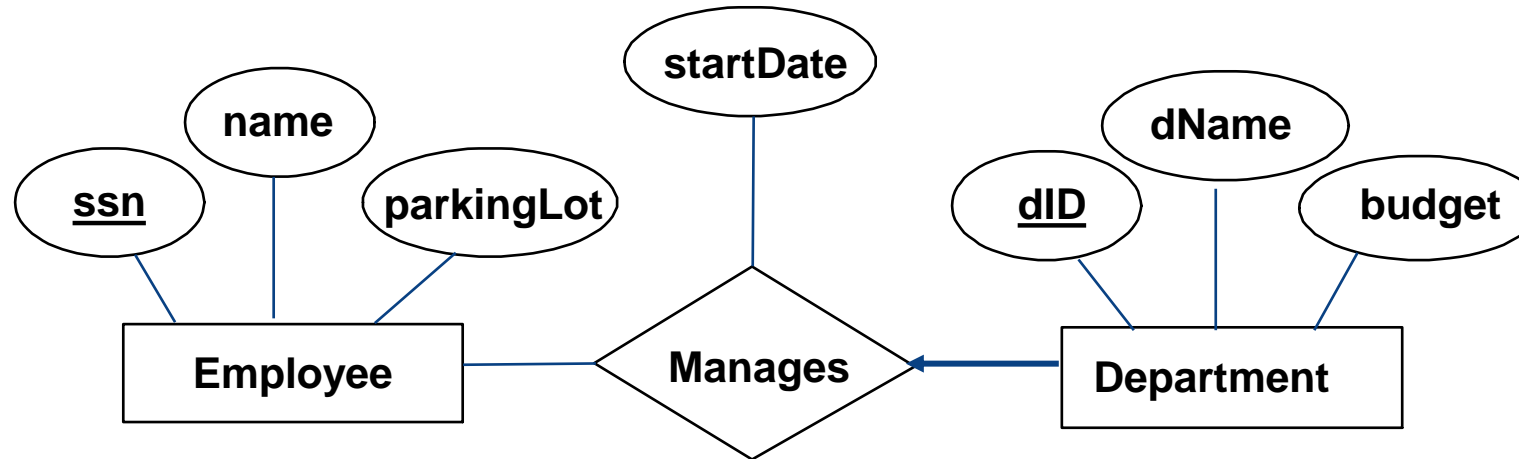
### Implementation:

```
CREATE TABLE Contract (  
    supplier_id INTEGER,  
    part_id INTEGER,  
    department_id INTEGER,  
    PRIMARY KEY (supplier_id, part_id, department_id),  
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(id),  
    FOREIGN KEY (part_id) REFERENCES Parts(id),  
    FOREIGN KEY (department_id) REFERENCES Departments(id))
```



# Key Constraints: Logical design

Each department has at most one manager, according to the key constraint on Manages.



## Logical Design:

Employee = (ssn, name, parkingLot)

Department = (dID, dName, budget)

Manages = (ssn, *dID*, startDate)

VS.

Employee = (ssn, name, parkingLot)

Department = (dID, dName, budget, *ssn*, startDate)

Note:

Underline = PK,

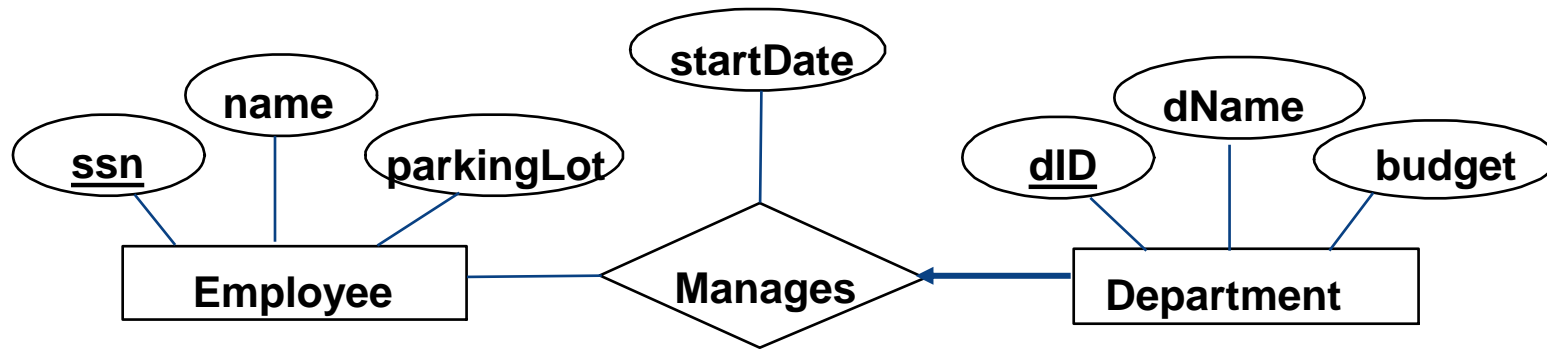
Italic = FK,

Underline + Italic = PFK



# Key Constraints in SQL

## Implementation:



```
CREATE TABLE Employee
(ssn CHAR(11),
 name CHAR(20),
 parkingLot INTEGER,
 PRIMARY KEY (ssn))
```

```
CREATE TABLE Manages(
 ssn CHAR(11),
 dID INTEGER,
 startDate DATE,
 PRIMARY KEY (ssn, did),
 FOREIGN KEY (ssn) REFERENCES Employee,
 FOREIGN KEY (dID) REFERENCES Department)
```

VS.

```
CREATE TABLE Department
(dID INTEGER,
 dName CHAR(20),
 budget FLOAT,
 ssn CHAR(11),
 startDate DATE,
 PRIMARY KEY (dID),
 FOREIGN KEY (ssn) REFERENCES
 Employee (ssn))
```

Which one is better?

# Key Constraints rule

RULE: Primary key from the *one* side becomes a foreign key on the *many* side

This is the way to ensure that the key constraint holds

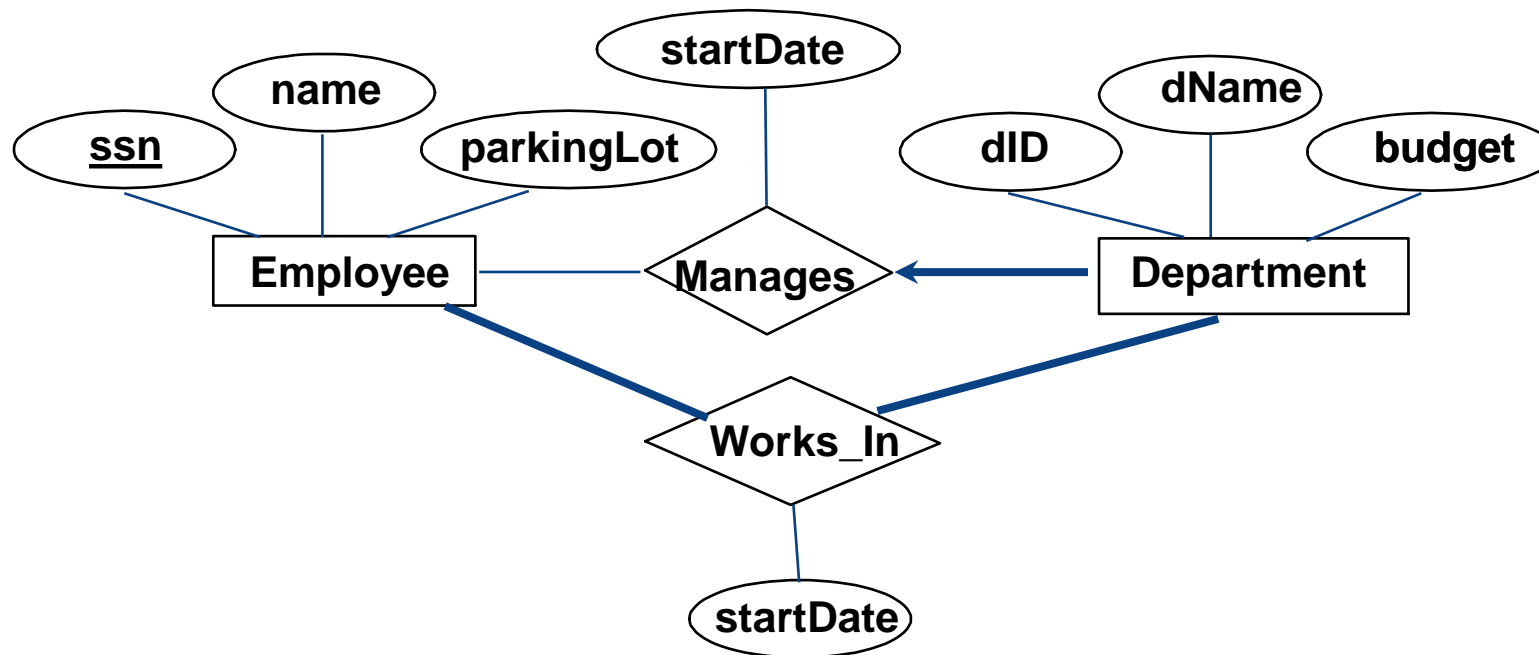
```
CREATE TABLE Department
(dID INTEGER,
 dName CHAR(20),
 budget FLOAT,
 ssn CHAR(11),
 startDate DATE,
 PRIMARY KEY (dID)
 FOREIGN KEY (ssn)
 REFERENCES Employee (ssn))
```

Each department will  
have a *single* manager

# Review: Participation Constraints

Does every department have a manager?

- If so, this is a participation constraint: the participation of Departments in Manages is said to be *total* (vs. *partial*).



# Participation Constraints in SQL

We specify total participation with key words NOT NULL

- NOT NULL = this field cannot be empty

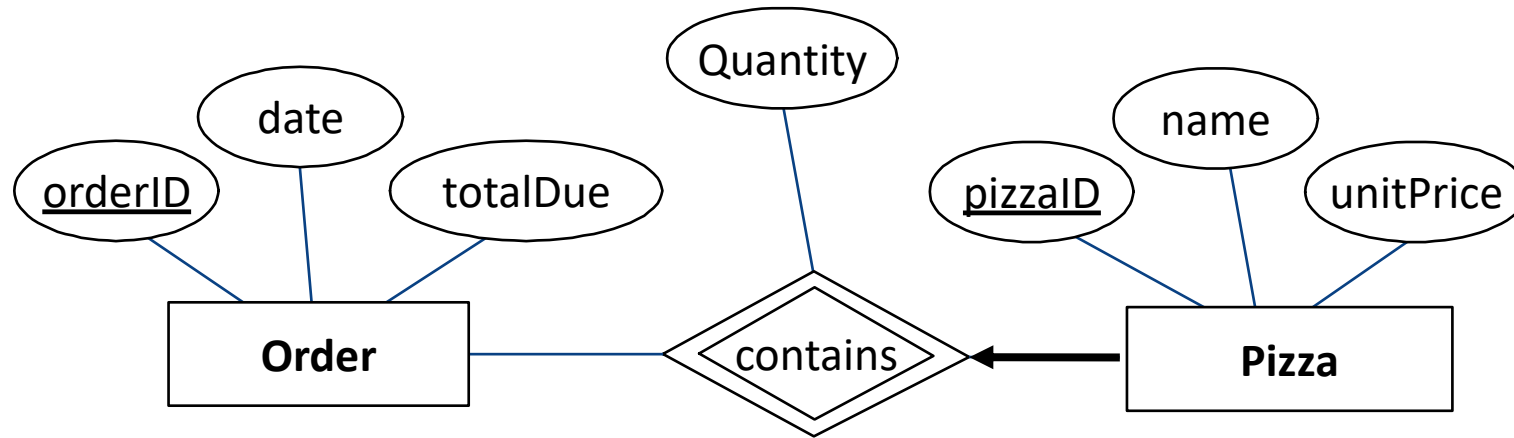
```
CREATE TABLE Departments(  
    dID    INTEGER NOT NULL,  
    dName  CHAR(20),  
    budget REAL,  
    ssn    CHAR(11) NOT NULL,  
    startDate DATE,  
    PRIMARY KEY (dID),  
    FOREIGN KEY (ssn) REFERENCES Employee (ssn),  
    ON DELETE NO ACTION)
```

Note, **NO ACTION** specification

It is the default and need not be explicitly specified; it ensures that an Employee tuple cannot be deleted while it is pointed to by a Department Manager tuple. If we wish to delete such an Employee tuple, we must first change the Department Manager tuple to have a new employee as manager.

# Review: Weak Entities

A **weak entity** can be identified uniquely only by considering the primary key of another (*owner*) entity.



When an owner entity is deleted, we want all owned weak entities to be deleted.

# Translating Weak Entity Sets

Weak entity set and identifying relationship set are translated into a single table.

- Contains is not the best entity name when we convert a relationship into an entity
- When the owner entity is deleted, all owned weak entities must also be deleted.

## Logical Design:

OrderItem = (pizzaID, qty, orderNo)

## Implementation:

```
CREATE TABLE OrderItem (  
  pizzaID CHAR(3) NOT NULL,  
  qty INTEGER,  
  orderNo CHAR(5) NOT NULL,  
  PRIMARY KEY (pizzaID, orderNo),  
  FOREIGN KEY (pizzaID) REFERENCES Pizza, ON DELETE CASCADE)  
  FOREIGN KEY (orderNo) REFERENCES Order, ON DELETE CASCADE)
```

Note:

Underline = PK,

Italic = FK,

Underline + Italic = PFK



# What's examinable\*

**Be able to model a case study from conceptual to instance and all stages in between (conceptual, logical, physical, implementation and instance)**

**Translate conceptual (ER) into logical & physical design**

**Understand integrity constraints**

**Use DDL of SQL to create tables with constraints**

\* All material is examinable – these are the suggested key skills you would need to demonstrate in an exam scenario



THE UNIVERSITY OF  
MELBOURNE

# Thank you