1

Stages of Database Design

- S1 • Requirement Analysis
- S2 • Conceptual Database Design
- S3 • Logical Database Design
- S4 • Refine the Schemas
- S5 • Physical Database Design
- S6 • Application and Security Design

ER model is relevant to the first three stages of database design.

2

Database Design Background

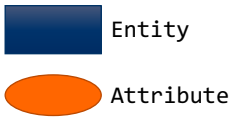
- Real-world application can be **large** and **complicated**.
  - We need our clients and domain experts to clearly tell us the requirements.
  - 100% clarity may never come.
- With the requirements, questions to be answered:
  - How to **represent** the information?
  - What **operations** do we need?
  - Any **constraints** between the items?

- Popular solution: a **graph**, a great choice to model the items and relationships, to model the conceptual schema.
- Such graph representation later guides us to create databases / tables.
- Pay attention to two directions:
  - **Missing info & redundancy.**

https://www.itprotoday.com/

3

ER Model – Entity



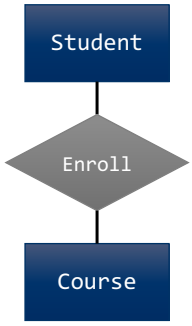
- Full name: **entity-relationship model**.
  - Allows us to model the data as a **graph**.
    - Pictures speak a thousand words, so much easier to debug.
  - Simple and **intuitive**, but also **powerful**.
  - Natively compatible with the **relational data model**.
    - There are even tools to automatically convert ER model into the database.
- **Entity**: a unique **object** in real-world.
  - e.g., student Judy Hops and course CSC2008.
  - Each entity has a set of **attributes**.
  - Each attribute has a **data type**.
  - A subset of the attributes can **uniquely identify** the entity.

- **Entity set**: a collection of the entities, with the **same properties**.
  - Same attributes, same data type, same order, same key attributes (only 1).
  - Values are different, at least for the key attributes.
  - Possible to have **several entity sets** with the same properties.
    - e.g., StudentAY19 and StudentAY20.

4

ER Model – Relationship

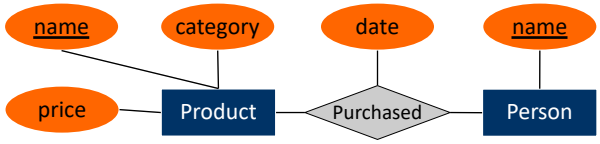
- **Relationship**: **association** among two or more entities.
  - **Uniquely identified** by the keys of its entities.
- Attribute: relationships may have **attributes** as well.
  - e.g., since and in.
  - Associated with a data type or feasible value set.
  - The data type can be quite complex.
    - e.g., address with country, city, postal code, unit no., etc.
- **Relationship Set**: **collection** of similar relationships.
  - **n-array relationship** set R relates **n entity sets** E1, ..., En.
  - **Same entity set** could participate in **>1 relationship sets**.
  - For simplicity, often use short words to describe and name.



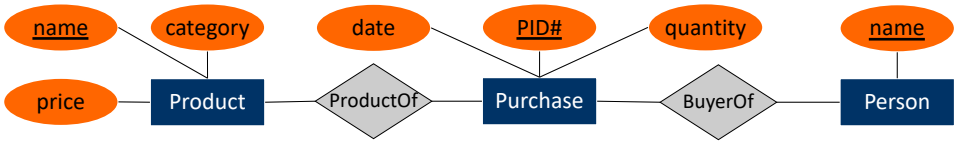
5

Design Decision: Relationship vs. Entity?

- Concept is not hard. But it can be challenging for real applications.
- Modeling as a relationship makes it unique; what if **not appropriate**?
  - A person can only buy a specific product once (on one day)?



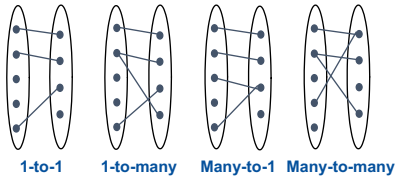
- Better design: have multiple relationships to capture the purchases per product & person pair.



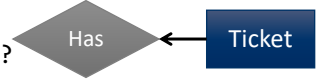
6

Key Constraints and Participation Constraints

- Example: the Booking entity appears in at most one Makes relationship.
- Makes relationship: 1-to-many relationship.
- Different relationships.
  - 1-to-1.
  - 1-to-many.
  - Many-to-1.
  - Many-to-many.



- **Key constraints** on the Makes relationship tells us that a booking belongs to **at most one** (≤ 1) customer.
- What if every booking **must be from someone** (≥ 1)?

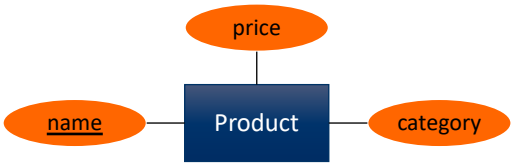


- If so, this requirement is an example of a **participation constraint**.

7

From ER Diagrams to Relational Schema

- Idea: **entity set** and **relationship set** both become **relations** or tables.
- Entity set -> a relation.
  - Each **tuple** is one **entity** in the entity set.
  - Each tuple is composed of the entity's **attributes.**
  - The same **primary key** for both entity set and table.



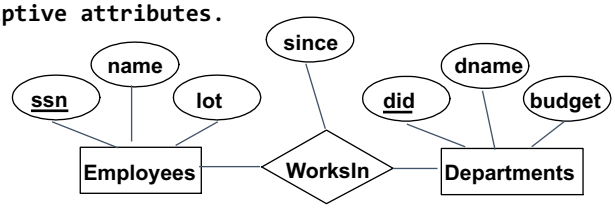

```
CREATE TABLE Product(
        name            CHAR(50) PRIMARY KEY,
        price           DOUBLE,
        category        VARCHAR(30)
)
```

name	price	category
apple	3.95	fruit
pear	2.95	fruit
pencil	0.80	stationary

8

Relationship Set to Tables

- In translating a **relationship** set to a relation, **attributes** of the relation must include:
  - **Keys** for all the participating **entity** sets, as **foreign keys**.
  - All **descriptive attributes**.



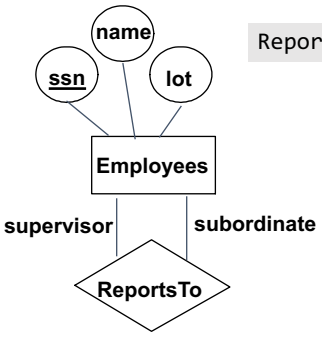

```
CREATE TABLE WorksIn(
        ssn  CHAR(11),
        did  INTEGER,
        since  DATE,
        PRIMARY KEY (ssn, did),
        FOREIGN KEY (ssn) REFERENCES Employees,
        FOREIGN KEY (did) REFERENCES Departments)
```

WorksIn(ssn: char(11), did: int, since: date)

ssn	did	since
123	1	2011-08-01
145	2	2015-08-01
199	2	2019-07-01

9

Relationship Set to Tables



ReportsTo(supervisor_ssn: char(11), subordinate_ssn: char(11))

Same entity set for both the left side and right side of a relationship set. The key is to **rename**.

Supervisor_ssn	Subordinate_ssn
123	145
123	199
199	255

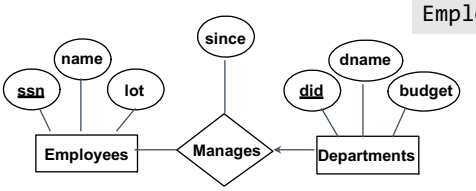
```
CREATE TABLE Reports_To(
        supervisor_ssn  CHAR(11),
        subordinate_ssn  CHAR(11),
        PRIMARY KEY (supervisor_ssn, subordinate_ssn),
        FOREIGN KEY (supervisor_ssn) REFERENCES Employees(ssn)
        FOREIGN KEY (subordinate_ssn) REFERENCES Employees(ssn))
```

10

SIT Internal

Relationship Set with Key Constraint

- **Key constraint** on Manages -> each department has at most 1 manager.



`Employee(ssn: char(11), name: char(50), lot: int)`

Employee

ssn	name	lot
1	Jogn	2
2	Eric	2
3	Lydia	1

`Manages(did: int, ssn: char(11), since: date)`

Manages

did	ssn	since
1	1	05/09/2013
2	2	04/03/2015
3	2	02/06/2014
3	3	06/03/2015

`Departments(did: int, dname: char(50), budget: int)`

Departments

did	dname	budget
1	Finance	10k
2	HR	5k

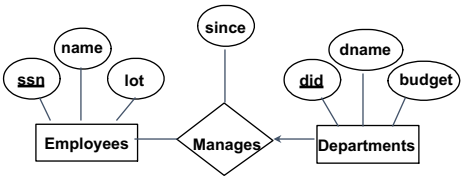
Invalid, does not satisfy key constraint

11

SIT Internal

Relationship Set with Key Constraint

- **Key constraint** on Manages -> each department has at most 1 manager.
- did should be the **key** for Manages. Not both did and ssn.



`Manages(did: int, ssn: char(11), since: date)`

Can further improve?

Manages

did	ssn	since
1	1	05/09/2013
2	2	04/03/2015
3	2	02/06/2014

Only 1 entry per did.

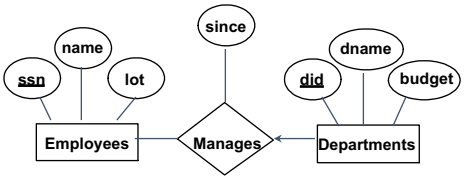
```
CREATE TABLE  Manages(
        ssn  CHAR(11),
        did  INTEGER,
        since  DATE,
        PRIMARY KEY (did),
        FOREIGN KEY (ssn) REFERENCES Employees,
        FOREIGN KEY (did) REFERENCES Departments)
```

12

Relationship Set with Key Constraint

- **Relationships** with **key constraints** are **merged** with the constrained entity to form **one relation/table.**
- Since each department has a unique manager, we could instead combine Manages and Departments -> **save space** for key duplications.

DeptMgr(did: int, dname: char(11), budget: real, ssn: char(11), date: date)



did	dname	budget	ssn	since
1	Finance	10k	1	05/09/2013
2	HR	5k	2	04/03/2015
3	Transport	20k	2	02/06/2014

```
CREATE TABLE DeptMgr(
      did INTEGER,
      dname CHAR(20),
      budget REAL,
      ssn CHAR(11),
      since DATE,
      PRIMARY KEY (did),
      FOREIGN KEY (ssn) REFERENCES Employees)
```

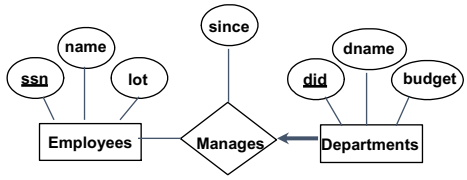
13

Relationship Set with Participation Constraint

- Relationships with **key participating** constraints are merged and must indicate that the other entity key must be **not null.**
- Does every department have a manager?
  - If so, this is a **participation constraint.**

```
CREATE TABLE DeptMgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11) NOT NULL,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE NO ACTION) -- default
```

DeptMgr(did: int, dname: char(11),
budget: real, ssn: char(11), date: date)



14

ER Diagram Summary

- ER design:
  - An **important pre-step** before jumping in to develop the database schema.
  - **Different solutions** to a problem.
    - **Analyze** the **application** to find a suitable or better one.
  - Most importantly, **ask questions** (clarify with customers).
    - May practice during your internships.
- One step closer towards a good database design.
  - Requirements -> conceptual design (ER).
  - ER -> logical design (relation schema).
  - Logical design -> further **refinement**.
    - Functional dependencies and database normalization.
  - We march to the next stage in the coming week.

https://www.techradar247.com/

15