Database

Design

ER Diagrams

Case Study

❯ Game Store
Requirement
Design

Game Store

Requirement

Game Store Requirement

Our company, **Apasaja Pte Ltd**, has been commissioned to develop an application to manage the data of an online app store. We want to store several items of information about our customers such as their first name, last name, date of birth, e-mail, date and country of registration to our online sales service and the customer identifier that they have chosen.

We also want to manage the list of our products, games, their name, their version, and their price. The price is fixed for each version of each game. Finally, our customers buy and download games. We record which version of which game each customer has downloaded. It is not essential to keep the download date for this application.

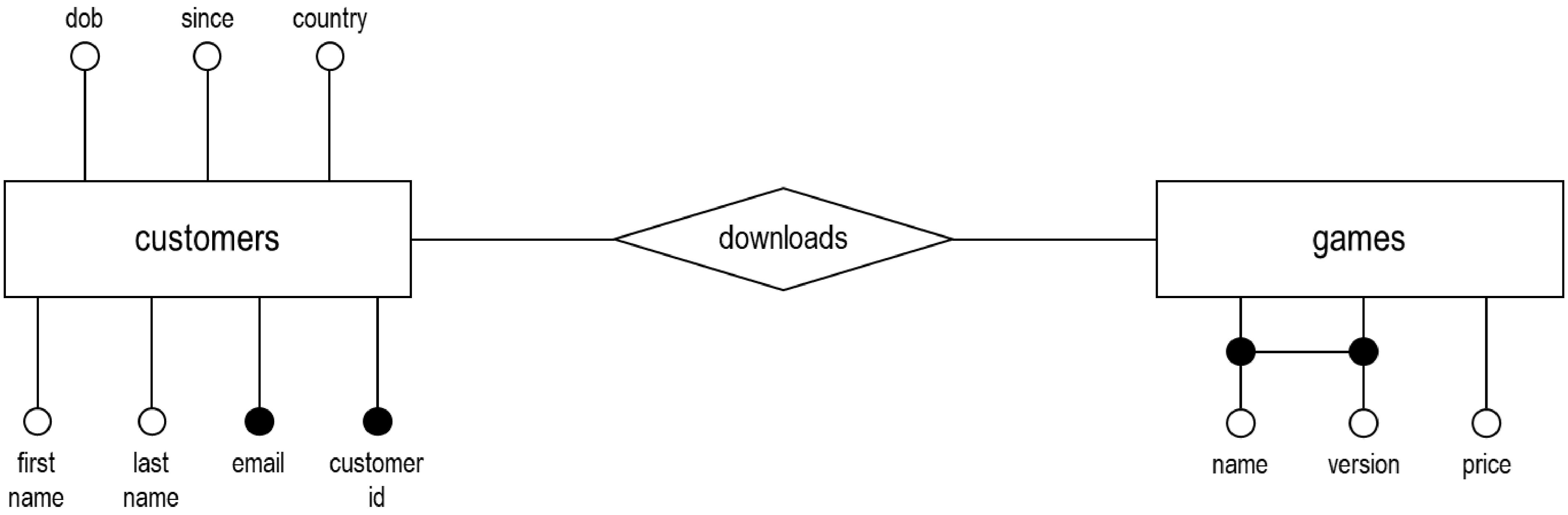
Case Study

Requirement

❯ Design

Design

Entity-Relationship Diagram



Diagram

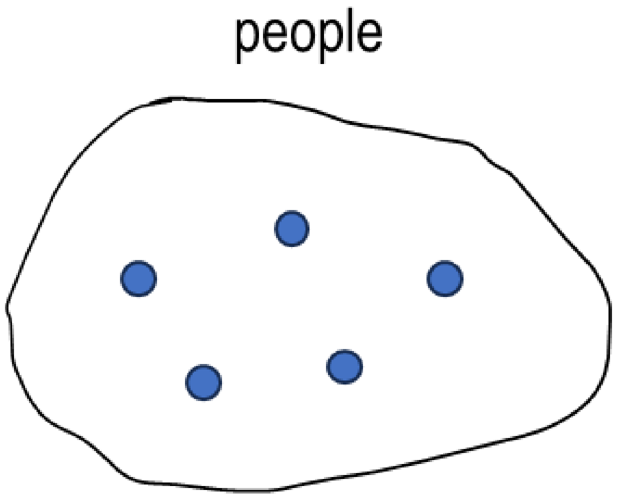
Entities

Sets

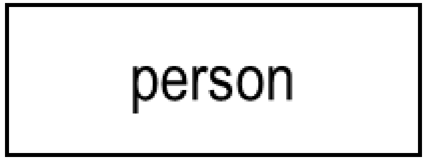
Entities and Entity Sets

Entities are identifiable things. The named box represents a set of entities or ***entity set***.

Concept



Diagram



Diagram

Entities

Attributes

Attributes, Values, and Value Sets

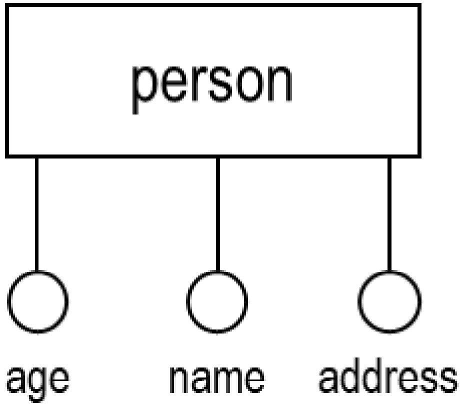
The ER model is value-oriented. Values can be integer, strings, or atoms.

Attributes of Entities

Entities can have attributes. All entities in one entity set have the same attributes.

However, the attributes take different values for each entity.

Diagram



Diagram

- Entities
- ❱ Relationships
  - *Sets*
  - *Attributes*
  - *Same Entity Set*
  - *n-Ary*
- Aggregation
- Consideration

Relationships

Sets

Note

In our convention, rectangle **can only connect** to diamond *(and vice versa)*. The only **exception** is connection to the attributes.

Relationships and Relationship Sets

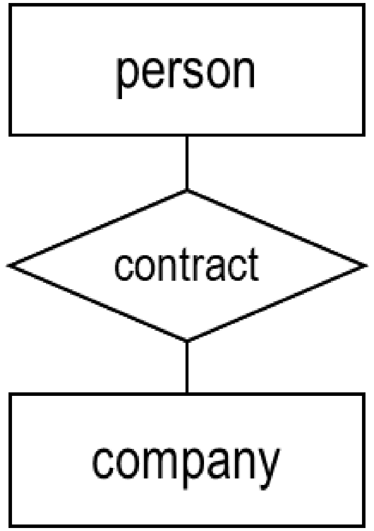
Relationship associates two entities *(can be fewer or more)*. The named diamond represents a set of relationships or ***relationship set***.

Association

A relationship set is a set of relationships associating entities from the same entity sets.

In the example on the right, we exclude attributes for simplicity. As it is a set, a person P can only be associated with a company C at most ***once***. But they can be associated with different company/person.

Diagram



Diagram

Relationships

Sets

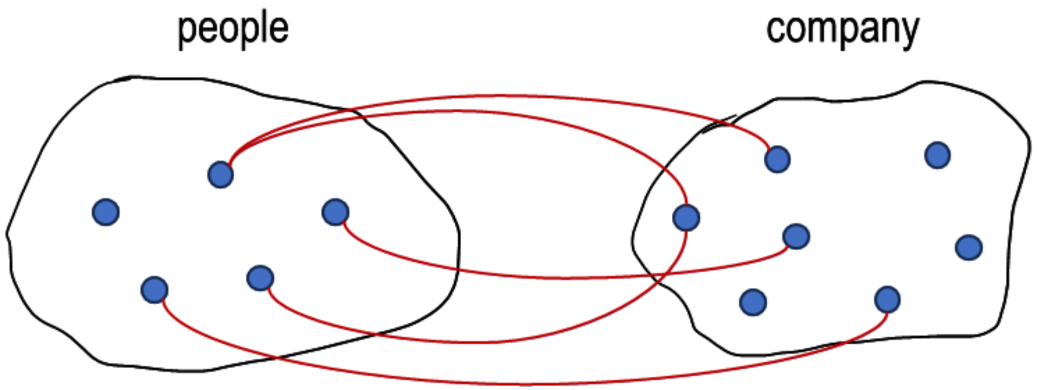
Note

In our convention, rectangle **can only connect** to diamond *(and vice versa)*. The only **exception** is connection to the attributes.

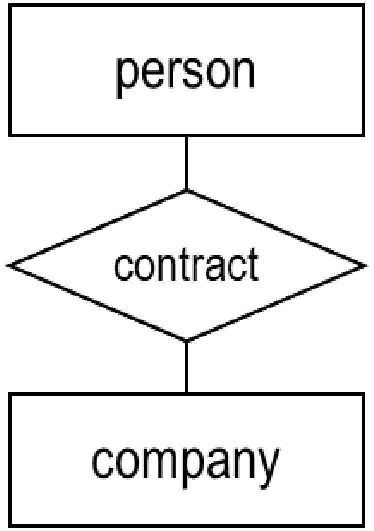
Relationships and Relationship Sets

Relationship associates two entities *(can be fewer or more)*. The named diamond represents a set of relationships or ***relationship set***.

Idea



Diagram



Diagram

Relationships

Attributes

Distinguishing Relationships

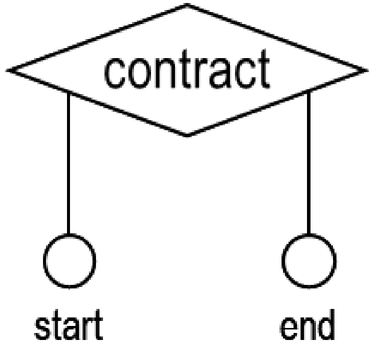
Relationship are distinguished not by their attributes but by their participating entities.

Attributes of Relationships

Relationship can have attributes. All relationships in one relationship set have the same attributes *(but different values)*.

The attributes are **dependent** on the **entities** being associated.

Diagram



Diagram

Relationships

Same Entity Set

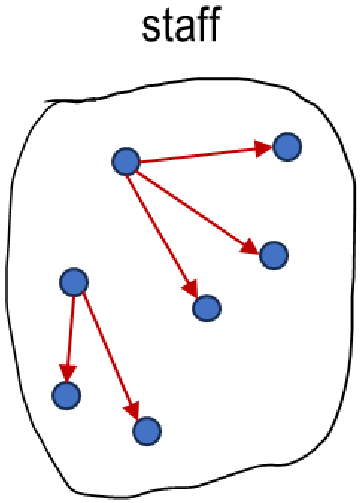
Note

The participation can always be **named** even when not associating the same entity set. But using a meaningful name for relationship set is better.

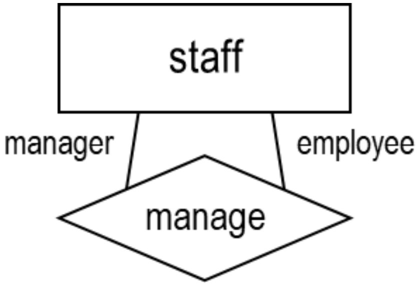
Associating the Same Entity Set

Relationships can associate entities from the same entity set. In this case *(and in general)*, participation *(or role)*, in the relationship can be named.

Idea



Diagram



Diagram

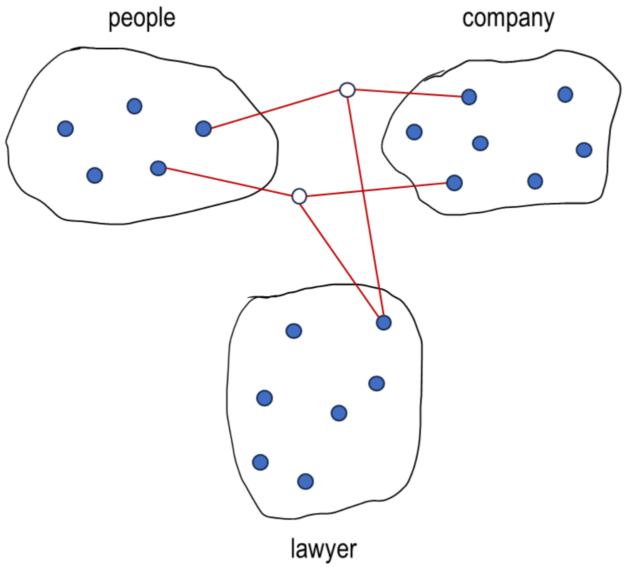
Relationships

n-Ary

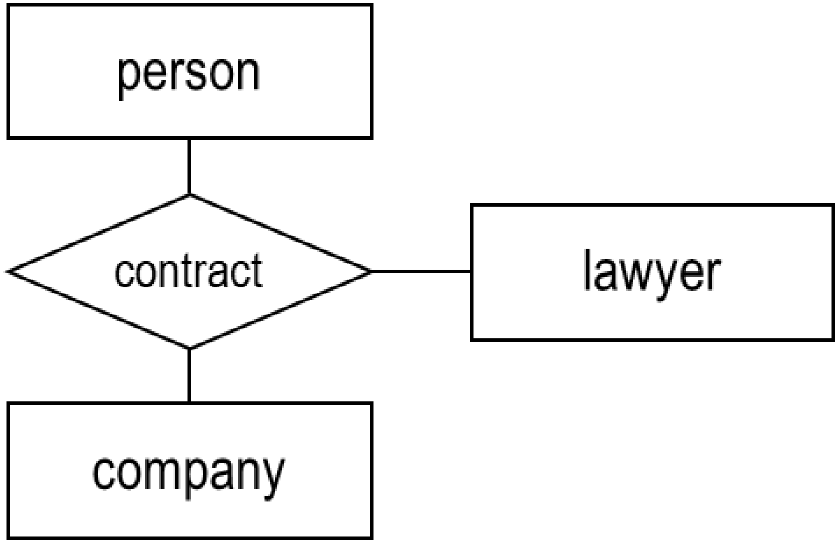
More Than 2 Entity Sets

Relationship sets can associate more than 2 entity sets. We call the relationship as n-ary relationships.

Idea



Diagram



Diagram

Entities
Relationships
❱ Aggregation
Consideration

Aggregation

Relationship Sets as Entity Sets

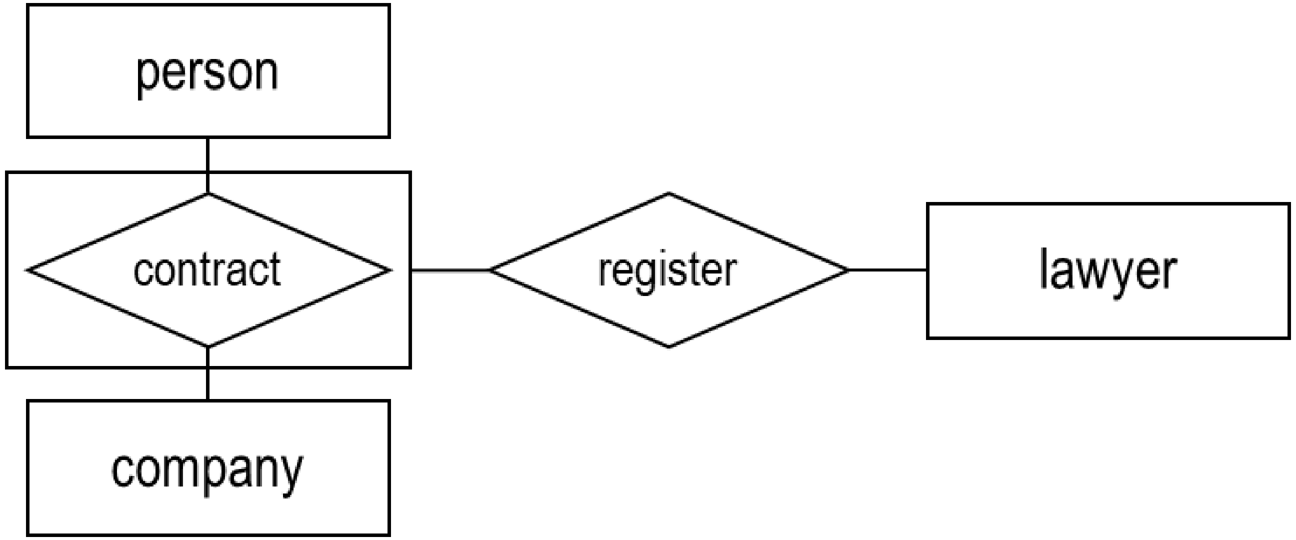
Note

Rectangle still connects only to diamond. The gap between rectangle and diamond ensures **no ambiguity**.

Associating with Relationship Set

In some instances, we want to associate an entity set with a relationship set. We represent this by wrapping the relationship set in a box.

Diagram

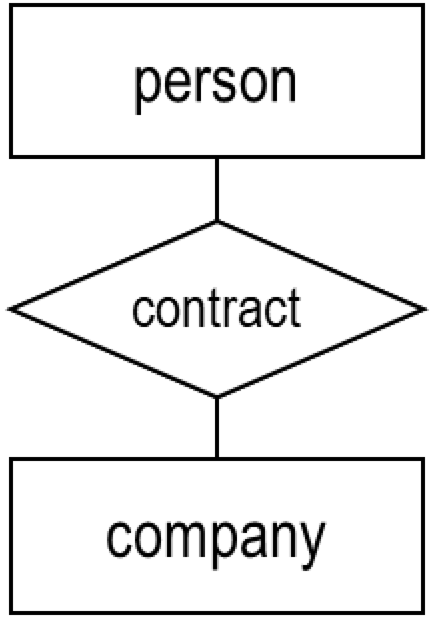


Diagram

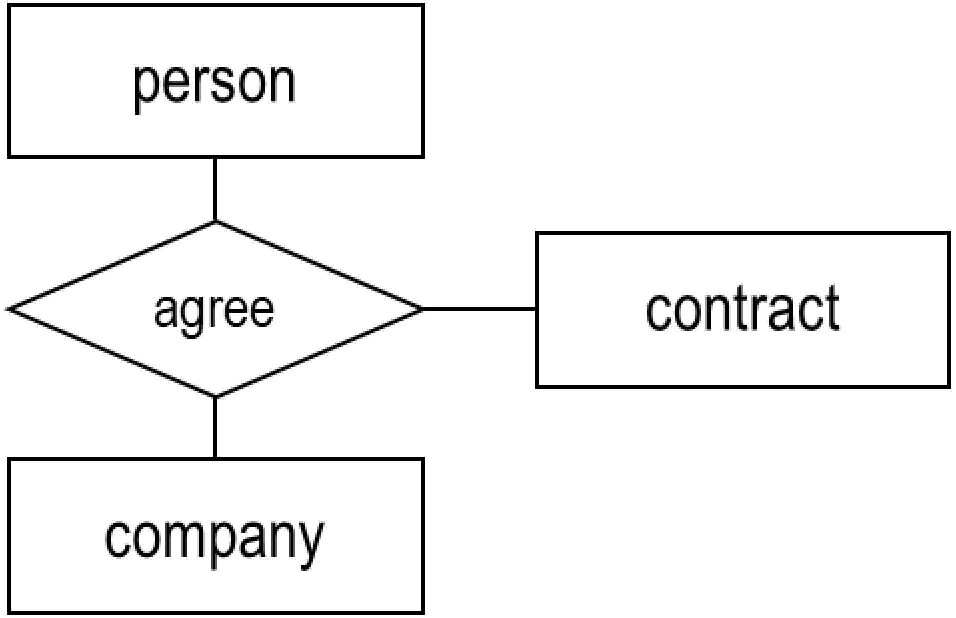
Consideration

Entity or Relationship?

Alternative #1



Alternative #2



Identities and Cardinalities

- Identities
  - *Key Attribute*
  - *Multi Key*
  - *Partial Key*
- Cardinality

Note

The set of all **identities** is also called **candidate keys**. Only **entity sets** can have identities in ER diagram *(i.e., no black dots on relationship set)*.

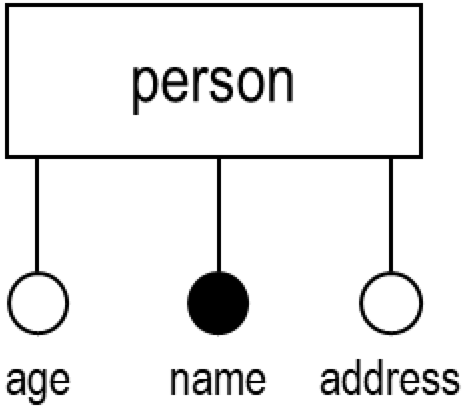
Identities

Key Attribute

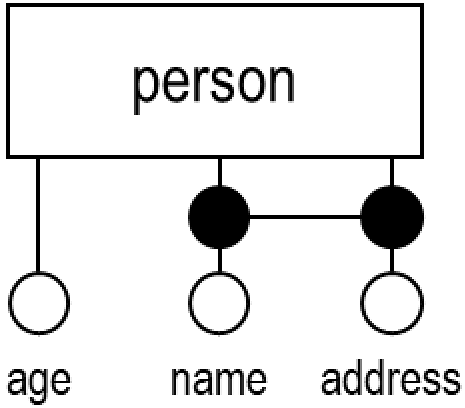
Entities' Identity

One or more attributes can identify the entity. This is a property of all entities in an entity set*. We use ***black dots*** to differentiate the key attributes.

Single Attribute



Multiple Attributes



*At the very least, the combination of all attributes identifies the entity.

Identities and Cardinalities

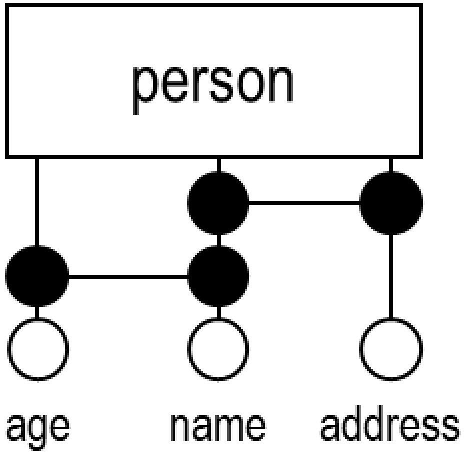
Identities

Multi Key

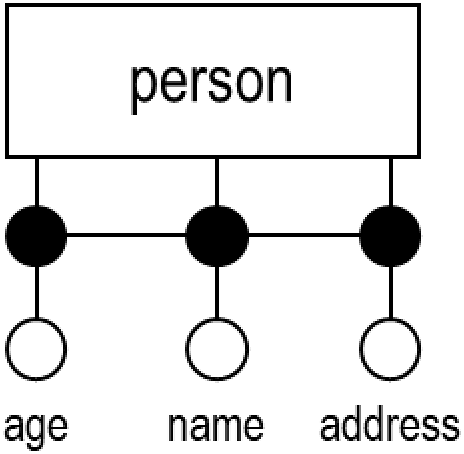
Entities' Identity

One or more attributes can identify the entity. This is a property of all entities in an entity set*. We use ***black dots*** to differentiate the key attributes.

Several Keys



Worst-Case



*We prefer the collection of **minimal** set of attributes.

Identities and Cardinalities

- ❯ Identities
  - *Key Attribute*
  - *Multi Key*
  - *Partial Key*
- Cardinality

Identities

Partial Key

Partial Identification

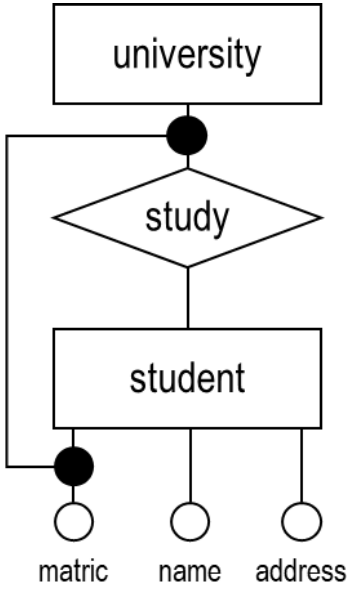
Some entities can only be identified within the scope of a relationship with another entity set. **Note:** The relationship must exist and be unique for each entity in the set.

Weak Entities

Matric numbers are given by the universities. The same number can be used by different universities.

- University is a dominant entity.
- Student is a weak entity *(cannot be identified by its attributes alone)*.

Diagram



Identities and Cardinalities

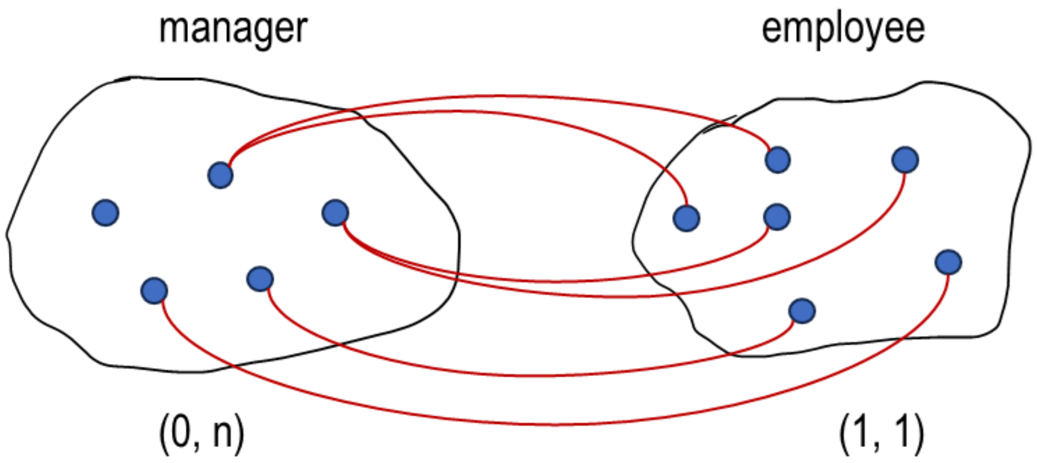
Cardinality

Participation

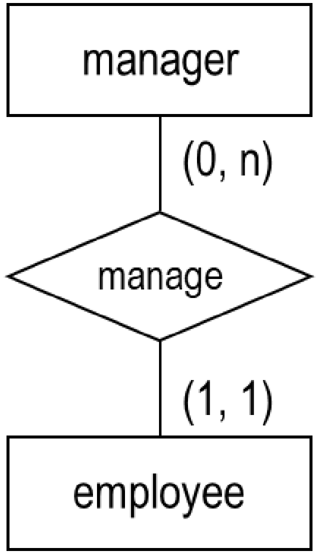
Kinds of Participation

The cardinality of the participation in a relationship can be constrained by a **minimum** and **maximum** value as `(min, max)`. For example, `(1, 1)`, `(0, n)`, `(2, 5)`, *etc*.

Idea



Diagram



Identities and Cardinalities

Identities
❱ Cardinality
Participation
Classifications
Examples

Cardinality

Classifications

Common Names

- `(1, x)` characterizes a ***mandatory*** participation.
- `(0, x)` characterizes an ***optional*** participation.
- `(x, 1)` may characterizes a ***one-to-one*** relationship
  if `(x, 1)` for all entities involved
- `(x, 1)` may characterizes a ***one-to-many*** relationship.
  if `(x, 1)` for one but `(x, n)` or `(x, y)` for y > 1 for other
- `(x, n)` characterizes a ***many-to-many*** relationship.
  if `(x, n)` for all

Note

This classification is not so useful for relationship set associating **more than two** entity sets. We use ***look here convention***. The cardinality describes the participation of the entity set it is attached.

Identities and Cardinalities

Identities
❱ Cardinality
Participation
Classifications
Examples

Cardinality

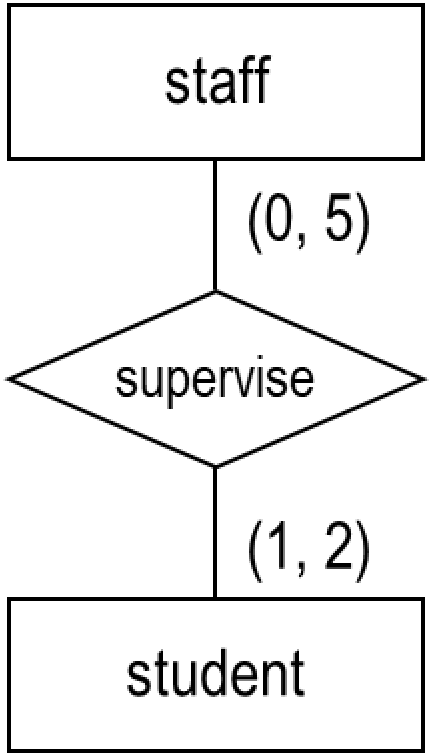
Examples

Example 1

Academic staff can supervise up to 5 research students. Some staff do not supervise students. Research students can have one or two supervisors.

Note

Here, the **staff** can participate 0 times in **supervise**. At most, **staff** can participate up to 5 times in **supervise**. This can be easily generalized to n-ary relationship.



*By default, if cardinality is omitted, we have optional many-to-many relationships *(i.e., `(0, n)`)*.

Identities and Cardinalities

Cardinality

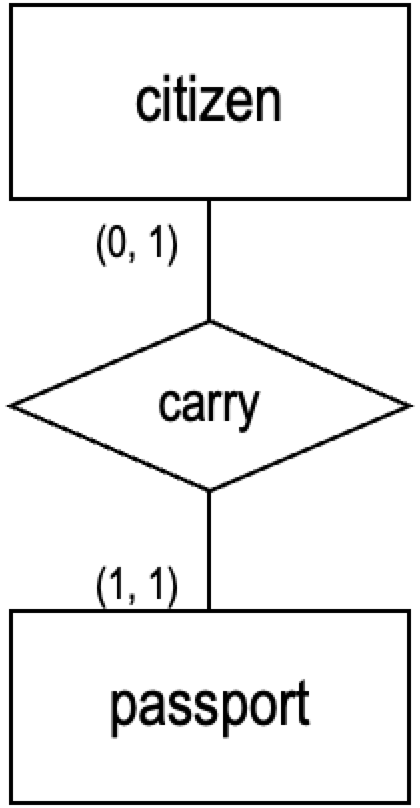
Examples

Example 2

Example of one-to-one relationship. We are assuming that old passports are **removed** from the database *(i.e., operational database and not historical database)*.

Note

The alternative of having passport information as part of **citizen** is less desirable as it introduces `null` values. This design **avoids** `null` values.



Identities and Cardinalities

Cardinality

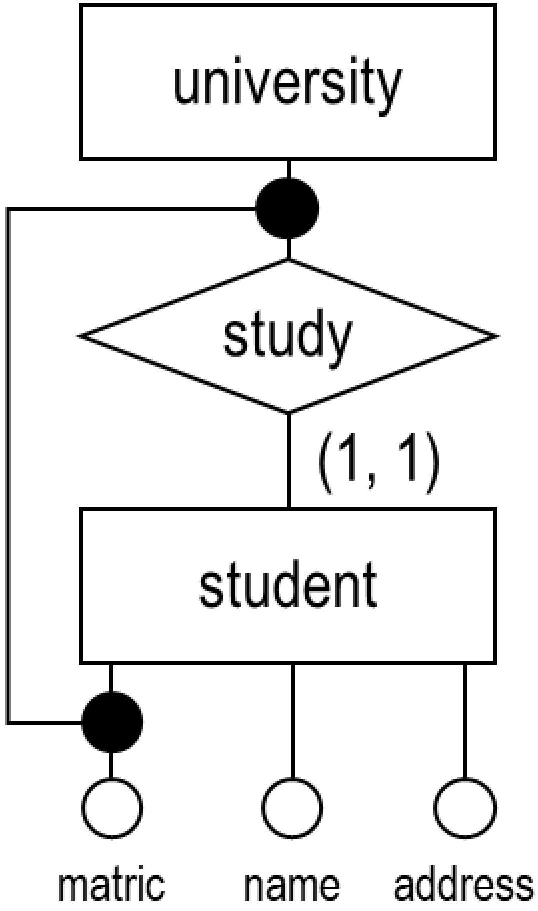
Examples

Example 3

Weak entities can only be defined for a participation constrained by `(1, 1)` cardinalities *(also called mandatory one-to-many relationships)*.

Note

The student **matric** no longer uniquely identifies a student because there can be two **different** students with the **same matric**. But they have to be in **different universities**.



Schema Translation

- **Rules**
  - *Value Sets*
  - *Entity Sets*
  - *Relationship Sets*
- Exceptions
- Limitations

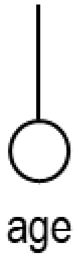
Rules

Value Sets

Rule #1

Value sets are mapped to **domains**. In practice, this is a first step towards the physical design. ER attributes are mapped to attributes of relations **with meaningful type**.

```
age INTEGER
```



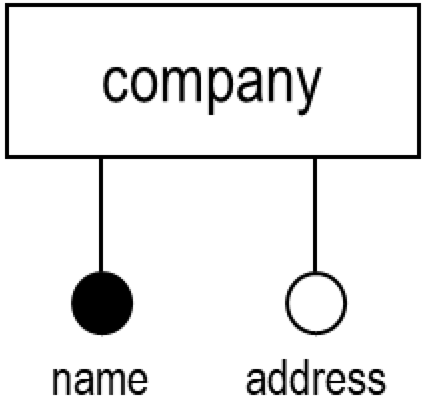
Schema Translation

Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to ***primary keys*** and `UNIQUE NOT NULL`.



Candidate keys** are all sets of attributes that uniquely identify the entities. But there **must** be at least one ***primary keys

Schema Translation

- Rules
  - *Value Sets*
  - *Entity Sets*
  - *Relationship Sets*
- Exceptions
- Limitations

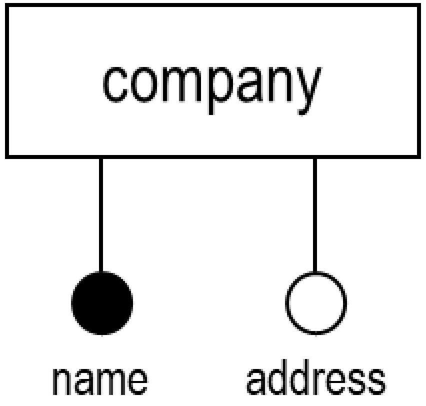
Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to *primary keys* and `UNIQUE NOT NULL`.

```sql
CREATE TABLE company (
  name VARCHAR(64) PRIMARY KEY,
  address VARCHAR(128)
);
```



***Candidate keys** are all sets of attributes that uniquely identify the entities. But there **must** be at least one *primary keys*

Schema Translation

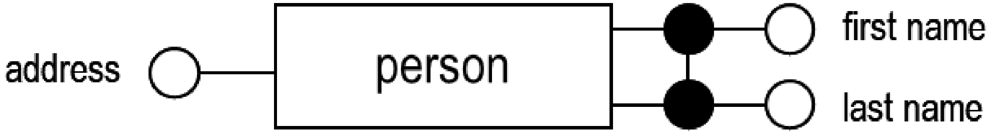
- **Rules**
  - *Value Sets*
  - ***Entity Sets***
  - *Relationship Sets*
- Exceptions
- Limitations

Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to ***primary keys*** and `UNIQUE NOT NULL`.



Schema Translation

- **Rules**
  - Value Sets
  - *Entity Sets*
  - Relationship Sets
- Exceptions
- Limitations

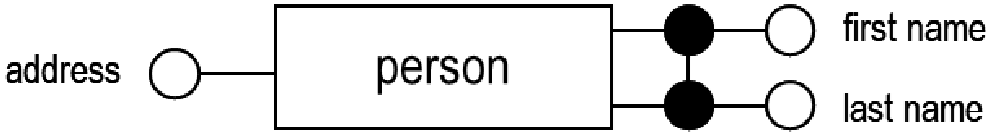
Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to ***primary keys*** and `UNIQUE NOT NULL`.

```sql
CREATE TABLE person (
  first_name VARCHAR(32),
  last_name VARCHAR(32),
  address VARCHAR(128) NOT NULL,
  PRIMARY KEY (first_name, last_name)
);
```



Schema Translation

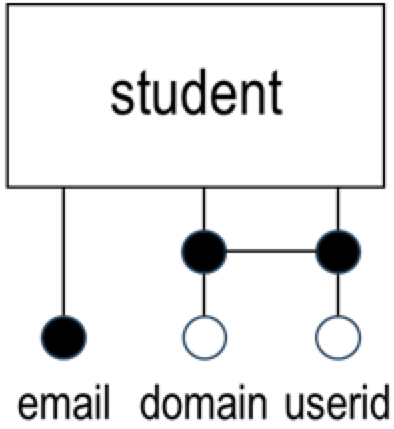
- **Rules**
  - *Value Sets*
  - ***Entity Sets***
  - *Relationship Sets*
- Exceptions
- Limitations

Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to ***primary keys*** and `UNIQUE NOT NULL`.



Schema Translation

❱ Rules
Value Sets
Entity Sets
Relationship Sets
Exceptions
Limitations

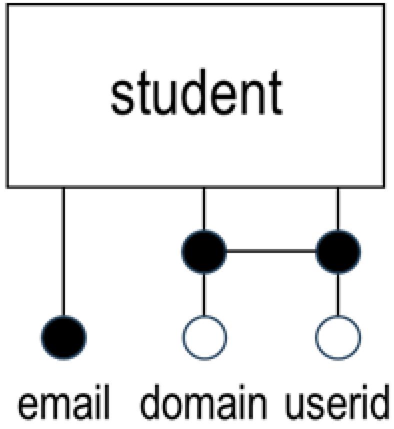
Rules

Entity Sets

Rule #2

Entity sets are mapped to **relations**. The entity set attributes are mapped to attributes of the relation. The **candidate keys** are mapped to ***primary keys*** and `UNIQUE NOT NULL`.

```
CREATE TABLE student (
  email VARCHAR(64) PRIMARY KEY,
  domain VARCHAR(12) NOT NULL,
  userid VARCHAR(50) NOT NULL,
  UNIQUE (domain, userid)
);
```



Schema Translation

- **❱ Rules**
  - *Value Sets*
  - *Entity Sets*
  - ***Relationship Sets***
- Exceptions
- Limitations

Rules

Relationship Sets

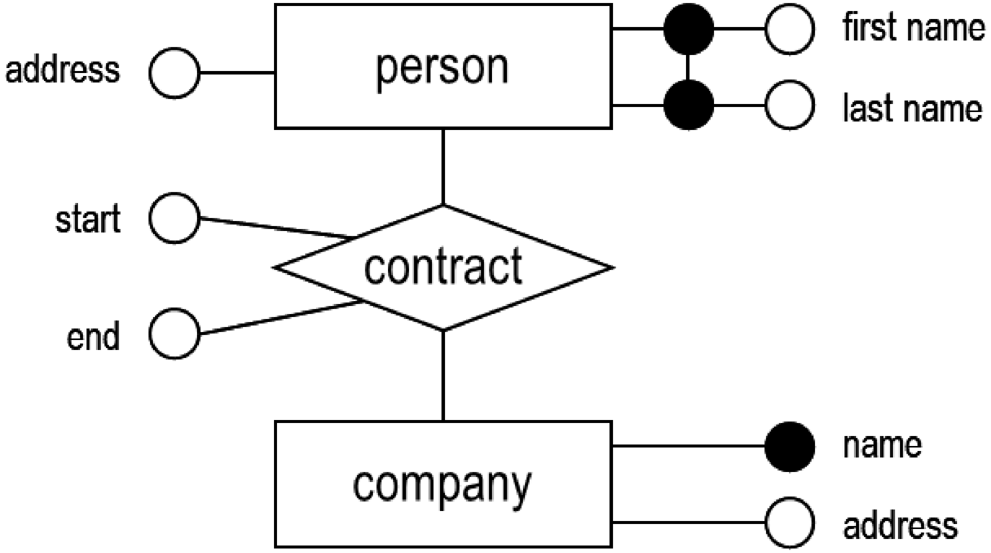
Rule #3

Relationship sets are mapped to relations. The attributes of the relation consist of the attributes of the relationship set. The keys are the keys of the participating entities.

Note

Aggregate is simply a **relationship set**. So the rule for relationship set applies.

But it can be used as **entity set** in a sense that the keys can be **referenced** by another relationship set.



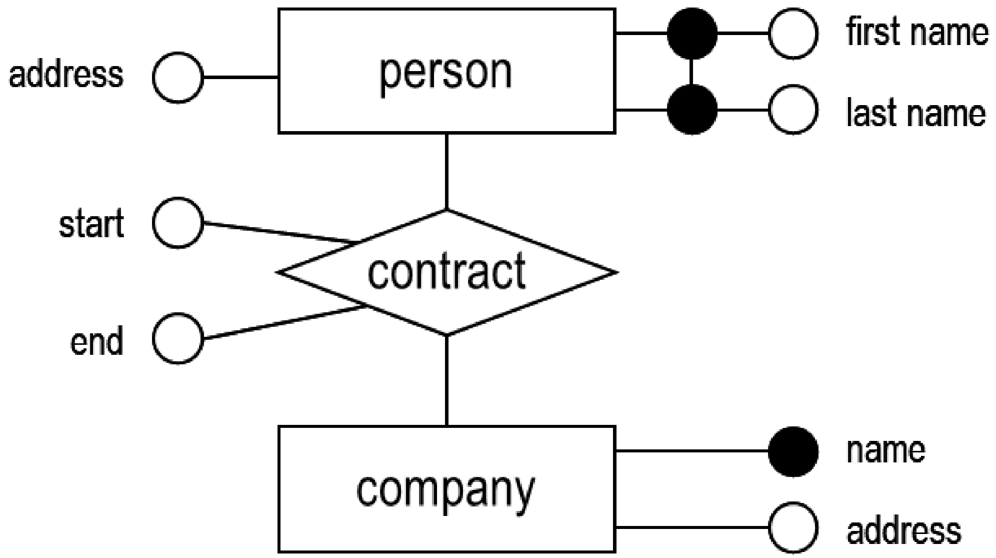
Schema Translation

- **Rules**
  - *Value Sets*
  - *Entity Sets*
  - ***Relationship Sets***
- Exceptions
- Limitations

Rules

Relationship Sets

```
CREATE TABLE contract(
  start DATE NOT NULL,
  end DATE NOT NULL,
  first_name VARCHAR(32),
  last_name VARCHAR(32),
  name VARCHAR(64),
  PRIMARY KEY (first_name, last_name, name),
  FOREIGN KEY (first_name , last_name)
    REFERENCES person(first_name, last_name),
  FOREIGN KEY (name) REFERENCES company(name)
);
```



Note

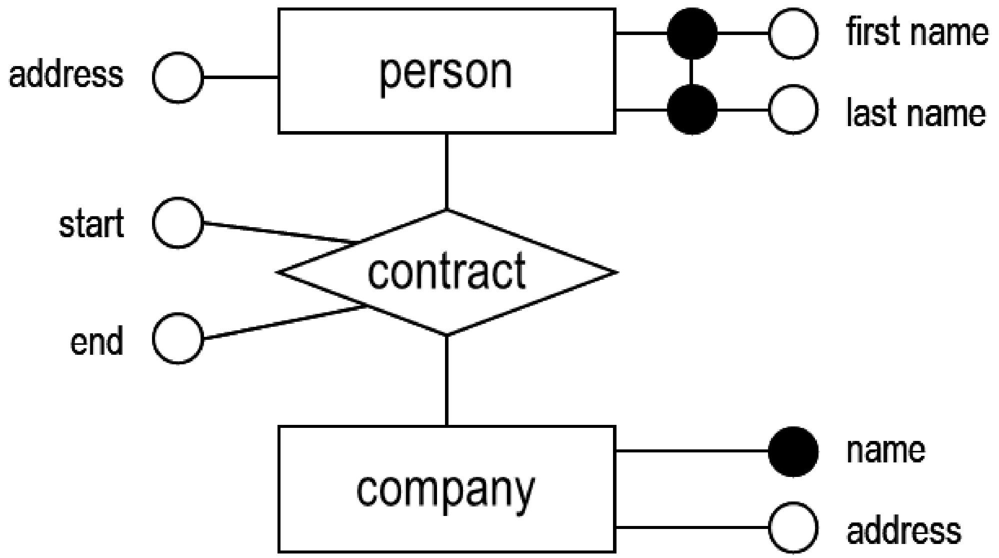
The attributes of the relationship set.

Schema Translation

Rules

Relationship Sets

```
CREATE TABLE contract(
  start DATE NOT NULL,
  end DATE NOT NULL,
  first_name VARCHAR(32),
  last_name VARCHAR(32),
  name VARCHAR(64),
  PRIMARY KEY (first_name, last_name, name),
  FOREIGN KEY (first_name , last_name)
    REFERENCES person(first_name, last_name),
  FOREIGN KEY (name) REFERENCES company(name)
);
```



Note

The keys of the participating entity sets.

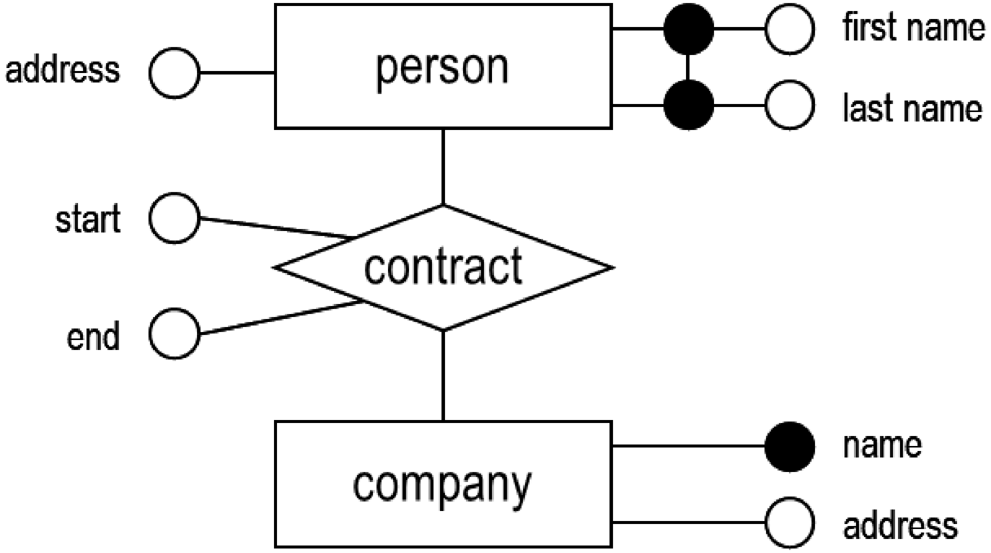
Schema Translation

- **Rules**
  - *Value Sets*
  - *Entity Sets*
  - ***Relationship Sets***
- Exceptions
- Limitations

Rules

Relationship Sets

```
CREATE TABLE contract(
  start DATE NOT NULL,
  end DATE NOT NULL,
  first_name VARCHAR(32),
  last_name VARCHAR(32),
  name VARCHAR(64),
  PRIMARY KEY (first_name, last_name, name),
  FOREIGN KEY (first_name , last_name)
    REFERENCES person(first_name, last_name),
  FOREIGN KEY (name) REFERENCES company(name)
);
```



Note

Reference to the participating entity sets.

Schema Translation

- **Rules**
  - *Value Sets*
  - *Entity Sets*
  - ***Relationship Sets***
- Exceptions
- Limitations

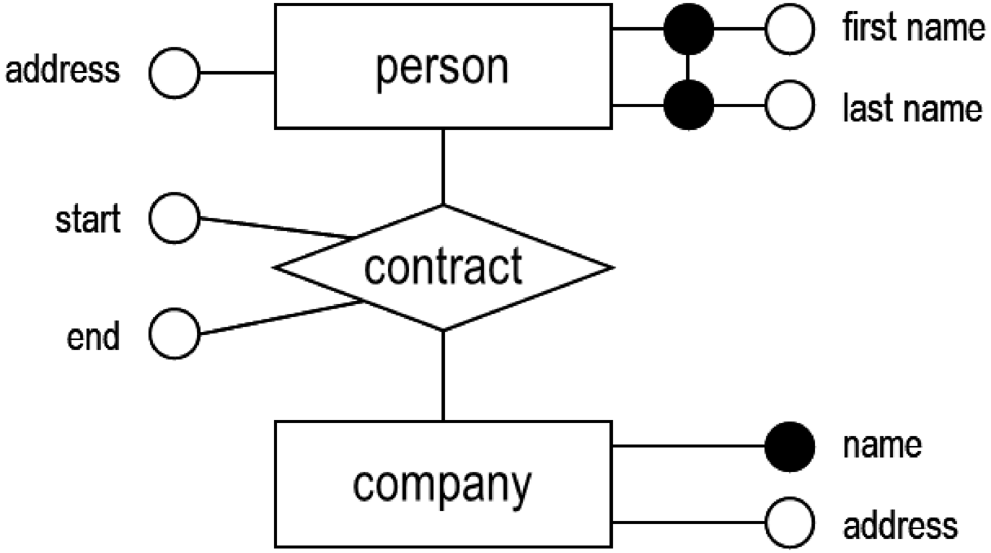
Rules

Relationship Sets

```sql
CREATE TABLE contract(
  start DATE NOT NULL,
  end DATE NOT NULL,
  first_name VARCHAR(32),
  last_name VARCHAR(32),
  name VARCHAR(64),
  PRIMARY KEY (first_name, last_name, name),
  FOREIGN KEY (first_name , last_name)
    REFERENCES person(first_name, last_name),
  FOREIGN KEY (name) REFERENCES company(name)
);
```

Note

The keys are the keys of the participating entity sets.



Schema Translation

Exceptions

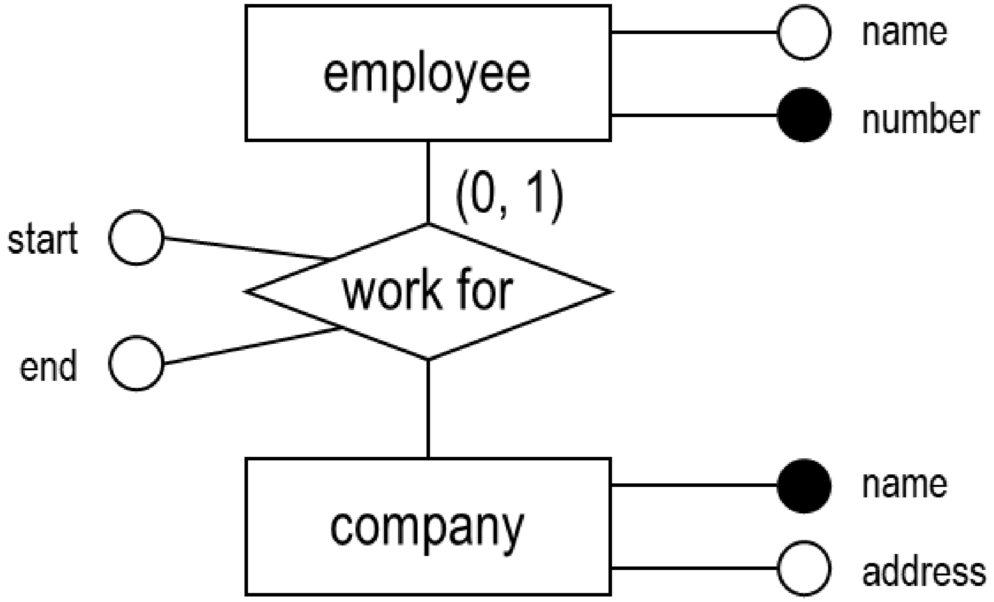
One-to-Many

Incorrect

```sql
CREATE TABLE work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8),
  cname VARCHAR(32),
  PRIMARY KEY (enumber, cname),
  FOREIGN KEY (enumber)
    REFERENCES employee(number),
  FOREIGN KEY (cname)
    REFERENCES company(name)
);
```

Issue

We can have an employee working for two different companies.



Schema Translation

Rules

❱ Exceptions

One-to-Many

(1, 1)

Weak Entity

Limitations

Exceptions

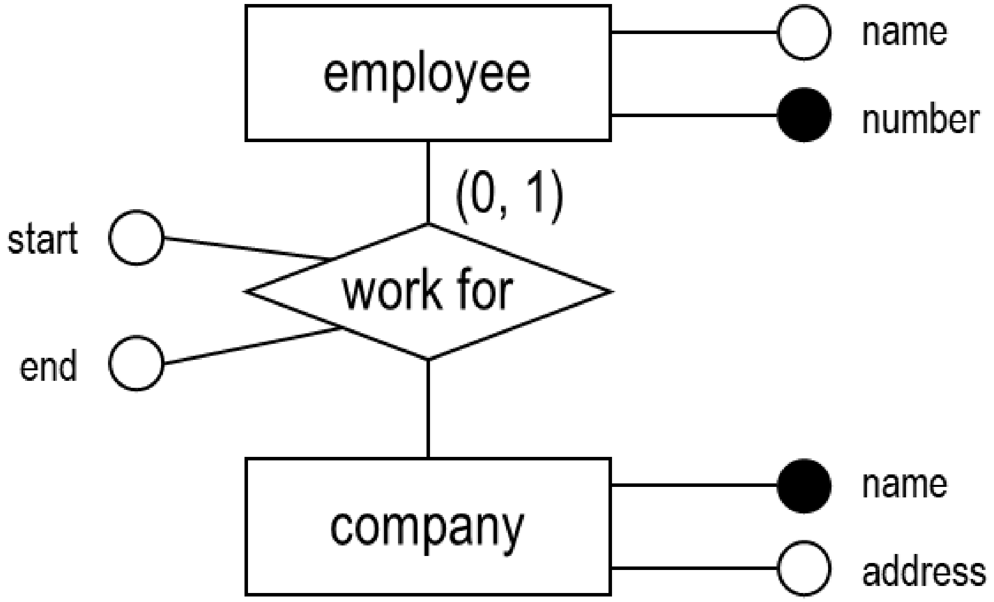
One-to-Many

Correct

```sql
CREATE TABLE work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8) PRIMARY KEY,
  cname VARCHAR(32) NOT NULL,
  -- NOT NULL to ensure existence
  FOREIGN KEY (enumber)
    REFERENCES employee(number),
  FOREIGN KEY (cname)
    REFERENCES company(name)
);
```

Correction

We restrict the primary key to the entity set with `(0, 1)` cardinality to avoid the issue.



Schema Translation

Rules
❱ Exceptions
One-to-Many
(1, 1)
Weak Entity
Limitations

Exceptions

(1, 1)

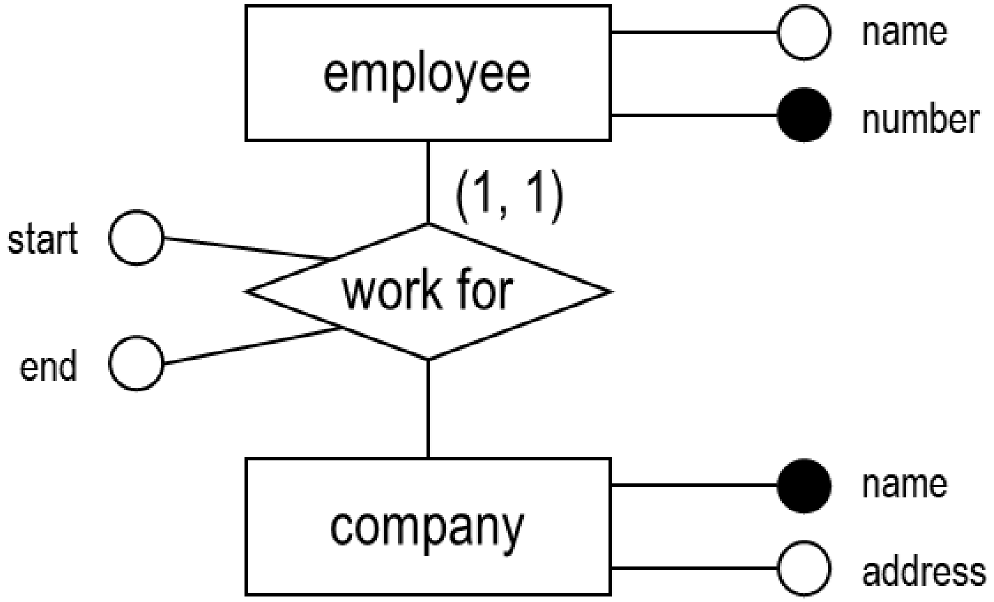
Incorrect

```
CREATE TABLE work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8) PRIMARY KEY,
  cname VARCHAR(32) NOT NULL,

  FOREIGN KEY (enumber)
    REFERENCES employee(number),
  FOREIGN KEY (cname)
    REFERENCES company(name)
);
```

Issue

We can insert into `employee` but forgot to insert into `work_for`, violating the **minimum**.



Schema Translation

- Rules
- ❯ Exceptions
  - *One-to-Many*
  - *(1, 1)*
  - *Weak Entity*
- Limitations

Exceptions

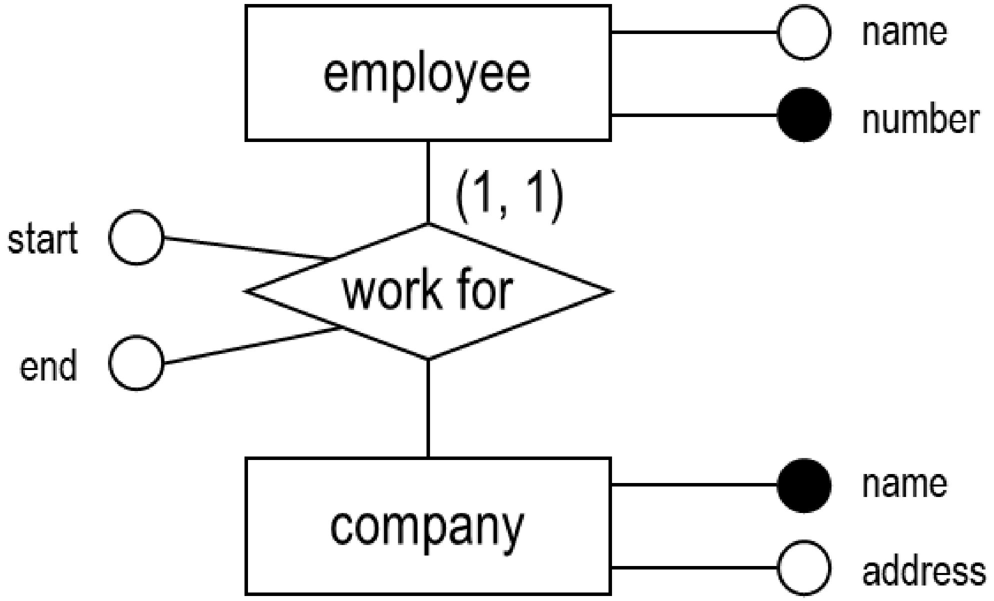
(1, 1)

Correct

```
CREATE TABLE employee_work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8) PRIMARY KEY,
  ename CHAR(32) NOT NULL,
  cname VARCHAR(32) NOT NULL,
  FOREIGN KEY (cname)
    REFERENCES company(name)
);
-- After merging, we choose a good
--   name (e.g., mix of both)
```

Correction

We merge `employee` and `work_for` so all employees **must** work for **at least one** company.



Schema Translation

Exceptions

Weak Entity

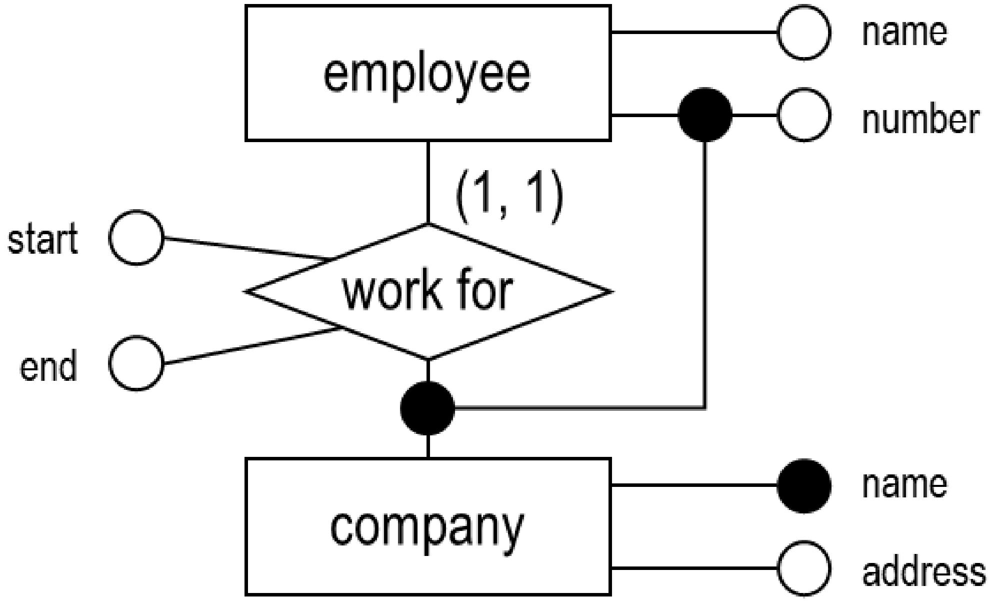
Incorrect

```
CREATE TABLE employee_work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8) PRIMARY KEY,
  ename CHAR(32) NOT NULL,
  cname VARCHAR(32) NOT NULL,

  FOREIGN KEY (cname)
    REFERENCES company(name)
);
```

Issue

This is only a **partial key**, it should **NOT** uniquely identify the entity.



Schema Translation

Exceptions

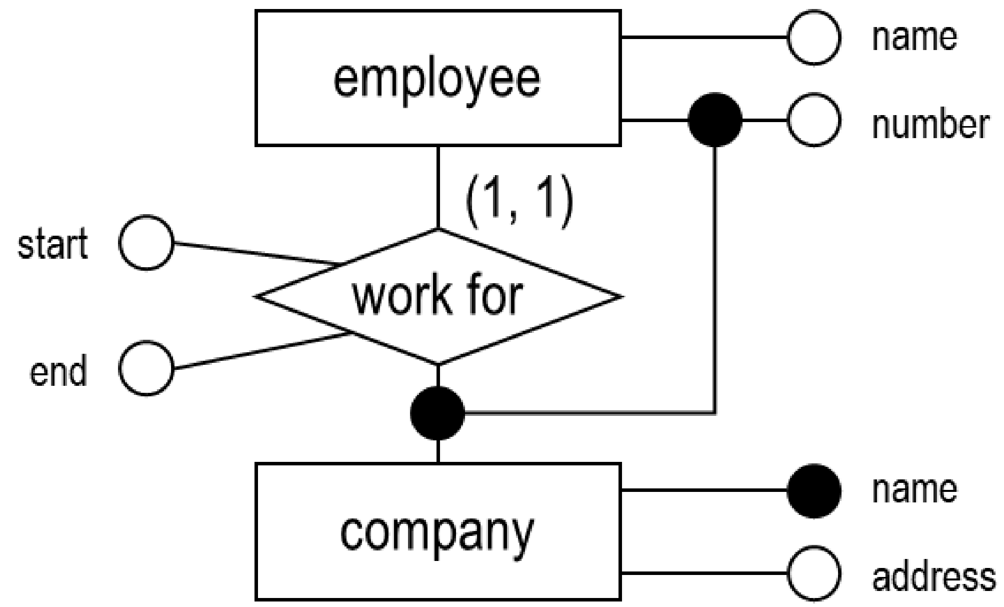
Weak Entity

Correct

```
CREATE TABLE employee_work_for (
  start DATE NOT NULL,
  end DATE NOT NULL,
  enumber CHAR(8),
  ename CHAR(32) NOT NULL,
  cname VARCHAR(32) NOT NULL,
  PRIMARY KEY (enumber, cname),
  FOREIGN KEY (cname)
    REFERENCES company(name)
);
```

Correction

We add the **primary key** of the **dominant entity set**. We still need to **merge** the table.



Schema Translation

Limitations

Other?

Inability to Translate

Our translation scheme is **simple** *(3 rules + 3 exceptions)*. However, it cannot translate other cardinalities *(e.g., `(1, n)`)*.

In such cases, think about the following constraints and enforce **as much as possible**.

- Ensure that **identities can uniquely identify** the entity set.
- Ensure that **minimum cardinality** is satisfied.
- Ensure that **maximum cardinality** is satisfied.

Not all constraints can be enforced, so enforce as much as possible.

```
postgres=# exit
Press any key to continue . . .
```