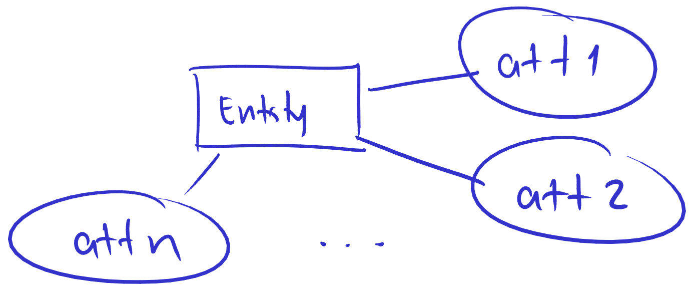
High Level Database Models
Chapter 4

Entity / Relationship Model (E/R)

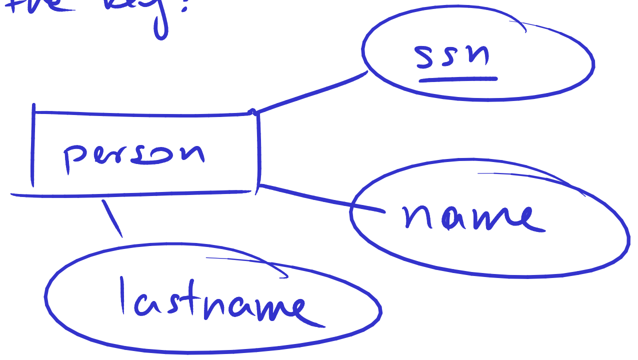
2 parts

1) Entity.

An entity has at least one attribute

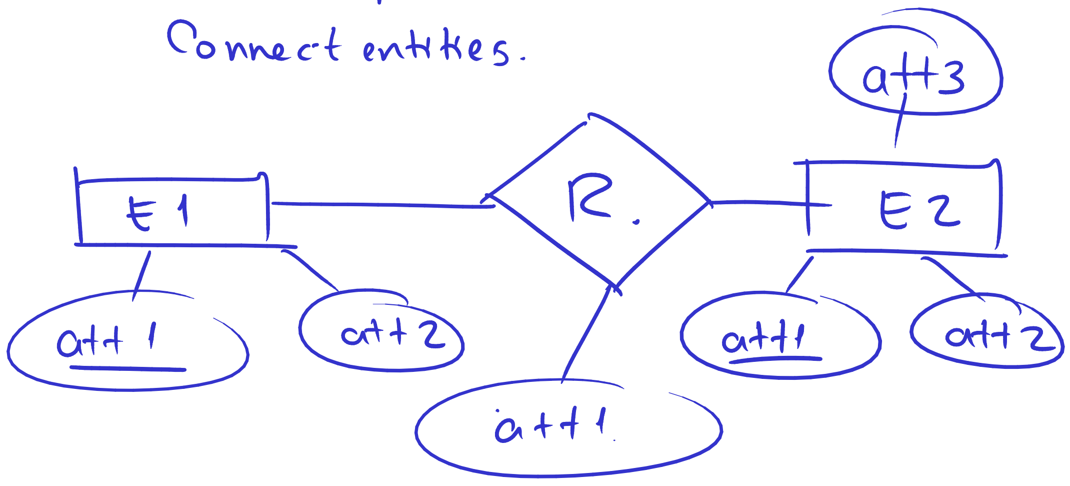


Underscore attributes that are part of the key:



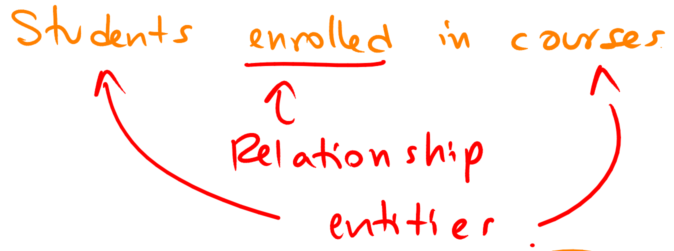
2) Relationships

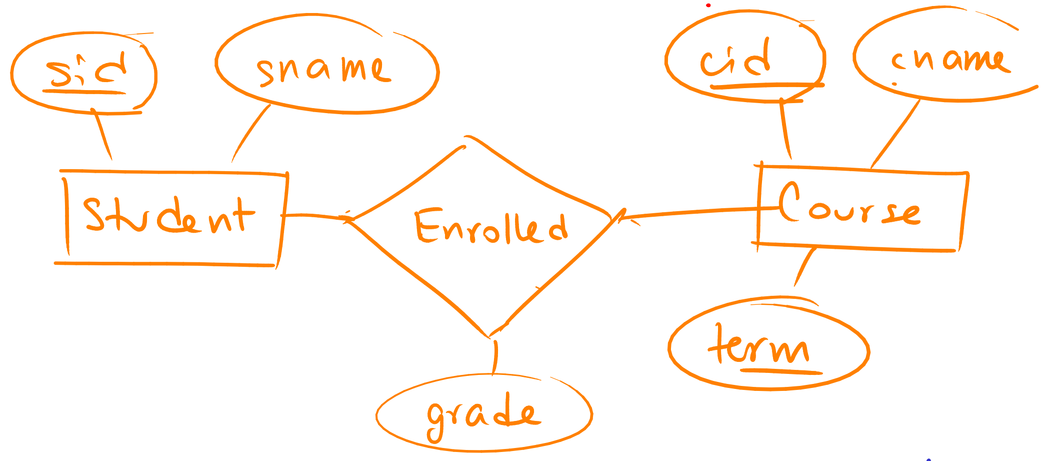
Connect entities.



Relationships can have attributes.

Ex: Students enrolled in courses





One entity relates to any number of entities via a relationship.

• Each Dependent has exactly one employee associated with it.

• If employee does not exist we don't care for her/his dependents.

```
CREATE TABLE Dependents (
    eid    CHAR(10),
    dname  CHAR(30),
    age    INTEGER,
    PRIMARY KEY (eid, dname),
    FOREIGN KEY (eid) REFERENCES
        Employees ON DELETE CASCADE
);
```

if referenced employee is deleted, then Dependents are deleted too !!

• More on this later.

Ex 2:

See Figure 4.2.2 for a Contracts entity as a weak entity

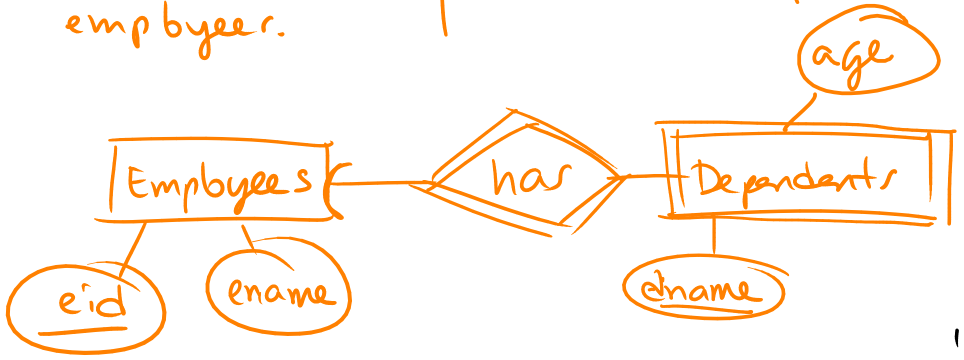
```
CREATE TABLE Cartoons (
    title CHAR(30),
    year INTEGER,
    PRIMARY KEY (title, year),
    FOREIGN KEY (title, year) REFERENCES
      Movies
);

CREATE TABLE Voices (
   ... as usual but reference Cartoons ...
```

Weak Entities (4.4)

Some times an entity that do not have an identifying attribute of their own.

- We need another entity to properly identify them

Ex: Employees and their dependent.
We do not care for dependents of non-employees.



14

dname does not need to be uniqe in Dep.

Both entities and relations become each a SQL relation.

- Entities are simply SQL relations

Ex:
```
CREATE TABLE Student (
    sid CHAR(10),
    sname VARCHAR
    PRIMARY KEY (sid)
);

CREATE TABLE Course (
    cid CHAR(10),
    cname VARCHAR,
    term char(3)
    PRIMARY KEY (cid, term)
);
```

Relationships

Their attributes are

- the Primary keys of its participating relations
- their own attributes

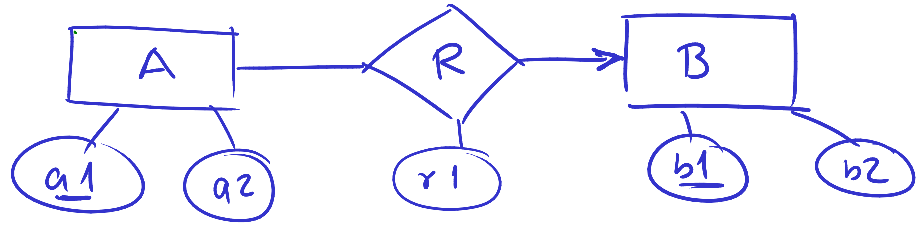
Their primary key is the attributes in the PKs of the participating relations.

```
CREATE TABLE Enrolled (
    sid    CHAR(10),
    cid    CHAR(10),
    term   CHAR(3),
    grade  INTEGER,
    PRIMARY KEY (sid, cid, term),
    FOREIGN KEY (sid) REFERENCES
        Students,
    FOREING KEY (cid, term) REFERENCES
        Courses
);
```

FOREIGN KEY constraint guarantees that we only keep in Enrolled students and courses that exist (More on that later)

Participation Constraints (4.1.6)

An entity relates to 0 ~~or~~ 1 entity

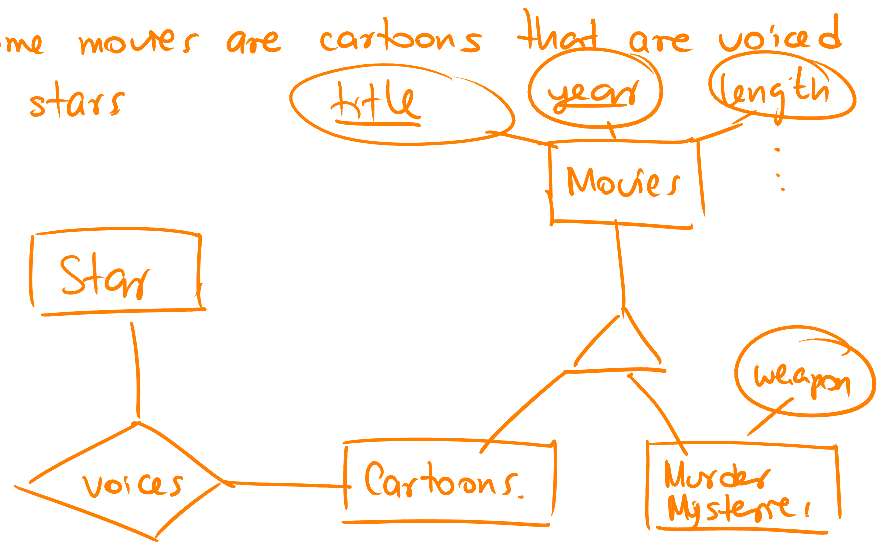


In this example R(a1, b1, r1)

Arrow in diagram implies $a1 \rightarrow b1, r1$

Ex:

Some movies are cartoons that are voiced by stars



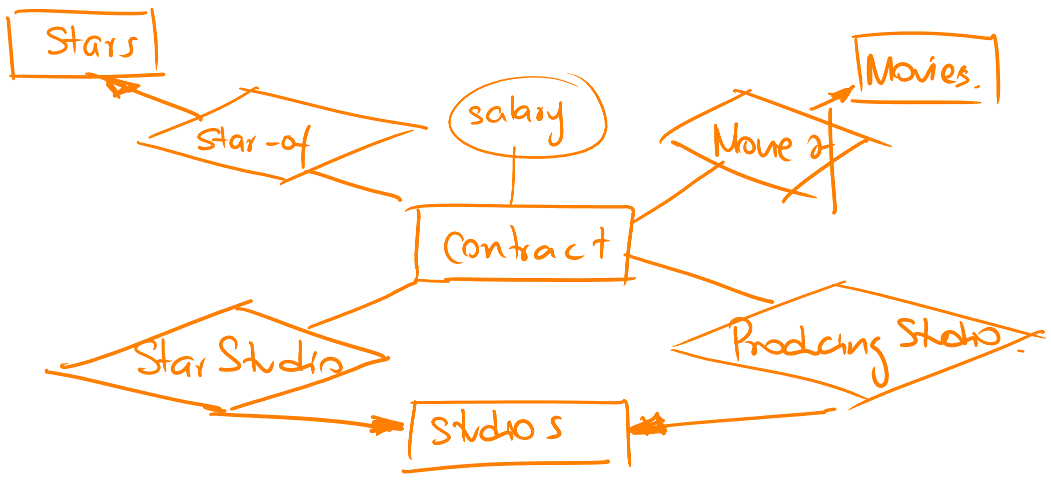
To convert to relations

- create relation of main entity
- each sub-entity has the same PK that main entity. plus any extra attributes.

Ignore 4.6.1 in textbook
Use only 4.6.2

Ex:

```
CREATE TABLE Movies ( . . .
      . . as usual . . .
);
CREATE TABLE MurderMysteries (
    title    CHAR(30),
    year     INTEGER,
    weapon   VARCHAR,
    PRIMARY KEY (title, year),
    FOREIGN KEY (title, year) REFERENCES
        Movies
);
```



The arrows imply that for every contract there is ∅ or 1 participating entity

They could be further constraint to be exactly 1.

Inheritance (4.1.11)

- Some type some entities in an entity set have special properties (extra attributes)

or

- Only a subset of entities is involved in a relationship

In SQL Assume attr are integer

```
CREATE TABLE R(
   a1 integer,
   b1 integer NOT NULL,
   r1 integer,
   PRIMARY KEY (a1)
   FOREIGN KEY(a1) REFERENCES A,
   FOREIGN KEY(b1) REFERENCES B
);
```

must not be empty.

A(a1, a2) $a1 \rightarrow a2$

R(a1, b1, r1) $a1 \rightarrow b1, r1$

Hence we can combine A and R

AR(a1, a2, b1, r1) $a_1 \rightarrow a2, b1, r1$

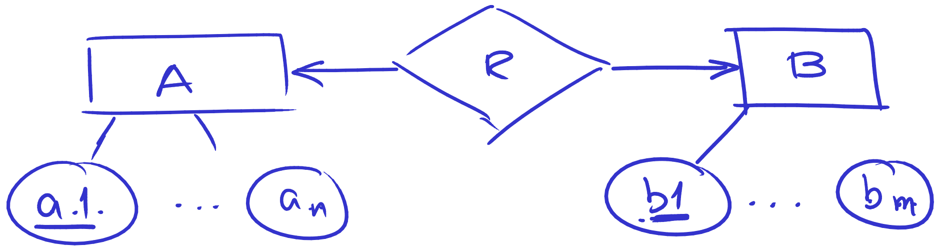
Instead of 2 relations we create one

```
CREATE TABLE AR(
   a1 integer,
   a2 integer,
   b1 integer,
   r1 integer,
   PRIMARY KEY (a1),
   FOREIGN KEY(b1) REFERENCES B
);
```

can be NULL (empty).

Primary keys can <u>never</u> be NULL.

We can have:



It means $R(a1, b1)$
has FD $a1 \to b1$, $b1 \to a1$

Can we merge R with A or B?

Say we choose A; so we create AR as above. This guarantees $a1 \to b1$.

But what about $b1 \to a1$?

b1 is also a CK for AR

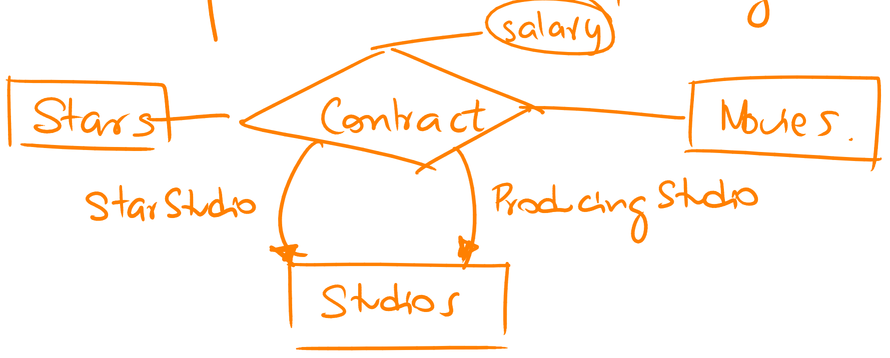
Make b1 unique: and add to AR:

```
:
UNIQUE (b1)
:
```

or if key of B is one attribute add it after its declaration:

Ex. 2:

Stars work on a movie, but now there is a studio of the star and the producing studio.



This implies:

Star, Movie $\to$ StarStudio
Star, Movie $\to$ Producing Studio

Often binary relationships are preferred:

To convert a n-way relationship to binary

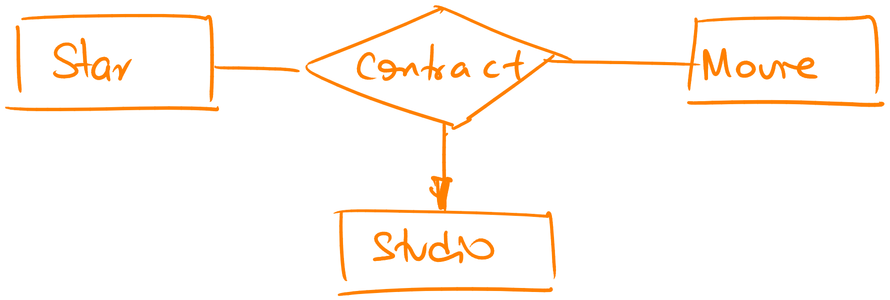
- convert relationship to entity.
- give it an primary key (perhaps artificial)
- Create a relationship between new entity and old entity.
  - many-to-one
    newEntity $\to$ entity1, entity2 ...

Multi way relationships

- Relationships can have 2 or more participating entities.
- Same type of participating constraints as with binary relationships.
- PK of relationship is the union of PKs of participating entities.

Ex: Ternary

A star has a contract with a studio to work on a movie.

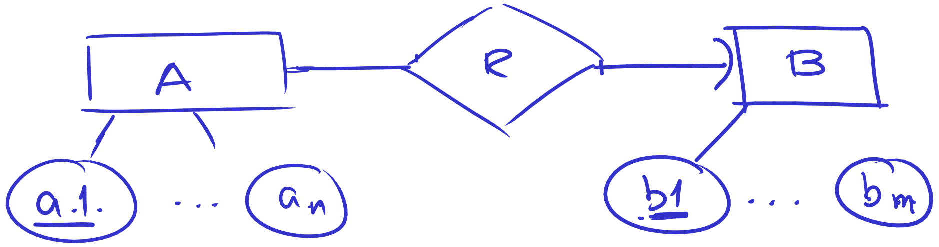


Star, Movie → Studio

(Not showing attributes of entities for simplicity)

⋮
b1 integer UNIQUE .
⋮

An entity relates to exactly <u>one</u> entity only



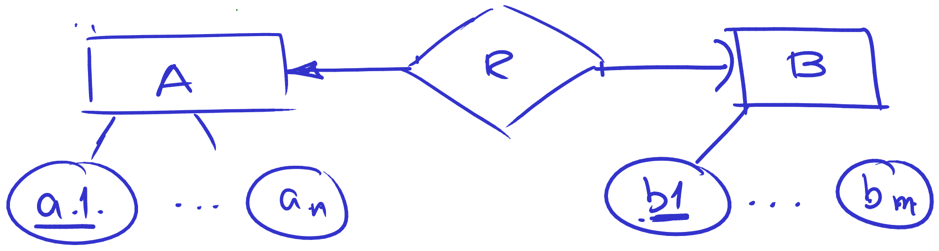
R(a1, b1) still a1 → b1

and ∀ value a1 in A: ∃ at most one corresponding value b1 in B.
(zero or one)

SQL: same schema as AR above, but b1 <u>can not</u> be NULL:

⋮
b1 integer NOT NULL
⋮

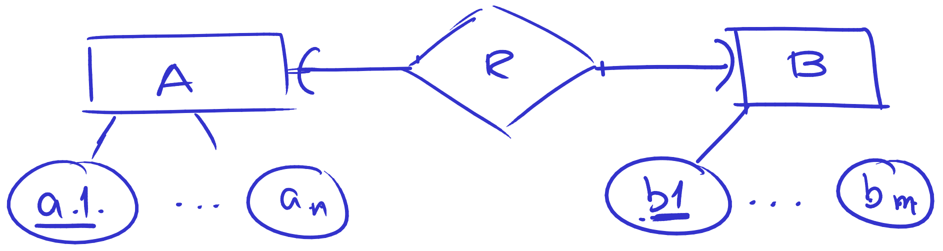
Some Combinations



$a1 \rightarrow b1 \quad b1 \rightarrow a1$

$\forall$ values of a1 $\Rightarrow \exists$ a value of b1.

Create AR, make key of B in AR unique and not NULL.



$a1 \rightarrow b1, \quad b1 \rightarrow a1$

$\forall$ value of a1 $\Rightarrow \exists$ value of b1

$\forall$ value of b1 $\Rightarrow \exists$ vale of a1

$\Rightarrow |A| = |B|$

tuples in A # tuples in B

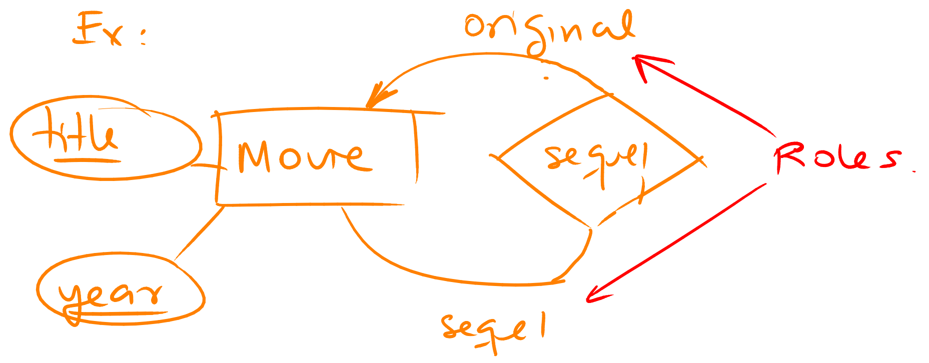
Make A, B and R one relation

key? a1 or b1, make the other unique, not null.

Roles

Sometimes an entity participates more than once in a relationship:

Ex:



sequelTitle, sequelYear $\rightarrow$ originalTitle, originalYear

The name of the role allows to identify each of the two entities involved in the relationship.

Useful to name attributes of relationship.