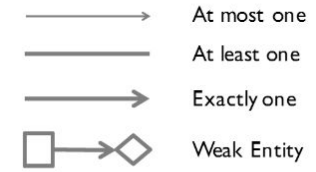
Entity: Object distinguishable from other objects of the same 'type', described as set of attributes and their values

Entity Set: Collection of entities with same attributes; **Attribute**: Information that describes the entity;
Attribute domain: Range of permissible values (e.g. int 1 - 20)

Key: Minimal set of attributes that uniquely identify an entity. **Overlap / Covering constraints**;

Candidate key : A unique key; **Primary key** : A designated unique identifier;
Relationship: Association between 2 or more entities. **Relationship set**: Collection of similar relationships
Key constraints: One-to-many / Many-to-one; **Participation Constraints**: at least one = total;
Weak entity: Can only be uniquely identified by using primary key of its owner;
Aggregation: Relationships between (entities - relationships); **Partial key**: Attributes that identify the weak entity, if given owning entity.
Data redundancy: Same data is stored in two separate places; **Physical/Logical Data Independence**; **High level language**;
Database: A set of relations; **Relation**: A table with rows and columns; **Schema**: Name of relation + name & type of each column;
Instance: Specific set of rows; **Integrity Constraints**: A condition that is true for any instance.
An instance is legal if all ICs are satisfied. **Domain Constraints**: (attr types)
Candidate Keys: Distinct, and minimal; **Superkey**: Distinct, but not minimal;
If >1 candidate keys in relation, admin needs to assign a primary key. SQL: Structured Query Lang.
Foreign Keys: set of fields in Relation R1 used to refer to tuple in R2 via R2's primary key.



Formal Name	Synonyms
Relation	Table
Tuple	Row, Record
Attribute	Column, Field
Domain	Type
Cardinality	# of tuple
Degree	# of attributes

Referential Integrity

A database instance has *referential integrity* if all foreign key constraints are enforced no dangling references

Examples where referential integrity is not enforced
- HTML links
- Yellow page listing
- Restaurant menus
- Some relational databases!

How to Enforce Integrity Constraints

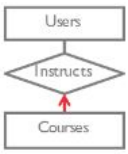
Run checks anytime database changes

On INSERT
- what if new Enrolled tuple refers to non-existent student?
- Reject insertion

On DELETE (many options)
- what if Students tuple is deleted?
  - delete dependent Enrolled tuples
  - reject deletion
  - set Enrolled.sid to default value or null
  - (null means 'unknown' or 'inapplicable' in SQL)

At Most One → Relation

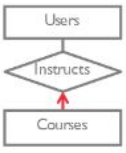
Add relationship attributes (none here)
Add keys for entity set as foreign keys
What is the primary key?




```
CREATE TABLE Instructs(
    uid int,
    cid int,
    PRIMARY KEY (cid),
    FOREIGN KEY (uid) REFERENCES Users,
    FOREIGN KEY (cid) REFERENCES Courses
)
```

At Most One: Combine?

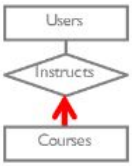
Zero or 1 courses, cid is primary key; Similar to ???
Combine Instructs attributes into Courses (preferred)
How to represent courses without instructor?
NULL uid (and other Instructs attributes)




```
CREATE TABLE Course_Instructs(
    cid int
    uid int,  Maybe rename?
    name text,
    loc text,
    PRIMARY KEY (cid),
    FOREIGN KEY (uid) REFERENCES Users
)
```

Exactly One Constraint → Relation

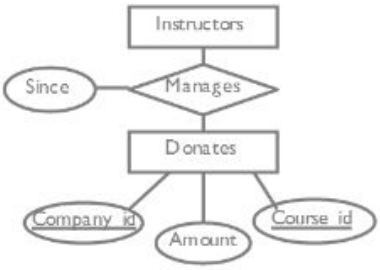
Represent: Course must have Instructor?
Combine relationship into Courses + NOT NULL
What happens if we delete User who is Instructor?
Default ON DELETE NO ACTION: Don't permit




```
CREATE TABLE Course_Instructs(
    cid int
    uid int, NOT NULL,
    name text,
    loc text,
    PRIMARY KEY (cid),
    FOREIGN KEY (uid) REFERENCES Users
)    ON DELETE NO ACTION
)
```

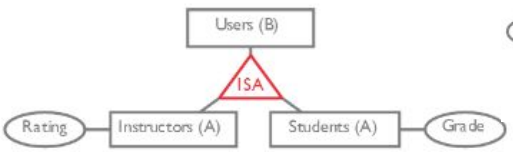
Aggregation

Convert the aggregated relationship into an entity
Convert like any other entity
E.g. Donates: PRIMARY KEY (Company_id, Course_id)
Manages: References this key



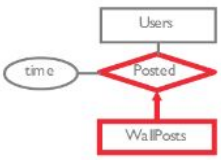
ISA Hierarchies

```
CREATE TABLE Users(uid int, name text, PRIMARY KEY(uid))
CREATE TABLE Instructors(uid int, rating int,
    PRIMARY KEY(uid), FOREIGN KEY (uid) REFERENCES Users)
CREATE TABLE Students(uid int, grade char(2),
    PRIMARY KEY(uid), FOREIGN KEY (uid) REFERENCES Users)
```



Weak Entity → Relation

Weak entity set and identifying relationship set are translated into a single table.
When the owner entity is deleted, all owned weak entities must also be deleted.




```
CREATE TABLE Wall_Posted(
    uid int,
    post_text text,
    posted_time DATE
    PRIMARY KEY (uid, posted_time),
    FOREIGN KEY (uid) REFERENCES Users
        ON DELETE CASCADE
)
```