37.1 Entities, relationships, and attributes

The entity-relationship model

Database design begins with verbal or written requirements for the database. Requirements are formalized as an entity-relationship model and then implemented in SQL.

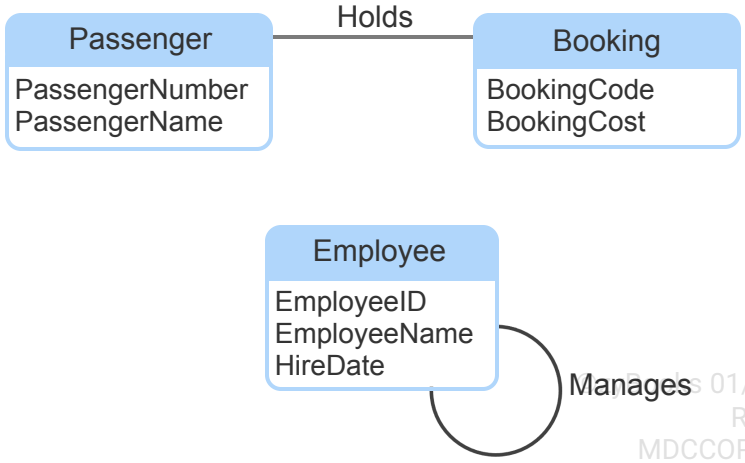
An ***entity-relationship model*** is a high-level representation of data requirements, ignoring implementation details. An entity-relationship model guides implementation in a particular database system, such as MySQL.

An entity-relationship model includes three kinds of objects:

- An ***entity*** is a person, place, product, concept, or activity.
- A ***relationship*** is a statement about two entities.
- An ***attribute*** is a descriptive property of an entity.

A relationship is usually a statement about two different entities, but the two entities may be the same. A ***reflexive relationship*** relates an entity to itself.

PARTICIPATION ACTIVITY 37.1.1: Entities, relationships, and attributes.



Animation content:

Step 1: In an airline reservation system, Passenger and Booking are entities. There are two entities named Passenger and Booking. An entity is represented by a colored box with the name inside.

Step 2: Holds is a relationship between Passenger and Booking. A line named Holds connects the two entities.

Step 3: PassengerNumber, PassengerName, BookingCode, BookingCost are attributes. Two rows are added to the Passenger entity as attributes with values Passenger Number and Passenger Name. Two rows are added to the Booking entity as attributes with values Booking Code and Booking Cost.

Step 4: In a human resources systems, Employee-Manages-Employee is a reflexive relationship. An entity Employee appears, with attributes EmployeeID, EmployeeName, and HireDate. A line named Manages connects Employee to itself.

Animation captions:

1. In an airline reservation system, Passenger and Booking are entities.
2. Holds is a relationship between Passenger and Booking.
3. PassengerNumber, PassengerName, BookingCode, BookingCost are attributes.
4. In a human resources systems, Employee-Manages-Employee is a reflexive relationship.

When the model is implemented in SQL, entities typically become tables. Relationships and attributes typically become foreign keys and columns, respectively. However, some relationships and attributes become tables. Since the conversion is indirect, requirements are documented as entities, relationships, and attributes rather than tables, keys, and columns.

Terminology

Attribute *is used in both entity-relationship and relational models. In the relational model, attribute is a formal term for column. Since entity-relationship attributes typically become relational columns, the meaning of attribute is similar in both models.*

PARTICIPATION ACTIVITY 37.1.2: Entities, relationships, and attributes.

How are the following requirements represented in an entity-relationship model?

The Widgets and Digits company has projects that are composed of multiple tasks. Projects have a limited duration, and tasks have a starting date.

1) A project is a(n) _____.

Check Show answer

2) The duration of a project is a(n) _____.

Check Show answer

3) "Task belongs to project" is a(n) _____.

Check Show answer

4) The starting date of a task is a(n) _____.

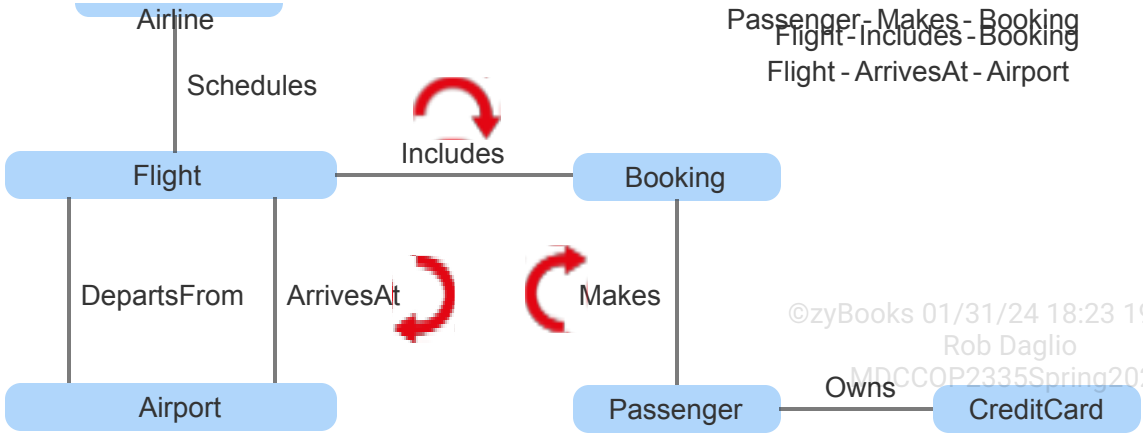
Check Show answer

Entity-relationship diagram and glossary

Entities, relationships, and attributes are depicted in an ***entity-relationship diagram***, commonly called an ***ER diagram***. An ER diagram depicts entities as rectangles with rounded corners, relationships as lines connecting rectangles, and attributes within entity rectangles. ER diagrams always include entities and relationships. Attributes are optional and only appear when additional detail is needed.

The full name of a relationship includes the related entities and is written "Entity-Relationship-Entity". The full name is read clockwise around the center of the relationship, as illustrated in the animation below.

PARTICIPATION ACTIVITY 37.1.3: The clockwise rule.



Animation content:

Step 1: This ER diagram of an airline reservation system has entities and relationships but no attributes. There are six entities named Airline Flight Airport Booking Passenger and Credit Card. A line named Schedules connects entities Airline and Flight. Two lines named Departs From and Arrives At connect entities Flight and Airport. A line named Includes connects entities Flight and Booking. A line Makes connects entities Booking and Passenger. A line named Owns connects entities Passenger and Credit Card.

Step 2: The full relationship name Passenger-Makes-Booking is read clockwise, around the center of the relationship line. A red arrow pointing up going from entity Passenger to entity Booking appears next to line Makes. Text duplicates and moves above from line Makes and entities Booking and Passenger to create the text Passenger hyphen Makes hyphen Booking.

Step 3: All relationships are read following the clockwise rule. A red arrow pointing right going from entity Flight to entity Booking appears next to line Includes. Text duplicates and moves above from line Includes and entities Flight and Booking to create the text Flight hyphen Includes hyphen Booking. A red arrow pointing down going from entity Flight to entity Airport appears next to line Arrives At. Text duplicates and moves above from line Arrives At and entities Flight and Airport to create the text Flight hyphen Arrives At hyphen Airport.

Animation captions:

1. This ER diagram of an airline reservation system has entities and relationships but no attributes.
2. The full relationship name Passenger-Makes-Booking is read clockwise, around the center of the relationship line.
3. All relationships are read following the clockwise rule.

A ***glossary***, also known as a ***data dictionary*** or ***repository***, documents additional detail in text format. A glossary includes names, synonyms, and descriptions of entities, relationships, and attributes. For simple databases with few users, a database designer may record the glossary with a text editor. For more complex databases, the designer may use a database or software tool specifically designed for glossaries.

The ER diagram and glossary are complementary and, together, completely describe an entity-relationship model.

PARTICIPATION ACTIVITY 37.1.4: ER diagram and glossary.

1) Referring to the animation above, how is the "schedules" relationship read?
   - Flight-Schedules-Airline
   - Airline-Schedules-Flight
   - Flight-IsScheduledBy-Airline

2) An entity-relationship model is completely described by:
   - An ER diagram
   - A glossary
   - Both an ER diagram and a glossary

3) What is included in a glossary?
   - Descriptions only
   - Names and synonyms only
   - Names, synonyms, and descriptions only
   - Names, synonyms, descriptions, and ER diagrams

Types and instances

In entity-relationship modeling, a type is a set:

- An ***entity type*** is a set of things. Ex: All employees in a company.
- A ***relationship type*** is a set of related things. Ex: Employee-Manages-Department is a set of (employee, department) pairs, where the employee manages the department.
- An ***attribute type*** is a set of values. Ex: All employee salaries.

Entity, relationship, and attribute types usually become tables, foreign keys, and columns, respectively.

An instance is an element of a set:

- An ***entity instance*** is an individual thing. Ex: The employee Sam Snead.
- A ***relationship instance*** is a statement about entity instances. Ex: "Maria Rodriguez manages Sales."
- An ***attribute instance*** is an individual value. Ex: The salary $35,000.

Entity, relationship, and attribute instances usually become rows, foreign key values, and column values, respectively

Table 37.1.1: Example types and instances.

	Type	Instance
Entity	Passenger	Muhammed Ali
Attribute	BookingCode	39240
Relationship	Passenger-Holds-Booking	Muhammed Ali holds 39240

PARTICIPATION ACTIVITY 37.1.5: Types and instances.

Match the term to the example.

If unable to drag and drop, refresh the page.

attribute instance | relationship instance | entity instance | attribute type | entity type | relationship type

	Students at San Antonio Community College
	Eleanor Rigby, a student at San Antonio community college

	Students take exams
	Eleanor Rigby takes the final exam in calculus
	Student record number
	324A21

Reset

Database design

Complex databases are developed in three phases:

1. ***Analysis*** develops an entity-relationship model, capturing data requirements while ignoring implementation details.
2. ***Logical design*** converts the entity-relationship model into tables, columns, and keys for a particular database system.
3. ***Physical design*** adds indexes and specifies how tables are organized on storage media.

Analysis is particularly important for complex databases with many users when documenting requirements is challenging. For small databases with just a few tables and users, analysis is less important and often omitted.

Analysis and logical design steps are summarized in the table below. Although these steps are presented in sequence, in practice execution is not always sequential. Often an early step is revisited after a later step is completed.

Physical design is dependent on specific index and table structures, which vary greatly across relational databases. Physical design is discussed elsewhere in this material.

Table 37.1.2: Analysis steps.

Step	Name
1	Discover entities, relationships, and attributes
2	Determine cardinality
3	Distinguish strong and weak entities

4	Create supertype and subtype entities

Table 37.1.3: Logical design steps.

Step	Name
5	Implement entities
6	Implement relationships
7	Implement attributes
8	Apply normal form

PARTICIPATION ACTIVITY 37.1.6: Analysis and logical design.

1) Analysis considers implementation issues related to a specific database system.
   - True
   - False

2) An entity-relationship model is developed for all database design projects.
   - True
   - False

3) Entities, relationships, and attributes always map directly to tables, foreign keys, and columns, respectively.
   - True
   - False

37.2 Discovery

Discovery

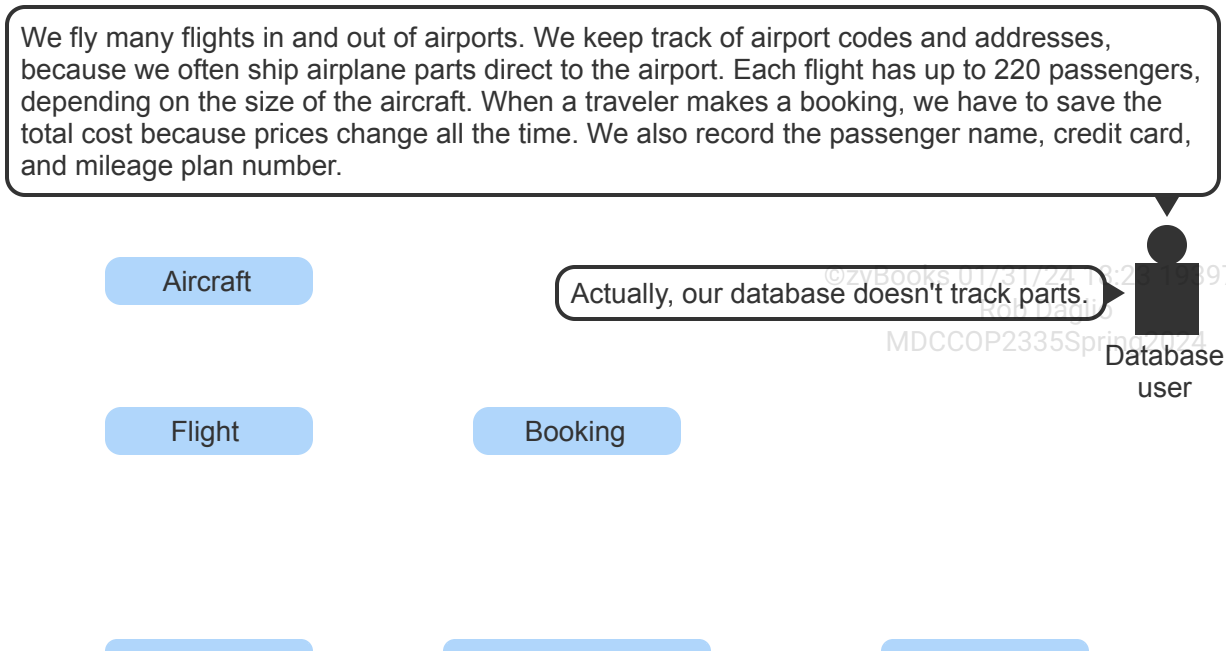
Database requirements are determined by interviewing the database users and managers. Users and managers are usually familiar with requirements from an old database, or perhaps a manual process with paper records. When users are difficult to reach, a database designer may communicate with surrogates. Ex: A sales representative might communicate on behalf of prospective customers.

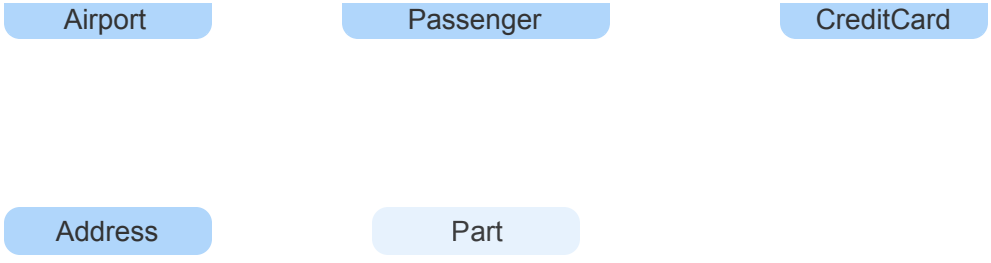
Entities, relationships, and attributes surface as nouns and verbs in an interview:

- Entities are usually nouns, but not all nouns are entities. Designers should ignore nouns that denote specific data or are not relevant to the database.
- Relationships are usually verbs. Designers should ignore statements that are not about entities, not relevant to the database, or redundant to other relationships. Designers should look for relationships that are not explicitly stated, since users may overlook important information.
- Attributes are usually nouns that denote specific data, such as names, dates, quantities, and monetary values.

In addition to interviews, written documents are a good source of data requirements. Ex: The user manual for an older version of the database is a good source of requirements.

PARTICIPATION ACTIVITY 37.2.1: Discover entities.





Animation content:

Step 1: In an interview with a database user, flight, airport, aircraft, passenger, and booking are entities. There is a database user who states We fly many flights in and out of airports. We keep track of airport codes and addresses, because we often ship airplane parts direct to the airport. Each flight has up to 220 passengers, depending on the size of the aircraft. When a traveler makes a booking, we have to save the total cost because prices change all the time. We also record the passenger name, credit card, and mileage plan number. From this statement keyword Aircraft Flight Airport Passenger and Booking are underlined and entities are created with these keywords.

Step 2: Airplane is another word for aircraft. Traveler is another word for passenger. Keywords airplane and traveler get underlined in the statement by the database user. Entities Aircraft and Passenger are boxed.

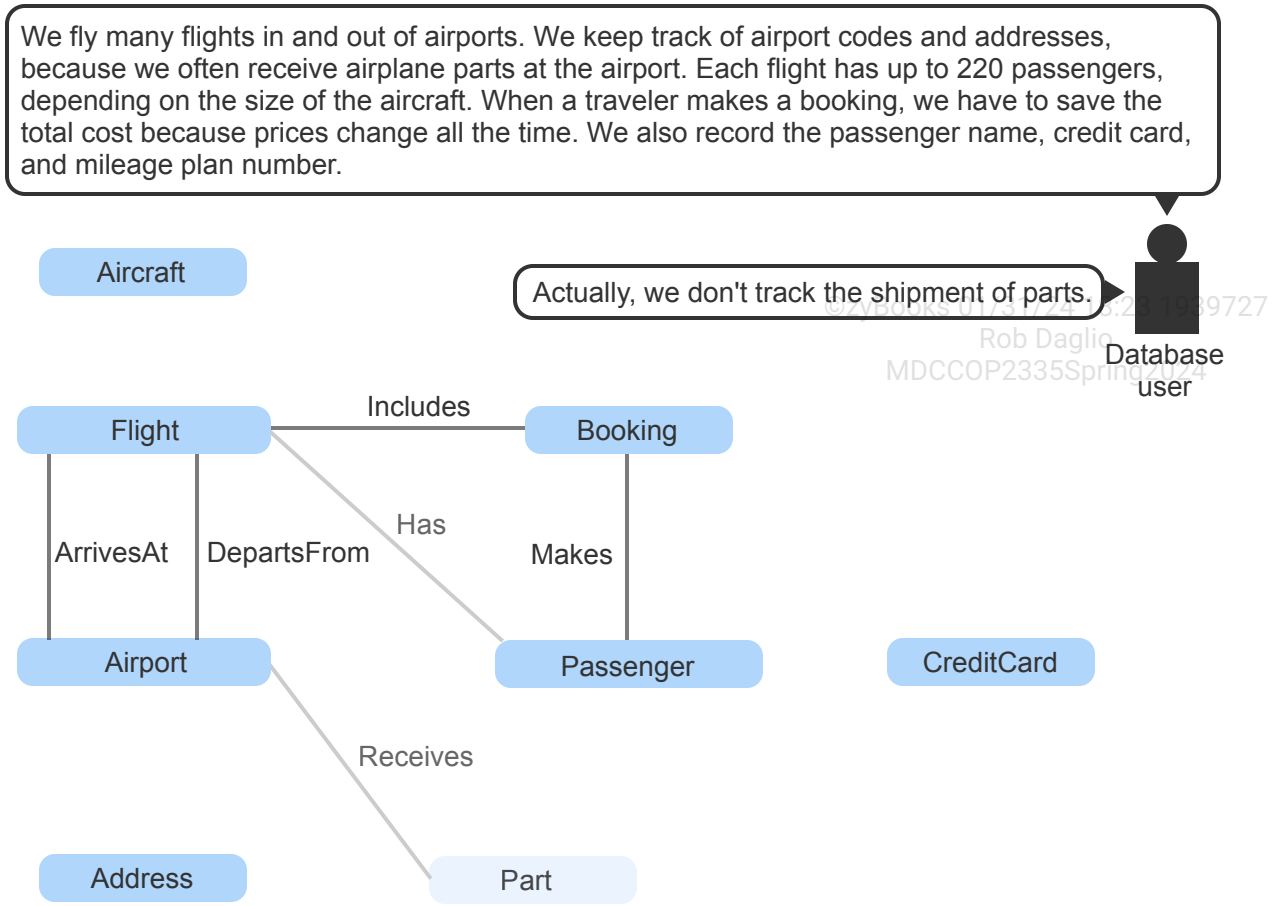
Step 3: Usually, address and credit card have many details and are entities. Keywords Address and Credit Card are underlined in the statement by the database user and entities are created for these keywords.

Step 4: Further interviews determine the database does not track parts. Part is not an entity. Keyword parts is underlined in the statement by the database user and an entity is created for this keyword but is then faded away.

Animation captions:

1. In an interview with a database user, flight, airport, aircraft, passenger, and booking are entities.
2. Airplane is another word for aircraft. Traveler is another word for passenger.
3. Usually, address and credit card have many details and are entities.
4. Further interviews determine the database does not track parts. Part is not an entity.

PARTICIPATION ACTIVITY 37.2.2: Discover relationships.



Animation content:

Step 1: Flight-ArrivesAt-Airport, Flight-DepartsFrom-Airport, and Passenger-Makes-Booking are relationships. There is a database user who states We fly many flights in and out of airports. We keep track of airport codes and addresses, because we often ship airplane parts direct to the airport. Each flight has up to 220 passengers, depending on the size of the aircraft. When a traveler makes a booking, we have to save the total cost because prices change all the time. We also record the passenger name, credit card, and mileage plan number. There are also eight entities named Aircraft, Flight, Airport, Address, Booking, Passenger, Part and Credit Card. Keywords fly, in, out, and makes are underlined in the statement by the database user. Two lines named Arrives At and Departs From connect entities Flight and Airport. A line named Makes connects entities Booking and Passenger.

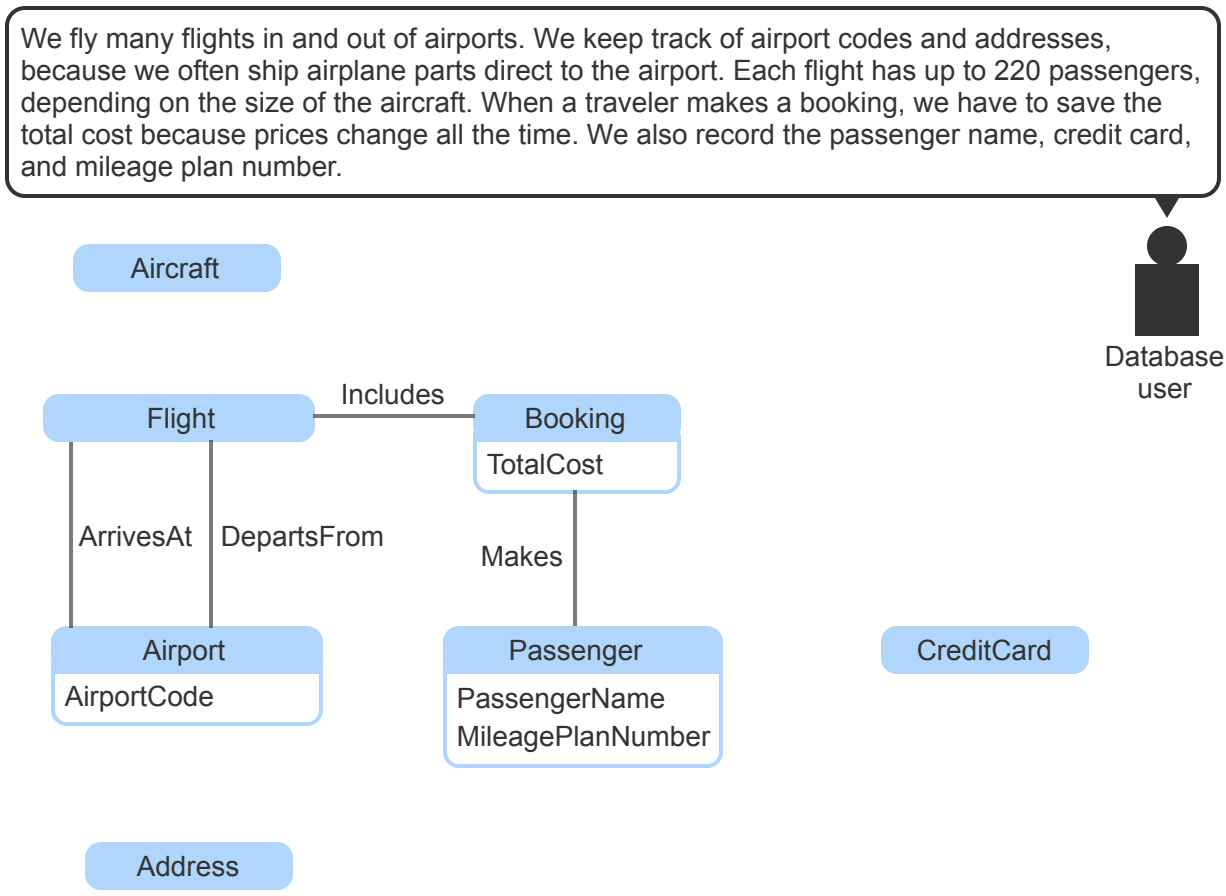
Step 2: Passenger can be determined from booking. Replace Flight-Has-Passenger with Flight-Includes-Booking. Keyword Has is underlined in the statement by the database user. A line named Has connects entities Flight and Passenger. A line named Includes connects entities Flight and Booking and causes the line Has to fade.

Step 3: Further interviews determine the database does not track parts. Airport-Receives-Part is not a relationship. Keyword ship is underlined in the statement by the database user. A line named Shipped To connects entities Airport and Part and then the line Shipped To and entity Part fades.

Animation captions:

1. Flight-ArrivesAt-Airport, Flight-DepartsFrom-Airport, and Passenger-Makes-Booking are relationships.
2. Passenger can be determined from booking. Replace Flight-Has-Passenger with Flight-Includes-Booking.
3. Further interviews determine the database does not track parts. Airport-Receives-Part is not a relationship.

PARTICIPATION ACTIVITY 37.2.3: Discover attributes.



Animation content:

Step 1: Airport code, total cost, and name are attributes of Airport, Booking, and Passenger. There is a database user who states We fly many flights in and out of airports. We keep track of airport codes and addresses, because we often ship airplane parts direct to the airport. Each flight has up

to 220 passengers, depending on the size of the aircraft. When a traveler makes a booking, we have to save the total cost because prices change all the time. We also record the passenger name, credit card, and mileage plan number. There are also seven entities named Aircraft, Flight, Airport, Address, Booking, Passenger, and Credit Card. Two lines named Arrives At and Departs From connect entities Flight and Airport. A line named Makes connects entities Booking and Passenger. A line named Includes connects entities Flight and Booking. Keywords airport codes are highlighted and Airport Code are added as an attribute to entity Airport. Keywords total cost is highlighted and Total Cost is added as an attribute to entity Booking. Keywords passenger name is highlighted and Passenger Name is added as an attribute to entity Passenger.

Step 2: Mileage plan number is an attribute of Passenger or, if additional mileage plan information is tracked, a separate entity. Keywords mileage plan number are highlighted and Mileage Plan Number are added as an attribute to entity Passenger.

Animation captions:

1. Airport code, total cost, and name are attributes of Airport, Booking, and Passenger.
2. Mileage plan number is an attribute of Passenger or, if additional mileage plan information is tracked, a separate entity.

PARTICIPATION ACTIVITY 37.2.4: Discovery.

Refer to the following interview:

Our department tracks student information. We have a record of every course taken by students and the professor who taught the course. We also keep track of the name and number of credits for each course. For professors, we always store the name and current title, such as "Assistant Professor" or "Visiting Lecturer".

1) Which noun is an entity?

- information
- name
- professor

2) Which verb is a relationship?

- taken
- track
- have

3) Which noun is an attribute?

- record
- Assistant Professor
- title
- course

Names

Entity names are a singular noun. Ex: Employee rather than Employees. The best names are commonly used and easily understood by database users.

Relationship names have the form `Entity-Verb-Entity`, such as Division-Contains-Department. When the related entities are obvious, in ER diagrams or informal conversation, `Verb` is sufficient and entity names can be omitted. The verb should be active rather than passive. Ex: Manages rather than IsManagedBy. Occasionally, the same verb relates different entity pairs. Ex: Order-Contains-LineItem and Division-Contains-Department.

Attribute names have the form `EntityQualifierType`, such as EmployeeFirstName:

- `Entity` is the name of the entity that the attribute describes. When the entity is obvious, in ER diagrams or informal conversation, `QualifierType` is sufficient and the entity name can be omitted.
- `Qualifier` describes the meaning of the attribute. Ex: First, Last, and Alternate. Sometimes a qualifier is unnecessary and can be omitted. Ex: StudentNumber.
- `Type` is chosen from a list of standard attribute types such as Name, Number, and Count. Attribute types are not identical to SQL data types. Ex: "Amount" might be an attribute type representing monetary values, implemented as the MONEY data type in SQL. "Count" might be an attribute type representing quantity, implemented as NUMBER in SQL.

Standard attribute types are documented in the glossary and applied uniformly to all attribute names.

Table 37.2.1: Example names.

	Formal name	Informal name
Entity	Vehicle	Vehicle
Relationship	Vehicle-BelongsTo-Person	BelongsTo
Attribute	VehicleLicenseNumber	LicenseNumber

PARTICIPATION ACTIVITY 37.2.5: Naming conventions.

1) "Students" is a good entity name.
   - True
   - False

2) "License" is a good attribute name.
   - True
   - False

3) "Employee-Manages-Employee" is a good relationship name.
   - True
   - False

4) "People" is a good entity name.
   - True
   - False

5) "PassengerMileagePlanCode" is a good attribute name.
   - True
   - False

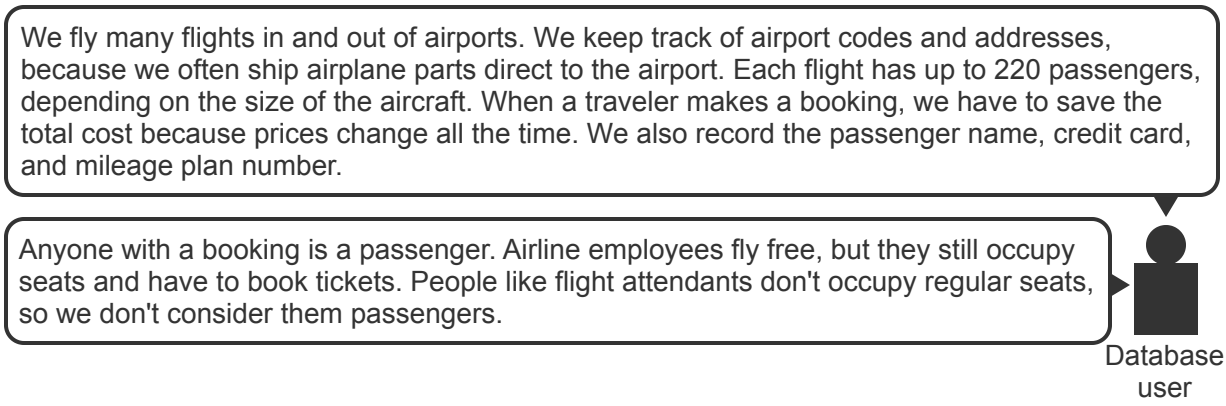
6) "Department-IsManagedBy-Employee" is a good relationship name.
   - True
   - False

Synonyms and descriptions

Often, entity, relationship, and attribute names have synonyms. Ex: Representative may be a synonym for SalesAgent. Synonyms are common in informal communications. To avoid confusion, one official name is selected for each entity, relationship, and attribute. Other names are documented in the glossary as synonyms.

The glossary also contains complete descriptions of entities, relationships, and attributes. The description states the meaning of each entity, relationship, or attribute in complete sentences. The description begins with the name and includes examples and counterexamples to illustrate usage.

PARTICIPATION ACTIVITY 37.2.6: Synonyms and descriptions.




```
Glossary

Entity Name: Passenger

Synonyms: Traveler, Customer

Description: A passenger is any person who occupies a seat on a flight.
Passengers include both paid and unpaid seat occupants, but excludes
flight officers, flight attendants, and in-cabin pets.
```

Animation content:

Step 1: Traveler is a synonym of passenger. There is a database user who states We fly many flights in and out of airports. We keep track of airport codes and addresses, because we often ship airplane parts direct to the airport. Each flight has up to 220 passengers, depending on the size of the aircraft. When a traveler makes a booking, we have to save the total cost because prices change all the time. We also record the passenger name, credit card, and mileage plan number. Keyword traveler is underlined and text with Title Glossary appears. Underneath the title is a line that separates it from the two lines Entity Name colon Passenger and Synonyms colon Traveler comma Customer.

Step 2: An entity description begins with a name and definition, followed by examples and counterexamples. The database user also states Anyone with a booking is a passenger. Airline employees fly free, but they still occupy seats and have to book tickets. People like flight attendants don't occupy regular seats, so we don't consider them passengers. This causes text to appear underneath the existing lines of text and states Description colon A passenger is any person who occupies a seat on a flight. Passengers include both paid and unpaid seat occupants, but excludes flight officers, flight attendants, and in-cabin pets.

Animation captions:

1. Traveler is a synonym of passenger.
2. An entity description begins with a name and definition, followed by examples and counterexamples.

PARTICIPATION ACTIVITY 37.2.7: Synonyms and descriptions.

1) Must entity, relationship, and attribute synonyms follow naming conventions?

- Yes
- Only attribute synonyms must follow naming conventions.
- No

2) What is wrong with the following entity description?

The difference between a course and a class is that a course refers to the catalog description, while a class is an individual offering of a course in a specific term.

- Description does not begin with entity name.
- Description does not include counterexamples.
- Description does not use complete sentences.

3) What is wrong with the following relationship description?

"Student-Takes-Course" describes all course instances taken by a student instance.

- Description should begin with an entity name.
- Description does not use complete sentences.
- Description does not include examples and counterexamples.

Database design

The first step of the analysis phase is discovery of entities, relationships, and attributes in interviews and document review. As discovery proceeds, the designer draws an ER diagram, determines standard attribute types, and documents names, synonyms, and descriptions in the glossary.

Although the step numbers suggest a sequence, database designers commonly move back and forth between steps. As names, synonyms, and descriptions are documented, additional entities, relationships, and attributes are discovered. The ER diagram and glossary are usually developed in parallel.

Table 37.2.2: Discover entities, relationships, and attributes.

Step	Activity
1A	Identify entities, relationships, and attributes in interviews.
1B	Draw ER diagram.
1C	List standard attribute types in glossary.
1D	Document names, synonyms, and descriptions in glossary.

PARTICIPATION ACTIVITY 37.2.8: Database design.

1) Discovery is a step in which phase?

- Analysis
- Logical design
- Database design

2) Identification of entities, relationships, and attributes precedes documentation.

- Always
- Usually
- Never

3) Standard attribute types are determined after the ER diagram is drawn.

- Always
- Usually
- Never

CHALLENGE ACTIVITY 37.2.1: Discovery.

539740.3879454.qx3zqy7

Start

Our business has a number of factories. Each factory has a name and an address. Each facto several products. Each product has a product ID, name, and description. Many kinds of parts manufacturing each product. Each kind of part is identified by number. Each part has a bin nu indicating where the part is stored.

What are the entities?

- [ ] Name
- [ ] Business
- [ ] Part
- [ ] Bin
- [ ] Factory
- [ ] Product

1 2 3

Check Next

37.3 Cardinality

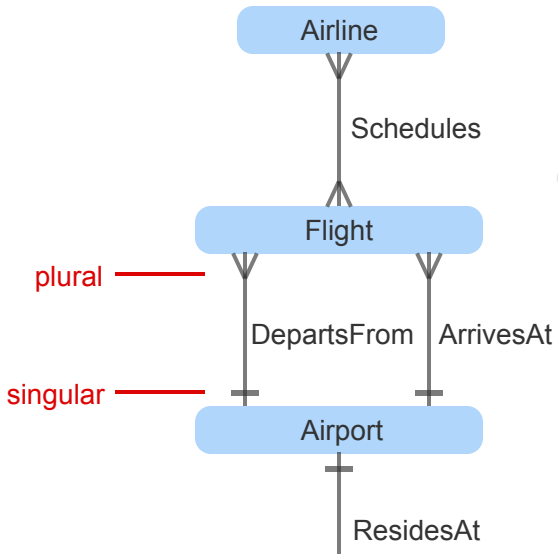
Relationship maximum

In entity-relationship modeling, ***cardinality*** refers to maxima and minima of relationships and attributes.

Relationship maximum is the greatest number of instances of one entity that can relate to a single instance of another entity. A relationship has two maxima, one for each of the related entities. Maxima are usually specified as one or many. A related entity is ***singular*** when the maximum is one and ***plural*** when the maximum is many.

On ER diagrams, maximum of one is shown as a short bar across the relationship line. Maximum of many is shown as three short lines that converge at a point. The three lines look like a bird's foot, so this convention is called ***crow's foot*** notation.

PARTICIPATION ACTIVITY 37.3.1: Relationship maximum.



Animation content:

Step 1: Each flight departs from at most one airport. Airport is singular in Flight-DepartsFrom-Airport. There are four entities named Airline, Flight, Airport, and Address. A line Schedules connects Airline and Flight. Lines ArrivesAt and DepartsFrom connect Flight and Airport. A line ResidesAt connects Airport and Address. A short line across line DepartsFrom appears above Airport. The caption singular appears next to the short line.

Step 2: Each airport has many departing flights. Flight is plural in Flight-DepartsFrom-Airport. Along line DepartsFrom, the crow's foot symbol appears below entity Flight. The caption plural appears next to the crow's foot symbol.

Step 3: ArrivesAt and DepartsFrom have the same maxima. Along the ArrivesAt line, a crow's foot symbol appears below Flight and a short line appears above Airport.

Step 4: Each airline schedules many flights. When multiple airlines share the same flights, each flight can be scheduled by many airlines. Along the Schedules line, the crow's foot symbol appears below Airline and above Flight.

Step 5: Each airport resides at most one official address. Each address has at most one airport. Along the ResidesAt line, a short line appears below Airport and above Address.

Animation captions:

1. Each flight departs from at most one airport. Airport is singular in Flight-DepartsFrom-Airport.
2. Each airport has many departing flights. Flight is plural in Flight-DepartsFrom-Airport.
3. ArrivesAt and DepartsFrom have the same maxima.
4. Each airline schedules many flights. When multiple airlines share the same flights, each flight can be scheduled by many airlines.
5. Each airport resides at most one official address. Each address has at most one airport.

Occasionally, a plural entity has a fixed numeric maximum. Ex: In Employee-Has-Telephone, if each employee has at most three telephone numbers, the maximum of Telephone is three. Fixed numeric maxima are documented in the glossary.

PARTICIPATION ACTIVITY 37.3.2: Relationship maximum.

Determine the maxima for each relationship. The correct answer may depend on business rules.

1) Student-Takes-Course

- one-one
- one-many
- many-one
- many-many

2) City-IsCapitalOf-State

- one-one
- one-many
- many-one
- many-many

3) Airport-Has-Runway

- one-one
- one-many
- many-one
- many-many

4) Person-Owns-Vehicle

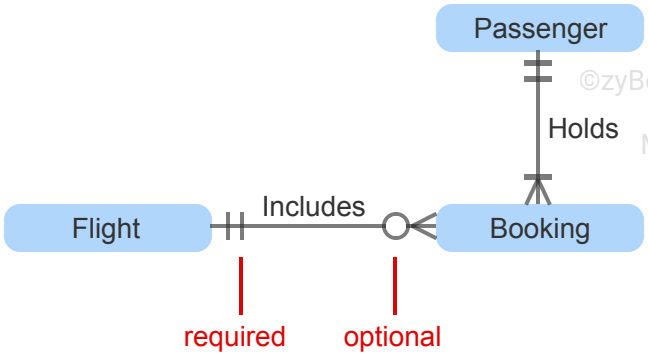
- one-one
- one-many
- many-one
- many-many

Relationship minimum

Relationship minimum is the least number of instances of one entity that can relate to a single instance of another entity. A relationship has two minima, one for each of the related entities. Minima are usually specified as zero or one. A related entity is ***optional*** when the minimum is zero and ***required*** when the minimum is one.

On ER diagrams, minimum of one is shown as a short bar across the relationship line. Minimum of zero is shown as a circle. Maxima symbols always appear next to the entity and minima symbols appear further from the entity.

PARTICIPATION ACTIVITY 37.3.3: Relationship minimum.



Animation content:

Step 1: The maximum of Flight-Includes-Booking is one-many. There are three entities named Passenger, Booking, and Flight. A line named Holds connects Passenger and Booking. A line named Includes connects Flight and Booking. Along the Includes line, a short line appears next to Flight and a crow's foot symbol appears next to Booking.

Step 2: Each booking is included on at least one flight. Flight is required in Flight-Includes-Booking. Across the Includes line, a second short line, with caption required, appears next to Flight.

Step 3: A new flight may have no bookings. Booking is optional in Flight-Includes-Booking. Across the Includes line, a circle with caption optional appears next to Booking.

Step 4: The maximum of Passenger-Holds-Booking is one-many. Along the Holds line, a short line appears next to Passenger and a crow's foot symbol appears next to Booking.

Step 5: Passenger and Booking are both required in Passenger-Holds-Booking. Across the Holds line, short lines appear next to Passenger and Booking.

Animation captions:

1. The maximum of Flight-Includes-Booking is one-many.
2. Each booking is included on at least one flight. Flight is required in Flight-Includes-Booking.
3. A new flight may have no bookings. Booking is optional in Flight-Includes-Booking.
4. The maximum of Passenger-Holds-Booking is one-many.
5. Passenger and Booking are both required in Passenger-Holds-Booking.

Occasionally, a required entity has a minimum greater than one. Ex: In Customer-Has-Identification, if two forms of identification are required for every customer, the minimum of Identification is two. Minima greater than one are documented in the glossary.

PARTICIPATION ACTIVITY 37.3.4: Relationship minimum.

Determine the minima for each relationship. The correct answer may depend on business rules.

1) Person-Marries-Person

- zero-zero
- zero-one
- one-zero
- one-one

2) Person-Has-Passport

- zero-zero
- zero-one
- one-zero
- one-one

3) Flight-ArrivesAt-Airport

- zero-zero
- zero-one
- one-zero
- one-one

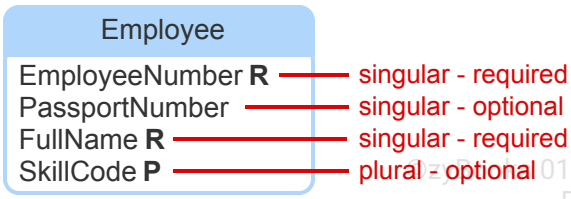
Attribute maximum and minimum

Attribute maximum is the greatest number of attribute values that can describe each entity instance. Attribute maximum is specified as one (singular) or many (plural).

Attribute minimum is the least number of attribute values that can describe each entity instance. Attribute minimum is specified as zero (optional) or one (required).

In ER diagrams, attributes are presumed singular and optional. A "P" following the attribute indicates the attribute is plural. An "R" indicates the attribute is required.

PARTICIPATION ACTIVITY 37.3.5: Attribute maximum and minimum.



Animation content:

Step 1: Each employee has exactly one employee number. There is an entity named Employee with attributes EmployeeNumber, PassportNumber, FullName, and SkillCode. R appears after EmployeeNumber and is labeled singular-required.

Step 2: Each employee has at most one United States passport number. Some employees do not have a passport. PassportNumber is labeled singular-optional.

Step 3: Each employee has exactly one name. R appears after FullName and is labeled singular-required.

Step 4: An employee may have many skills. Some employees record no skills in the database. P appears after SkillCode and is labeled plural-optional.

Animation captions:

1. Each employee has exactly one employee number.
2. Each employee has at most one United States passport number. Some employees do not have a passport.
3. Each employee has exactly one name.
4. An employee may have many skills. Some employees record no skills in the database.

Occasionally, attribute maximum or minimum is a fixed number other than zero, one, or many. Ex: If each employee must speak at least two languages, the Language attribute of Employee has a minimum of two. Numeric minimum and maximum are documented in the glossary.

PARTICIPATION ACTIVITY 37.3.6: Attribute maximum and minimum.

Determine the maximum and minimum of each attribute. The correct answer may depend on business rules.

1) Entity Airport, attribute AirportCode

- singular-required
- singular-optional
- plural-required

2) Entity Person, attribute SpouseName

- singular-required
- plural-required
- singular-optional

3) Entity Employee, attribute LanguageCode

- singular-required
- plural-optional
- plural-required

Unique attributes

Each value of a ***unique attribute*** describes at most one entity instance. Ex: Vehicle identification number (VIN) is a unique attribute of Vehicle.

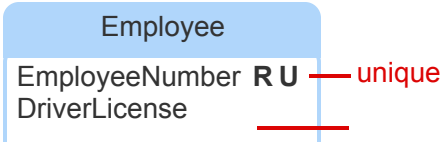
Occasionally, a composite of several attributes is unique, although the individual attributes are not. Ex: (AirlineCode, FlightNumber) is a unique composite attribute of Flight. However, AirlineCode and FlightNumber are not individually unique, since flights on different airlines may have the same flight number.

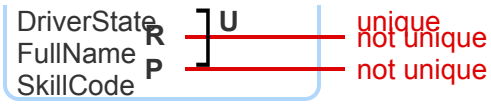
Unique is not the same as a singular:

- A unique attribute has at most one entity instance for each attribute value.
- A singular attribute has at most one attribute value for each entity instance.

In ER diagrams, attributes are presumed not unique. A "U" following the attribute indicates the attribute is unique. Unique composite attributes are grouped with a brace before the "U" symbol.

PARTICIPATION ACTIVITY 37.3.7: Unique attributes.





Animation content:

Step 1: Each employee number describes at most one employee. An entity named Employee has attributes EmployeeNumber R, DriverLicense, DriverState, FullName R, and SkillCode P. U, with caption unique, appears next to EmployeeNumber R.

Step 2: License number and state are not separately unique, but the composite is unique. One bracket appears next to attributes DriverLicense and DriverState. U, with caption unique, appears next to the bracket.

Step 3: Each name and each skill can describe many employees. Caption not unique appears next to FullName and SkillCode.

Step 4: Unique is not the same as singular. Ex: FullName is singular but not unique. FullName is highlighted.

Animation captions:

1. Each employee number describes at most one employee.
2. License number and state are not separately unique, but the composite is unique.
3. Each name and each skill can describe many employees.
4. Unique is not the same as singular. Ex: FullName is singular but not unique.

Entity-Has-Attribute relationship

Entities have an implicit relationship with their attributes, called Entity-Has-Attribute. Attribute maximum and minimum are the cardinality of Attribute in this relationship. Ex: VIN is a required attribute of Vehicle. In the relationship Vehicle-Has-VIN, every vehicle must have a VIN..

An attribute is unique when Entity is singular in this relationship. Ex: Each VIN has at most one vehicle. So VIN is a unique attribute of Vehicle.

PARTICIPATION ACTIVITY 37.3.8: Unique attributes.

Match the term to the database requirement.

If unable to drag and drop, refresh the page.

unique | optional | singular | required | plural

	We track at most one contact telephone number for each student. TelephoneNumber is a(n) _____ attribute of Student.
	Students can major in several subjects. MajorSubjectName is a(n) _____ attribute of Student.
	Sometimes students do not provide a telephone number on registration forms, so this information is left blank in the database. TelephoneNumber is a(n) _____ attribute of Student.
	All students must have an official email address. EmailAddress is a(n) _____ attribute of Student.
	Students are assigned an eight-digit number, used to identify students on forms and records. StudentNumber is a(n) _____ attribute of Student.

Reset

Database design

Relationship and attribute cardinality depends on business rules. Ex: Usually Employee-WorksIn-Department maxima are many-one. If a company assigns employees to multiple departments, however, the maxima are many-many.

During the analysis phase, the designer looks for cardinality business rules in interviews and document review. The designer then converts business rules into zero, one, and many specifications, and documents specifications in the ER diagram and glossary.

Depending on the desired level of detail, cardinality does not always appear on ER diagrams. Usually, ER diagrams are drawn with software tools that can automatically show or hide cardinality.

Table 37.3.1: Determine cardinality.

Step	Activity
2A	Determine relationship maxima and minima.
2B	Determine attribute maxima and minima.
2C	Identify unique attributes.
2D	Document cardinality in glossary and, optionally, on ER diagram.

PARTICIPATION ACTIVITY 37.3.9: Database design.

1) 'Cardinality' refers to relationships only.
   - True
   - False

2) ER diagrams indicate maxima and minima for relationships only.
   - True
   - False

3) Maxima and minima usually depend on business rules.
   - True
   - False

4) In the analysis phase, cardinality is always determined after discovery is complete.

- True
- False

CHALLENGE ACTIVITY 37.3.1: Cardinality.

539740.3879454.qx3zqy7

Start

Hundreds of students are in one course.
Each student takes multiple courses.

Determine the relationship maxima.

Student-Takes-Course

Select

1 2 3 4 5

Check Next

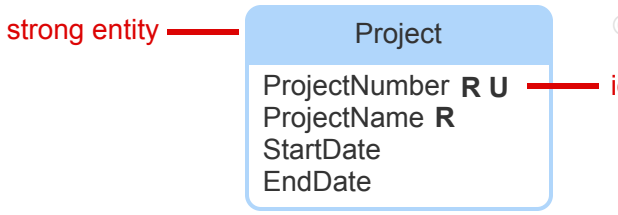
37.4 Strong and weak entities

Strong entities

An ***identifying attribute*** is unique, singular, and required. Identifying attribute values correspond one-to-one to, or ***identify***, entity instances.

A ***strong entity*** has one or more identifying attributes. When a strong entity is implemented as a table, one of the identifying attributes may become the primary key.

PARTICIPATION ACTIVITY 37.4.1: Strong entities.



Animation content:

Static figure:
The Project entity has attributes ProjectNumber R U, ProjectName R, StartDate, and EndDate. The Project entity has caption strong entity. The ProjectNumber attribute has caption identifying attribute.

Step 1: Each number describes at most one project, so ProjectNumber is unique. The U following ProjectNumber is highlighted.

Step 2: Each project has exactly one number, so ProjectNumber is singular and required. The R following ProjectNumber is highlighted.

Step 3: ProjectNumber is an identifying attribute. Project numbers correspond one-to-one to projects. The caption identifying attribute appears next to ProjectNumber.

Step 4: An entity with an identifying attribute is strong. The caption strong entity appears next to Project.

Animation captions:

1. Each number describes at most one project, so ProjectNumber is unique.
2. Each project has exactly one number, so ProjectNumber is singular and required.
3. ProjectNumber is an identifying attribute. Project numbers correspond one-to-one to projects.
4. An entity with an identifying attribute is strong.

PARTICIPATION ACTIVITY 37.4.2: Identifying attributes.

The Project entity has a ProjectCode attribute.

1) Each project has at most one code. The ProjectCode attribute is _____.
   - unique
   - singular
   - plural
   - required
   - optional

2) Each project code describes at most one project. The ProjectCode attribute is _____.
   - unique
   - singular
   - plural
   - required
   - optional

3) A project may have no code. The ProjectCode attribute is _____.
   - unique
   - singular
   - plural
   - required
   - optional

4) ProjectCode is an identifying attribute of the Project entity.
   - True
   - False

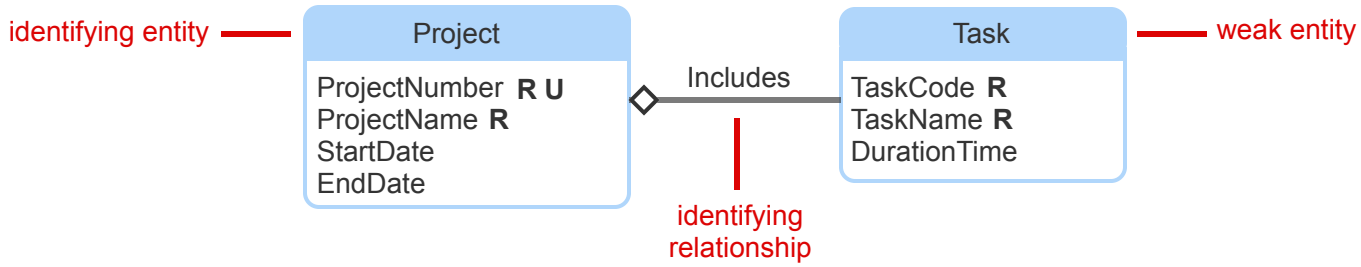
©zyBooks 01/31/24 18:23 1939727
Rob Daglio
MDCCOP2335Spring2024

Weak entities

A ***weak entity*** does not have an identifying attribute. Instead, a weak entity usually has a relationship, called an ***identifying relationship***, to another entity, called an ***identifying entity***. The identifying entity must be singular and required in an identifying relationship.

In an ER diagram, an identifying relationship has a diamond next to the identifying entity. Since an identifying entity is always singular and required, the diamond replaces the entity's cardinality symbols.

PARTICIPATION ACTIVITY 37.4.3: Weak entities.



Animation content:

Static figure:
The Project entity has attributes ProjectNumber R U, ProjectName R, StartDate, and EndDate. The Project entity has caption identifying entity.

The Task entity has attributes TaskCode R, TaskName R, DurationTime. The Task entity has caption weak entity.

The relationship Project-Includes-Task has a diamond next to Project. The relationship has caption identifying relationship.

Step 1: TaskCode, TaskName, and DurationTime are not unique. Since Task does not have an identifying attribute, Task is a weak entity. Task appears with caption weak entity.

Step 2: Task is weak and Project is singular and required, so Project-Includes-Task is an identifying relationship. Project appears. The Includes relationship appears with caption identifying relationship. Along the Includes line, two short lines appear next to Project.

Step 3: A diamond indicates an identifying relationship. The diamond is next to the identifying entity and replaces the cardinality symbols. The two short lines are replaced with a diamond. The caption identifying entity appears next to Project.

Animation captions:

1. TaskCode, TaskName, and DurationTime are not unique. Since Task does not have an identifying attribute, Task is a weak entity.
2. Task is weak and Project is singular and required, so Project-Includes-Task is an identifying relationship.
3. A diamond indicates an identifying relationship. The diamond is next to the identifying entity and replaces the cardinality symbols.

For weak entities, identifying relationships replace identifying attributes. Ex: In the animation above, If each project has at most one task, ProjectNumber identifies Task. If each project has many tasks, (ProjectNumber, TaskName) identifies Task. The second attribute, TaskName, must be singular, required, and unique within each project.

PARTICIPATION ACTIVITY 37.4.4: Strong and weak entities.

Refer to the following database requirements:

A model has entities Department, Course, and Exam. Each academic department has a unique name.

Courses are offered by academic departments. Courses are identified by a course number and the name of the department that offers the course. Different departments use the same course numbers. Ex: Math 101, Philosophy 101, Economics 200A.

Each course has several exams. Each exam is labeled with a course name and exam sequence number. The sequence number begins at 1 for all courses. Ex: Math 101-1, Math 101-2, Math 101-3.

If unable to drag and drop, refresh the page.

Strong entity **Weak entity** **Identifying attribute** **Identifying relationship** **Identifying entity**

	DepartmentName
	Course, Exam
	Department
	Department, Course

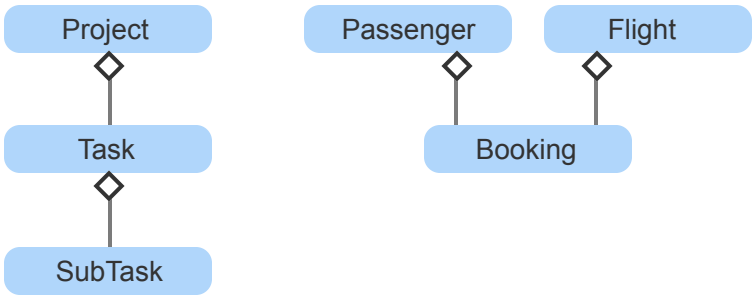
	Department-Offers-Course, Course-Gives-Exam

Reset

Identifying entities

A weak entity is usually identified by a strong entity. However, a weak entity can be identified by another weak entity or by several entities.

PARTICIPATION ACTIVITY 37.4.5: Identifying entities.



Animation content:

Static figure:
Entities Project, Task, and SubTask appear. Project and Task are connected by an unnamed relationship, with a diamond next to Project. Task and SubTask are connected by an unnamed relationship, with a diamond next to Task.

Entities Passenger, Flight, and Booking appear. Passenger and Booking are connected by an unnamed relationship, with a diamond next to Passenger. Flight and Booking are connected by an unnamed relationship, with a diamond next to Flight.

Step 1: Project has an identifying attribute, such as ProjectNumber, and is a strong entity. The Project entity appears.

Step 2: Task is a weak entity, identified by the Project entity. The Task entity appears. The unnamed relationship between Project and Task appears with a diamond next to Project.

Step 3: The weak entity Subtask is identified by the weak entity Task. The SubTask entity appears.

The unnamed relationship between Task and SubTask appears with a diamond next to Task.

Step 4: The weak entity Booking is identified by Passenger and Flight. Passenger, Flight, and Booking appear, along with the associated relationships and diamonds.

Animation captions:

1. Project has an identifying attribute, such as ProjectNumber, and is a strong entity.
2. Task is a weak entity, identified by the Project entity.
3. The weak entity Subtask is identified by the weak entity Task.
4. The weak entity Booking is identified by Passenger and Flight.

When a weak entity is identified by a weak entity or multiple entities, the identifying attribute may be complex. Ex: In the animation above:

- If each task has many subtasks, a composite attribute such as (ProjectNumber, TaskName, SubtaskCode) identifies Subtask.
- If each person and each flight has many bookings, a composite attribute such as (FlightNumber, PassengerID, BookingDate) identifies Booking.

In these cases, the identifying attribute depends on business rules and may not be apparent in the ER diagram.

PARTICIPATION ACTIVITY 37.4.6: Identifying entities.

1) How often is an identifying entity also a weak entity?
   - Never
   - Sometimes
   - Always

2) How many relationships can identify one weak entity?
   - Zero or one
   - Exactly one
   - One or many

3) How many weak entities can one entity identify?

- Zero or one
- One or many
- Zero, one, or many

Database design

After entities, relationships, attributes, and cardinality are determined, the database designer distinguishes strong and weak entities. For each weak entity, the identifying relationship is noted. Weak entities and identifying relationships are documented in the glossary and ER diagram.

Most database designers use software tools to manage ER diagrams. Software tools usually allow users to choose from alternative conventions and automatically switch between conventions.

Table 37.4.1: Distinguish strong and weak entities.

Step	Activity
3A	Identify strong and weak entities.
3B	Determine the identifying relationship(s) for each weak entity.
3C	Document weak entities and identifying relationships in glossary and ER diagram.

PARTICIPATION ACTIVITY 37.4.7: Distinguish strong and weak entities.

1) Distinguishing strong and weak entities is a logical design activity.

- True
- False

2) Determining cardinality always precedes distinguishing strong and weak entities.

- True
- False

3) Database designers may look for identifying relationships first, before noting weak entities.

- True
- False

CHALLENGE ACTIVITY 37.4.1: Strong and weak entities.

539740.3879454.qx3zqy7

Start

An athletic association has a database of track meets. Each track meet has events. Each eve preliminary rounds called "heats". Track meets have unique names but usually use the same codes. Heats are named with an event code and a heat number.

Select the correct identifying entity.

TrackMeet Select

Event Select

Heat Select

1 2

Check Next

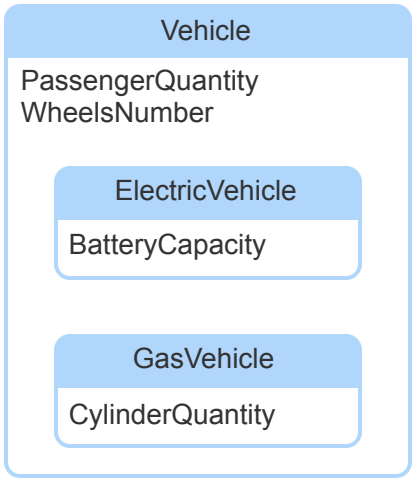
37.5 Supertype and subtype entities

Supertype and subtype entities

An entity type is a set of entity instances. A ***subtype entity*** is a subset of another entity type, called the ***supertype entity***. Ex: Managers are a subset of employees, so Manager is a subtype entity of the Employee supertype entity. On ER diagrams, subtype entities are drawn within the supertype.

A supertype entity usually has several subtypes. Attributes of the supertype apply to all subtypes. Attributes of a subtype do not apply to other subtypes or the supertype.

PARTICIPATION ACTIVITY | 37.5.1: Supertype and subtype entities.



Animation content:

Step 1: Vehicle supertype has ElectricVehicle and GasVehicle subtypes. Entity Vehicle appears. Entities ElectricVehicle and GasVehicle appear inside entity Vehicle.

Step 2: Number of passengers and number of wheels apply to both electric and gas vehicles. Attributes PassengerQuantity and WheelsNumber appear in Vehicle.

Step 3: Battery capacity applies to electric vehicles only. Attribute BatteryCapacity appears in ElectricVehicle.

Step 4: Number of cylinders applies to gas vehicles only. Attribute CylinderQuantity appears in GasVehicle.

Animation captions:

1. Vehicle supertype has ElectricVehicle and GasVehicle subtypes.
2. Number of passengers and number of wheels apply to both electric and gas vehicles.
3. Battery capacity applies to electric vehicles only.
4. Number of cylinders applies to gas vehicles only.

Supertype and subtype entities have several benefits:

- Subtypes highlight subsets of data with different attributes and relationships than the supertype. This clarifies database semantics and behavior.
- Optional supertype attributes become required subtype attributes. This results in compact tables with fewer NULL values.
- An attribute that is repeated in several subtypes becomes a single supertype attribute. This ensures a single name, description, cardinality, and type for the attribute, rather than one for each subtype.

Examples of optional and repeated attributes appear in the subsection *Similar entities and optional attributes*, below.

PARTICIPATION ACTIVITY 37.5.2: Supertype and subtype entities.

Refer to the following database requirements:

Some passengers enroll in mileage plans and get a mileage plan number. Travelers who fly 100,000 miles or more in a calendar year are gold mileage plan members. Between 25,000 miles and 100,000 miles is silver status, and below 25,000 miles flown is bronze status. Gold members designate the name of a family member who can fly for free.

1) What is passenger?

- Supertype entity
- Subtype entity
- Weak entity

2) What is gold plan member?

- Supertype entity
- Subtype entity
- Attribute

3) What is mileage plan number?

- Subtype entity
- Attribute of Passenger
- Attribute of GoldPlanMember

4) What is the name of the family member designated by gold status passengers?

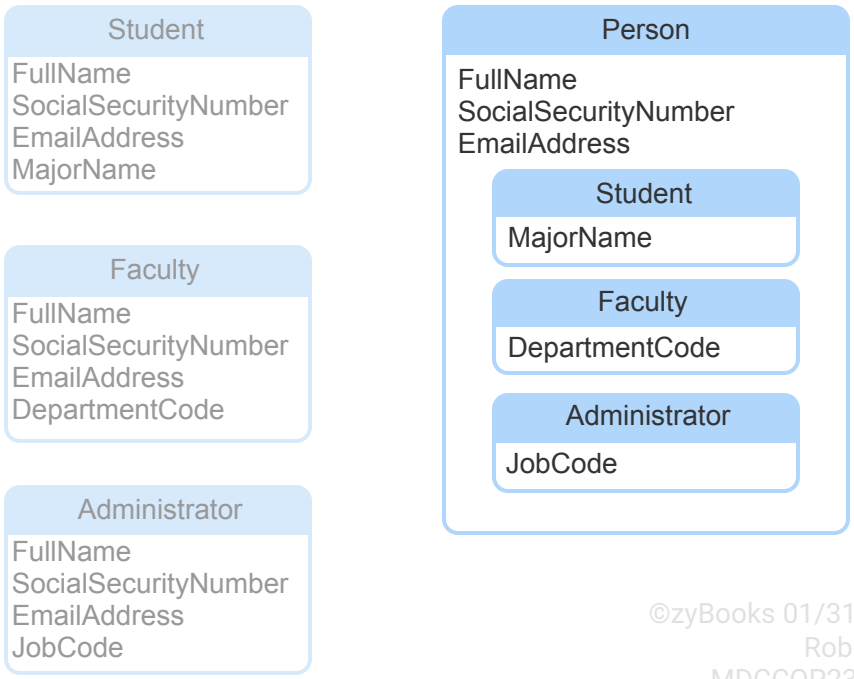
- Subtype entity
- Attribute of Passenger
- Attribute of GoldPlanMember

Similar entities and optional attributes

Supertype and subtype entities are often created from similar entities and optional attributes.

Similar entities are entities that have many common attributes and relationships. Similar entities become subtypes of a new supertype entity, as in the animation below. Common attributes and relationships move to the new supertype entity. Attributes and relationships that are not shared remain with the subtype entities.

PARTICIPATION ACTIVITY 37.5.3: Similar entities become supertype entity.



Animation content:

Static figure:
Two entity-relationship diagrams appear.

In the first diagram, entity Student has attributes FullName, SocialSecurityNumber, EmailAddress, and MajorName. Entity Faculty has attributes FullName, SocialSecurityNumber, EmailAddress, and DepartmentCode. Entity Administrator has attributes FullName, SocialSecurityNumber, EmailAddress, and JobCode The first diagram is faded out.

In the second diagram, entity Person has attributes FullName, SocialSecurityNumber, and EmailAddress. Entities Student, Faculty, and Administrator are inside Person. Student has attribute MajorName. Faculty has attribute DepartmentCode. Administrator has attribute JobCode.

Step 1: Student, Faculty, and Administrator are similar entities with common attributes FullName, SocialSecurityNumber, and EmailAddress. The first diagram appears.

Step 2: Common attributes move to a new supertype entity called Person. In the first diagram, attributes FullName, SocialSecurityNumber, and EmailAddress are highlighted in all three entities. The Person entity appears with attributes FullName, SocialSecurityNumber, and EmailAddress.

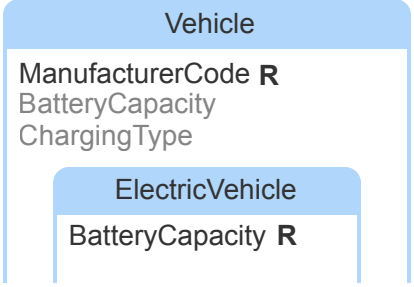
Step 3: Subtype-specific attributes remain attributes of each subtype. In the second diagram, entities Student, Faculty, and Administrator appear inside Person. Student has attribute MajorName. Faculty has attribute DepartmentCode. Administrator has attribute JobCode. The first diagram fades out.

Animation captions:

1. Student, Faculty, and Administrator are similar entities with common attributes FullName, SocialSecurityNumber, and EmailAddress.
2. Common attributes move to a new supertype entity called Person.
3. Subtype-specific attributes remain attributes of each subtype.

An entity with many optional attributes also suggests new supertype and subtype entities. The entity becomes a supertype entity and retains all required attributes. Optional attributes become required attributes of new subtype entities.

PARTICIPATION ACTIVITY 37.5.4: Optional attributes become subtype entities.



ChargingType R

Animation content:

Step 1: Vehicle has required attribute ManufacturerCode and optional attributes BatteryCapacity and ChargingType. Entity Vehicle appears with attributes ManufacturerCode R, BatteryCapacity, and ChargingType.

Step 2: Optional attributes become required attributes of the new subtype entity ElectricVehicle. Entity ElectricVehicle appears inside Vehicle. Attributes BatteryCapacity and ChargingType move from Vehicle to ElectricVehicle. An R appears after both attributes.

Animation captions:

1. Vehicle has required attribute ManufacturerCode and optional attributes BatteryCapacity and ChargingType.
2. Optional attributes become required attributes of the new subtype entity ElectricVehicle.

Creating a new supertype for similar entities, or a new subtype for optional attributes, is neither an automatic nor objective decision. Similar entities with many common attributes are good candidates for a new supertype. Entities with many optional attributes are good candidates for a new subtype.

PARTICIPATION ACTIVITY 37.5.5: Similar entities and optional attributes.

Refer to the following database requirements:

The university tracks the grade point average and telephone number for all students. The university also tracks the phone number for faculty and administrative staff. Faculty have an academic ranking, like "Assistant Professor" or "Adjunct Professor". Administrative staff are either hourly or salaried, which the university calls employment status.

The requirements are modeled as a supertype entity Person with subtypes Student, Faculty, and Administrator.

1) Which entity does GradePointAverage belong to?
   - ○ Person
   - ○ Student
   - ○ Faculty
   - ○ Administrator

2) Which entity does TelephoneNumber belong to?

- Person
- Student
- Faculty
- Administrator

3) Which entity does EmploymentStatus belong to?

- Person
- Student
- Faculty
- Administrator

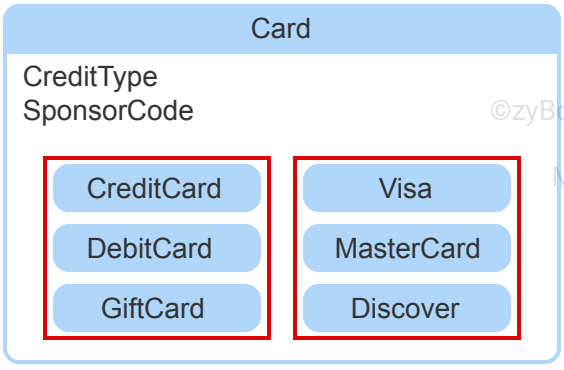
Partitions

A ***partition*** of a supertype entity is a group of mutually exclusive subtype entities. A supertype entity can have several partitions. Subtype entities within each partition are disjoint and do not share instances. Subtype entities in different partitions overlap and do share instances.

In diagrams, subtype entities within each partition are vertically aligned. Subtype entities in different partitions are horizontally aligned.

Each partition corresponds to an optional ***partition attribute*** of the supertype entity. The partition attribute indicates which subtype entity is associated with each supertype instance.

PARTICIPATION ACTIVITY 37.5.6: Partitions.



Animation content:

Step 1: Subtypes CreditCard, DebitCard, and GiftCard are mutually exclusive. These subtypes partition the supertype Card. Entity Card appears. Entities CreditCard, DebitCard, and GiftCard appear inside Card, stacked vertically.

Step 2: CreditType is a partition attribute. Values are "Credit", "Debit", "Gift". Attribute CreditType appears in Card.

Step 3: Visa, MasterCard, and Discover are another partition of Card. The partition attribute is SponsorCode. Entities Visa, MasterCard, and Discover appear inside Card, stacked vertically. Attribute SponsorCode appears in Card.

Step 4: Card has two partitions. Subtypes within each partition are disjoint. Subtypes in different partitions overlap. A box surrounds CreditCard, DebitCard, and GiftCard. Another box surrounds entities Visa, MasterCard, and Discover.

Animation captions:

1. Subtypes CreditCard, DebitCard, and GiftCard are mutually exclusive. These subtypes partition the supertype Card.
2. CreditType is a partition attribute. Values are "Credit", "Debit", "Gift".
3. Visa, MasterCard, and Discover are another partition of Card. The partition attribute is SponsorCode.
4. Card has two partitions. Subtypes within each partition are disjoint. Subtypes in different partitions overlap.

PARTICIPATION ACTIVITY 37.5.7: Partitions and partition attributes.

1) An entity instance can be in two subtypes of the same partition.
   - ○ True
   - ○ False

2) An entity instance can be in two subtypes of different partitions.
   - ○ True
   - ○ False

3) Each partition attribute value corresponds to one subtype.

- True
- False

4) One partition attribute can correspond to several partitions.

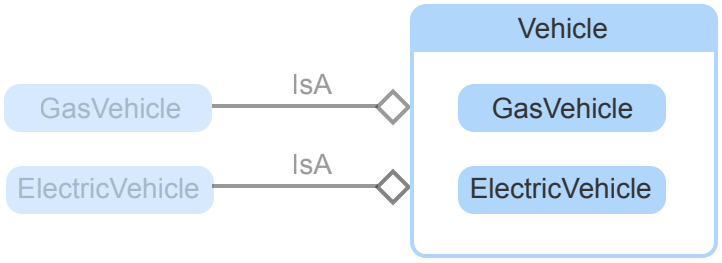
- True
- False

IsA relationship

A supertype entity identifies its subtype entities. The identifying relationship is called an ***IsA relationship***. Ex: Manager-IsAn-Employee relates each manager instance to the corresponding employee instance. Employee numbers, which identify all employees, also identify managers.

Since every subtype has an IsA relationship to the supertype, the relationship is assumed and may be omitted from the ER diagram.

PARTICIPATION ACTIVITY 37.5.8: IsA relationship.



Animation content:

Step 1: Subtype entities have an identifying relationship, called IsA, to the supertype. Entity Vehicle appears. Entities GasVehicle and ElectricVehicle appear outside of Vehicle. Relationships GasVehicle-IsA-Vehicle and ElectricVehicle-IsA-Vehicle appear. Along both relationships, a diamond appears next to Vehicle.

Step 2: IsA relationships are not shown in an ER diagram. GasVehicle and ElectricVehicle move inside Vehicle. The IsA relationships disappear.

Animation captions:

1. Subtype entities have an identifying relationship, called IsA, to the supertype.
2. IsA relationships are not shown in an ER diagram.

PARTICIPATION ACTIVITY 37.5.9: IsA relationship.

1) Subtype entities must have an identifying attribute
   - True
   - False

2) The identifying attribute of a supertype entity also identifies the subtype entities.
   - True
   - False

3) All subtype entities are weak entities.
   - True
   - False

4) All weak entities are subtype entities.
   - True
   - False

Database design

After entities, relationships, attributes, cardinality, and strong and weak entities are determined, the database designer looks for supertype and subtype entities. Similar entities and optional attributes suggest new supertype and subtype entities and warrant special attention. Mutually exclusive subtype entities are grouped into partitions. For each partition, a partition attribute is added to the supertype entity.

Table 37.5.1: Create supertype and subtype entities.

Step	Activity
4A	Identify supertype and subtype entities.
4B	Replace similar entities and optional attributes with supertype and subtype entities.
4C	Identify partitions and partition attributes.
4D	Document supertypes, subtypes, and partitions in glossary and ER diagram.

Creating supertype and subtype entities is the last of four analysis steps:

1. Discover entities, relationships, and attributes
2. Determine cardinality
3. Distinguish strong and weak entities
4. Create supertype and subtype entities

Logical design follows analysis. Logical design converts an entity-relationship model to tables, columns, and keys for a specific database system.

PARTICIPATION ACTIVITY 37.5.10: Analysis activities.

Match high-level step names to detailed activity descriptions.

If unable to drag and drop, refresh the page.

Determine cardinality **Create supertype and subtype entities**

Distinguish strong and weak entities **Discover entities, relationships, and attributes**

	List standard attribute types in glossary.
	Determine attribute maxima and minima.
	Determine the identifying relationship for each weak entity.

Identify partitions and partition attributes.

Reset

CHALLENGE ACTIVITY 37.5.1: Supertype and subtype entities.

539740.3879454.qx3zqy7

Start

A library provides different kinds of resources. Ex: Print books, digital books, and audiobooks. Each print book has a call number. Each resource has a title and a publisher name. Each audiobook has a listening time. Each digital book has one of several file formats. Ex: DOCX and PDF.

Select the correct description for each attribute.

Format Select

PublisherName Select

1

Check Try again

37.6 Alternative modeling conventions

Diagram conventions

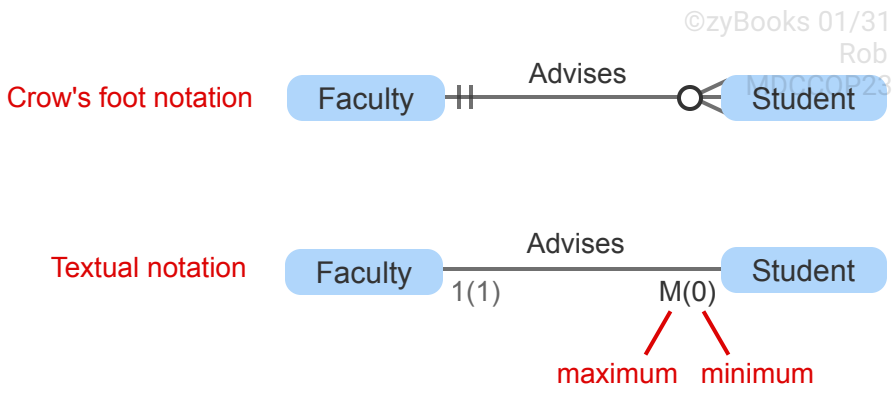
ER diagram conventions vary widely. Ex: Some ER diagrams may:

- Depict relationship names inside a diamond.
- Depict weak entities and identifying relationships with double lines.
- Depict subtype entities with IsA relationships rather than inside of supertype entities.
- Use color, dashed lines, or double lines to convey additional information.

Variations in cardinality symbols are common. Ex: Textual notation depicts optional as 0, singular and required as 1, and plural as M. Maximum cardinality appears first, followed by minimum cardinality in

parentheses, as in the animation below.

PARTICIPATION ACTIVITY 37.6.1: Alternative cardinality notation.



Animation content:

Step 1: Each faculty member advises zero or many students. Entities Faculty and Student appear. The relationship Faculty-Advises-Student appears. Along the relationship, crow's foot and circle symbols appear next to Student. The caption crow's foot notation appears.

Step 2. Each student has exactly one faculty advisor. Along the relationship, two short lines appear next to Faculty.

Step 3: In textual notation, M represents maximum many. 0 represents minimum zero. The Faculty-Advises-Student relationship appears a second time, without the crow's foot, circle, or short lines. In the second copy of Faculty-Advises-Student, M(0) appears next to Student. The caption textual notation appears.

Step 4: The maximum always appears first. Minimum follows in parentheses. The caption maximum appears under the M. The caption minimum appears under the 0.

Step 5: 1(1) indicates both minimum and maximum are one. In the second copy of Faculty-Advises-Student, 1(1) appears next to Faculty.

Animation captions:

1. Each faculty member advises zero or many students.
2. Each student has exactly one faculty advisor.
3. In textual notation, M represents maximum many. 0 represents minimum zero.
4. The maximum always appears first. Minimum follows in parentheses.

5. 1(1) indicates both minimum and maximum are one.

PARTICIPATION ACTIVITY 37.6.2: Alternative diagram conventions.

Match the modeling concept to the alternative diagram notation.

If unable to drag and drop, refresh the page.

plural and optional | **singular and required** | **singular and optional**

plural and required | **identifying relationship**

	1(1)
	M(1)
	Solid double lines
	1(0)
	M(0)

Reset

Model conventions

ER modeling concepts also vary. Ex: Some ER models may:

- Allow relationships between three or more entities.
- Decompose a complex model into a group of related entities, called a ***subject area***.
- Refer to strong entities as ***independent*** and weak entities as ***dependent***.

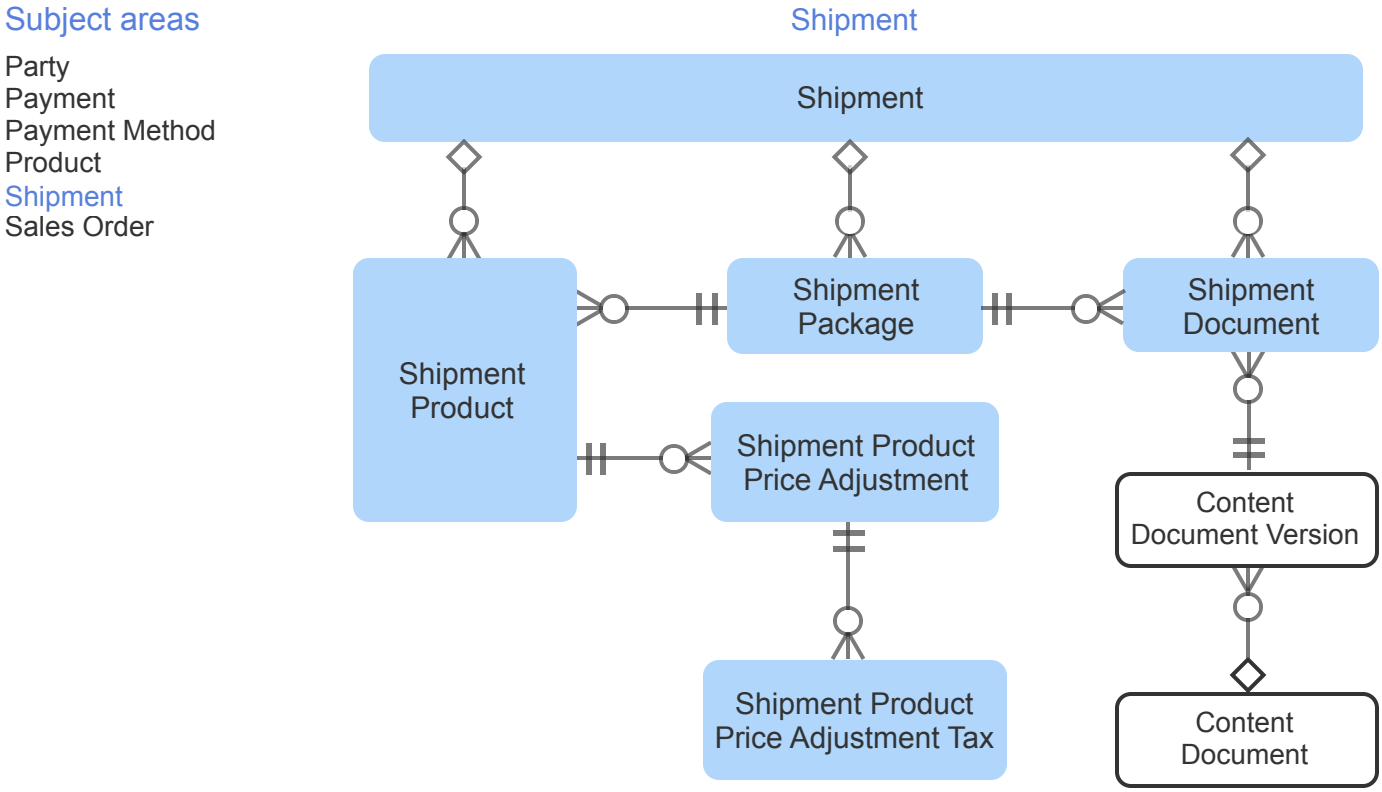
Several model conventions are standardized and widely used. Leading conventions include:

- ***Unified Modeling Language***, or ***UML***, is commonly used for software development. Software data structures are similar to database structures, so UML includes ER conventions.
- ***IDEF1X*** stands for Information DEFinition version 1X. IDEF1X became popular, in part, due to early adoption by the United States Department of Defense.

- ***Chen notation*** appeared in an early ER modeling paper by Peter Chen. Chen notation is not standardized but often appears in literature and tools.

By and large, differences between conventions are stylistic rather than substantial. The choice of convention does not usually affect the resulting database design.

PARTICIPATION ACTIVITY 37.6.3: Cloud Information Model.



Animation content:

Static figure:
The caption Subject Areas appears. Six subject areas appear under this caption: Party, Payment, Payment Method, Product, Shipment, and Sales Order. Shipment is highlighted.

The caption Shipment appears. Eight entities appear under this caption. Entities Shipment, ShipmentProduct, ShipmentPackage, ShipmentDocument, ShipmentProductPriceAdjustment, and ShipmentProductPriceAdjustmentTax appear with a shaded fill. Entities ContentDocumentVersion and ContentDoucment appear with no fill. The eight entities are connected by unnamed relationships. Cardinality symbols appear in crow's foot notation.

Step 1: The Cloud Information Model is an industry standard for sales fulfillment. The model is divided into subject areas. The subject area names appear.

Step 2: The Shipment subject area contains entities related to product shipments. The Shipment subject area is highlighted. The Shipment caption appears. The entities with shaded fill appear below the caption.

Step 3: Cardinality is depicted with crow's foot notation. Identifying relationships are depicted with diamonds. Relationships and cardinality symbols appear.

Step 4: Related entities outside the Shipment subject area are shown in a different color. The two entities with no fill, and their relationships, appear.

Animation captions:

1. The Cloud Information Model is an industry standard for sales fulfillment. The model is divided into subject areas.
2. The Shipment subject area contains entities related to product shipments.
3. Cardinality is depicted with crow's foot notation. Identifying relationships are depicted with diamonds.
4. Related entities outside the Shipment subject area are shown in a different color.

Source: github.com

PARTICIPATION ACTIVITY 37.6.4: ER model and diagram conventions.

1) In some models, weak entities are called _____ entities.

Check Show answer

2) A group of related entities is often called a/an _____ .

Check Show answer

3) _____ is a modeling standard intended for software development

Check Show answer

37.7 Implementing entities

Selecting primary keys

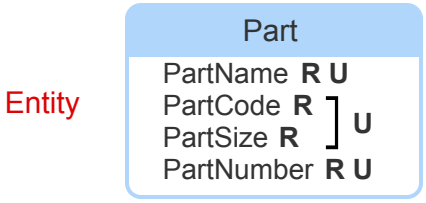
In the first step of logical design, entities become tables and attributes become columns. As tables and columns are specified, primary keys are selected. Primary keys must be unique and required (not NULL). Primary keys should also be:

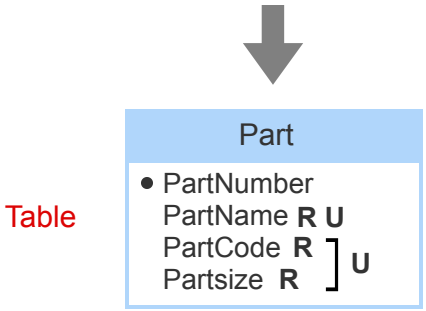
- *Stable*. Primary key values should not change. When a primary key value changes, statements that specify the old value must also change. Furthermore, the new primary key value must cascade to matching foreign keys.
- *Simple*. Primary key values should be easy to type and store. Small values are easy to specify in an SQL WHERE clause and speed up query processing. Ex: A 2-byte integer is easier to type and faster to process than a 15-byte character string.
- *Meaningless*. Primary keys should not contain descriptive information. Descriptive information occasionally changes, so primary keys containing descriptive information are unstable.

Stable, simple, and meaningless primary keys are desirable but not necessary. Occasionally, these guidelines may be violated.

In table diagrams, a bullet (●) indicates a primary key column.

PARTICIPATION ACTIVITY 37.7.1: Selecting primary keys.





Animation content:

Static figure:
Entity Part has attributes PartName R U, PartCode R, PartSize R, and PartNumber R U. (PartCode R, PartSize R) is followed by a bracket and U.

Table Part has columns PartName, PartCode R, PartSize R, and PartNumber R U. (PartCode R, PartSize R) is followed by a bracket and U. PartNumber is the primary key.

An arrow indicates that entity Part is implemented as table Part.

Step 1: PartName is unique and required, but also complex and meaningful. PartName is not a good primary key. The Part entity appears. Attribute PartName R U is highlighted.

Step 2: Each part code may come in several sizes, so PartCode is not individually unique. Attribute PartCode R is highlighted.

Step 3: (PartCode, PartSize) is unique but not a good primary key because part size is meaningful and may change. Attributes PartCode R and PartSize R are highlighted. The bracket and U following these attributes are highlighted.

Step 4: PartNumber is unique, required, stable, simple, and meaningless. PartNumber is a good primary key. PartNumber R U is highlighted. The Part table appears with primary key PartNumber.

Step 5: Primary keys are always required and unique, so R and U are unnecessary after PartNumber. In the Part table, the R and U following PartNumber disappear.

Animation captions:

1. PartName is unique and required, but also complex and meaningful. PartName is not a good primary key.
2. Each part code may come in several sizes, so PartCode is not individually unique.
3. (PartCode, PartSize) is unique but not a good primary key because part size is meaningful and may change.

4. PartNumber is unique, required, stable, simple, and meaningless. PartNumber is a good primary key.
5. Primary keys are always required and unique, so R and U are unnecessary after PartNumber.

PARTICIPATION ACTIVITY 37.7.2: Selecting primary keys.

1) A _____ attribute becomes a column that is never NULL.

Check Show answer

2) A _____ primary key is easy to specify in a WHERE clause.

Check Show answer

3) _____ columns contain no descriptive information and make good primary keys.

Check Show answer

4) A _____ primary key reduces cascading updates in the database.

Check Show answer

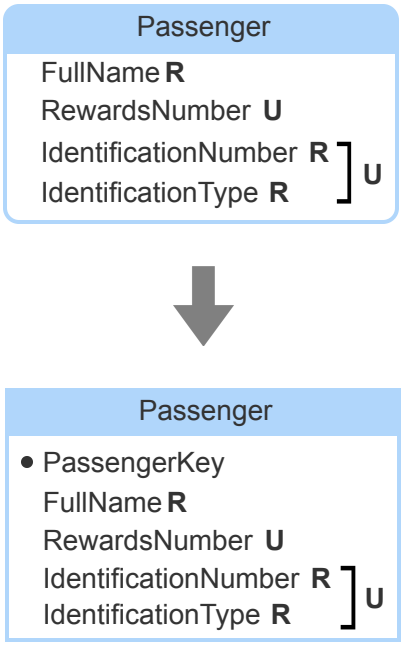
Implementing strong entities

A strong entity becomes a ***strong table***. The primary key must be unique and required, and should be stable, simple, and meaningless.

Simple primary keys are best for strong tables. If no simple primary key is available, a composite primary key may be selected. Alternatively, the database designer may create an artificial primary key. An ***artificial key*** is a simple primary key created by the database designer. Usually artificial keys are

integers, generated automatically by the database as new rows are inserted to the table. Artificial keys are stable, simple, and meaningless.

PARTICIPATION ACTIVITY 37.7.3: Implementing strong entities.



Animation content:

Static figure:
Entity Passenger has attributes FullName R, RewardsNumber U, IdentificationNumber R, and IdentificationType R. (IdentificationNumber R, IdentificationType R) is followed by a bracket and U.

Table Passenger has columns PassengerKey, FullName R, RewardsNumber U, IdentificationNumber R, and IdentificationType R. (IdentificationNumber R, IdentificationType R) is followed by a bracket and U. PassengerKey is the primary key.

An arrow indicates entity Passenger is implemented as table Passenger.

Step 1: In the initial design, every Passenger attribute becomes a table column. The Passenger entity appears. The Passenger table appears without the PassengerKey column.

Step 2: FullName is not unique and hence cannot be the primary key. In the Passenger table, FullName R is highlighted.

Step 3: RewardsNumber is not required and hence cannot be the primary key. In the Passenger table, RewardsNumber U is highlighted.

Step 4: A passenger may submit different identification over time. So (IdentificationNumber, IdentificationType) is not a good primary key. In the Passenger table, IdentificationNumber R and IdentificationType R are highlighted.

Step 5: Since no suitable primary key exists, an artificial key called PassengerKey is created. The PassengerKey column, preceded by a bullet, is added to the Passenger table.

Animation captions:

1. In the initial design, every Passenger attribute becomes a table column.
2. FullName is not unique and hence cannot be the primary key.
3. RewardsNumber is not required and hence cannot be the primary key.
4. A passenger may submit different identification over time. So (IdentificationNumber, IdentificationType) is not a good primary key.
5. Since no suitable primary key exists, an artificial key called PassengerKey is created.

PARTICIPATION ACTIVITY 37.7.4: Implementing strong entities.

Vehicle is a strong entity:

Vehicle
ManufacturerName **R**
MakeCode **R**
YearNumber **R**
LicensePlateNumber **R U**
VehicleIdentificationNumber **R U**
DealerInvoiceNumber **U**

Which of the following is a good primary key for the Vehicle table?

1) DealerInvoiceNumber

- ○ True
- ○ False

2) LicensePlateNumber

- ○ True
- ○ False

3) VehicleIdentificationNumber

- True
- False

4) (ManufacturerName, MakeCode, YearNumber)

- True
- False

5) A new artificial key

- True
- False

Implementing weak entities

A weak entity becomes a ***weak table***. A weak table has a foreign key that references the identifying table and implements the identifying relationship.

The primary key depends on the cardinality of the identifying relationship:

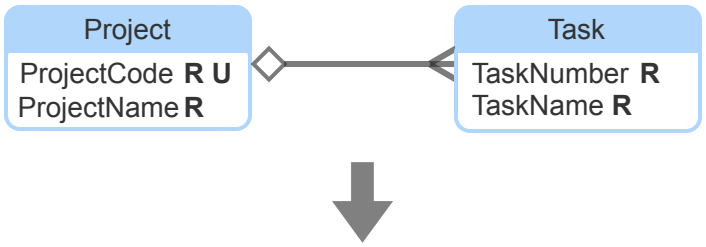
- Usually, the weak entity is plural. The primary key is the composite of the foreign key and another column.
- Occasionally, the weak entity is singular. The primary key is the foreign key only.

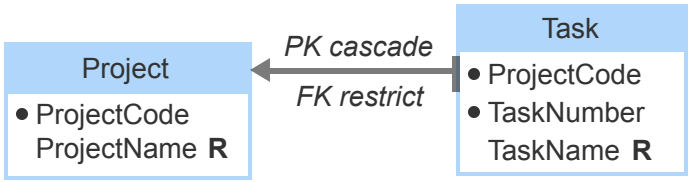
The foreign key usually has the following referential integrity actions:

- Cascade on primary key update and delete
- Restrict on foreign key insert and update

In table diagrams, an arrow indicates a foreign key. The arrow starts at the foreign key and points to the table containing the referenced primary key.

PARTICIPATION ACTIVITY 37.7.5: Implementing weak entities.





Animation content:

Static figure:
The Project entity has attributes ProjectCode R U and ProjectName R. The Task entity has attributes TaskNumber R and TaskName R. An unnamed relationship connects Project and Task, with a diamond on the Project side and a crow's foot symbol on the Task side.

The Project table has columns ProjectCode and ProjectName R. ProjectCode is the primary key. The Task table has columns ProjectCode, TaskNumber, and TaskName R. (ProjectCode, TaskNumber) is the primary key. An arrow points from ProjectCode in Task to the Project table. The arrow has captions PK cascade and FK restrict.

A larger arrow indicates that the Project and Task entities are implemented as the Project and Task tables.

Step 1: Project is a strong entity. Task is a weak entity identified by Project. The Project and Task entities appear. The unnamed relationship appears with the diamond but without the crow's foot symbol.

Step 2: ProjectCode is the primary key of the Project table. The Project table appears. The ProjectCode column is highlighted.

Step 3: The Task table has a foreign key ProjectCode that references Project. The Task table appears, without bullets next to ProjectCode and TaskNumber. ProjectCode of Task is highlighted. The arrow from ProjectCode of Task to ProjectCode of Project appears.

Step 4: If each project has at most one task, ProjectCode is unique in Task and becomes the primary key. A short bar appears across the unnamed relationship next to the Task entity. A bullet appears next to ProjectCode in Task.

Step 5: If each project has many tasks, ProjectCode is not unique and the primary key is composite: (ProjectCode, TaskNumber). The short bar next to the Task entity is replaced by a crow's foot symbol. A second bullet appears, next to TaskNumber in Task.

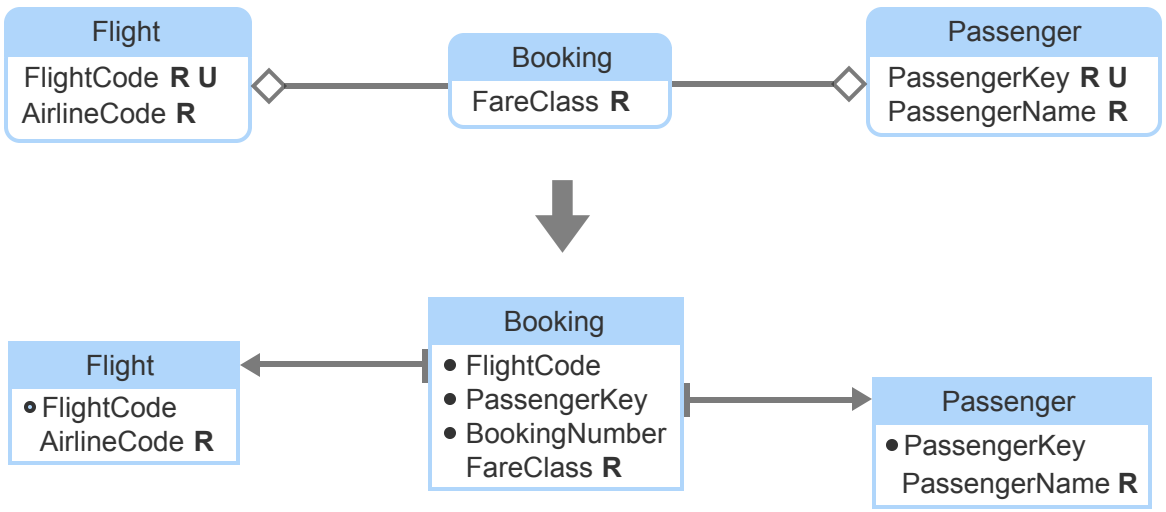
Step 6: Optionally, foreign key actions may appear on the diagram. The captions PK cascade and FK restrict appear next to the arrow.

Animation captions:

1. Project is a strong entity. Task is a weak entity identified by Project.
2. ProjectCode is the primary key of the Project table.
3. The Task table has a foreign key ProjectCode that references Project.
4. If each project has at most one task, ProjectCode is unique in Task and becomes the primary key.
5. If each project has many tasks, ProjectCode is not unique and the primary key is composite: (ProjectCode, TaskNumber).
6. Optionally, foreign key actions may appear on the diagram.

If a weak entity has several identifying relationships, the primary key includes one foreign key for each identifying relationship. The primary key may include an additional column, if necessary for uniqueness.

PARTICIPATION ACTIVITY 37.7.6: Implementing weak entities with several identifying relationships.



Animation content:

Static figure:
Entity Flight has attributes FlightCode R U and AirlineCode R. Entity Booking has attribute FareClass R. Entity Passenger has attributes PassengerKey R U and PassengerName R. An unnamed relationship connects Flight and Booking, with a diamond next to Flight. An unnamed relationship connects Booking and Passenger, with a diamond next to Passenger.

Table Flight has columns FlightCode and AirlineCode R. FlightCode is the primary key. Table Booking has columns FlightCode, PassengerKey, BookingNumber, and FareClass R. (FlightCode, PassengerKey, BookingNumber) is the primary key. Table Passenger has columns PassengerKey and PassengerName R. PassengerKey is the primary key. An arrow points from FlightCode of Booking to Flight. Another arrow points from PassengerKey of Booking to Passenger.

A larger arrow indicates the Flight, Booking, and Passenger entities are implemented as the Flight, Booking, and Passenger tables.

Step 1: Flight and Passenger are strong entities. Booking is a weak entity, identified by Flight and Passenger. The Flight, Booking, and Passenger entities appear. The unnamed relationships and diamonds appear.

Step 2: Flight and Passenger become strong tables. Primary keys are FlightCode and PassengerKey. The Flight and Passenger tables appear. Columns FlightCode and PassengerKey are highlighted, with bullets.

Step 3: Booking becomes a weak table, with foreign keys that reference the Flight and Passenger tables. The Booking table appears without column BookingNumber and without bullets next to FlightCode and PassengerKey. In Booking, columns FlightCode and PassengerKey are highlighted. Arrows appear, from FlightCode of Booking to Flight and from PassengerKey of Booking to Passenger.

Step 4: The primary key of Booking is (FlightCode, PassengerKey). In Booking, bullets appear next to FlightCode and PassengerKey.

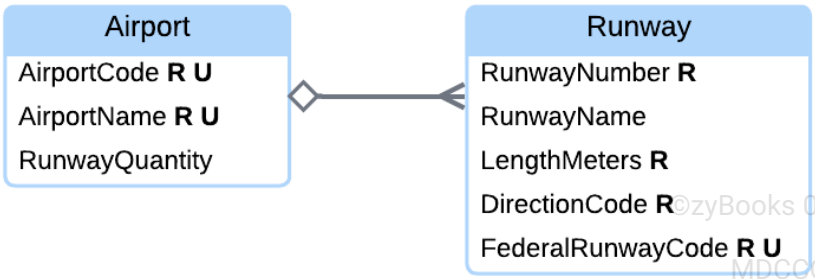
Step 5: If a passenger can make several bookings on the same flight, a third column is necessary in the primary key. The BookingNumber column, with a bullet, is added to Booking and highlighted.

Animation captions:

1. Flight and Passenger are strong entities. Booking is a weak entity, identified by Flight and Passenger.
2. Flight and Passenger become strong tables. Primary keys are FlightCode and PassengerKey.
3. Booking becomes a weak table, with foreign keys that reference the Flight and Passenger tables.
4. The primary key of Booking is (FlightCode, PassengerKey).
5. If a passenger can make several bookings on the same flight, a third column is necessary in the primary key.

PARTICIPATION ACTIVITY 37.7.7: Implementing weak entities.

Runway is a weak entity, identified by the strong entity Airport:



AirportCode is the primary key of the Airport table. At each airport, runway number is unique.

Which of the following is a good primary key for the Runway table?

1) (AirportCode, RunwayName)

- True
- False

2) (AirportCode, RunwayNumber)

- True
- False

3) (AirportName, RunwayNumber)

- True
- False

4) FederalRunwayCode

- True
- False

5) (AirportCode, DirectionCode)

- True
- False

Implementing supertype and subtype entities

A supertype entity becomes a ***supertype table***. A supertype entity that has an identifying attribute is implemented like a strong entity. A supertype entity that has an identifying relationship is implemented like a weak entity.

A subtype entity becomes a ***subtype table***:

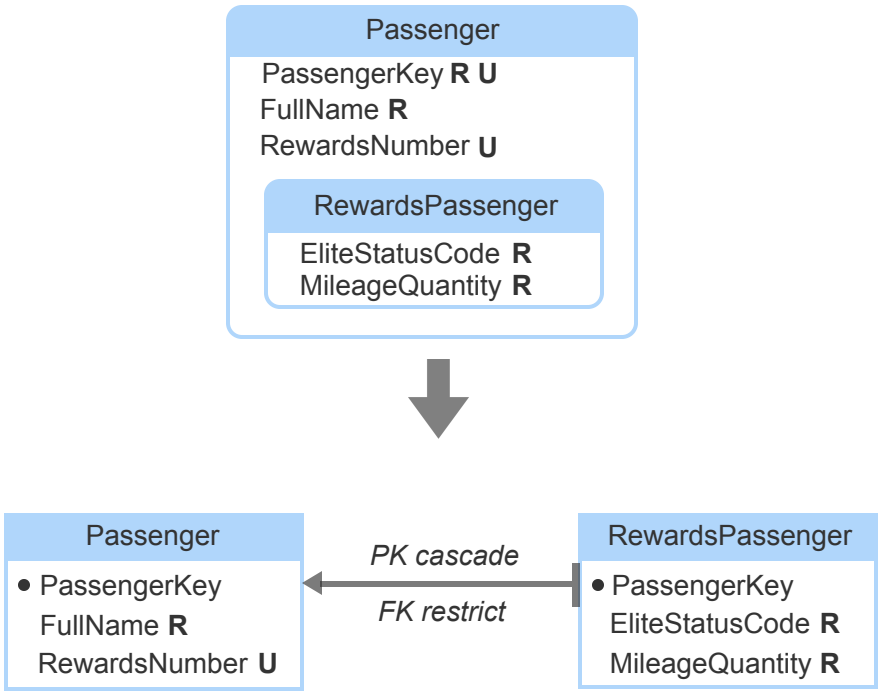
- The primary key is identical to the supertype primary key.
- The primary key is also a foreign key that references the supertype primary key.

The foreign key usually has the following referential integrity actions:

- Cascade on primary key update and delete
- Restrict on foreign key insert and update

The foreign key implements the IsA relationship between subtype and supertype entities.

PARTICIPATION ACTIVITY 37.7.8: Implementing subtype entities.



Animation content:

Static figure:
The Passenger entity has attributes PassengerKey R U, FullName R, and RewardsNumber U. The RewardsPassenger entity has attributes EliteStatusCode R and MileageQuantity R. RewardsPassenger is inside Passenger.

The Passenger table has columns PassengerKey, FullName R, and RewardsNumber U. PassengerKey is the primary key. The RewardsPassenger table has columns PassengerKey, EliteStatusCode R, and MileageQuantity R. PassengerKey is the primary key. An arrow points from PassengerKey of RewardsPassenger to Passenger, with captions PK cascade and FK restrict.

A larger arrow indicates the Passenger and RewardsPassenger entities are implemented as the Passenger and RewardsPassenger tables.

Step 1: RewardsPassenger is a subtype entity, containing passengers enrolled in an airline's mileage plan. The Passenger and RewardsPassenger entities appear.

Step 2: Passenger becomes a supertype table. The identifying attribute PassengerKey becomes the primary key. The Passenger table appears. The PassengerKey column is highlighted.

Step 3: The subtype primary key is identical to the supertype primary key. The RewardsPassenger table appears. The PassengerKey column of RewardsPassenger is highlighted.

Step 4: The subtype primary key is also a foreign key that references the supertype. The arrow from PassengerKey of RewardsPassenger to Passenger appears.

Step 5: Optionally, foreign key actions may appear on the diagram. The captions PK cascade and FK restrict appear next to the arrow.

Animation captions:

1. RewardsPassenger is a subtype entity, containing passengers enrolled in an airline's mileage plan.
2. Passenger becomes a supertype table. The identifying attribute PassengerKey becomes the primary key.
3. The subtype primary key is identical to the supertype primary key.
4. The subtype primary key is also a foreign key that references the supertype.
5. Optionally, foreign key actions may appear on the diagram.

PARTICIPATION ACTIVITY 37.7.9: Implementing subtype entities.

1) The subtype table primary key is identical to the _____ table primary key.

Check Show answer

2) The primary key of a subtype table is also a _____.

Check Show answer

3) The foreign key in the subtype table usually has the referential integrity action _____ on primary key delete.

Check Show answer

4) The foreign key in a subtype table implements the _____ relationship between subtype and supertype entities.

Check Show answer

Database design

The implement entities step creates an initial table design and specifies primary keys. If no suitable primary key is available, an artificial key is specified. The design is augmented in subsequent steps, as relationships and attributes are implemented. The final SQL specification stabilizes as tables are reviewed for normal form.

Some implementation decisions are affected by the database system. Ex: Some database systems have tools to automatically generate new artificial key values. A database designer might choose artificial primary keys more often when these tools are available.

Table 37.7.1: Implement entities.

Step	Activity
5A	Implement strong entities as tables.

5B	Create an artificial key when no suitable primary key exists.
5C	Implement weak entities as tables.
5D	Implement supertype and subtype entities as tables.

PARTICIPATION ACTIVITY 37.7.10: Database design.

1) After the 'implement entities' step is completed, table and column specifications are final.
   - True
   - False

2) All design decisions in the 'implement entities' step are affected by the database system.
   - True
   - False

3) Occasionally, database design skips a formal analysis phase and begins with logical design.
   - True
   - False

4) The activities of the 'implement entities' step are always executed in sequential order.
   - True
   - False

CHALLENGE ACTIVITY 37.7.1: Implementing entities.

539740.3879454.qx3zqy7

Start

Assume the following rules:

An Ethernet card's hardware address is long and hard to type.
No two Ethernet cards have the same hardware address.
An Ethernet card's hardware address does not contain descriptive information.
Each Ethernet card has a hardware address.
An Ethernet card's hardware address always stays the same.

Which of the following properties does the HardwareAddress attribute have?

- [ ] Unique
- [ ] Required
- [ ] Stable
- [ ] Simple
- [ ] Meaningless

1 2

Check Next

37.8 Implementing relationships

Implementing many-one relationships

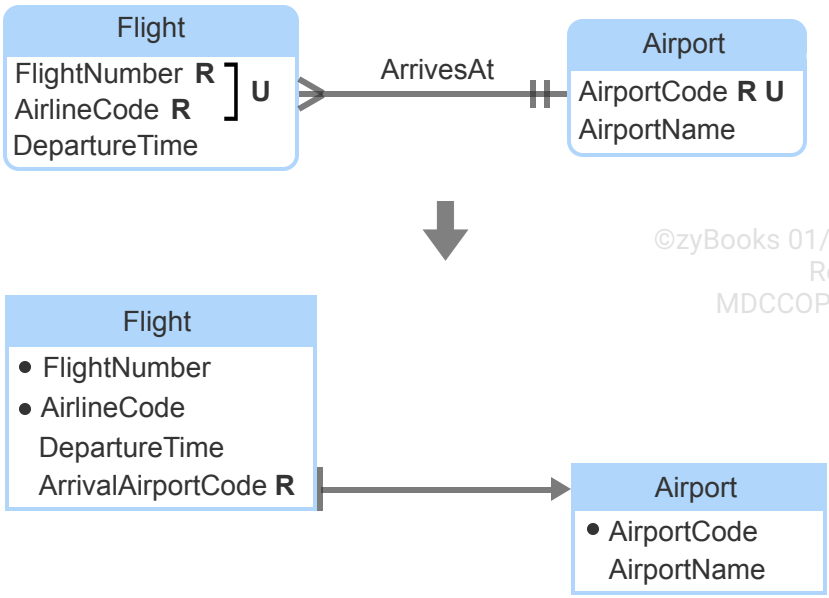
The 'implement entities' step converts identifying relationships into foreign keys. The 'implement relationships' step converts all other relationships into foreign keys or tables.

A many-one or one-many relationship becomes a foreign key:

- The foreign key goes in the table on the 'many' side and refers to the table on the 'one' side.
- If the entity on the 'one' side is required, the foreign key column is also required.

The foreign key name is the name of the referenced primary key, with an optional prefix. The prefix is usually derived from the relationship name and clarifies the meaning of the foreign key.

PARTICIPATION ACTIVITY 37.8.1: Implementing many-one relationships.



Animation content:

Static figure:
An ER diagram and a table diagram appear. A large arrow indicates that the ER diagram is implemented as the table diagram.

In the ER diagram, entity Flight has attributes FlightNumber R, AirlineCode R, and DepartureTime. (FlightNumber R, AirlineCode R) is followed by a bracket and U. Entity Airport has attributes AirportCode R U and AirportName. Relationship Flight-ArrivesAt-Airport has a crow's foot symbol on the Flight side and two short lines on the Airport side.

In the table diagram, table Flight has columns FlightNumber, AirlineCode, DepartureTime, and ArrivalAirportCode R. (FlightNumber, AirlineCode) is the primary key. Table Airport has columns AirportCode and AirportName. AirportCode is the primary key. An arrow points from ArrivalAirportCode of the Flight table to the Airport table.

Step 1: Each flight arrives at one airport. Each airport has many arriving flights. The ER diagram appears with only one short line next to Flight. The crow's foot symbol and short line are highlighted.

Step 2: Entities are implemented as tables. The primary key of Airport is AirportCode. The Flight table appears without ArrivalAirportCode R. The Airport table appears. AirportCode in the Airport table is highlighted.

Step 3: The many-one relationship becomes the foreign key ArrivalAirportCode in the table on the 'many' side. ArrivalAirportCode is added to the Flight table. The arrow from ArrivalAirportCode to the Airport table appears.

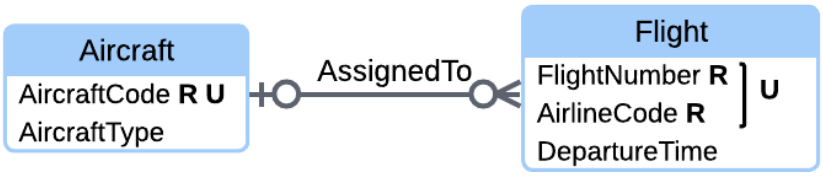
Step 4: If Airport is required in the ArrivesAt relationship, the foreign key ArrivalAirportCode is also required. In the ER diagram, the second short line appears next to Flight. In the table diagram, the R appears after ArrivalAirportCode.

Animation captions:

1. Each flight arrives at one airport. Each airport has many arriving flights.
2. Entities are implemented as tables. The primary key of Airport is AirportCode.
3. The many-one relationship becomes the foreign key ArrivalAirportCode in the table on the 'many' side.
4. If Airport is required in the ArrivesAt relationship, the foreign key ArrivalAirportCode is also required.

PARTICIPATION ACTIVITY 37.8.2: Implementing many-one relationships.

Refer to the following relationship:



The primary key of the Aircraft table is AircraftCode. The primary key of the Flight table is (FlightNumber, AirlineCode).

1) In which table is the foreign key placed?
   - Aircraft
   - Flight
   - The table with fewer rows

2) What is the name of the new foreign key?
   - FlightNumber
   - AircraftCode
   - AssignedAircraftCode

3) Are NULLs allowed in the foreign key column?

- Yes
- No
- Cannot be determined from the diagram.

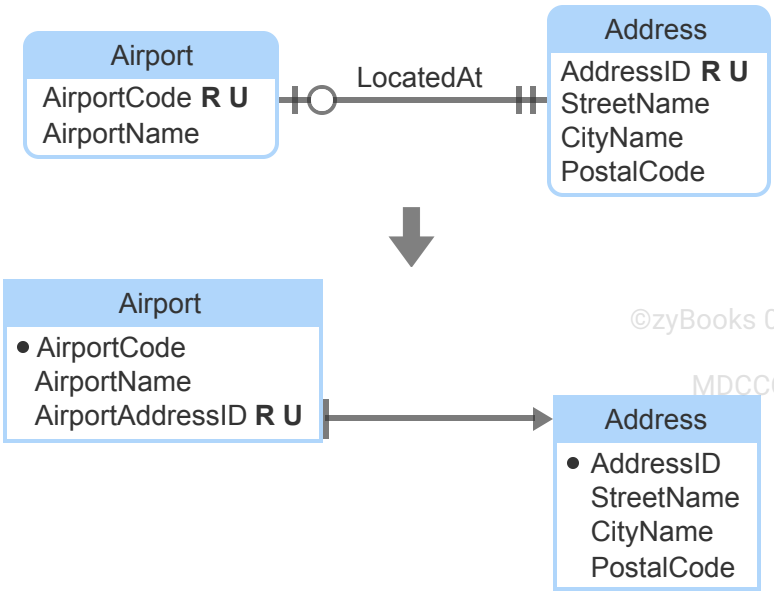
Implementing one-one relationships

A one-one relationship becomes a foreign key. The foreign key can go in the table on either side of the relationship. Usually, the foreign key is placed in the table with fewer rows, to minimize the number of NULL values.

- The foreign key refers to the table on the opposite side of the relationship.
- The foreign key column is unique.
- If the entity on the opposite side of the relationship is required, the foreign key column is also required.

The foreign key name is the name of the referenced primary key, with an optional prefix. The prefix is usually derived from the relationship name and clarifies the meaning of the foreign key.

PARTICIPATION ACTIVITY 37.8.3: Implementing one-one relationships.



Animation content:

Static figure:
An ER diagram and a table diagram appear. A large arrow indicates that the ER diagram is implemented as the table diagram.

In the ER diagram, entity Airport has attributes AirportCode R U and AirportName. Entity Address has attributes AddressID R U, SteeetName, CityName, and PostalCode. Relationship Airport-LocatedAt-Address has a short line and circle on the Airport side and two short lines on the Address side.

In the table diagram, table Airport has columns AirportCode, AirportName, and AirportAddressID R U. AirportCode is the primary key. Table Address has columns AddressID, StreetName, CityName and PostalCode. AddressID is the primary key. An arrow points from AirportAddressID of the Airport table to the Address table.

Step 1: Each airport has at most one address. Each address has at most one airport. The ER diagram appears. On the relationship, the short lines directly next to Airport and Address are highlighted.

Step 2: Every airport has an address, but some addresses are for people, not airports. So the Airport table has fewer rows. On the relationship, the circle next to Airport and second short line next to Address are highlighted. The Airport table appears without AirportAddressID RU. The Address table appears.

Step 3: The foreign key goes in the table with fewer rows. AirportAddressID is added to the Airport table. The arrow from AirportAddressID in the Airport table to the Address table appears.

Step 4: In this case, the prefix is derived from the table name instead of the relationship name. In AirportAddressID, the word Airport is highlighted.

Step 5: The Address entity is required, so the foreign key is also required. On the relationship, the second short line next to Address is highlighted. The R following AirportAddressID appears.

Step 6: The Airport entity is singular, so the foreign key is unique. The U following AirportAddressID R appears.

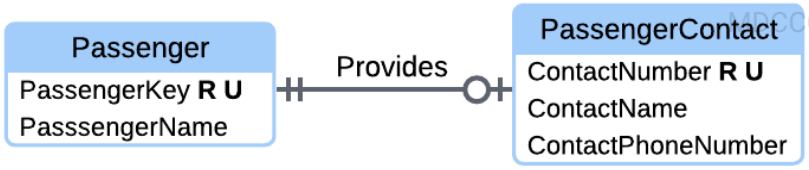
Animation captions:

1. Each airport has at most one address. Each address has at most one airport.
2. Every airport has an address, but some addresses are for people, not airports. So the Airport table has fewer rows.
3. The foreign key goes in the table with fewer rows.
4. In this case, the prefix is derived from the table name instead of the relationship name.

5. The Address entity is required, so the foreign key is also required.
6. The Airport entity is singular, so the foreign key is unique.

PARTICIPATION ACTIVITY 37.8.4: Implementing one-one relationships.

Some passengers provide a personal contact to the airline, in case of emergency:



The primary key of the Passenger table is PassengerKey. The primary key of the PassengerContact table is ContactNumber.

1) In which table is the foreign key placed?
   - Passenger
   - PassengerContact
   - Cannot be determined from the diagram.

2) What is the name of the new foreign key?
   - Passenger
   - ContactNumber
   - PassengerKey

3) Are NULLs allowed in the foreign key column?
   - Yes
   - No
   - Cannot be determined from the diagram.

Implementing many-many relationships

A many-many relationship becomes a new weak table:

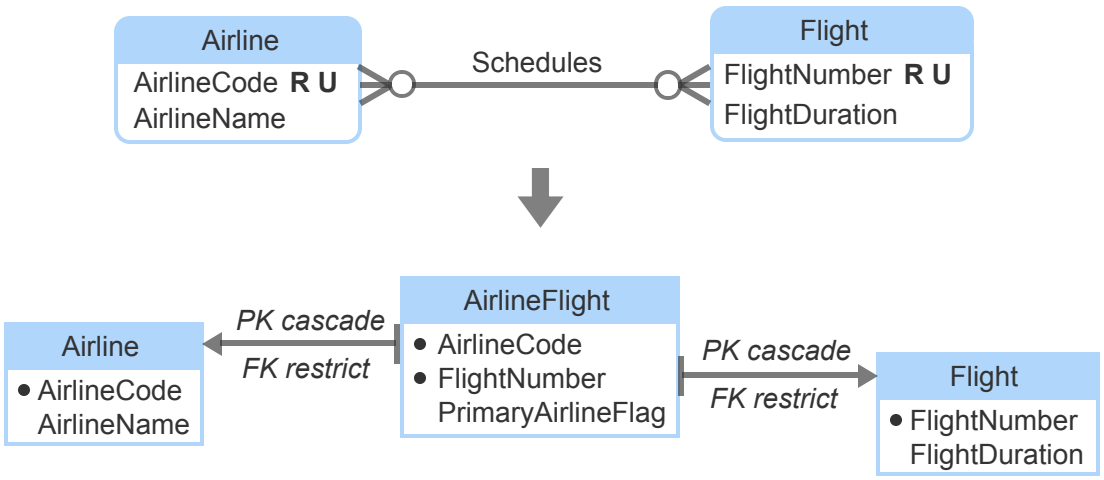
- The new table contains two foreign keys, referring to the primary keys of the related tables.

- The primary key of the new table is the composite of the two foreign keys.
- The new table is identified by the related tables, so primary key cascade and foreign key restrict rules are usually specified.

Occasionally, an attribute describes a many-many relationship and becomes a column of the new weak table.

The new table name consists of the related table names with an optional qualifier in between. The qualifier is usually derived from the relationship name and clarifies the meaning of the table.

PARTICIPATION ACTIVITY 37.8.5: Implementing many-many relationships.



Animation content:

Static figure:
An ER diagram and a table diagram appear. A large arrow indicates the ER diagram is implemented as the table diagram.

In the ER diagram, entity Airline has attributes AirlineCode R U and AirlineName. Entity Flight has attributes FlightNumber R U and FlightDuration. Relationship Airline-Schedules-Flight has a crow's foot symbol and circle next to both Airline and Flight.

In the table diagram, table Airline has attributes AirlineCode and AirlineName. AirlineCode is the primary key. Table Flight has attributes FlightNumber and FlightDuration. FlightNumber is the primary key. Table AirlineFlight has attributes AirlineCode, FlightNumber, and PrimaryAirlineFlag. (AirlineCode, FlightNumber) is the primary key. An arrow points from AirlineCode in the AirlineFlight

table to the Airline table. Another arrow points from FlightNumber in the AirlineFlight table to the Flight table. Both arrows have captions PK cascade and FK restrict.

Step 1: Each airline has many flights. Occasionally, the same flight is associated with several partner airlines. The ER diagram appears. The crow's foot symbols next to both entities are highlighted.

Step 2: Entities are implemented as tables. Primary keys are AirlineCode and FlightNumber. The Airline and Flight tables appear. The primary keys of both tables are highlighted.

Step 3: The many-many relationship becomes a new weak table. Foreign keys refer to the related tables. The AirlineFlight table appears without PrimaryAirlineFlag. In the AirlineFlight table, AirlineCode and FlightNumber are highlighted. Both arrows appear.

Step 4: The primary key is the composite of the foreign keys. Bullets appear before AirlineCode and FlightNumber in the AirlineFlight table.

Step 5: Occasionally, the new table has additional columns. PrimaryAirlineFlag is true for the primary airline that operates the flight. PrimaryAirlineFlag is added to the AirlineFlight table.

Step 6: Optionally, foreign key rules may appear on the diagram. The captions PC cascade and FK restrict appear next to both arrows.

Animation captions:

1. Each airline has many flights. Occasionally, the same flight is associated with several partner airlines.
2. Entities are implemented as tables. Primary keys are AirlineCode and FlightNumber.
3. The many-many relationship becomes a new weak table. Foreign keys refer to the related tables.
4. The primary key is the composite of the foreign keys.
5. Occasionally, the new table has additional columns. PrimaryAirlineFlag is true for the primary airline that operates the flight.
6. Optionally, foreign key rules may appear on the diagram.

PARTICIPATION ACTIVITY 37.8.6: Implementing many-many relationships.

A passenger may record several credit cards, and a credit card may be shared by family members:

The primary key of the Passenger table is PassengerKey. The primary key of the CreditCard table is CardNumber.

1) What is (are) the foreign key(s) in the new table?

- The new table contains no foreign keys.
- PassengerKey
- CardNumber
- PassengerKey and CardNumber

2) What is the primary key of the new table?

- CardNumber
- (PassengerKey, CardNumber)
- The composite of PassengerKey, CardNumber, and a third column of the new table.

3) What is the name of the new table?

- PassengerCreditCard
- Owns
- PassengerCreditCardOwnership

Database design

Identifying relationships become foreign keys in the 'implement entities' step. All other relationships become foreign keys or tables in the 'implement relationships' step.

Each many-one and one-one relationship becomes a new foreign key. Foreign key names are the referenced primary key name, with an optional prefix derived from the relationship name.

Each many-many relationship becomes a new weak table. The table contains two foreign keys that refer to the related tables. The table name consists of the related table names, with an optional qualifier derived from the relationship name.

Table 37.8.1: Implement relationships.

Step	Activities

6A	Implement many-one relationships as a foreign key on the 'many' side.
6B	Implement one-one relationships as a foreign key in the table with fewer rows.
6C	Implement many-many relationships as new weak tables.

PARTICIPATION ACTIVITY 37.8.7: Implementing relationships.

1) Foreign keys always have the same name as the referenced primary key.
   - True
   - False

2) Many-one and one-one relationships are always implemented before many-many relationships.
   - True
   - False

3) The primary key of a table that implements a many-many relationship is composite.
   - True
   - False

CHALLENGE ACTIVITY 37.8.1: Implementing relationships.

539740.3879454.qx3zqy7

Start

These entities and attributes become tables and primary keys.

Implement the relationship as a foreign key.

In which table is the foreign key placed?

Pick

What is the name of the new foreign key?

Pick

Are NULLs allowed in the foreign key column?

Pick

1 2 3 4

Check Next

37.9 Implementing attributes

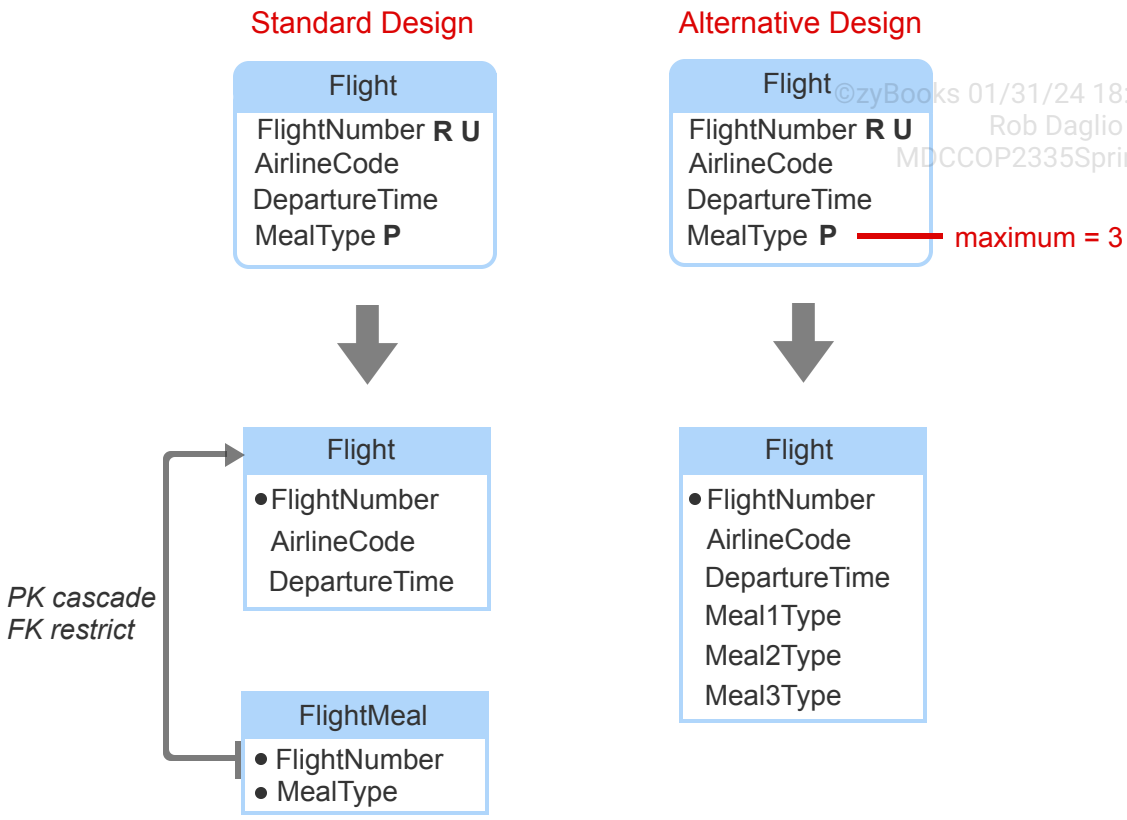
Implementing plural attributes

In the 'implement entities' step, entities become tables and attributes become columns. Singular attributes remain in the initial table, but plural attributes move to a new weak table:

- The new table contains the plural attribute and a foreign key referencing the initial table.
- The primary key of the new table is the composite of the plural attribute and the foreign key.
- The new table is identified by the initial table, so primary key cascade and foreign key restrict rules are specified.
- The new table name consists of the initial table name followed by the attribute name.

If a plural attribute has a small, fixed maximum, the plural attribute can be implemented as multiple columns in the initial table. However, implementing plural attributes in a new table simplifies queries and is usually a better solution.

PARTICIPATION ACTIVITY 37.9.1: Implementing plural attributes.



Animation content:

Static figure:
Two diagrams appear, with captions Standard Design and Alternative Design. In both diagrams, the Flight entity appears above table(s). A large arrow indicates the entity is implemented as the table(s).

In the Standard Design diagram, entity Flight has attributes FlightNumber R U, AirlineCode, DepartureTime, and MealType P. Table Flight has columns FlightNumber, AirlineCode, and DepartureTime. FlightNumber is the primary key. Table FlightMeal has columns FlightNumber and MealType. (FlightNumber, MealType) is the primary key. An arrow points from FlightNumber of table FlightMeal to the Flight table, with captions PK cascade and FK restrict.

In the Alternate Design diagram, the Flight entity is the same. Column MealType P has caption maximum = 3. Only the Flight table appears, with columns FlightNumber, AirlineCode, DepartureTime, Meal1Type, Meal2Type, and Meal3Type. FlightNumber is the primary key.

Step 1: Each flight offers several kinds of in-flight meals, so MealType is a plural attribute. The

caption Standard Design and entity Flight appear. Attribute MealType P is highlighted.

Step 2: Singular attributes are implemented in the Flight table. FlightNumber is the primary key. Table Flight appears.

Step 3: The plural attribute is implemented in a new table, along with a foreign key that refers to Flight. Table FlightMeal appears without primary key bullets. The arrow from FlightNumber in table FlightMeal to table Flight appears, with captions PK cascade and FK restrict.

Step 4: The primary key of the new table is the composite of the plural attribute and the foreign key. The primary key bullets in FlightMeal appear.

Step 5: If a flight has at most three kinds of meals, then MealType has a small, fixed maximum. The caption Alternative Design and the second Flight entity appear. Caption maximum = 3 appears next to attribute MealType P.

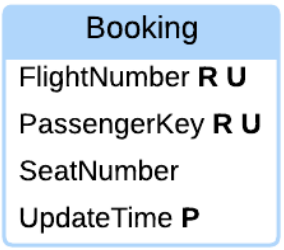
Step 6: Plural attributes with a small, fixed maximum can be implemented as multiple columns in Flight. The second Flight table appears, with additional columns Meal1Type, Meal2Type, and Meal3Type.

Animation captions:

1. Each flight offers several kinds of in-flight meals, so MealType is a plural attribute.
2. Singular attributes are implemented in the Flight table. FlightNumber is the primary key.
3. The plural attribute is implemented in a new table, along with a foreign key that refers to Flight.
4. The primary key of the new table is the composite of the plural attribute and the foreign key.
5. If a flight has at most three kinds of meals, then MealType has a small, fixed maximum.
6. Plural attributes with a small, fixed maximum can be implemented as multiple columns in Flight.

PARTICIPATION ACTIVITY 37.9.2: Implementing plural attributes.

The plural attribute UpdateTime records dates and times when passengers change their seat number:



Singular attributes are implemented in the Booking table, which has primary key (FlightNumber, PassengerKey). The plural attribute is implemented in a new weak table.

1) What is the name of the new table?

- Booking
- UpdateTime
- BookingUpdateTime

2) What is the primary key of the new table?

- (FlightNumber, PassengerKey)
- (FlightNumber, PassengerKey, UpdateTime)
- UpdateTime

3) Which column is a foreign key in the new table?

- FlightNumber only
- (FlightNumber, PassengerKey)
- The new table does not contain a foreign key

Implementing attribute types

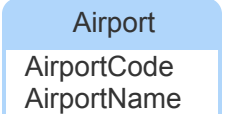
During analysis, a list of standard attribute types is established. During logical design, an SQL data type is defined for each attribute type. Attribute types and the corresponding data types are documented in the glossary.

Each attribute name includes a standard attribute type as a suffix. The attribute type determines the data type of the corresponding column.

PARTICIPATION ACTIVITY 37.9.3: Implementing attribute types.

Glossary

```
Attribute Type: Code
Data Type: CHAR(3)
Description: 'Code' is used for
three-character attributes which
```



```
identify objects. Characters in
codes are alphabetic or numeric.
Punctuation is not allowed. Examples
are airport codes, such as SFO,
and time zones, such as PDT.

Attribute Type: Name

Data Type: VARCHAR(30)

Description: 'Name' describes
attributes which label information
with free-form names up to 30
characters. Examples include people,
product, and company names.
```

CountryCode
CityName

```
CREATE TABLE Airport (
  AirportCode CHAR(3),
  AirportName VARCHAR(30),
  CountryCode CHAR(3),
  CityName VARCHAR(30)
);
```


Animation content:

Static figure:
The caption Glossary appears above the following text:
Attribute Type: Code
Data Type: CHAR(3)
Description: 'Code' is used for three-character attributes which identify objects. Characters in codes are alphabetic or numeric. Punctuation is not allowed. Examples are airport codes, such as SFO, and time zones, such as PDT.
Attribute Type: Name
Data Type: VARCHAR(30)
Description: 'Name' describes attributes which label information with free-form names up to 30 characters. Examples include people, product, and company names.

An entity appears above an SQL statement. A large arrow indicates the entity is implemented as the SQL statement:

Entity Airport has columns AirportCode, AirportName, CountryCode, and CityName.

Begin SQL code:

```
CREATE TABLE Airport (
  AirportCode CHAR(3),
  AirportName VARCHAR(30),
  CountryCode CHAR(3),
  CityName VARCHAR(30)
);
```

End SQL code.

Step 1: The glossary specifies data types CHAR(3) and VARCHAR(30) for attribute types Code and

Name. The text under caption Glossary appears.

Step 2: Two attributes have type Code and two have type Name. The Airport entity appears. Arrows point from AttributeType: Code to attributes AirportCode and CountryCode. Arrows point from AttributeType: Name to attributes AirportName and CityName.

Step 3: Attribute type Code is implemented as data type CHAR(3). The SQL statement appears. Under the caption Glossary, Code and CHAR(3) are highlighted. In the SQL statement, the Code suffixes and CHAR(3) data types are highlighted.

Step 4: Attribute type Name is implemented as data type VARCHAR(30). Under the caption Glossary, Name and VARCHAR(30) are highlighted. In the SQL statement, the Name suffixes and VARCHAR(30) data types are highlighted.

Animation captions:

1. The glossary specifies data types CHAR(3) and VARCHAR(30) for attribute types Code and Name.
2. Two attributes have type Code and two have type Name.
3. Attribute type Code is implemented as data type CHAR(3).
4. Attribute type Name is implemented as data type VARCHAR(30).

PARTICIPATION ACTIVITY 37.9.4: Implementing attribute types.

1) Attribute names always include an attribute type.
   - True
   - False

2) Data types are independent of the database system.
   - True
   - False

3) Data type depends on attribute cardinality.
   - True
   - False

Implementing attribute cardinality

Since plural attributes are implemented as singular columns, as described above, all columns are singular. Required or unique attributes become required or unique columns. Like attributes, columns are presumed optional and not unique unless followed by R or U in the table diagram.

Relationship cardinality determines constraints on foreign key columns:

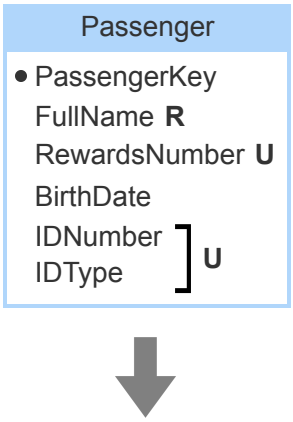
- If the table *referenced by* the foreign key implements a *required* entity, the column is required.
- If the table *containing* the foreign key implements a *singular* entity, the column is unique.

Table diagrams are implemented as CREATE TABLE statements:

- NOT NULL is specified for required columns.
- UNIQUE is specified for unique columns.
- PRIMARY KEY is specified for primary key columns.

Composite unique columns and composite primary keys cannot be specified in a column definition clause. Composite constraints require an additional clause in the CREATE TABLE statement.

PARTICIPATION ACTIVITY 37.9.5: Implementing attribute cardinality.




```
CREATE TABLE Passenger(
  PassengerKey INT PRIMARY KEY,
  FullName VARCHAR(30) NOT NULL,
  RewardsNumber INT UNIQUE
  BirthDate DATE,
  IDNumber INT,
  IDType CHAR(1),
  UNIQUE (IDNumber, IDType)
);
```


Animation content:

Static figure:
A table appears above an SQL statement. A large arrow indicates the table is implemented as the SQL statement:

Table Passenger has columns PassengerKey, FullName R, RewardsNumber U, BirthDate, IDNumber, and IDType. PassengerKey is the primary key. (IDNumber, IDType) is followed by a bracket and U.

Begin SQL code:

```
CREATE TABLE Passenger(
  PassengerKey INT PRIMARY KEY,
  FullName VARCHAR(30) NOT NULL,
  RewardsNumber INT UNIQUE
  BirthDate DATE,
  IDNumber INT,
  IDType CHAR(1),
  UNIQUE (IDNumber, IDType)
);
```

End SQL code.

Step 1: The Passenger table is implemented as a CREATE TABLE statement. Entity Passenger appears. The first and last line of the SQL statement appear.

Step 2: A bullet is implemented with keywords PRIMARY KEY. The PassengerKey column definition appears in the SQL statement.

Step 3: R is implemented with keywords NOT NULL. The FullName column definition appears in the SQL statement.

Step 4: U is implemented with keyword UNIQUE. The RewardsNumber column definition appears in the SQL statement.

Step 5: A column with no R, U, or bullet may be NULL and not unique. The BirthDate column definition appears in the SQL statement.

Step 6: A constraint on a composite column is implemented as a separate clause. The IDNumber and IDType column definitions, and the UNIQUE clause, appear in the SQL statement.

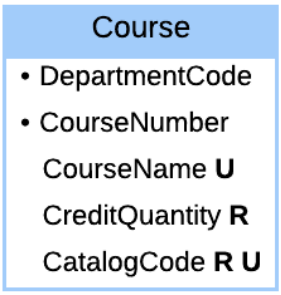
Animation captions:

1. The Passenger table is implemented as a CREATE TABLE statement.
2. A bullet is implemented with keywords PRIMARY KEY.
3. R is implemented with keywords NOT NULL.
4. U is implemented with keyword UNIQUE.
5. A column with no R, U, or bullet may be NULL and not unique.

6. A constraint on a composite column is implemented as a separate clause.

PARTICIPATION ACTIVITY 37.9.6: Implementing attribute cardinality.

Refer to the following table diagram:



The table is implemented with the following statement:

```
CREATE TABLE Course (
  DepartmentCode CHAR(3),
  CourseNumber INT,
  CourseName VARCHAR(30) __A__,
  CreditQuantity TINYINT __B__,
  CatalogCode CHAR(3) __C__,
  __D__ (DepartmentCode, CourseName)
);
```

1) What is A?

Check Show answer

2) What is B?

Check Show answer

3) What is C?

Check Show answer

4) What is D?

Check Show answer

Database design

The 'implementing attributes' step specifies columns, column constraints, and data types. Plural attributes become new weak tables. Unique and required attributes are implemented with UNIQUE, NOT NULL, and PRIMARY KEY keywords.

After the 'implementing attributes' step, the database is completely specified as CREATE TABLE statements. The final step, 'review tables for third normal form', ensures that tables do not contain redundant data and fine-tunes the design if necessary.

Table 37.9.1: Implement attributes.

Step	Activities
7A	Implement plural attributes as new weak tables.
7B	Specify cascade and restrict rules on new foreign keys in weak tables.
7C	Specify column data types corresponding to attribute types.
7D	Enforce relationship and attribute cardinality with UNIQUE and NOT NULL keywords.

PARTICIPATION ACTIVITY 37.9.7: Implementing attributes.

1) Plural attributes are implemented as new weak tables.

- True
- False

2) The logical design phase ends when attributes have been implemented as columns.

- True
- False

3) An attribute name indicates the data type of the corresponding column.

- True
- False

CHALLENGE ACTIVITY 37.9.1: Implementing attributes.

539740.3879454.qx3zqy7

Start

Account
AccountID R U AccountName R U CreationTimestamp R

Which properties apply to each attribute?

	Plural	Required	Unique
AccountID	☐	☐	☐
AccountName	☐	☐	☐
CreationTimestamp	☐	☐	☐

1 2 3 4

Check Next

37.10 First, second, and third normal form

Functional dependence

Column A ***depends on*** column B means each B value is related to at most one A value. Columns A and B may be simple or composite. 'A depends on B' is denoted B → A.

Dependence of one column on another is called ***functional dependence***. Functional dependence reflects business rules. Ex: "Each student receives one letter grade in a course" indicates the Grade column depends on the composite column (StudentID, CourseCode).

Examples in this section illustrate functional dependence with static table data. However, functional dependence cannot be inferred from values in a table at one point in time. Today, each value of column B may relate to at most one value of column A, but future updates may alter the data.

PARTICIPATION ACTIVITY 37.10.1: Functional dependence.

Booking

● PassengerNumber →	PassengerName	● FlightCode	FareClass →	BoardingZoneNumber
222	Elvira Yin	AZ312	First	1
222	Elvira Yin	BF999	Economy	3
222	Elvira Yin	GC848	Business	3
333	Deepak Chopra	GC848	First	1
444	Mary Hutcher	GC848	First	1

Animation content:

Static figure:
Table Booking has columns PassengerNumber, PassengerName, FlightCode, FareClass, and BoardingZoneNumber. (PassengerNumber, FlightCode) is the primary key. An arrow points from PasengerNumber to PassengerName. An arrow points from FareClass to BoardingZoneNumber.
Booking has five rows:
222, Elvira Yin, AZ312, First, 1
222, Elvira Yin, BF999, Economy, 3
222, Elvira Yin, GC848, Business, 3
333, Deepak Chopra, GC848, First, 1
444, Mary Hutcher, GC848, First, 1

The three instances of (222, Elvira Yin) are highlighted. The three instances of (First, 1) are highlighted.

Step 1: Each passenger number always has the same name, so PassengerName depends on PassengerNumber. PassengerNumber, PassengerName, and the arrow between are highlighted.

Step 2: Ex: Passenger 222 is always named Elvira Yin. The three instances of (222, Elvira Yin) are highlighted.

Step 3: Each fare class always has the same boarding zone, so BoardingZoneNumber depends on FareClass. FareClass, BoardingZoneNumber, and the arrow between are highlighted.

Step 4: Ex: First class is always assigned zone 1. The three instances of (First, 1) are highlighted.

Animation captions:

1. Each passenger number always has the same name, so PassengerName depends on PassengerNumber.
2. Ex: Passenger 222 is always named Elvira Yin.
3. Each fare class always has the same boarding zone, so BoardingZoneNumber depends on FareClass.
4. Ex: First class is always assigned zone 1.

Functional dependence is not the only type of dependence. ***Multivalued dependence*** and ***join dependence*** entail dependencies between three or more columns. However, multivalued and join dependencies are complex, uncommon, and not discussed in this material.

PARTICIPATION ACTIVITY 37.10.2: Functional dependence.

Refer to the Booking table in the animation above.

1) B → A means each value of A relates to at most one value of B.

   - True
   - False

2) 'PassengerName depends on PassengerNumber' means each name is related to at most one number.

   - True
   - False

3) "Column A does *not* depend on column B" can, in some cases, be inferred from static data in columns A and B.

- True
- False

4) FareClass depends on BoardingZoneNumber.

- True
- False

5) FareClass depends on PassengerNumber.

- True
- False

6) FareClass depends on the primary key (PassengerNumber, FlightCode).

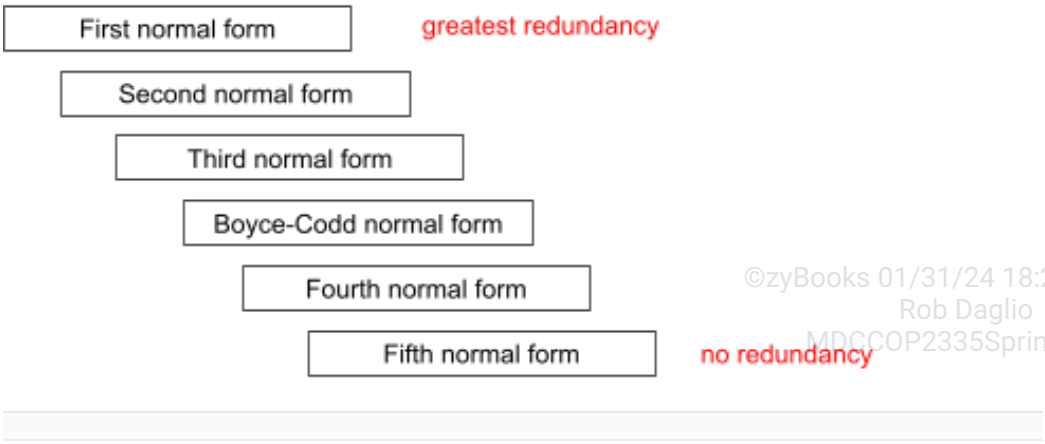
- True
- False

Normal forms

Redundancy is the repetition of related values in a table. Ex: In the Booking table, above, (222, Elvira Yin) is repeated. Redundancy causes database management problems. When related values are updated, all copies must be changed, which makes queries slow and complex. If copies are not updated uniformly, the copies become inconsistent and the correct version is uncertain.

Normal forms are rules for designing tables with less redundancy. Normal forms are numbered, first through fifth. An additional normal form, Boyce-Codd, is an improved version of third normal form. The six normal forms comprise a sequence, with each successive normal form allowing less redundancy.

Figure 37.10.1: Normal forms.



Redundancy occurs when a dependence is on a column that is not unique. Ex: In the Booking table, (222, Elvira Yin) is repeated because PassengerName depends on PassengerNumber, which is not unique. Boyce-Codd normal form eliminates all dependencies on non-unique columns and, in practice, is the most important normal form.

Fourth and fifth normal forms eliminate multivalued and join dependencies, respectively. Since multivalued and join dependencies are complex and uncommon, fourth and fifth normal forms are primarily of theoretical interest.

First, second, and third normal forms are described below. Boyce-Codd normal form is described elsewhere in this material. Fourth and fifth normal forms are not described in this material.

PARTICIPATION ACTIVITY 37.10.3: Redundancy.

1) Redundancy is the repetition of an individual value.
   - True
   - False

2) Tables that allow redundancy might contain inconsistent data.
   - True
   - False

3) Dependence on a primary key can cause redundancy.
   - True
   - False

PARTICIPATION ACTIVITY 37.10.4: Normal forms.

Match each normal form to a description.

If unable to drag and drop, refresh the page.

Fifth normal form **Boyce-Codd normal form** **Fourth normal form**

First normal form

	Eliminates multivalued dependencies and associated redundancy.
	Eliminates all redundancy arising from dependencies on non-unique columns.
	Allows the most redundancy of any normal form.
	Eliminates join dependencies and associated redundancy.

Reset

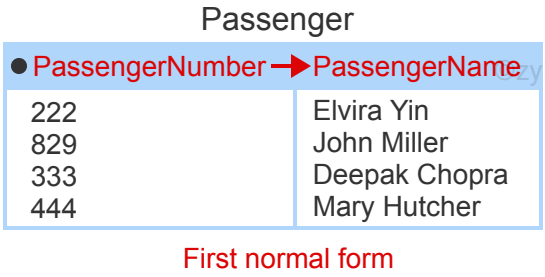
First normal form

Every cell of a table contains exactly one value. A table is in ***first normal form*** when, in addition, the table has a primary key. This definition has two corollaries:

- *In a first normal form table, every non-key column depends on the primary key*. Each primary key value appears in exactly one row, and each non-key cell contains exactly one value. So each primary key value is related to exactly one non-key value.
- *A first normal form table has no duplicate rows*. Every row contains a different primary key value and therefore every row is different.

In practice, databases allow tables with duplicate rows and no primary key. However, such tables are usually temporary. Ex: A database user might load external data with duplicate rows into a temporary table. Normally, when data is moved to a permanent table, duplicate rows are removed and a primary key is created.

PARTICIPATION ACTIVITY | 37.10.5: In a first normal form table, every non-key column depends on the primary key.



Animation content:

Static figure:
Table Passenger has columns PassengerNumber and PassengerName. PassengerNumber is a primary key. An arrow points from PassengerNumber to PassengerName. Passenger and has four rows:
222, Elvira Yin
829, John Miller
333, Depak Chopra
444, Mary Hutcher

Passenger is labeled First normal form.

Step 1: The Passenger table has a primary key and therefore is in first normal form. PassengerNumber is highlighted.

Step 2: Since the primary key is unique, each primary key value appears on one row only. The values in the PassengerNumber column are highlighted.

Step 3: Since each cell contains one value, each primary key value is related to one PassengerName value. Each row is highlighted

Step 4: Therefore, PassengerName depends on PassengerNumber. PassengerNumber, PassengerName, and the arrow between are highlighted.

Animation captions:

1. The Passenger table has a primary key and therefore is in first normal form.
2. Since the primary key is unique, each primary key value appears on one row only.

3. Since each cell contains one value, each primary key value is related to one PassengerName value.
4. Therefore, PassengerName depends on PassengerNumber.

Alternative definitions of first normal form

First normal form is commonly defined in several ways:

- *The table has a primary key.*
- *Every non-key column depends on the primary key.*
- *The table cannot have duplicate rows.*
- *Every cell contains exactly one value.*

The first three definitions are equivalent, but the last is different. The last definition is true of any relational table, and allows for duplicate rows and no primary key.

PARTICIPATION ACTIVITY 37.10.6: First normal form.

1) Since a string contains multiple characters, a string represents multiple values in a table cell.
   - ○ True
   - ○ False

2) In a table that never contains duplicate rows, non-key columns always depend on the primary key.
   - ○ True
   - ○ False

3) In a first normal form table, non-key columns depend *only* on the primary key.
   - ○ True
   - ○ False

4) In relational theory, all tables are in first normal form.

- ○ True
- ○ False

Second normal form

A table is in ***second normal form*** when all non-key columns depend on the whole primary key. In other words, a non-key column cannot depend on part of a composite primary key. A table with a simple primary key is automatically in second normal form.

A table in second normal form is also in first normal form, by definition.

PARTICIPATION ACTIVITY 37.10.7: Second normal form eliminates some redundancy.

Booking

● PassengerNumber	PassengerName	● FlightCode	FareClass	BoardingZoneNumber
222	Elvira Yin	AZ312	First	1
222	Elvira Yin	BF999	Economy	3
222	Elvira Yin	GC848	Business	3
333	Deepak Chopra	GC848	First	1
444	Mary Hutcher	GC848	First	1

First normal form

Booking

● PassengerNumber	● FlightCode	FareClass	BoardingZoneNumber
222	AZ312	First	1
222	BF999	Economy	3
222	GC848	Business	3
333	GC848	First	1
444	GC848	First	1

Passenger

● PassengerNumber	Passenger
222	Elvira Yin
333	Deepak Ch
444	Mary Hutcl

Second normal form

Animation content:

Static figure:
Three tables appear, named Booking, Booking, and Passenger. A large arrow indicates the first Booking table is decomposed into the second Booking table and the Passenger table.

The first Booking table has columns PassengerNumber, PassengerName, FlightCode, FareClass, and BoardingZoneNumber. (PassengerNumber, FlightCode) is the primary key. Booking has caption First normal form and five rows:

222, Elvira Yin, AZ312, First, 1
222, Elvira Yin, BF999, Economy, 3
222, Elvira Yin, GC848, Business, 3
333, Deepak Chopra, GC848, First, 1
444, Mary Hutcher, GC848, First, 1

The three instances of (222, Elvira Yin) are highlighted.

The second Booking table is the same as the first, except the PassengerName column does not appear. The caption is Second normal form.

The Passenger table has columns PassengerNumber and PassengerName. PassengerNumber is the primary key. Passenger has three rows:
222, Elvira Yin, GC848
333, Deepak Chopra
444, Mary Hutcher

Step 1: PassengerName depends on PassengerNumber. Dependence on part of the primary key causes repetition of (222, Elvira Yin). The first Booking table appears. The three instances of (222, Elvira Yin) are highlighted.

Step 2: Removing PassengerName from Booking eliminates redundancy. Booking is now in second normal form. An X appears above column PassengerName. The second Booking table appears.

Step 3: PassengerName moves to the Passenger table. The Passenger table has no redundancies, since PassengerName depends on the whole primary key. The Passenger table appears.

Animation captions:

1. PassengerName depends on PassengerNumber. Dependence on part of the primary key causes repetition of (222, Elvira Yin).
2. Removing PassengerName from Booking eliminates redundancy. Booking is now in second normal form.
3. PassengerName moves to the Passenger table. The Passenger table has no redundancies, since PassengerName depends on the whole primary key.

PARTICIPATION ACTIVITY 37.10.8: Second normal form.

The table below lists courses taken by students, along with student scores in the course, student email addresses, and letter grades. Each student has exactly one email address.

StudentRecord

● StudentNumber	● CourseNumber	ScoreNumber	EmailAddress	GradeLetter
8034	Math 100	95	john@gmail.com	A
2111	Math 100	73	sammy@icloud.com	C
9930	Spanish 22A	89	abc@hotmail.com	B
8034	History 11	89	john@gmail.com	B
5091	Biology 200B	41	maria@sbc.net	F

1) Which pair of related values is repeated?

- (8034, john@gmail.com)
- (2111, Math 100)
- (Math 100)

2) Which non-key column does not depend on the whole primary key?

- EmailAddress
- StudentNumber
- ScoreNumber

3) Which column must be removed to achieve second normal form?

- ScoreNumber
- CourseNumber
- EmailAddress

4) A table with a simple primary key must be in _____.

- first but not second normal form
- second normal form or higher
- third normal form

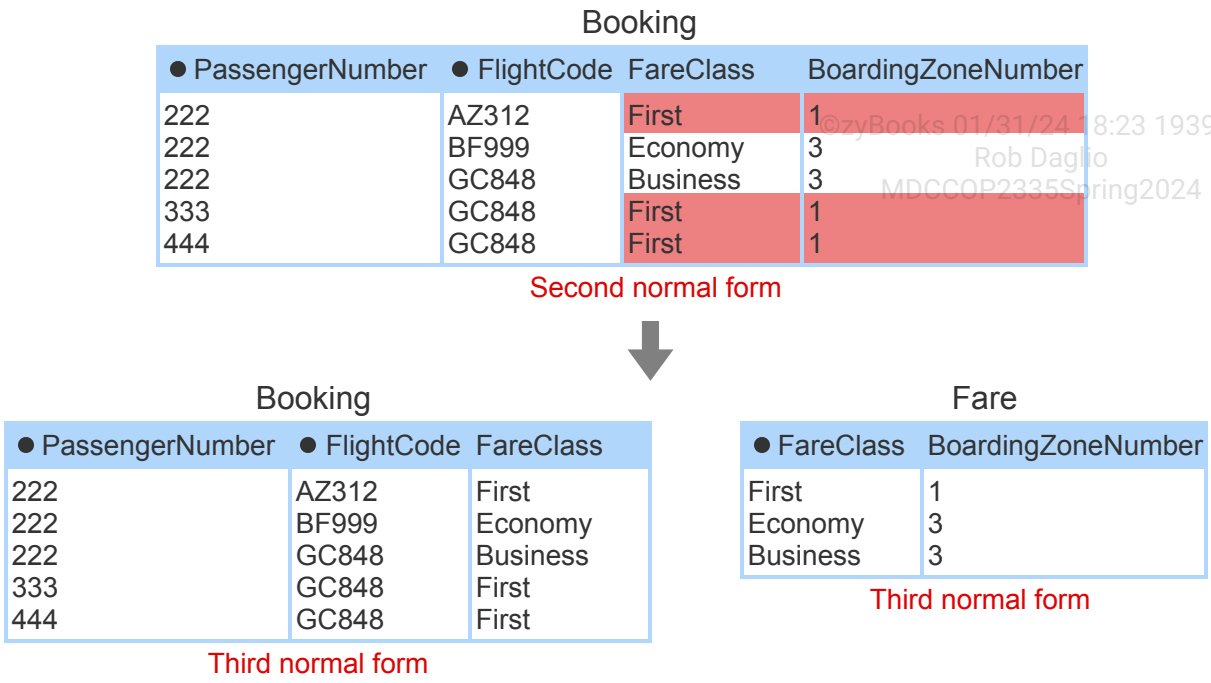
Third normal form

Redundancy can occur in a second normal form table when a non-key column depends on another non-key column. Informally, a table is in ***third normal form*** when all non-key columns depend on the key, the whole key, and nothing but the key. A formal definition appears elsewhere in this material.

A table in third normal form is also in first and second normal forms, by definition.

PARTICIPATION ACTIVITY

37.10.9: Third normal form eliminates most redundancy.



Animation content:

Static figure:
Three tables appear, named Booking, Booking, and Fare. A large arrow indicates the first Booking table is decomposed into the second Booking table and the Fare table.

The first Booking table has columns PassengerNumber, FlightCode, FareClass, and BoardingZoneNumber. (PassengerNumber, FlightCode) is the primary key. Booking has caption Second normal form and five rows:
222, AZ312, First, 1
222, BF999, Economy, 3
222, GC848, Business, 3
333, GC848, First, 1
444, GC848, First, 1

The three instances of (First, 1) are highlighted.

The second Booking table is the same as the first, except the BoardingZoneNumber column does not appear. The caption is Third normal form.

The Fare table has columns FareClase and BoardingZoneNumber. FareClass is the primary key. Fare

has caption Third normal form and three rows:
First, 1
Economy, 3
Business, 3

Step 1: BoardingZoneNumber depends on FareClass. Dependence on a non-key column causes the redundancy (First, 1). The first Booking table appears. The three instances of (First, 1) are highlighted.

Step 2: Removing BoardingZoneNumber from Booking eliminates redundancy. Booking is now in third normal form. An X appears above the BoardingZoneNumber column. The second Booking table appears.

Step 3: BoardingZoneNumber moves to the Fare table. The Fare table has no redundancies since BoardingZoneNumber depends on the primary key. The Fare table appears.

Animation captions:

1. BoardingZoneNumber depends on FareClass. Dependence on a non-key column causes the redundancy (First, 1).
2. Removing BoardingZoneNumber from Booking eliminates redundancy. Booking is now in third normal form.
3. BoardingZoneNumber moves to the Fare table. The Fare table has no redundancies since BoardingZoneNumber depends on the primary key.

PARTICIPATION ACTIVITY 37.10.10: Third normal form.

Refer to the second normal form table, below. Scores of 90 and above receive an A grade, 80 to 89 a B, and so on.

StudentRecord

• StudentNumber	• CourseNumber	ScoreNumber	GradeLetter
8034	Math 100	95	A
2111	Math 100	88	B
9930	Spanish 22A	72	C
8034	History 11	88	B
5091	Biology 200B	41	F

1) Which fact is repeated?

- (8034)
- (88, B)
- (9930, Spanish 22A)

2) Which non-key column depends on another non-key column?

- GradeLetter
- ScoreNumber
- CourseNumber

3) Which column must be removed to achieve third normal form?

- GradeLetter
- ScoreNumber
- CourseNumber

37.11 Boyce-Codd normal form

Redundancy and dependence

Column A depends on column B means each B value is related to at most one A value. Columns A and B may be simple or composite. 'A depends on B' is denoted B → A. Dependence of one column on another is called *functional dependence*.

Redundancy occurs when a column depends on another column that is not unique.

PARTICIPATION ACTIVITY 37.11.1: Redundancy and dependence.

Booking

● PassengerNumber →	PassengerName	● FlightCode	FareClass	BoardingZoneNumber
222	Elvira Yin	AZ312	First	1
222	Elvira Yin	BF999	Economy	3
222	Elvira Yin	GC848	Business	3
333	Deepak Chopra	GC848	First	1
444	Mary Hutcher	GC848	First	1

Animation content:

Static figure:
Table Booking has columns PassengerNumber, PassengerName, FlightCode, FareClass, and BoardingZoneNumber. (PassengerNumber, FlightCode) is the primary key. An arrow points from PasengerNumber to PassengerName. Booking has five rows:
222, Elvira Yin, AZ312, First, 1
222, Elvira Yin, BF999, Economy, 3
222, Elvira Yin, GC848, Business, 3
333, Deepak Chopra, GC848, First, 1
444, Mary Hutcher, GC848, First, 1

The three instances of (222, Elvira Yin) are highlighted.

Step 1: Since a passenger number always identifies the same name, PassengerName depends on PassengerNumber. PassengerNumber, PassengerName, and the arrow between are highlighted.

Step 2: Since a passenger may have bookings on several flights, PassengerNumber is not unique. The three instances of 222 are highlighted.

Step 3: The dependence of PassengerName on a non-unique column creates redundancy. The three instances of (222, Elvira Yin) are highlighted.

Animation captions:

1. Since a passenger number always identifies the same name, PassengerName depends on PassengerNumber.
2. Since a passenger may have bookings on several flights, PassengerNumber is not unique.
3. The dependence of PassengerName on a non-unique column creates redundancy.

In a Boyce-Codd normal form table, all dependencies are on unique columns. Dependence on a unique column never creates redundancy, so Boyce-Codd normal form eliminates all redundancy arising from functional dependence.

PARTICIPATION ACTIVITY 37.11.2: Redundancy and dependence.

Refer to the Booking table in the animation above.

1) What column does BoardingZoneNumber depend on?

- PassengerName
- FlightCode
- FareClass

2) Is FareClass unique?

- Yes
- No
- Cannot determine from table data

3) Does the dependency FareClass → BoardingZoneNumber create redundancy?

- Yes
- No
- Cannot determine from table data

4) A table with a dependency on a non-unique column _____ contains redundant data.

- always
- often
- never

Third normal form

Informally, a table is in third normal form when all non-key columns depend on the key, the whole key, and nothing but the key. This definition is accurate when the primary key is the only unique column. The formal definition, below, accounts for tables with several unique columns.

A ***candidate key*** is a simple or composite column that is unique and minimal. ***Minimal*** means all columns are necessary for uniqueness. A table may have several candidate keys. The database designer designates one candidate key as the primary key.

A ***non-key*** column is a column that is not contained in a candidate key.

A table is in ***third normal form*** if, whenever a non-key column A depends on column B, then B is unique. Columns A and B may be simple or composite. Although B is unique, B is not necessarily minimal and therefore is not necessarily a candidate key.

PARTICIPATION ACTIVITY 37.11.3: Third normal form.

Refer to the following table. Each department has one chair, but the same person occasionally chairs several departments. Course names can be repeated in different departments, but never within the same department.

Course

•CourseCode	DepartmentCode	CourseName	DepartmentChair
8451	MATH	Discrete Mathematics	Azim Rafiq
8452	CS	Discrete Mathematics	Je Soomin
2391	SPAN	Introduction to Spanish	Susan Williams
5505	BIO	Genetics and Evolution	Susan Williams
8449	MATH	Calculus I	Azim Rafiq
8036	MATH	Differential Equations	Azim Rafiq

1) Which fact is repeated?

- Discrete Mathematics
- (MATH, Differential Equations)
- (MATH, Azim Rafiq)

2) What are the candidate keys?

- CourseCode only
- CourseCode and (DepartmentCode, CourseName)
- CourseCode, (DepartmentCode, CourseName), and (CourseName, DepartmentChair)

3) What are the non-key columns?

- None
- DepartmentChair only
- CourseName and DepartmentChair

4) What dependency violates the definition of third normal form?

- DepartmentChair depends on DepartmentCode
- DepartmentChair depends on CourseName
- CourseName depends on CourseCode

5) Which column should be removed to achieve third normal form?

- DepartmentChair
- CourseCode
- DepartmentCode

Boyce-Codd normal form

The definition of third normal form applies to *non-key* columns only, which allows for occasional redundancy. Boyce-Codd normal form applies to *all* columns and eliminates this redundancy.

A table is in ***Boyce-Codd normal form*** if, whenever column A depends on column B, then B is unique. Columns A and B may be simple or composite. This definition is identical to the definition of third normal form with the term 'non-key' removed.

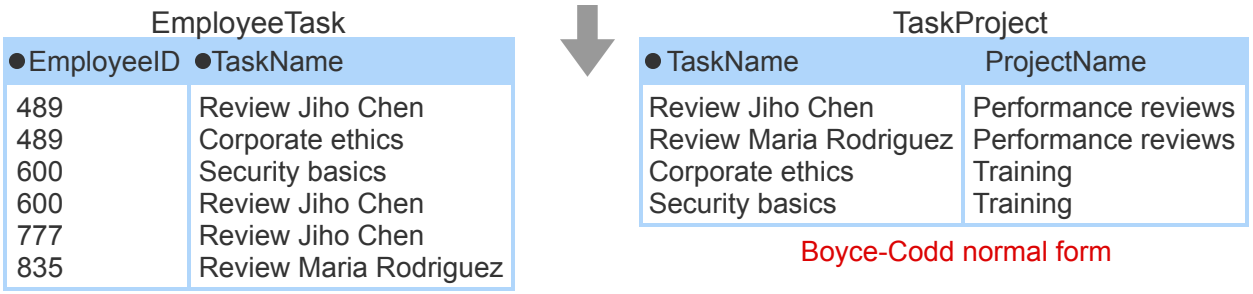
Boyce-Codd normal form is considered the gold standard of table design. Although fourth and fifth normal forms remove additional types of redundancy, these redundancies are uncommon and of little practical concern.

PARTICIPATION ACTIVITY 37.11.4: Boyce-Codd normal form eliminates all common redundancy.

EmployeeProjectTask

● EmployeeID	● ProjectName ← TaskName	
489	Performance reviews	Review Jiho Chen
489	Training	Corporate ethics
600	Training	Security basics
600	Performance reviews	Review Jiho Chen
777	Performance reviews	Review Jiho Chen
835	Performance reviews	Review Maria Rodriguez

Third normal form



Animation content:

Static figure:
Tables EmployeeProjectTask, EmployeeTask, and TaskProject appear. A large arrow indicates EmployeeProjectTask is decomposed to EmployeeTask and TaskProject.

The EmployeeProjectTask table has columns EmployeeID, ProjectName, and TaskName. (EmployeeID, ProjectName) is the primary key. An arrow points from TaskName to ProjectName. EmployeeProjectTask has caption Third normal form and six rows:
489, Performance reviews, Review Jiho Chen
489, Training, Corporate ethics
600, Training, Security basics
600, Performance reviews, Review Jiho Chen
777, Performance reviews, Review Jiho Chen
835, Performance reviews, Review Maria Rodriguez

The three instances of (Performance reviews, Review Jiho Chen) are highlighted.

The EmployeeTask table has columns EmployeeID and TaskName. (EmployeeID, TaskName) is the primary key. EmployeeTask has caption Boyce-Codd normal form and six rows:
489, Review Jiho Chen
489, Corporate ethics
600, Security basics
600, Review Jiho Chen
777, Review Jiho Chen
835, Review Maria Rodriguez

The TaskProject table has columns TaskName, ProjectName. (TaskName, ProjectName) is the primary key. TaskProject has caption Boyce-Codd normal form and four rows:
Review Jiho Chen, Performance reviews
Review Maria Rodriguez, Performance reviews
Corporate ethics, Training
Security basics, Training

Step 1: Each employee is assigned at most one task on each project. The EmployeeTaskProject table appears.

Step 2: TaskName depends on the primary key (EmployeeID, ProjectName) but not on EmployeeID or ProjectName. Parentheses surround EmployeeID and ProjectName. An arrow points from (EmployeeID, ProjectName) to TaskName.

Step 3: Since every non-key column depends on a unique column, EmployeeProjectTask is in third normal form. The caption Third normal form appears below EmployeeProjectTask.

Step 4: In this company, TaskNames are never repeated on different projects. So ProjectName depends on TaskName, which is not unique. TaskName, ProjectName, and the arrow between are highlighted.

Step 5: Since ProjectName depends on a non-unique column, EmployeeProjectTask is not in Boyce-Codd normal form and contains redundancy. The three instances of (Performance reviews, Review Jiho Chen) are highlighted.

Step 6: Decomposing to EmployeeTask and TaskProject eliminates the redundancy. The EmployeeTask and TaskProject tables appear.

Step 7: EmployeeTask contains no dependencies. TaskProject contains one dependency, ProjectName on the unique column TaskName. The caption Boyce-Codd normal form appears below the EmployeeTask and TaskProject tables.

Animation captions:

1. Each employee is assigned at most one task on each project.
2. TaskName depends on the primary key (EmployeeID, ProjectName) but not on EmployeeID or ProjectName.
3. Since every non-key column depends on a unique column, EmployeeProjectTask is in third normal form.
4. In this company, TaskNames are never repeated on different projects. So ProjectName depends on TaskName, which is not unique.
5. Since ProjectName depends on a non-unique column, EmployeeProjectTask is not in Boyce-Codd normal form and contains redundancy.
6. Decomposing to EmployeeTask and TaskProject eliminates the redundancy.
7. EmployeeTask contains no dependencies. TaskProject contains one dependency, ProjectName on the unique column TaskName.

PARTICIPATION ACTIVITY 37.11.5: Boyce-Codd normal form.

Refer to the following table. Assume all licenses and postal codes are in the United States.

Employee

• EmployeeID	DriversLicenseNumber	DriversLicenseState	Name	PostalCode
489	AB7325	IL	Lisa Ellison	60415
517	N3259211	CA	Sam Snead	90295
600	B16629045	CA	Malia Efrenza	90295
777	8242103	TX	Nadia Shah	75185
929	8242103	FL	Maria Rodriguez	32099
933	AX493200	CA	Jiho Chen	94701

1) EmployeeID is the only candidate key.

- True
- False

2) Name and PostalCode are the only non-key columns.

- True
- False

3) All non-key columns depend only on unique columns.

- True
- False

4) Employee is in third normal form.

- True
- False

5) All columns depend only on unique columns.

- True
- False

6) Employee is in Boyce-Codd normal form.

- True
- False

7) Removing DriversLicenseState eliminates all redundancy from Employee.

- True
- False

Trivial dependencies

*When the columns of A are a subset of the columns of B, A always depends on B. Ex: FareClass depends on (FlightCode, FareClass). These dependencies are called **trivial**.*

*Technically, trivial dependencies must be excluded in definitions of normal form: A table is in Boyce-Codd normal form if, for all **non-trivial** dependencies B → A, B is unique.*

CHALLENGE ACTIVITY 37.11.1: Normal form.

539740.3879454.qx3zqy7

Start

Country

• TLD	CountryName	CapitalName	ContinentName
.bi	Burundi	Gitega	Africa
.me	Montenegro	Podgorica	Europe
.se	Sweden	Stockholm	Europe
.mu	Mauritius	Port Louis	Africa
.sb	Solomon Islands	Honiara	Oceania

What columns depend on TLD?

- ☐ CountryName
- ☐ CapitalName
- ☐ ContinentName

What columns depend on CapitalName?

- ☐ TLD
- ☐ CountryName
- ☐ ContinentName

1 2 3

Check Next

Exploring further:

- Boyce-Codd normal form

37.12 Applying normal form

Normalization

Occasionally, implementing entities, relationships, and attributes generates tables that contain redundancy. This redundancy is eliminated with normalization, the last step of logical design. ***Normalization*** eliminates redundancy by decomposing a table into two or more tables in higher normal form.

Database designers usually normalize tables to Boyce-Codd normal form. In a ***Boyce-Codd normal form*** table, if column A depends on column B, then B must be unique. Normalizing a table to Boyce-Codd normal form involves three steps:

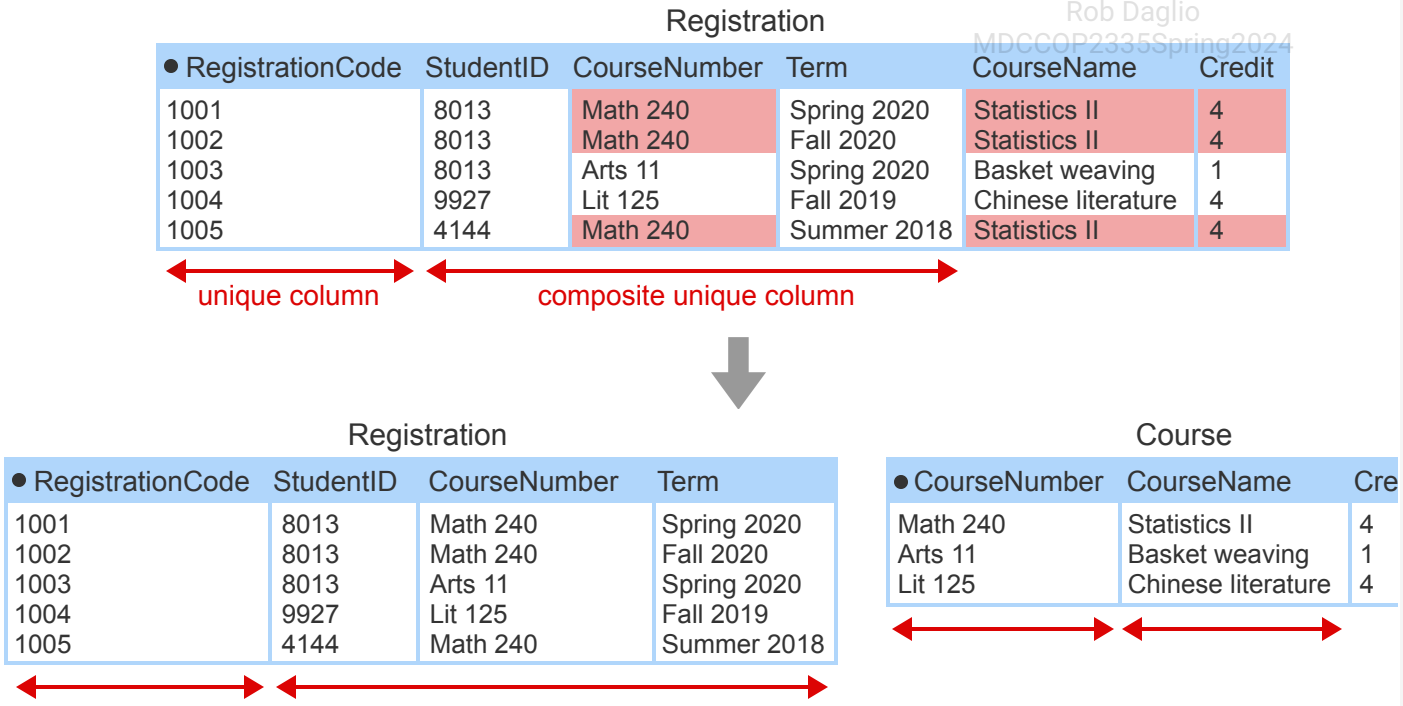
1. *List all unique columns*. Unique columns may be simple or composite. In composite columns, remove any columns that are not necessary for uniqueness.
2. *Identify dependencies on non-unique columns*. Non-unique columns are either *external* to all unique columns or *contained within* a composite unique column.
3. *Eliminate dependencies on non-unique columns*. If column A depends on a non-unique column B, A is removed from the original table. A new table is created containing A and B. B is a primary key in the new table and a foreign key in the original table.

Since the data relating columns A and B is recorded in a new table, no information is lost when A is removed from the original table.

Terminology

*In E. F. Codd's original paper on the relational model, **normalization** meant achieving first normal form. Over time, however, normalization has come to mean achieving higher normal forms.*

PARTICIPATION ACTIVITY 37.12.1: Normalizing tables.



Animation content:

Static figure:
Tables Registration, Registration, and Course appear. A large arrow indicates that the first Registration table is decomposed into the second Registration table and the Course table.

The first Registration table has columns RegistrationCode, StudentID, CourseNumber, Term, CourseName, and Credit. RegistrationCode is the primary key. RegistrationCode is labeled unique column. (StudentID, CourseNumber, Term) is labeled composite unique column. Registration has five rows:
1001, 8013, Math 240, Spring 2020, Statistics II, 4
1002, 8013, Math 240, Fall 2020, Statistics II, 4
1003, 8013, Arts 11, Spring 2020, Basket weaving, 1
1004, 9927, Lit 125, Fall 2019, Chinese literature, 4
1005, 4144, Math 240, Summer 2018, Statistics II, 4

The three instances of (Math 240, Statistics II, 4) are highlighted.

The second Registration table has columns RegistrationCode, StudentID, CourseNumber, and Term. RegistrationCode is the primary key. A highlight line spans RegistrationCode. Another highlight line spans (StudentID, CourseNumber, Term). The second Registration table has five rows:
1001, 8013, Math 240, Spring 2020
1002, 8013, Math 240, Fall 2020
1003, 8013, Arts 11, Spring 2020
1004, 9927, Lit 125, Fall 2019
1005, 4144, Math 240, Summer 2018

The Course table has columns CourseNumber, CourseName, and Credit. CourseNumber is the primary key. A highlight line spans CourseNumber. Another highlight line spans CourseName. Course has three rows:
Math 240, Statistics II, 4
Arts 11, Basket weaving, 1
Lit 125, Chinese literature, 4

Step 1: The Registration table lists student registration for courses by term. The first Registration table appears.

Step 2: RegistrationCode is a unique column. The RegistrationCode column is highlighted. The unique column caption appears..

Step 3: (StudentID, CourseNumber, Term) is a composite unique column. The (StudentID, CourseNumber, Term) columns are highlighted. The composite unique column caption appears.

Step 4: CourseName and Credit depend on CourseNumber, which is not unique. Registration is not in Boyce-Codd normal form. The three instances of (Math 240, Statistics II, 4) are highlighted.

Step 5: Redundancy is eliminated by removing CourseName and Credit. The new table is in Boyce-Codd normal form. The second Registration table appears with highlight lines.

Step 6: CourseNumber, CourseName, and Credit are tracked in a new Course table. The Course table appears.

Step 7: All dependencies in Course are on a unique column. Course is in Boyce-Codd normal form. The highlight lines for the Course table appear.

Animation captions:

1. The Registration table lists student registration for courses by term.
2. RegistrationCode is a unique column.
3. (StudentID, CourseNumber, Term) is a composite unique column.
4. CourseName and Credit depend on CourseNumber, which is not unique. Registration is not in Boyce-Codd normal form.

5. Redundancy is eliminated by removing CourseName and Credit. The new table is in Boyce-Codd normal form.
6. CourseNumber, CourseName, and Credit are tracked in a new Course table.
7. All dependencies in Course are on a unique column. Course is in Boyce-Codd normal form.

PARTICIPATION ACTIVITY 37.12.2: Normalization.

Refer to the Shipment table:

Shipment

•TrackingID	OrderCode	ProductNumber	ProductName	ShipperCode	ShipperName
201	A37	8024	Slip Sandal	FX	Federal Express
202	A37	9119	Zephyr T-Shirt	UPS	United Parcel Service
203	C28	9119	Zephyr T-Shirt	FX	Federal Express
204	R43	3558	Cool Hat	FX	Federal Express

1) Which simple or composite column(s) are unique?

- TrackingID only
- TrackingID and (OrderCode, ProductNumber)
- TrackingID and (OrderCode, ProductNumber, ShipperCode)

2) Which column depends on part of the composite unique column (OrderCode, ProductNumber)?

- ProductName
- ShipperCode
- ShipperName

3) Which column depends on a column that is external to unique columns?

- ProductName
- TrackingID
- ShipperName

4) When Shipment is normalized, which columns remain?

- TrackingID, OrderCode, ProductNumber, ShipperCode
- TrackingID, OrderCode, ProductNumber
- OrderCode, ProductNumber, ShipperCode

5) How many tables are in the normalized design?

- 2
- 3
- 4

Denormalization

Boyce-Codd normal form is ideal for tables with frequent inserts, updates, and deletes. In a database used primarily for reporting, changes are infrequent and redundancy is acceptable. In fact, redundancy can be desirable in reporting databases, as processing is faster and queries are simpler. Therefore, reporting databases may contain tables that, by design, are not in third normal form.

Denormalization means intentionally introducing redundancy by merging tables. Denormalization eliminates join queries and therefore improves query performance. Denormalization results in first and second normal form tables and should be applied selectively and cautiously.

In the figure below, the Booking, Passenger, and Fare tables are denormalized into a single Booking table. The red highlight indicates redundancy in the denormalized table.

Figure 37.12.1: Denormalization example.

Booking

PassengerNumber	FlightCode	FareClass
222	AZ312	First
222	BF999	Economy
222	GC848	Business
333	GC848	First
444	GC848	First

Passenger

PassengerNumber	PassengerName
222	Elvira Yin
333	Deepak Chopra
444	Mary Hutcher

Fare

FareClass	BoardingZoneNumber
First	1
Economy	3
Business	2

Booking

PassengerNumber	PassengerName	FlightCode	FareClass	BoardingZoneNumber
222	Elvira Yin	AZ312	First	1
222	Elvira Yin	BF999	Economy	3
222	Elvira Yin	GC848	Business	2
333	Deepak Chopra	GC848	First	1
444	Mary Hutcher	GC848	First	1

PARTICIPATION ACTIVITY 37.12.3: Denormalization.

1) Denormalization never results in second-normal-form tables.
   - True
   - False

2) Denormalization accelerates all SELECT queries.
   - True
   - False

3) Occasionally, tables are denormalized in a frequently updated database.
   - True
   - False

Database design

As tables and keys are specified, the database designer reviews each table for Boyce-Codd normal form. Dependencies and unique columns are identified. If any dependencies are not on unique columns, the table is decomposed into smaller tables in Boyce-Codd normal form. Tables that experience infrequent inserts, updates, and deletes may be denormalized to simplify and accelerate join queries.

Table 37.12.1: Applying normal form.

Step	Activity
8A	Identify dependencies on non-unique columns.
8B	Eliminate redundancy by decomposing tables.
8C	Consider denormalizing tables in reporting databases.

PARTICIPATION ACTIVITY 37.12.4: Applying normal form.

Match the activity with the resulting normal form.

If unable to drag and drop, refresh the page.

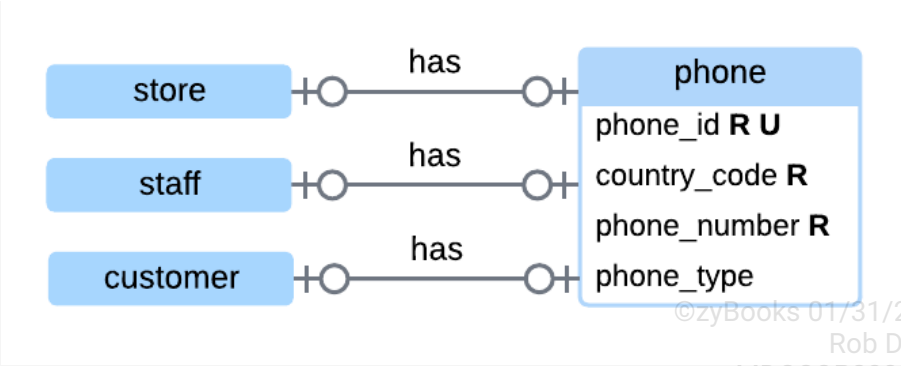
Boyce-Codd normal form | **First or second normal form** | **Second normal form**

	Consider denormalizing tables in reporting databases.
	Eliminate dependencies on columns contained within a composite unique column.
	Eliminate dependencies on non-unique columns.

Reset

37.13 LAB - Implement strong entity (Sakila)

Implement a new strong entity **phone** in the Sakila database. Attributes and relationships are shown in the following diagram:



The diagram uses Sakila naming conventions. Follow the Sakila conventions for your table and column names:

- All lower case
- Underscore separator between root and suffix
- Foreign keys have the same name as referenced primary key

Write CREATE TABLE and ALTER TABLE statements that:

1. Implement the entity as a new `phone` table.
2. Implement the `has` relationships as foreign keys in the Sakila `customer`, `staff`, and `store` tables.
3. Remove the existing `phone` column from the Sakila `address` table.

Step 2 requires adding a foreign key constraint to an existing table. Ex:

```
ALTER TABLE customer
  ADD FOREIGN KEY (phone_id) REFERENCES phone(phone_id)
  ON DELETE SET NULL
  ON UPDATE CASCADE;
```

Specify data types as follows:

- phone_id, phone_number, and country_code have data type INT.
- phone_type has date type VARCHAR(12) and contains strings like 'Home', 'Mobile', and 'Other'.

Apply these constraints:

- NOT NULL constraints correspond to cardinalities on the diagram above.
- Foreign key actions are SET NULL for delete rules and CASCADE for update rules.
- Specify a suitable column as the `phone` table primary key.

LAB ACTIVITY | 37.13.1: LAB - Implement strong entity (Sakila) | 0 / 10

Main.sql — Load default template...

```
1 -- Your CREATE TABLE and ALTER TABLE statements go here
2
```


Develop mode | Submit mode

Explore the database and run your program as often as you'd like, before submitting for grading. Click **Run program** and observe the program's output in the second box.

Run program

Main.sql (Your program) → Output (shown below)

Program output displayed here

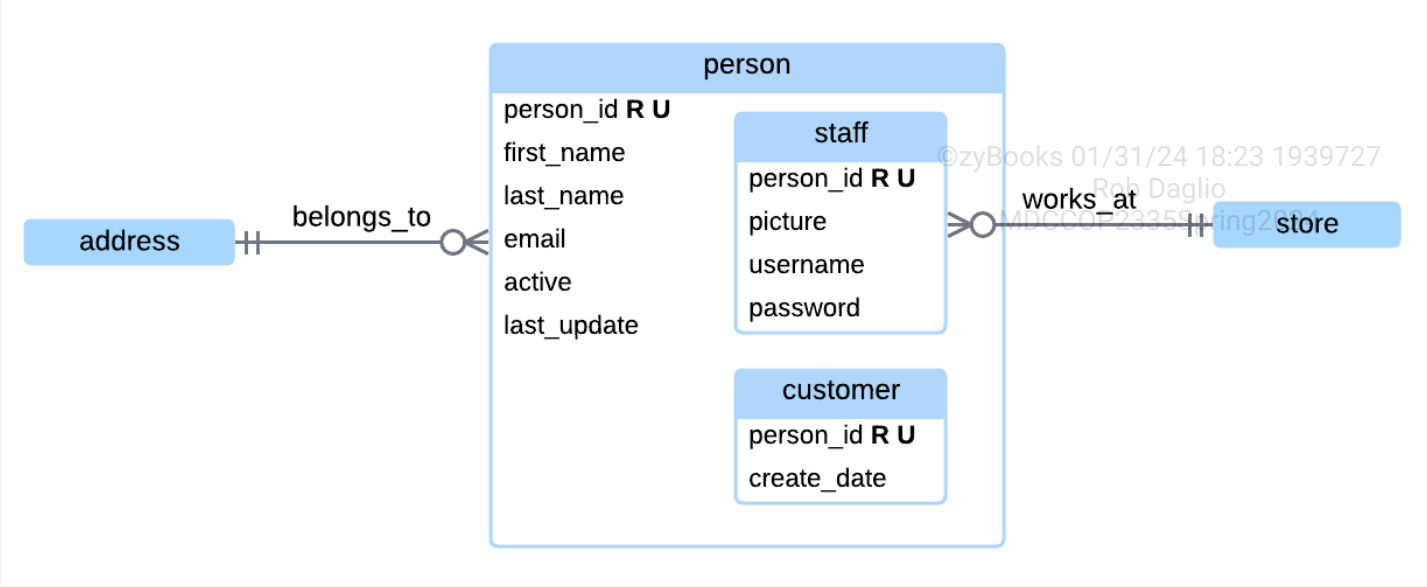
Coding trail of your work What is this?

```
History of your effort will appear here once you begin working
on this zyLab.
```

37.14 LAB - Implement supertype and subtype entities (Sakila)

Similar entities have many common attributes and relationships. Similar entities are often converted into subtypes of a supertype entity, as illustrated in this lab.

In the Sakila database, the `customer` and `staff` tables have several common columns. Convert these tables into subtypes of `person`. Specifically, write CREATE TABLE statements for `person`, `customer`, and `staff` that implement this ER diagram:



Follow Sakila conventions for table and column names:

- All lower case
- Underscore separator between root and suffix
- Foreign keys have the same name as referenced primary key

Implement attributes as columns:

- The primary key of all three tables is `person_id` with data type `SMALLINT UNSIGNED`.
- The `last_update` and `create_date` columns have data type `TIMESTAMP`.
- The `picture` column has data type `BLOB`.
- All other columns have data type `VARCHAR(20)`.

Implement the `belongs_to` and `works_at` relationships as foreign keys:

- `belongs_to` becomes an `address_id` foreign key in `person` with data type `SMALLINT UNSIGNED`.
- `works_at` becomes a `store_id` foreign key in `staff` with data type `TINYINT UNSIGNED`.
- Specify RESTRICT actions for both foreign keys.
- Required relationships become NOT NULL foreign keys.

Subtype entities have an `IsA` relationship to the supertype. Implement these relationships as foreign keys:

- The `person_id` columns of `customer` and `staff` become foreign keys referring to `person`.
- Specify CASCADE actions for both foreign keys.

NOTE: If you execute your solution with the Sakila database, you must first drop `customer`, `staff`, and all constraints that refer to these tables. Use the following statements with Sakila only, not in the zyLab environment:

```
ALTER TABLE payment
   DROP CONSTRAINT fk_payment_customer,
   DROP CONSTRAINT fk_payment_staff;
ALTER TABLE rental
   DROP CONSTRAINT fk_rental_customer,
   DROP CONSTRAINT fk_rental_staff;
ALTER TABLE store
   DROP CONSTRAINT fk_store_staff;
DROP TABLE customer, staff;
```


539740.3879454.qx3zqy7

LAB ACTIVITY | 37.14.1: LAB - Implement supertype and subtype entities (Sakila) 0 / 10

Main.sql Load default template...

```
1 -- Your CREATE TABLE statements go here
2
```

Develop mode | **Submit mode**

Explore the database and run your program as often as you'd like, before submitting for grading. Click **Run program** and observe the program's output in the second box.

Run program

Main.sql (Your program) → Output (shown below)

Program output displayed here