5.1 Implementing entities

Selecting primary keys

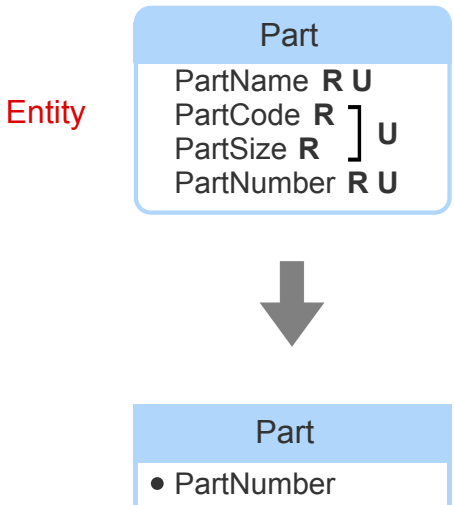
In the first step of logical design, entities become tables and attributes become columns. As tables and columns are specified, primary keys are selected. Primary keys must be unique and required (not NULL). Primary keys should also be:

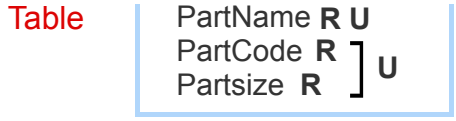
- *Stable*. Primary key values should not change. When a primary key value changes, statements that specify the old value must also change. Furthermore, the new primary key value must cascade to matching foreign keys.
- *Simple*. Primary key values should be easy to type and store. Small values are easy 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 5.1.1: 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 5.1.2: 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

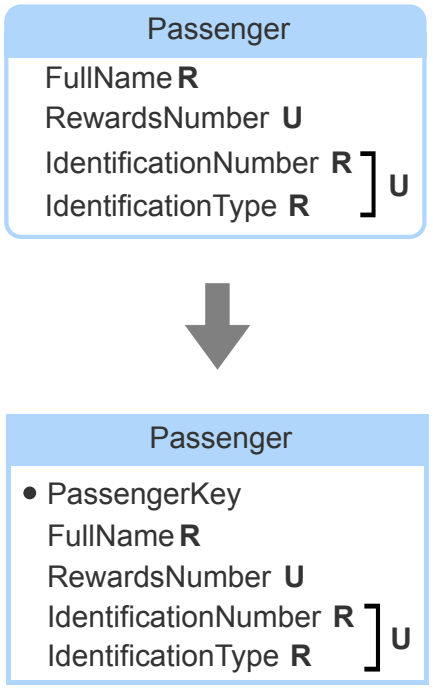
Implementing strong entities

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 5.1.3: 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

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 5.1.4: 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

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:

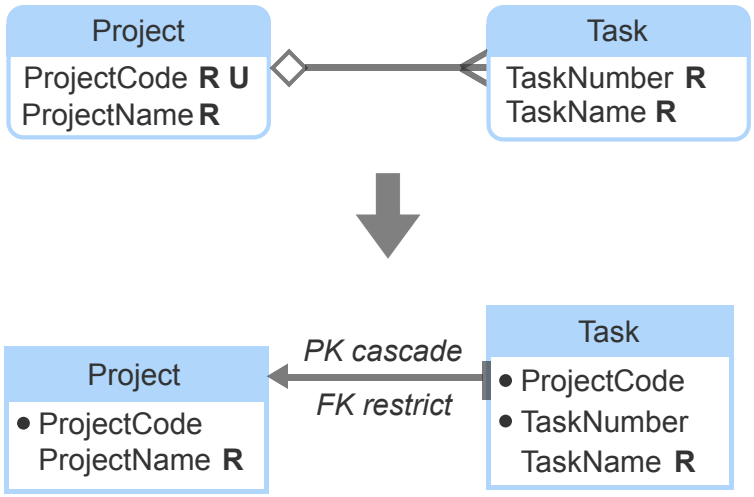
- Usually, the weak entity is plural. The primary key is the composite of the foreign key and another column.
- Occasionally, the weak entity is singular. The primary key is the foreign key only.

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 | 5.1.5: 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.

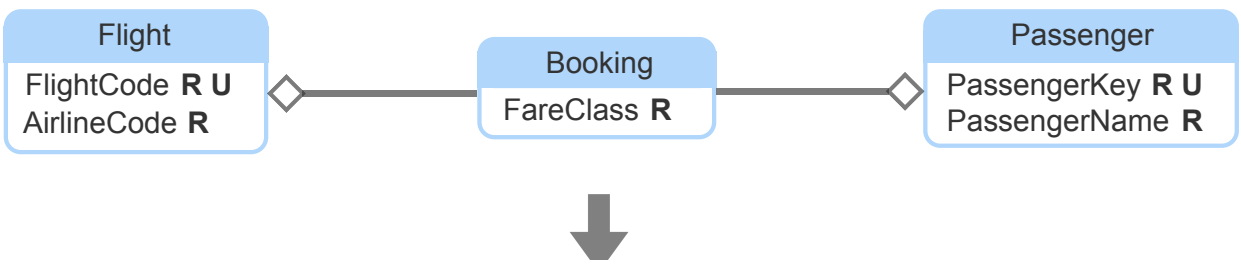
Step 6: Optionally, foreign key actions may appear on the diagram. The captions PK cascade and FK restrict appear next to the arrow.

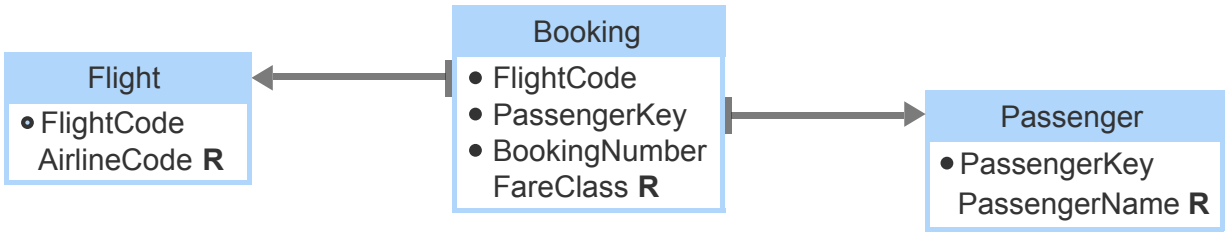
Animation captions:

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 5.1.6: 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.

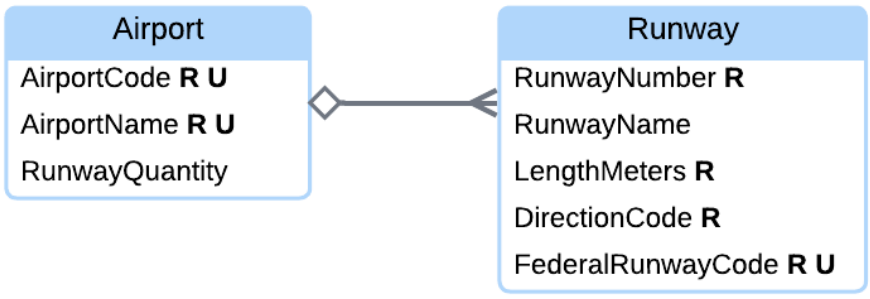
Step 5: If a passenger can make several bookings on the same flight, a third column is necessary in the primary key. The BookingNumber column, with a bullet, is added to Booking and highlighted.

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 5.1.7: 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, rather than an identifying attribute, 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.

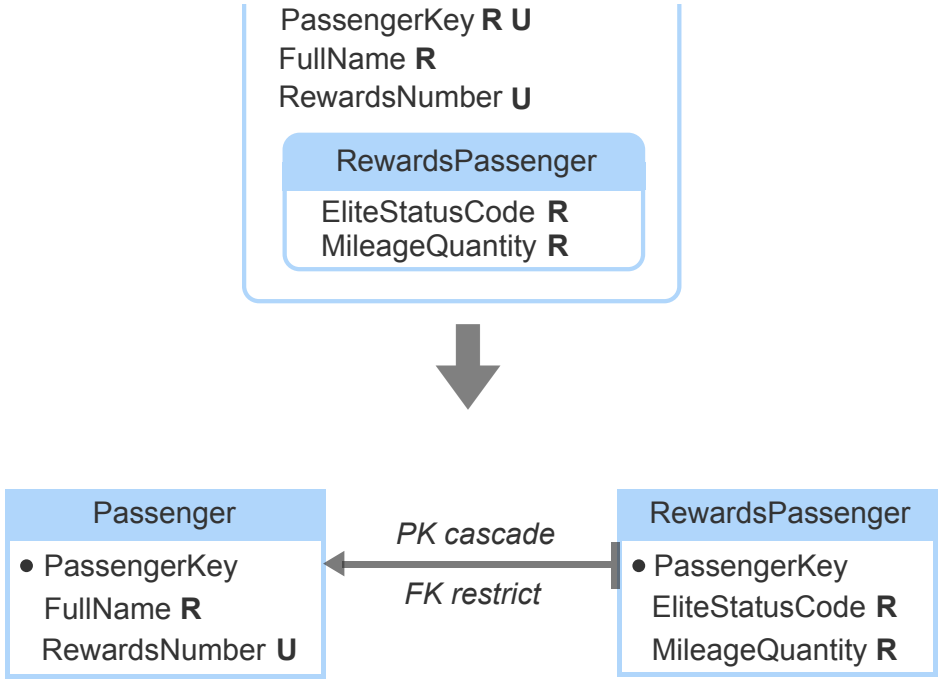
The foreign key usually has the following referential integrity actions:

- 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 5.1.8: Implementing subtype entities.

Passenger



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 5.1.9: 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 5.1.1: 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 5.1.10: 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 5.1.1: Implementing entities.

544874.3500394.qx3zqy7

Start

Assume the following rules:

No two people have the same ID number.
An ID number is generated randomly.
An ID number is an integer between 1 and 9999.
Each person is assigned an ID number.
A person's ID number never changes.

Which of the following properties does the IDNumber attribute have?

- [ ] Unique
- [ ] Required
- [ ] Stable

Simple

Meaningless

1 2

Check Next

5.2 Implementing relationships

Implementing many-one relationships

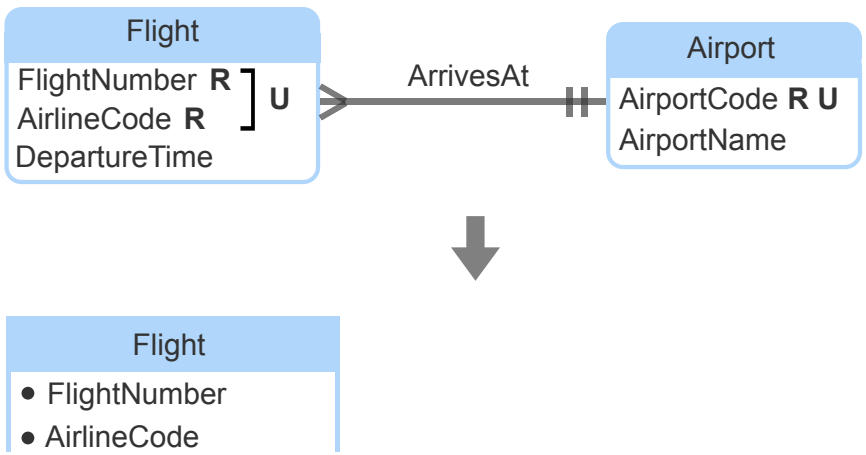
The 'implement entities' step converts identifying relationships into foreign keys. The 'implement relationships' step converts all other relationships into foreign keys or tables.

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 5.2.1: 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.

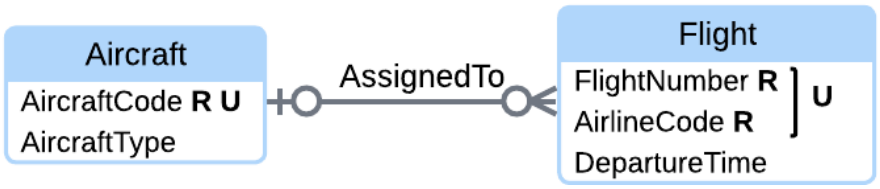
Animation captions:

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 5.2.2: 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.

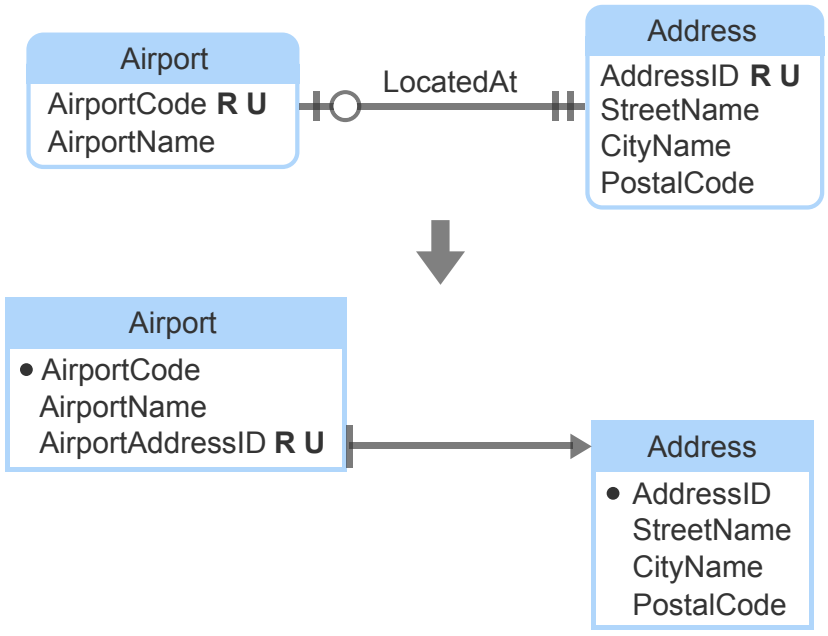
Implementing one-one relationships

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 5.2.3: 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 | 5.8.4: Implementing one-one relationships

ACTIVITY | 5.2.4: 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:

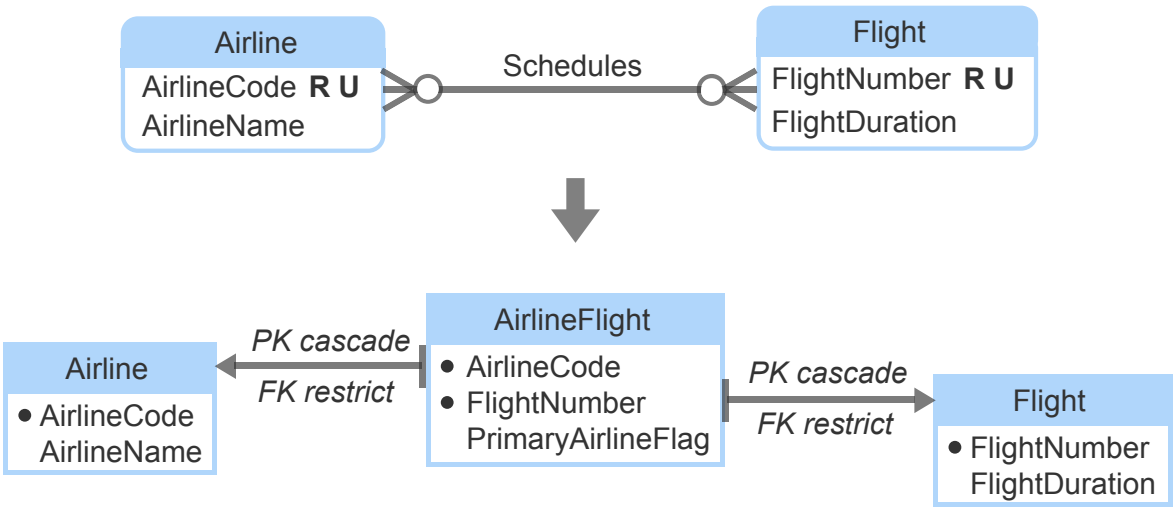
- The new table contains two foreign keys, referring to the primary keys of the related tables.
- The primary key of the new table is the composite of the two foreign keys.
- The new table is identified by the related tables, so primary key cascade and foreign key

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 5.2.5: 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.

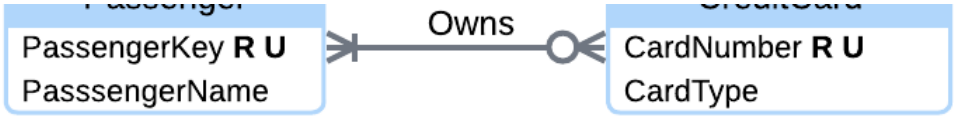
Step 6: Optionally, foreign key rules may appear on the diagram. The captions PC cascade and FK restrict appear next to both arrows.

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 5.2.6: 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.

qualifier derived from the relationship name.

Table 5.2.1: 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 5.2.7: 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

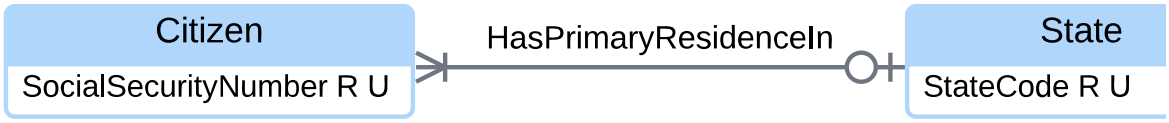
3) The primary key of a table that implements a many-many relationship is composite.
   - True
   - False

CHALLENGE ACTIVITY 5.2.1: Implementing relationships.

544874.3500394.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

5.3 Implementing attributes

Implementing plural attributes

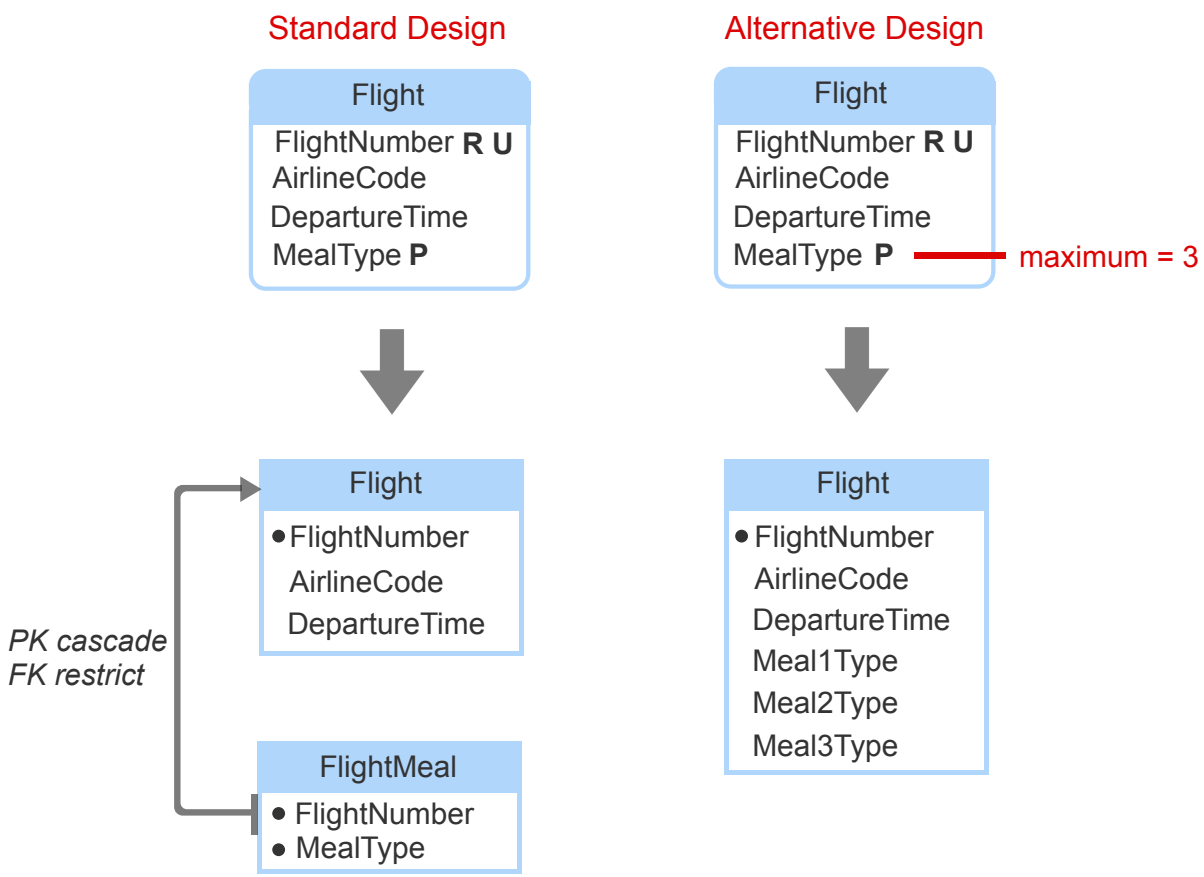
In the 'implement entities' step, entities become tables and attributes become columns. Singular attributes remain in the initial table, but plural attributes move to a new weak table:

- 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 5.3.1: 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.

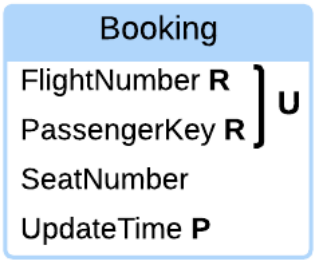
Step 6: Plural attributes with a small, fixed maximum can be implemented as multiple columns in Flight. The second Flight table appears, with additional columns Meal1Type, Meal2Type, and Meal3Type.

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 | 5.3.2: 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?

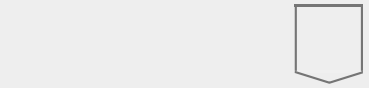
- ○ FlightNumber only
- ○ (FlightNumber, PassengerKey)
- ○ The new table does not contain a foreign key

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 5.3.3: 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.


```
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 5.3.4: 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:

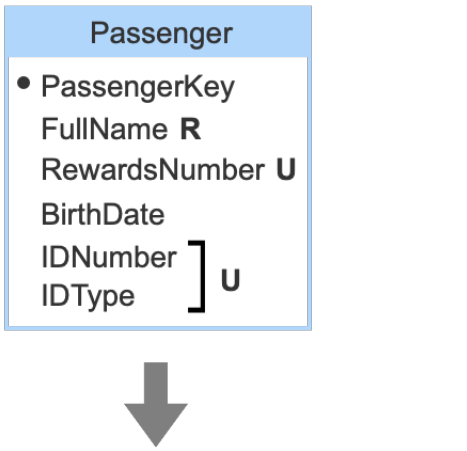
- If the table *referenced by* the foreign key implements a *required* entity, the column is required.
- If the table *containing* the foreign key implements a *singular* entity, the column is unique.

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 5.3.5: 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.

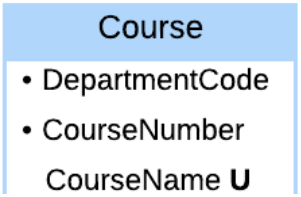
Step 6: A constraint on a composite column is implemented as a separate clause. The IDNumber and IDType column definitions, and the UNIQUE clause, appear in the SQL statement.

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 5.3.6: Implementing attribute cardinality.

Refer to the following table diagram:



CreditQuantity **R**
CatalogCode **R U**

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, CourseNumber)
);
```

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 5.3.1: 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 5.3.7: Implementing attributes.

1) Plural attributes are implemented as new weak tables.
   - True
   - False

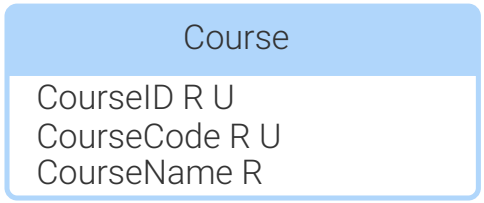
2) The logical design phase ends when attributes have been implemented as columns.
   - True
   - False

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

CHALLENGE ACTIVITY 5.3.1: Implementing attributes.

544874.3500394.qx3zqy7

Start



Which properties apply to each attribute?

	Plural	Required	Unique
CourseID	☐	☐	☐
CourseCode	☐	☐	☐
CourseName	☐	☐	☐

1 2 3 4

Check Next