24.1 Case study: Discovery

Database requirements

The Department of Motor Vehicles (DMV) for the state of California is developing a new database. Database requirements are revealed in interviews with key staff and review of DMV documents. The requirements become an ER model in the discovery step of database design.

Fictitious requirements

This case study is fictitious. The requirements are loosely based on California DMV regulations but do not reflect actual practices. The objective of this case study is to illustrate database and query design rather than describe California regulations.

Prior to discovery, the DMV develops a list of standard attribute types. Discovery then proceeds in three passes for each of three related database applications:

- Vehicle registration
- Driver licensing
- Accidents and citations

As each application is modeled, new entities, relationships, and attributes are added to the diagram. New elements discovered in each pass appear in red.

Standard attribute types

In preparation for discovery, the DMV database administrator develops a list of standard attribute types. An attribute type is appended to each attribute name and describes the attribute's format and use. Each attribute type is implemented as a specific SQL data type.

Table 24.1.1: Standard attribute types for DMV.

Attribute type	SQL data type	Description
Code	CHAR(N)	Fixed-length alpha-numeric string representing a fixed list of alternative values. Length is fixed for

Code	CHAR(N)	each attribute, between 1 and 100 characters. Ex: state codes, auto manufacturers.
Date	DATE	Year, month, and day. Time is not specified.
DateTime	DATETIME	Year, month, day, hour, minute, and, optionally, seconds. If seconds are not entered, the seconds value defaults to 0.
Desc	TEXT(M)	Textual descriptions of any length, up to 65,535 characters. Ex: accident reports, street addresses.
Flag	BOOLEAN	Represents a yes/no or true/false value. In MySQL, the BOOLEAN data type is a synonym for TINYINT. A nonzero value represents yes or true. A zero represents no or false.
ID	CHAR(N)	Fixed-length alphanumeric identifier of up to 30 characters. Implemented as a unique column. If the number of characters in values vary, blanks are appended. Ex: driver's licenses, vehicle identification numbers.
Name	VARCHAR(N)	Variable-length name of up to 30 characters. Represents a name. Ex: personal names, auto models.
Number	TINYINT UNSIGNED SMALLINT UNSIGNED INT UNSIGNED	Positive integer representing either a quantity or an identifying number. Ex: officer badges, points for citations.
Amount	DECIMAL(11, 2)	Decimal value representing US currency. Values are less than $1,000,000,000. Ex: price, fee, payment.
Score	TINYINT UNSIGNED	Integer value between 0 and 100. Ex: exam scores, integer ratings.

Vehicle registration

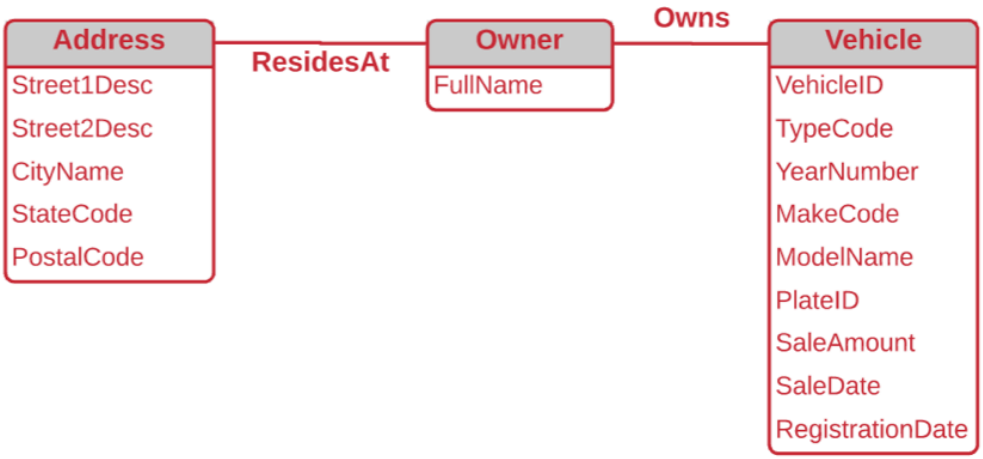
The vehicle registration unit of the DMV tracks data on automobiles, buses, motorcycles, trucks, aircraft, and boats. Every vehicle has a year, make, model, and manufacturer ID number stamped on the engine block. The manufacturer ID is usually referred to as the VIN (vehicle identification

on the engine block. The manufacturer ID is usually referred to as the VIN (vehicle identification number) by DMV staff. Most vehicles have license plates, with the exception of aircraft and boats.

In California, vehicles are registered annually; the DMV must record the vehicle's most recent registration date, full name and mailing address of owner(s), and sales price. Annual registration fees are calculated as the greater of $23 plus 4% of the vehicle's assessed value (sales price less 10% per year).

A smog test is required in odd years for vehicles with odd-numbered model years, and in even years for vehicles with even-numbered model years. The test is required regardless of the year when the registered owner changes. However, vehicles with model years before 1966 are exempt from smog tests.

Figure 24.1.1: Vehicle registration ER diagram.



Driver licensing

The DMV administers vehicle operator exams and maintains license records. Before taking the exam, operators must pass an eye test. Depending on the test results, the operator may be required to wear glasses while driving. Each exam consists of five parts: parallel parking, U-turn, Y-turn, use of turn signals, and speed control. A rating of 0 to 100 is assigned by the administering officer, except for the speed control portion, which is rated on a scale of 0 to 30.

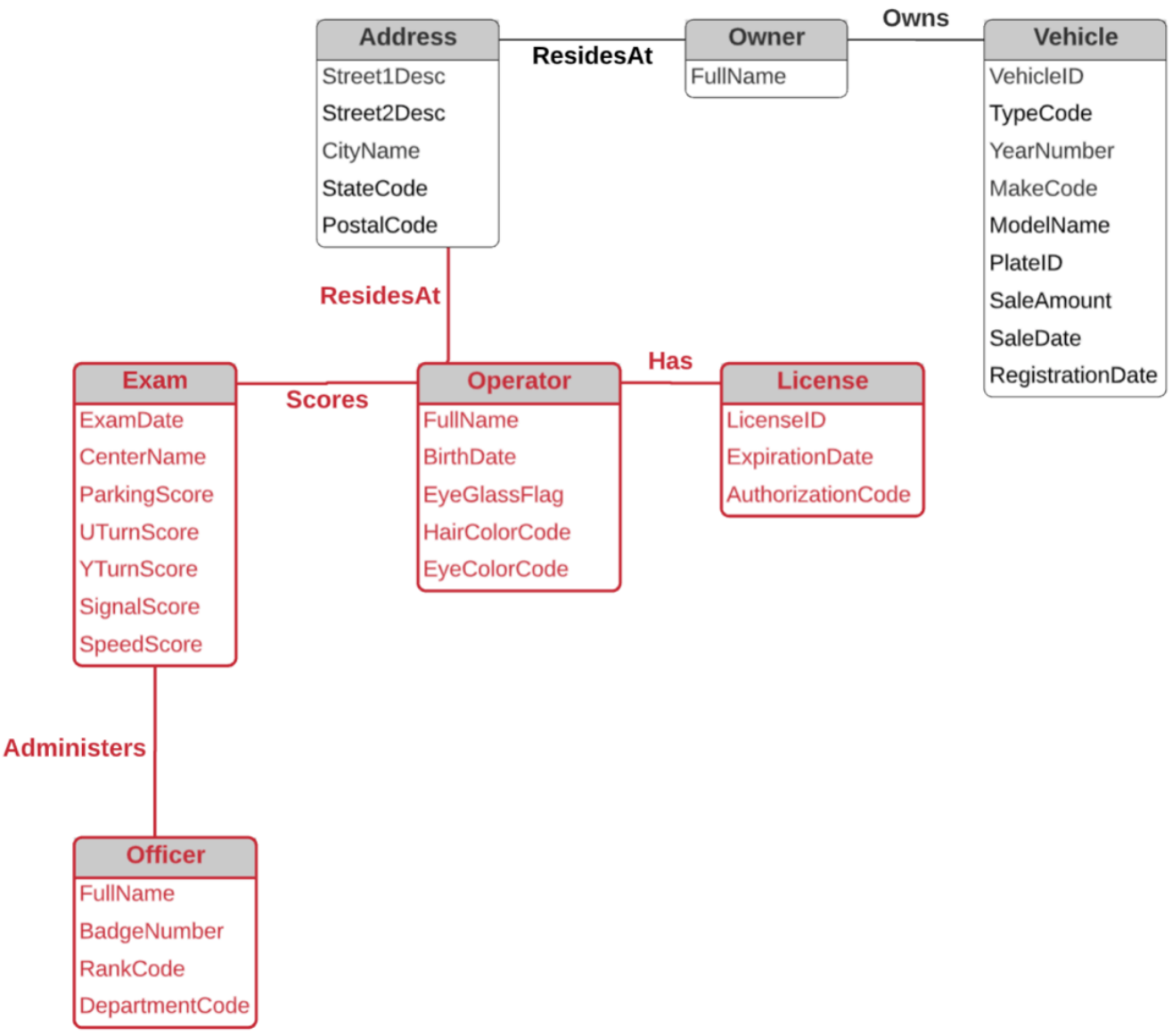
Vehicle operators occasionally fail the exam several times before passing. Each person is allowed one attempt per day and three attempts per year. Full historic records of exams, both passed and failed, must be stored. The DMV tracks the DMV center name and the police officer administering the exam.

When a new driver passes an exam, the full name, address, date of birth, hair and eye color are recorded. The operator is issued a driver's license with an expiration date four years from the date of issue. Driver's licenses are issued one or more authorization codes:

- NC — non-commercial
- M — motorcycle
- CF — commercial - freight transport
- CP — commercial - passenger transport

The DMV maintains a list of all California police officers with badge number, police department, and name. Both badge number and department are necessary to identify a police officer, since different departments may have duplicate badge numbers. The officer's rank may be recorded when available to the DMV.

Figure 24.1.2: Driver licensing ER diagram.



Accidents and citations

The DMV Records Bureau records all accidents reported in California. An accident report is written

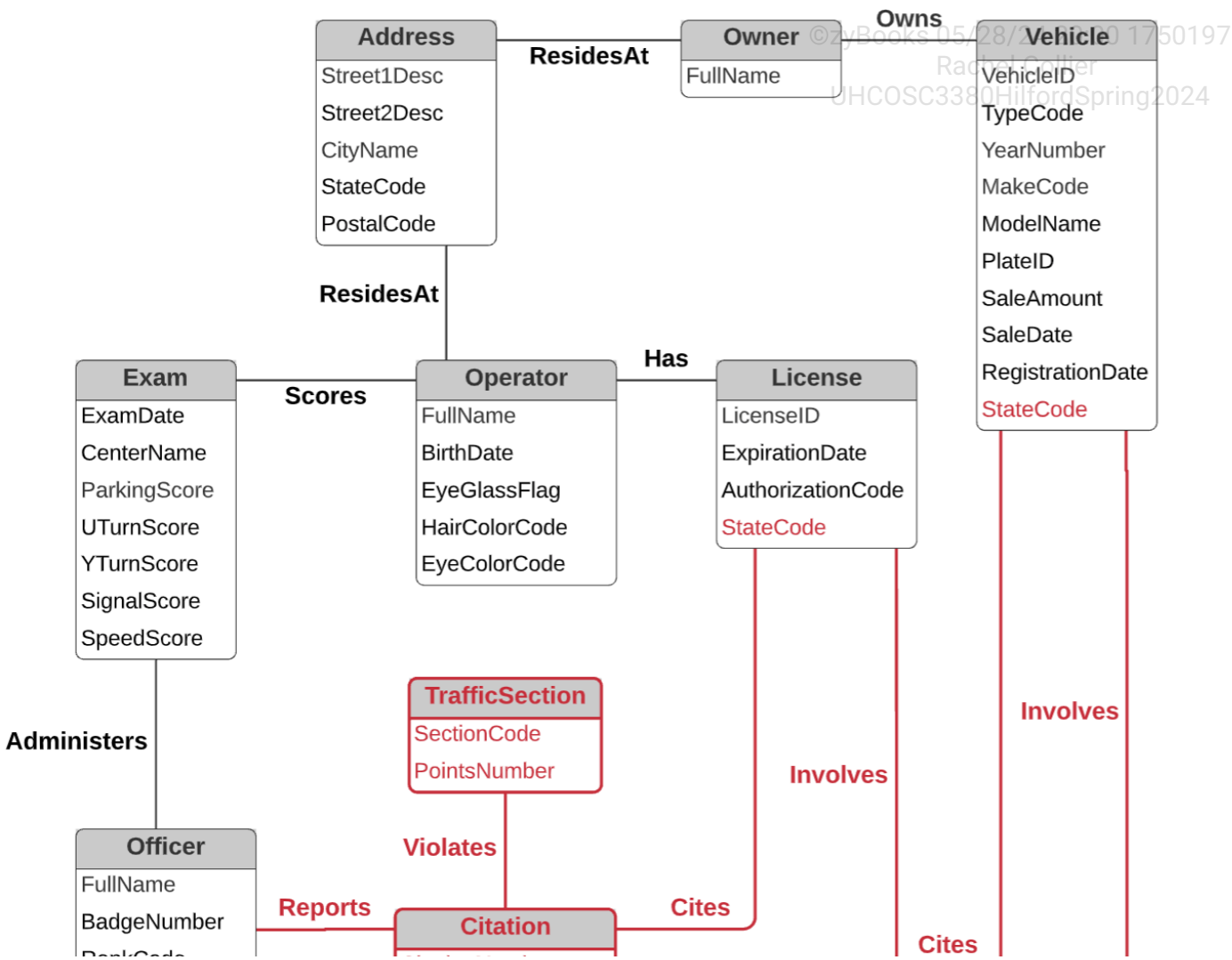
by an officer at the scene and includes all involved drivers' license numbers and vehicle plate numbers. The accident report also notes date, time, location, weather conditions (visibility and road condition), and a description of the accident's circumstances.

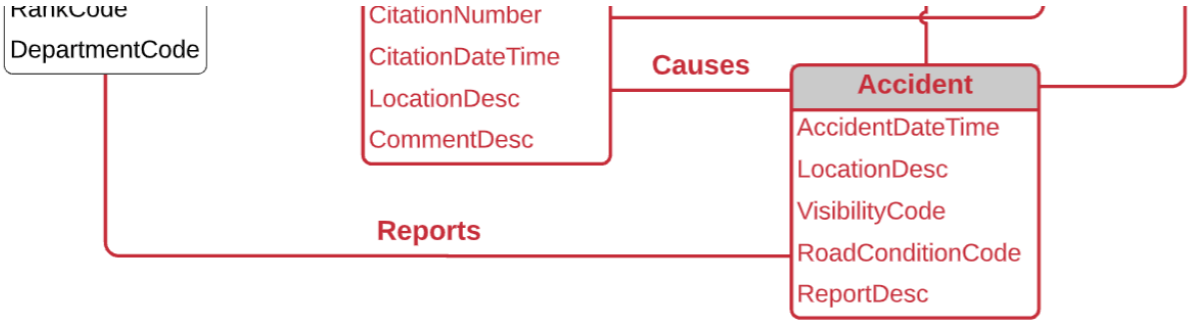
The DMV also records violations of the California Traffic Code, such as speeding tickets, illegal turns, and DUI infractions. Violation data is entered by DMV clerks from tickets written by officers. A ticket includes the citation number, officer identification, date, time, location, vehicle plate number, driver's license number, and optional comments. All violations are cited against sections of the California Traffic Code. For certain sections, points are charged against the driver's record.

For both accidents and violations, if a person has an out-of-state license or a vehicle has an out-of-state plate, the state is noted. If no license or plate is available, the person or vehicle is described in the accident report or violation comments.

Occasionally, an officer reports that a code violation causes one or more accidents. Ex: In an accident report, the officer notes that the accident was caused by a speeding violation. This information is extracted from the accident report and tracked in the database.

Figure 24.1.3: Accidents and citations ER diagram.





PARTICIPATION ACTIVITY 24.1.1: Discovery.

Match the attribute name to the data type. Refer to the standard attribute types.

If unable to drag and drop, refresh the page.

EyeGlassFlag **FullName** **VehicleID** **ReportDesc** **ParkingScore** **SaleAmount**

	DECIMAL(11, 2)
	CHAR(17)
	TEXT(50000)
	VARCHAR(40)
	BOOLEAN
	TINYINT UNSIGNED

Reset

24.2 Case study: Cardinality

Relationship cardinality

Relationship cardinality

Following discovery, minimum and maximum cardinality is determined for all relationships in the DMV model.

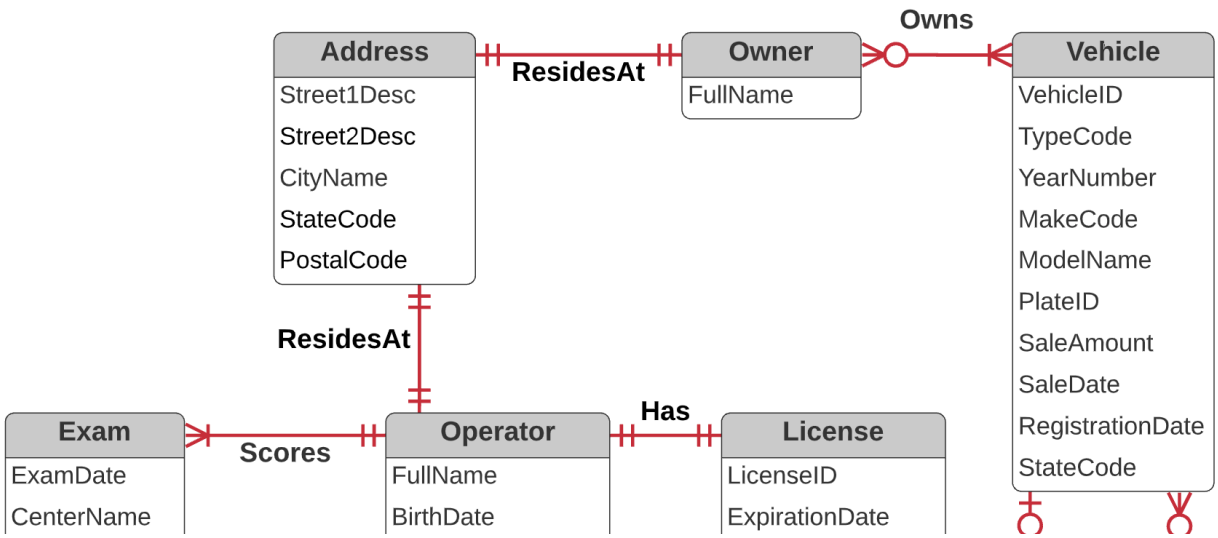
Cardinality is determined from requirements captured during discovery. Cardinality is sometimes inferred from common business practices. When requirements are uncertain, cardinality must be confirmed with staff in follow-up interviews.

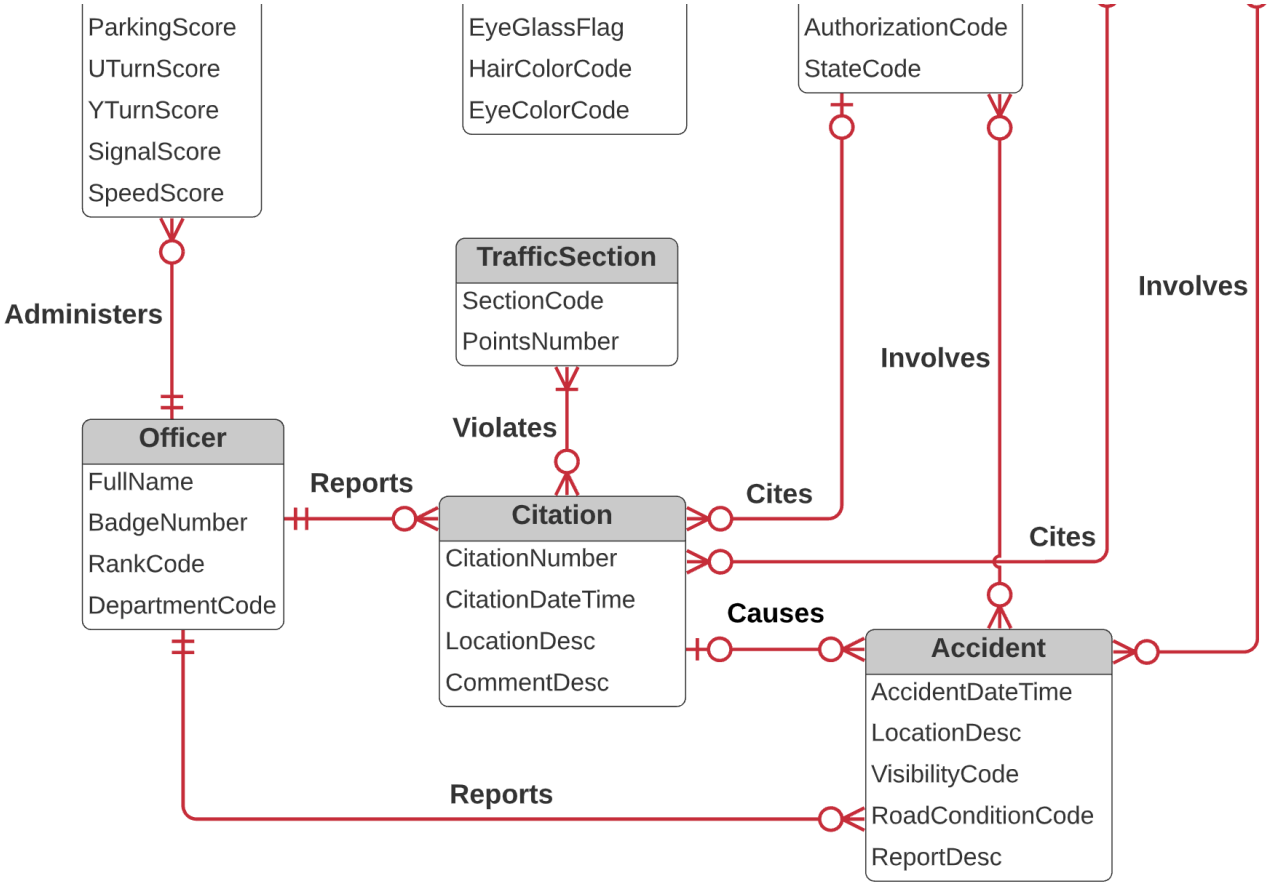
Cardinality appears below in crow's foot notation. The following relationships are noteworthy:

- *Owner-ResidesAt-Address* and *Operator-ResidesAt-Address*. Follow-up interviews confirm that each owner has exactly one address, and each address has exactly one owner. When several owners reside at the same address, The DMV duplicates the address information. The same is true of operators.
- *Owner-Owns-Vehicle*. DMV policy allows each vehicle to have many owners. When a vehicle is junked, the vehicle record is retained with no owner.
- *Accident-Involves-Vehicle* and *Accident-Involves-License*. Follow-up interviews confirm that an accident may involve a vehicle with no plate or an operator with no license. Hence, the minimum of vehicle and license is zero.
- *Citation-Cites-Vehicle* and *Citation-Cites-License*. As with accidents, a citation may cite a vehicle with no plate or an operator with no license, so the minimum of vehicle and license is zero.
- *Citation-Causes-Accident*. Follow-up interviews confirm that, per DMV policy, a citation may cause several accidents, and an accident is caused by at most one citation.

If cardinality cannot be confirmed, maximum many and minimum zero allow the greatest flexibility.

Figure 24.2.1: Relationship cardinality.





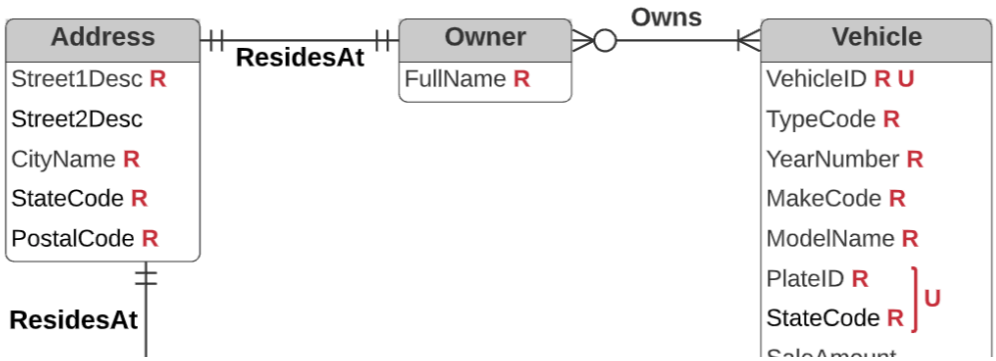
Attribute cardinality

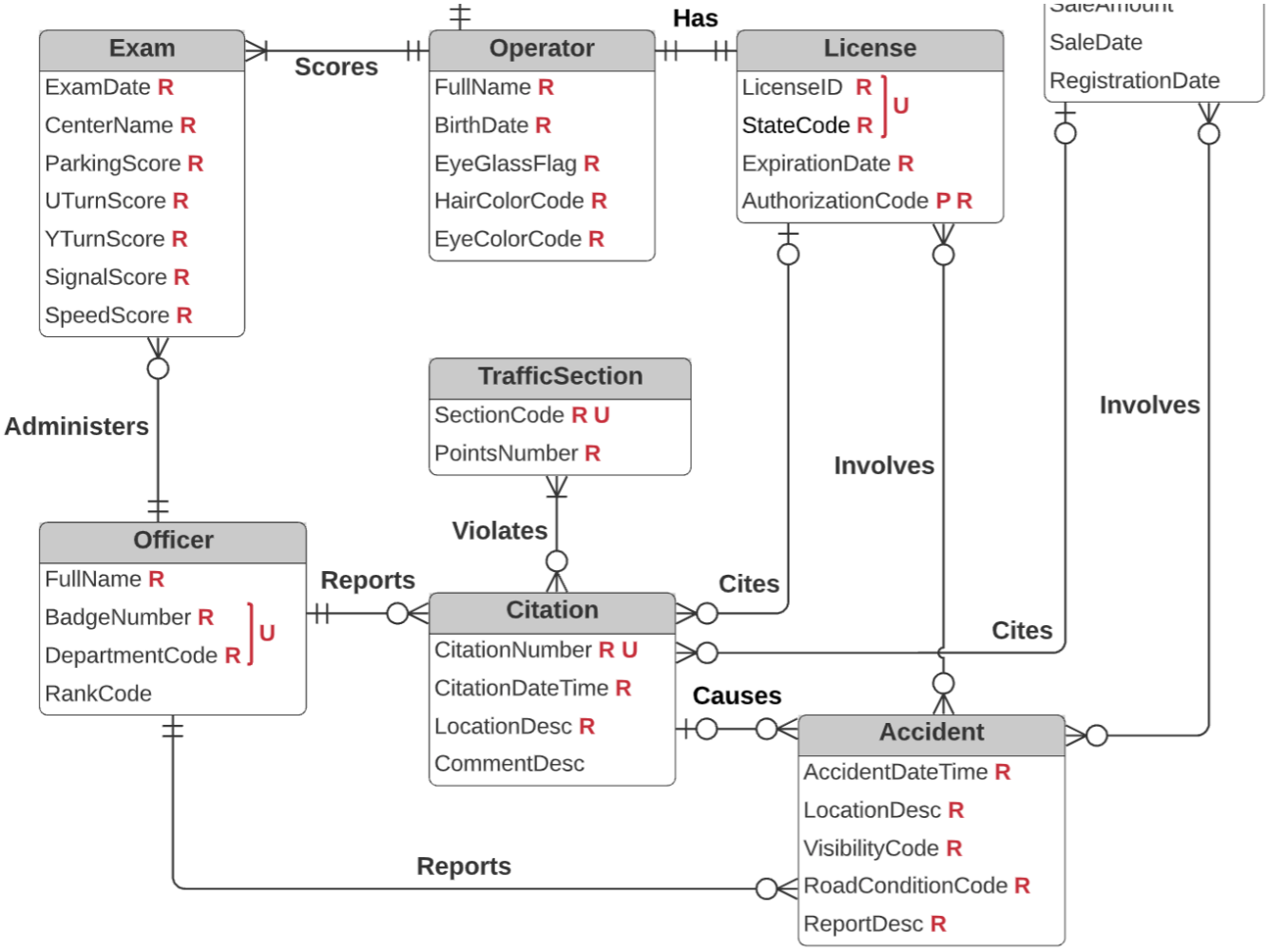
Like relationship cardinality, attribute cardinality is determined from requirements captured during discovery. Cardinality is sometimes inferred from common business practices. When requirements are uncertain, cardinality must be confirmed with staff in follow-up interviews.

Attribute cardinality appears in the figure below. Most attributes are followed by an R only, which indicates required, singular, and not unique.

Unique attributes, either simple or composite, may become primary keys in the logical design phase. Plural attributes become new tables. Required attributes become NOT NULL columns.

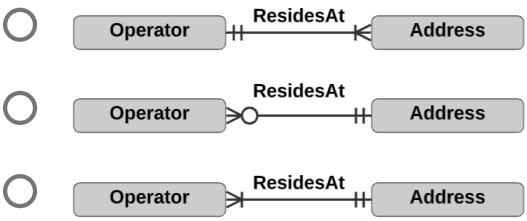
Figure 24.2.2: Attribute cardinality.





PARTICIPATION ACTIVITY 24.2.1: Cardinality.

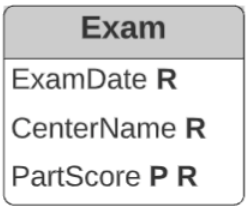
1) The model above indicates that each address belongs to exactly one operator. What is the correct model if each address belongs to one or more operators?



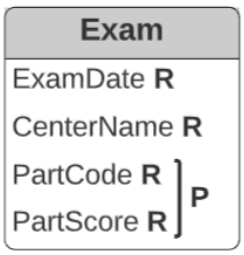
2) Can the five 'score' attributes of Exam be modeled as a plural attribute?

- No
- Yes, as a simple attribute

PartScore:



Yes, as a composite attribute (PartCode, PartScore):



24.3 Case study: Supertype and weak entities

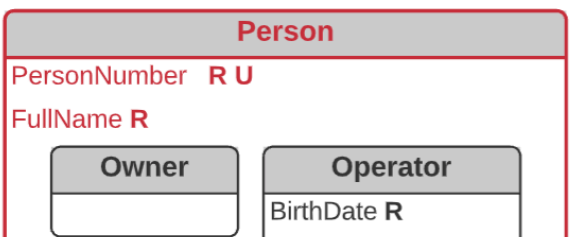
Person supertype entity

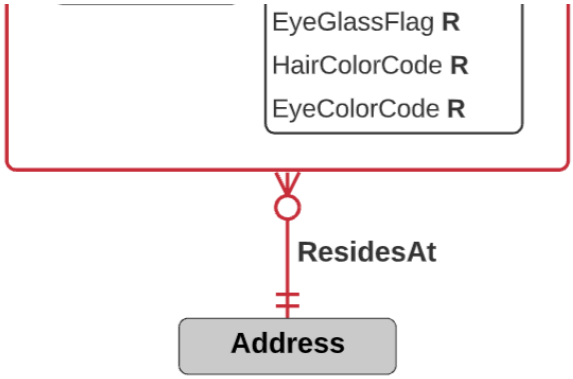
Following specification of cardinality, special entity types are determined. Special entity types include supertype, subtype, strong, and weak entities.

Owner and *Operator* are similar entities. Both have a *FullName* attribute and a *ResidesAt* relationship. Both consist of people tracked by the DMV. Follow-on interviews confirm that the DMV wants to identify owners and operators with a uniform, internal *PersonNumber*. For these reasons, a new supertype *Person* is created.

Attributes and relationships that are shared by *Owner* and *Operator* move to *Person*. Since most owners are also operators, *Owner* and *Operator* subtypes overlap. Therefore, *Owner* and *Operator* are in different partitions and are horizontally aligned on the diagram. Since most people are both owners and operators, no partition attributes are created.

Figure 24.3.1: Person supertype entity.





The *Officer* entity also includes people and might be considered a subtype of *Person*. However, *Officer* shares only the *FullName* attribute with *Owner* and *Operator*. Since the DMV does not track officer addresses, the *ResidesAt* relationship does not apply. Interviews confirm that the DMV does not want to assign a *PersonNumber* to officers. So, *Officer* is not modeled as a subtype of *Person*.

Event supertype entity

Accident and *Citation* are also similar entities:

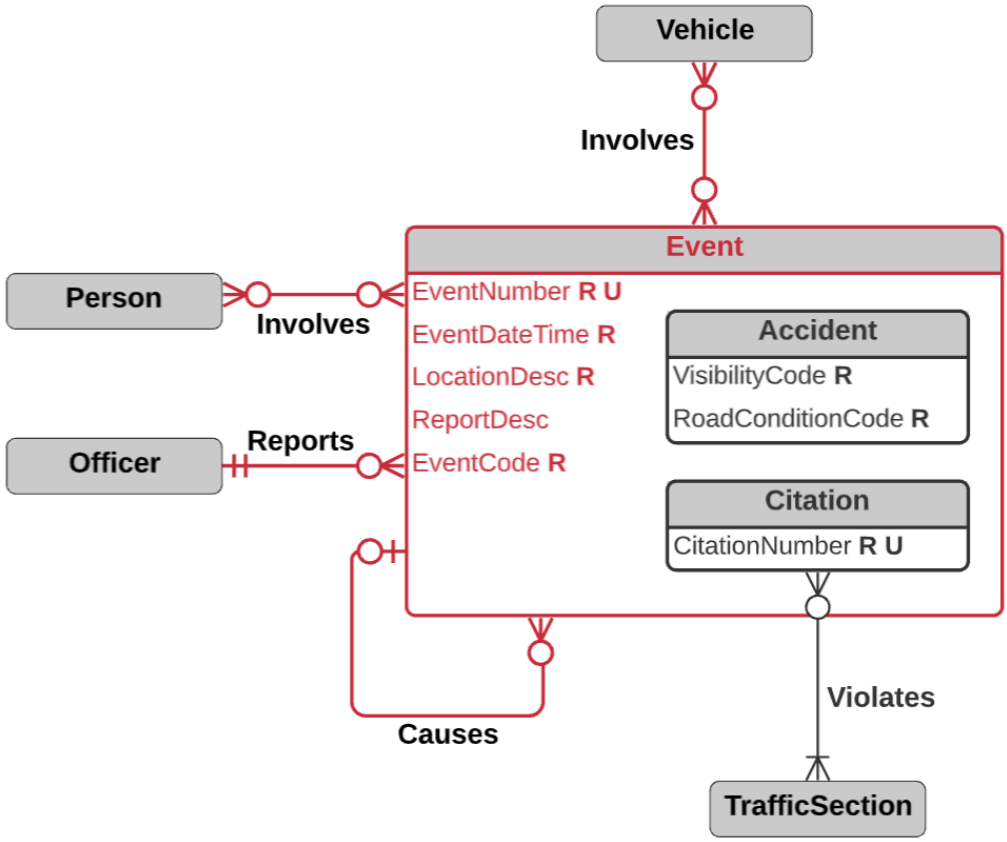
- Both are reported by an officer and have a date, time, and location.
- The *CommentDesc* and *ReportDesc* attributes have different names but are both free-form descriptive text.
- The *Cites* and *Involves* relationships to *Vehicle* have different names but similar meaning and cardinality.
- The *Cites* and *Involves* relationships to *License* also have similar meaning and cardinality.

For these reasons, a new supertype *Event* is created. Shared attributes and relationships move to *Event*. Several adjustments are made to the model:

- *ReportDesc* was required in *Accident*, but *CommentDesc* was optional in *Citation*. The new *ReportDesc* attribute of *Event* is optional.
- *Citation-Cites-License* and *Accident-Involves-License* become *Event-Involves-Person*. *Person* replaces *License* so that the DMV can track unlicensed participants in an event.
- *Citation-Causes-Accident* is generalized to a reflexive relationship *Event-Causes-Event*, for greater flexibility in tracking related accidents and citations.

A citation, such as speeding, may cause an accident. However, the DMV views related accidents and citations as distinct events. Consequently, *Accident* and *Citation* subtypes are disjoint, in the same partition, and vertically aligned on the diagram. The partition attribute *EventCode* has value 'A' for an event that is an accident and 'C' for an event that is a citation.

Figure 24.3.2: Event supertype entity.



Address and Exam weak entities

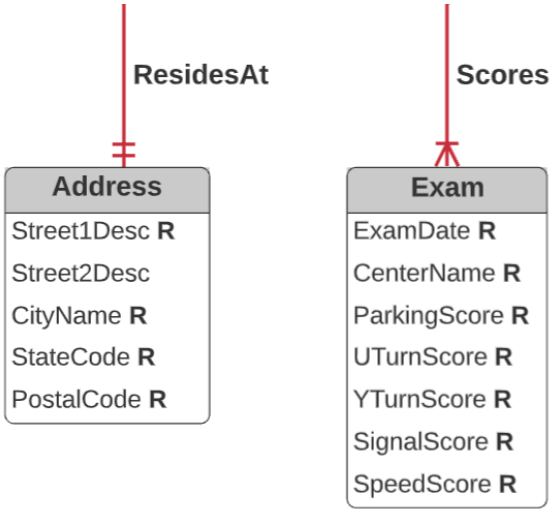
An identifying attribute is unique, singular, and required. Strong entities have identifying attributes. *Vehicle*, *Person*, *Event*, *License*, *Officer*, and *TrafficSection* are strong.

Weak entities have identifying relationships rather than identifying attributes. *Address* and *Exam* are weak:

- *Address* is identified via the *Person-ResidesAt-Address* relationship.
- *Exam* is identified via the *Operator-Scores-Exam* relationship.

Subtype entities have an identifying relationship to the supertype entity. The identifying relationship is implicit and not shown on the diagram. *Owner* and *Operator* are identified by *PersonNumber*. *Accident* and *Citation* are identified by *EventNumber*.

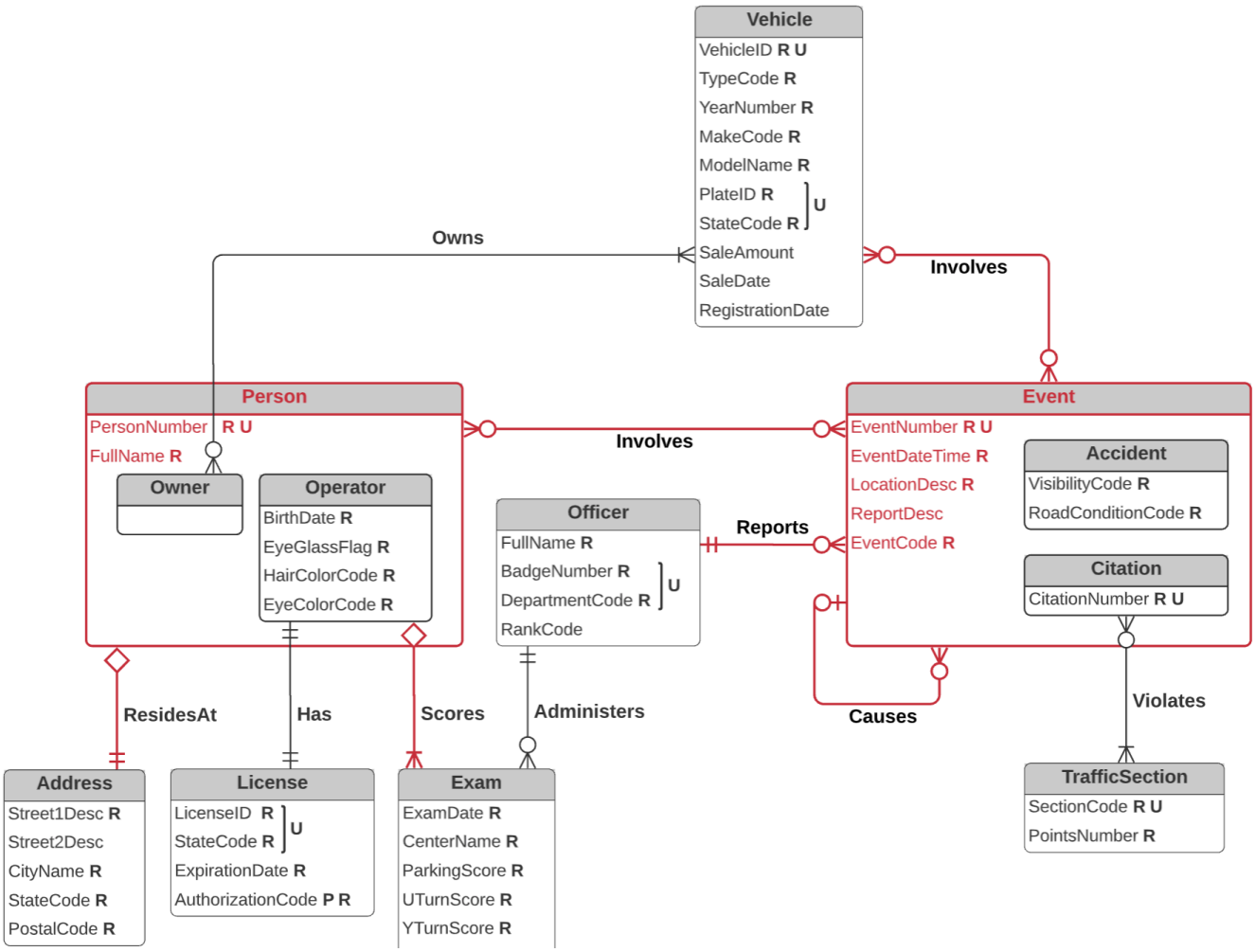
Figure 24.3.3: Address and Exam weak entities.

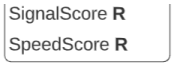


Completed ER diagram

The following diagram shows the completed ER model, with supertype and subtype entities, and identifying relationships.

Figure 24.3.4: Completed ER diagram.



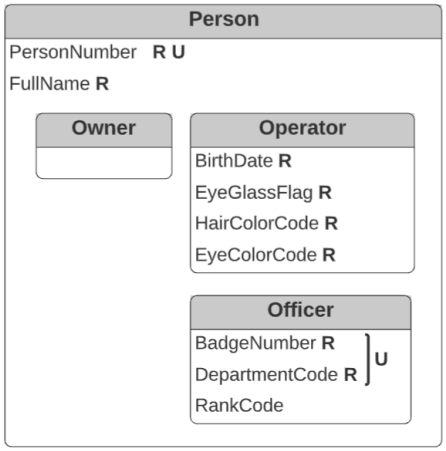


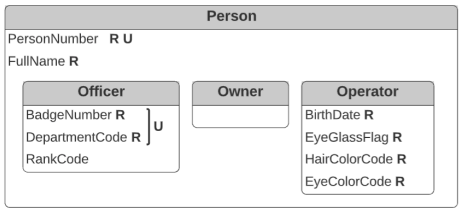
PARTICIPATION ACTIVITY 24.3.1: Supertype and weak entities.

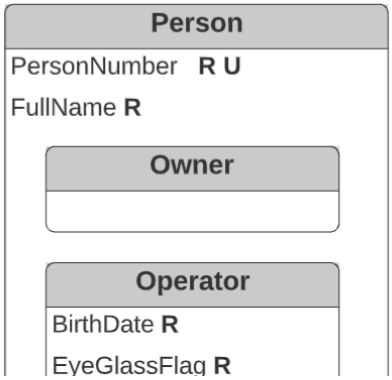
1) The DMV decides to assign a unique address ID to each address. The Address entity is now a _____ entity.

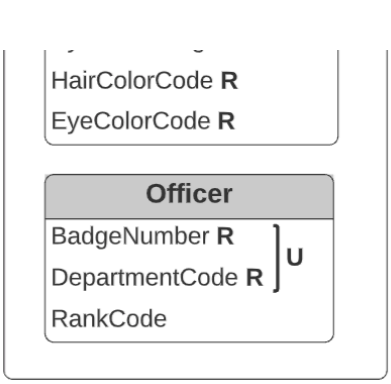
   - Strong
   - Weak
   - Subtype

2) The DMV decides to assign officers a person number and model Officer as a subtype of Person. Which diagram is correct?









24.4 Case study: Implementing entities

Strong entities

The first step of logical design is specification of primary keys. Primary keys must be singular, required, and unique. Primary keys should also be stable, simple, and meaningless, however these characteristics are not necessary. If no suitable primary key is available, the database designer may introduce an artificial key.

Strong entities become strong tables. Primary keys of strong tables are:

- *VehicleID*, or VIN, has 17 characters and encodes information such as manufacturer and model year. Since the VIN is meaningful and complex, a database designer may introduce an artificial primary key. However, a VIN never changes and is commonly used to identify vehicles, so *VehicleID* is selected as the primary key of *Vehicle*.
- *(LicenseID, StateCode)* is meaningful and complex, but also stable and commonly used to identify drivers. So *(LicenseID, StateCode)* is selected as the primary key of *License*.
- *(BadgeNumber, DepartmentCode)* is meaningful, complex, and unstable, since an officer's department may change. So an artificial primary key *OfficerNumber* is created for *Officer*.
- *SectionNumber* is stable and simple, but meaningful. Since section numbers rarely change, *SectionNumber* is selected as the primary key of *TrafficSection*.
- *Person* and *Event* are new supertype entities. No unique attributes were discovered for these entities, so identifying attributes *PersonNumber* and *EventNumber* were created in the analysis phase. These identifying attributes become artificial primary keys.

The table below summarizes the characteristics of these primary keys.

Table 24.4.1: Strong table primary keys.

Table	Primary key	Singular Required Unique	Stable	Simple	Meaningless	Artificial
Vehicle	VehicleID	✓	✓	—	—	No
License	(LicenseID, StateCode)	✓	✓	—	—	No
Officer	OfficerNumber	✓	✓	✓	✓	Yes
TrafficSection	SectionNumber	✓	✓	✓	—	No
Person	PersonNumber	✓	✓	✓	✓	Yes
Event	EventNumber	✓	✓	✓	✓	Yes

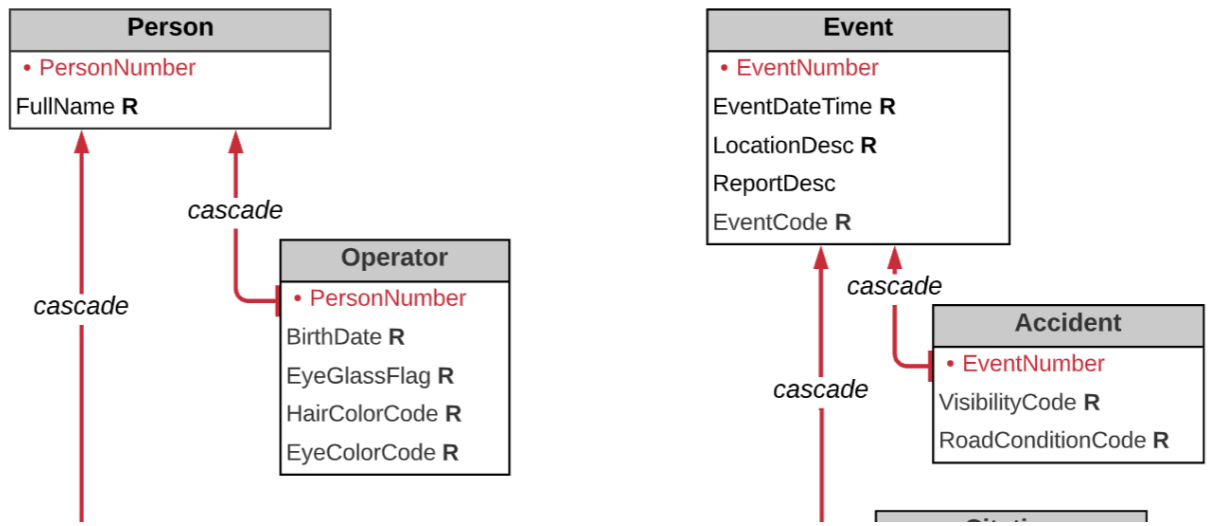
Subtype entities

Subtype entities become subtype tables. The subtype primary key is the same as the supertype primary key:

- *PersonNumber* is the primary key of *Owner* and *Operator*.
- *EventNumber* is the primary key of *Accident* and *Citation*.

PersonNumber and *EventNumber* are also foreign keys, referencing *Person* and *Event*, with cascade on primary key update and delete.

Figure 24.4.1: Subtype tables.



Weak entities

Weak entities become weak tables. The primary key of a weak table includes the primary key of the identifying table:

- A person can take the exam many times, but only once on each day, so the *Exam* primary key is *(PersonNumber, ExamDate)*.
- A person has at most one address in the DMV system, so the *Address* primary key is *PersonNumber*.

PersonNumber is also a foreign key in both tables. The foreign keys reference *Person* and *Operator* with cascade on primary key update and delete.

Figure 24.4.2: Weak tables.

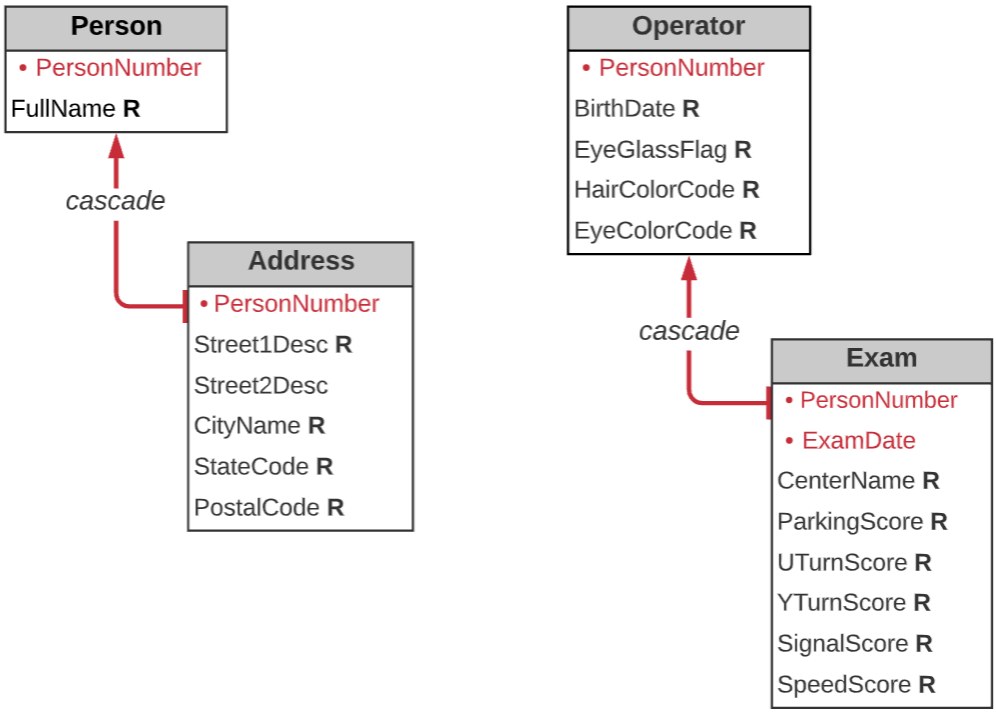
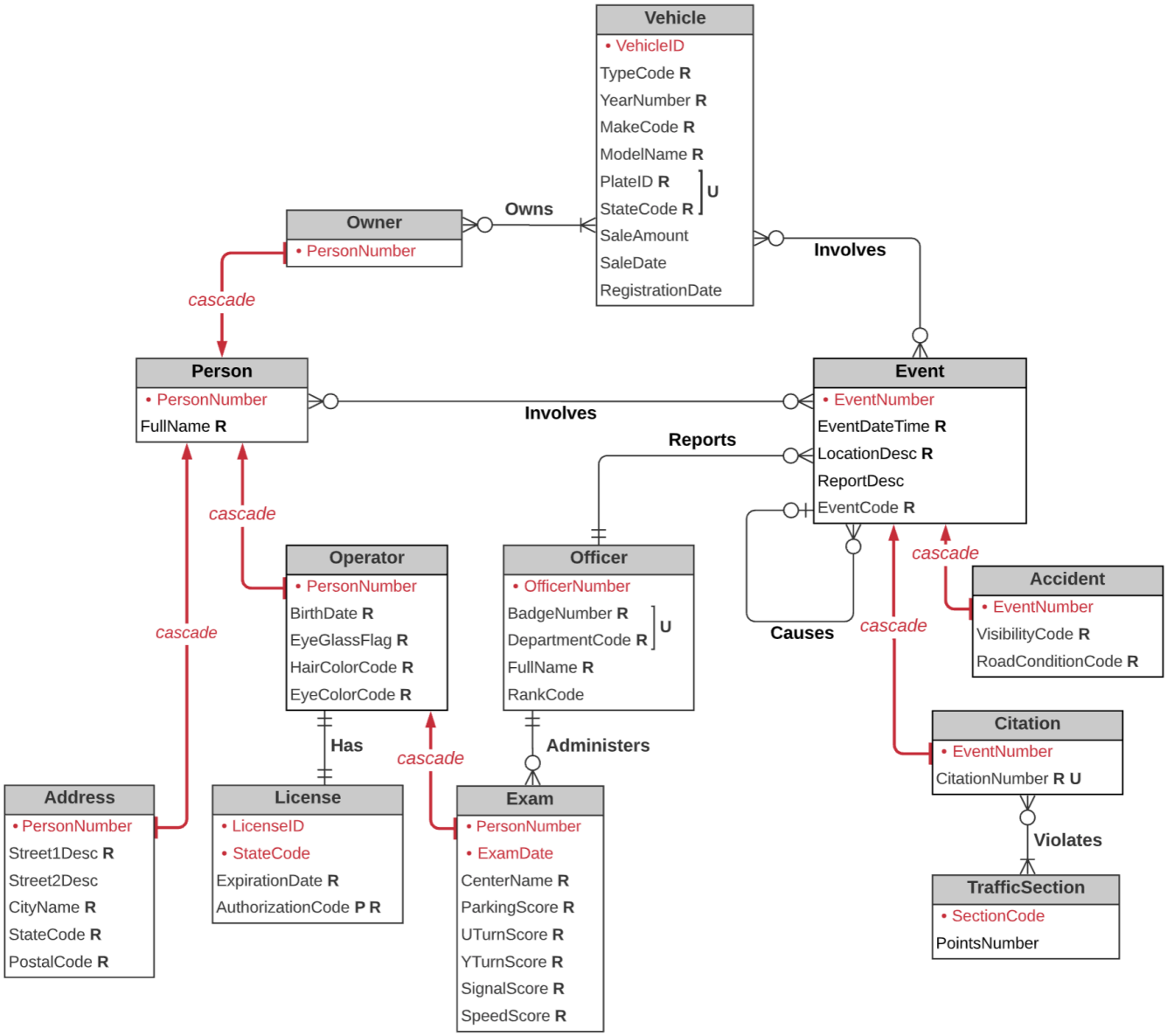


Table diagram

The table diagram below depicts the logical design after entities are implemented. Relationships and attributes have not yet been implemented, so the diagram is incomplete.

Primary and foreign keys appear in red. Relationships and attributes appear in black.

Figure 24.4.3: Table diagram after implementing entities.



PARTICIPATION ACTIVITY 24.4.1: Implementing entities.

1) When a foreign key is part of a primary key, which foreign key rule *cannot* be specified?

- RESTRICT
- CASCADE
- SET NULL

2) For a primary key, the update

2) For a given foreign key, the update and delete rules must _____.

- always be the same
- sometimes be the same
- always be different

3) The primary key of a strong entity is _____ stable.

- always
- usually
- never

4) The identifying relationship of a weak entity may be one-one maximum.

- True
- False

24.5 Case study: Implementing relationships

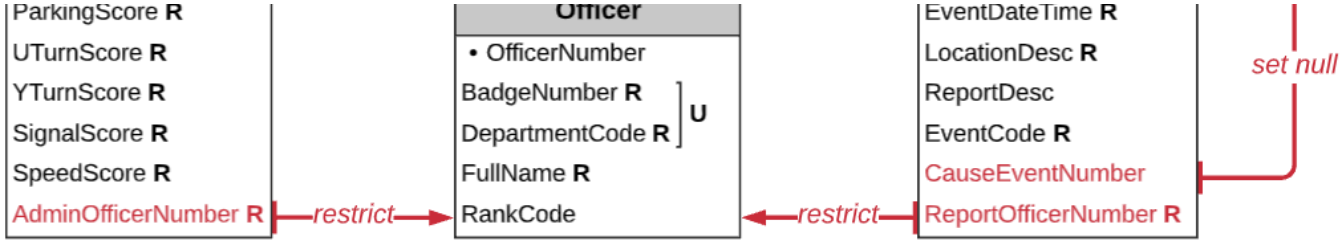
Many-one relationships

The first step of logical design implemented identifying relationships as foreign keys. The next step implements the remaining many-one relationships:

- *Officer-Administers-Exam*
- *Officer-Reports-Event*
- *Event-Causes-Event*

The foreign key is placed in the table on the many side of the relationship. The foreign key name is a short version of the relationship name followed by the referenced primary key.

Figure 24.5.1: Many-one relationships are foreign keys.



Foreign key cardinality and rules depend on the minimum cardinality of the opposite side of the relationship:

- If the minimum is one, the foreign key column is required and rules are either restrict or cascade.
- If the minimum is zero, the foreign key column is optional and foreign key rules are set null.

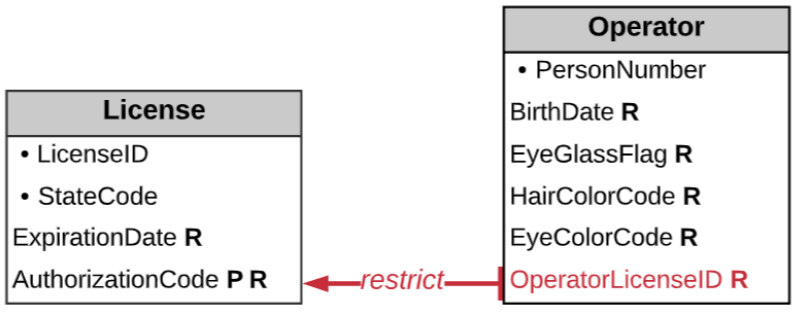
Ex: Each event is reported by an officer, so *ReportOfficerNumber* is required and rules are restrict. Some events are not caused by another event, so *CauseEventNumber* is optional and rules are set null.

One-one relationships

One-one relationships are implemented as a foreign key in the table with fewer rows. The only remaining one-one relationship is *Operator-Has-License*. This relationship is singular and required on both sides, so the tables are the same size and the foreign key can go in either. Placing the foreign key in *Operator* simplifies queries that need driver's license numbers but no additional license information.

Since the minimum on the opposite side of the relationship is one, the foreign key is required and rules are restrict.

Figure 24.5.2: One-one relationships are foreign keys.



Many-many relationships

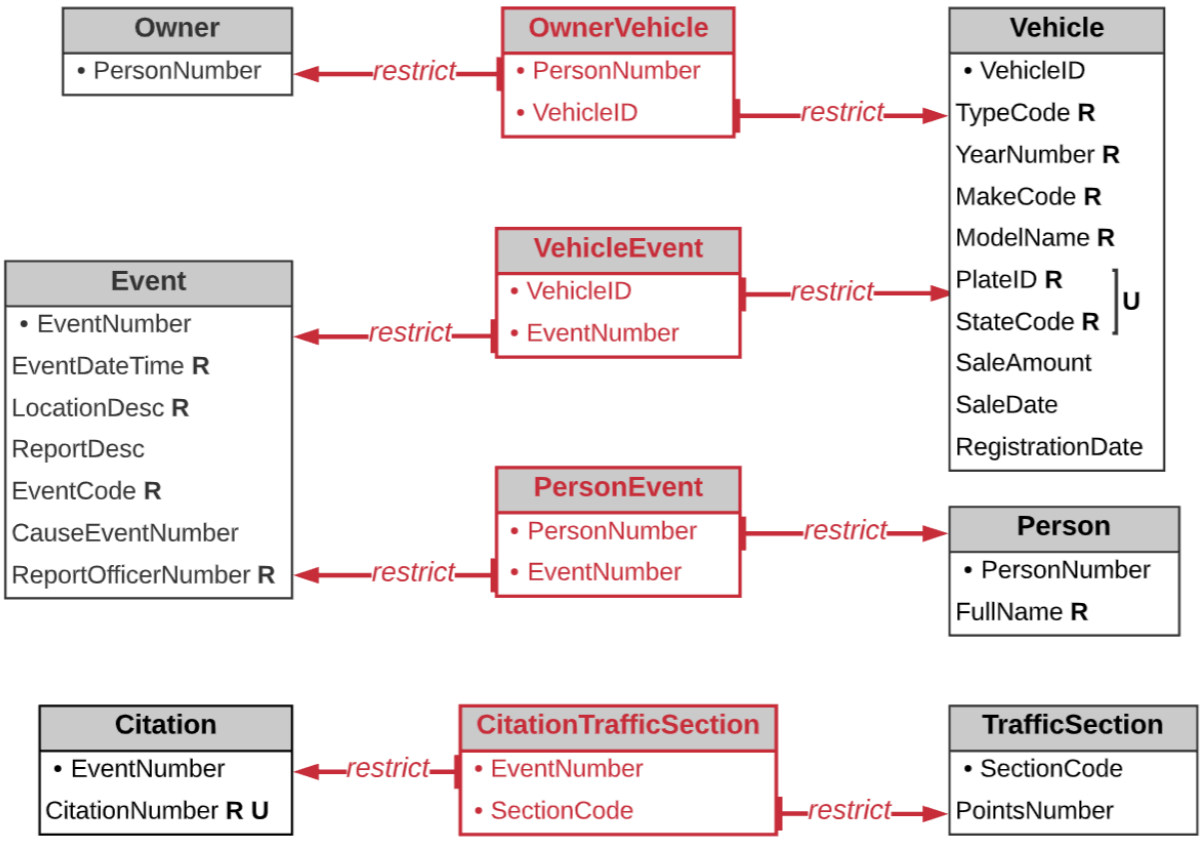
Many-many relationships become new tables. The many-many relationships are:

Many-many relationships become new tables. The many-many relationships are:

- *Owner-Owns-Vehicle*
- *Event-Involves-Vehicle*
- *Event-Involves-Person*
- *Citation-Violates-TrafficSection*

The name of a many-many table combines the names of the related tables.

Figure 24.5.3: Many-many relationships are new tables.



A many-many table contains foreign keys that reference the related tables. The primary key is the composite of these foreign keys. Since primary key columns may not be null, the foreign keys are required and rules may not be set null. DMV staff prefer manual, rather than automatic, deletion of rows in the many-many table. So restrict, rather than cascade, rules are specified.

Owner and Vehicle adjustments

After relationships are implemented, two adjustments are made to the design.

Owner has just one column, the primary key, and is unnecessary. The table is eliminated, *OwnerVehicle* is renamed *PersonVehicle*, and the *OwnerPersonNumber* foreign key references *Person*.

The DMV maintains historical records of vehicle sales. *SaleAmount* and *SaleDate* depend on both the owner and the vehicle and therefore move from *Vehicle* to *PersonVehicle*.

Figure 24.5.4: Owner and Vehicle adjustments.

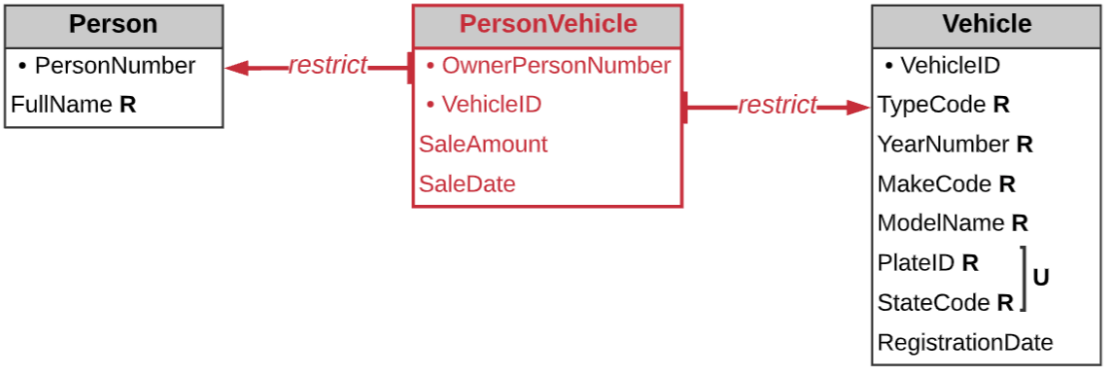
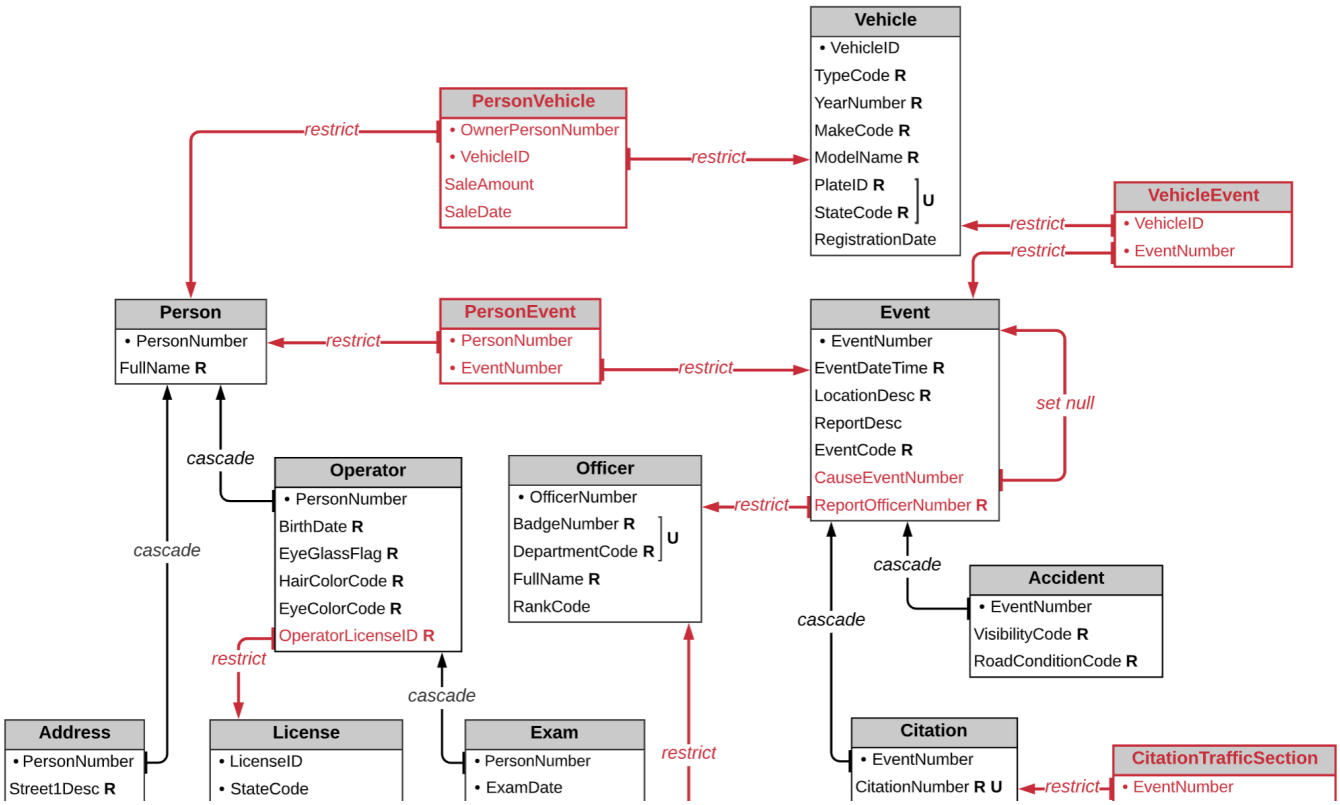


Table diagram

The table diagram below depicts the logical design after relationships are implemented. Plural attributes have not yet been implemented, so the diagram is incomplete.

Many-many tables, new foreign keys, and relocated columns are in red.

Figure 24.5.5: Table diagram after implementing relationships.



PARTICIPATION ACTIVITY 24.5.1: Implementing relationships.

Match the foreign key to the description.

If unable to drag and drop, refresh the page.

OwnerPersonNumber **OperatorLicenseID** **AdminOfficerNumber**

CauseEventNumber

	Implements a one-many, non-reflexive relationship
	Implements a many-many relationship
	Implements a one-many, reflexive relationship
	Implements a one-one relationship

Reset

24.6 Case study: Implementing attributes

Plural attributes

A plural attribute is usually implemented as a new weak table with two columns:

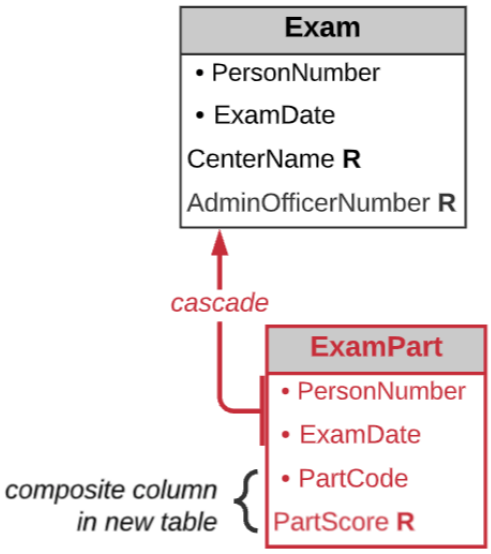
- A foreign key that references the initial table

- A column that implements the plural attribute

These columns may be simple or composite and, together, comprise the primary key of the new table. The new table name combines the names of the initial table and the new column.

The five score attributes of *Exam* might be modeled as a composite plural attribute *(PartCode, PartScore)*. Follow-up interviews reveal that the licensing exam changes periodically, with different parts from year to year. Since the maximum is not fixed, the composite plural attribute is implemented as a new table.

Figure 24.6.1: Plural attributes.

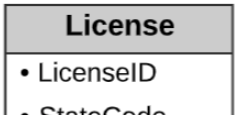


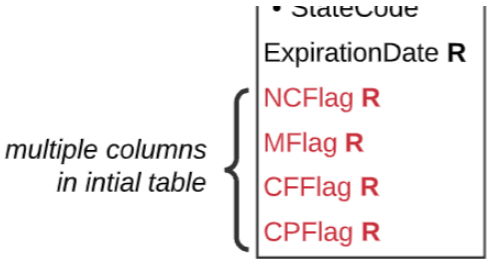
Alternative designs

Alternatively, a plural attribute with a small and fixed maximum may be implemented as multiple columns in the initial table. This design is less flexible but has fewer tables, resulting in fewer join queries.

The *AuthorizationCode* attribute of *License* is plural. A license has any of four authorization codes: NC, M, CF, and CP. Since the maximum is four and unlikely to change, *AuthorizationCode* is implemented as four columns in the *License* table. The columns have type Flag, a boolean value that indicates whether a code applies to the license.

Figure 24.6.2: Alternative design.



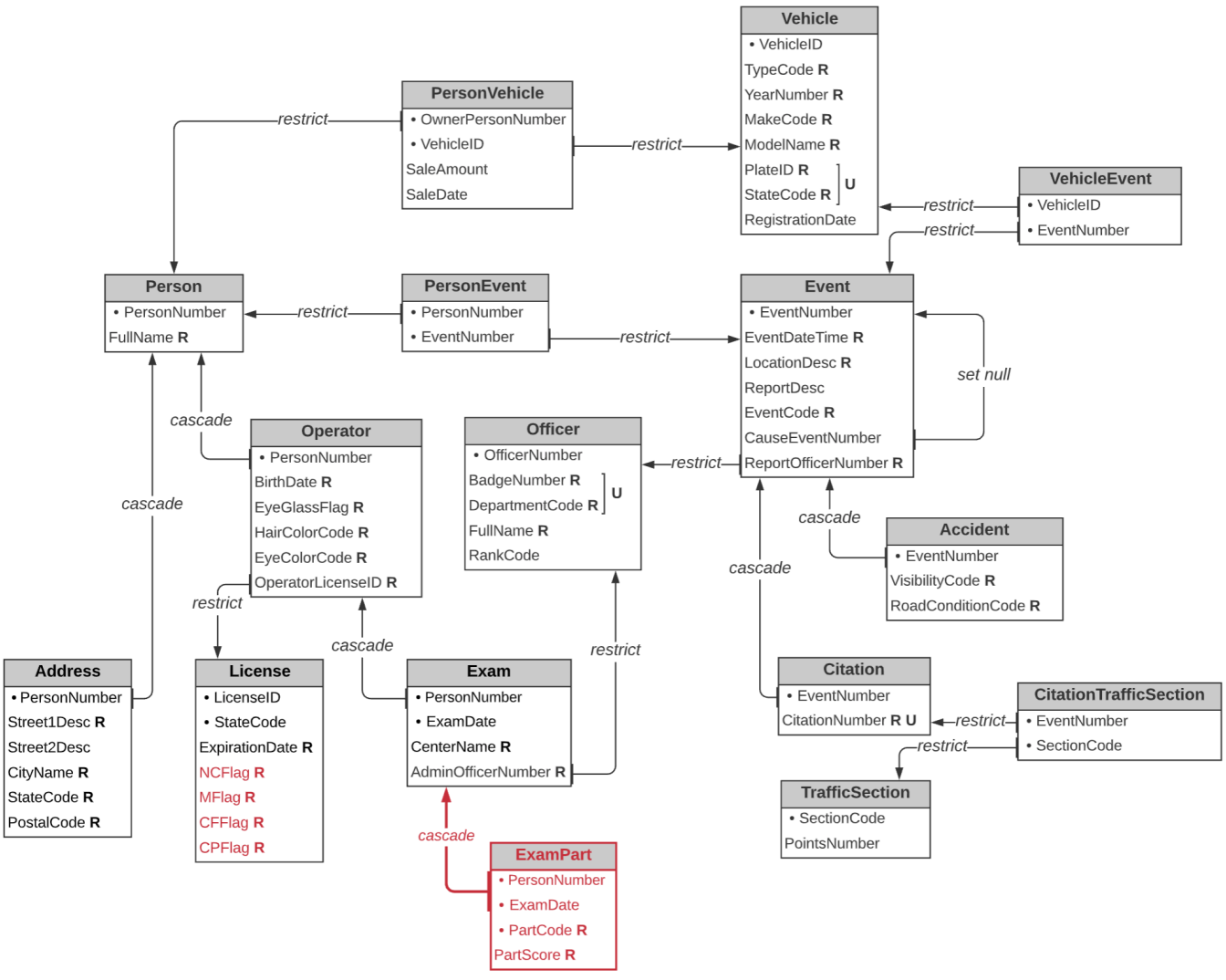


Some databases, including MySQL, support the SET data type. SET is a complex type that allows multiple values, selected from a fixed group of values, in one cell. A plural attribute with a small and fixed maximum may be implemented as a single column of type SET.

Completed table diagram

The table diagram below depicts the completed logical design. New columns and tables appear in red.

Figure 24.6.3: Completed table diagram.



CREATE TABLE statements

When the table diagram is complete, generating CREATE TABLE statements is straightforward. Table and column names are taken directly from the table diagram. Data types and constraints are determined as follows:

- Data types are derived from the column name suffix. Each suffix is a standard attribute type and is associated with a data type.
- Primary keys, denoted with bullets, have a PRIMARY KEY constraint.
- Artificial primary keys are designated UNSIGNED and AUTO_INCREMENT.
- Required columns, denoted with an R, have a NOT NULL constraint.
- Unique columns, denoted with a U, have a UNIQUE constraint.
- Foreign keys and rules, denoted with arrows in the diagram, are implemented with a FOREIGN KEY constraint.

Ex: CREATE TABLE statements for *Officer*, *Event*, and *ExamPart* appear below.

Figure 24.6.4: Officer table.

```
CREATE TABLE Officer (
   OfficerNumber INT UNSIGNED
AUTO_INCREMENT,
   BadgeNumber INT NOT NULL,
   DepartmentName VARCHAR(30) NOT NULL,
   FullName VARCHAR(30) NOT NULL,
   RankCode CHAR(3),
   PRIMARY KEY (OfficerNumber),
   UNIQUE (BadgeNumber, DepartmentName)
);
```

Figure 24.6.5: Event table.

```
CREATE TABLE Event (
   EventNumber INT UNSIGNED
AUTO_INCREMENT,
   EventDateTime DATETIME NOT NULL,
   LocationDesc TEXT(1000) NOT NULL,
   ReportDesc TEXT(5000),
   EventCode CHAR(2) NOT NULL,
   CauseEventNumber INT UNSIGNED,
   ReportOfficerNumber INT UNSIGNED NOT
```

```
    NULL,
       PRIMARY KEY (EventNumber),
       FOREIGN KEY (CauseEventNumber)
          REFERENCES Event (EventNumber)
          ON UPDATE SET NULL
          ON DELETE SET NULL,
       FOREIGN KEY (ReportOfficerNumber)
          REFERENCES Officer (OfficerNumber)
          ON UPDATE RESTRICT
          ON DELETE RESTRICT
    );
```

Figure 24.6.6: ExamPart table.

```
CREATE TABLE ExamPart (
   PersonNumber INT UNSIGNED,
   ExamDate DATE,
   PartCode CHAR(2),
   PartScore TINYINT UNSIGNED NOT NULL
      CHECK (PartScore <= 100),
   PRIMARY KEY (PersonNumber, ExamDate,
PartCode),
   FOREIGN KEY (PersonNumber, ExamDate)
      REFERENCES Exam (PersonNumber, ExamDate)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);
```

PARTICIPATION ACTIVITY 24.6.1: Implementing attributes.

The DMV classifies addresses as either residential, commercial, or government. Select the best implementation for each of the following scenarios.

1) Each address has exactly one of these values.

- AddressType column in the Address table
- ResidentialFlag, CommercialFlag, and GovernmentFlag columns in the Address table
- AddressType column, along with a foreign key referencing

Address, in a new AddressType table

2) Some addresses serve multiple functions and have two or three of these values.

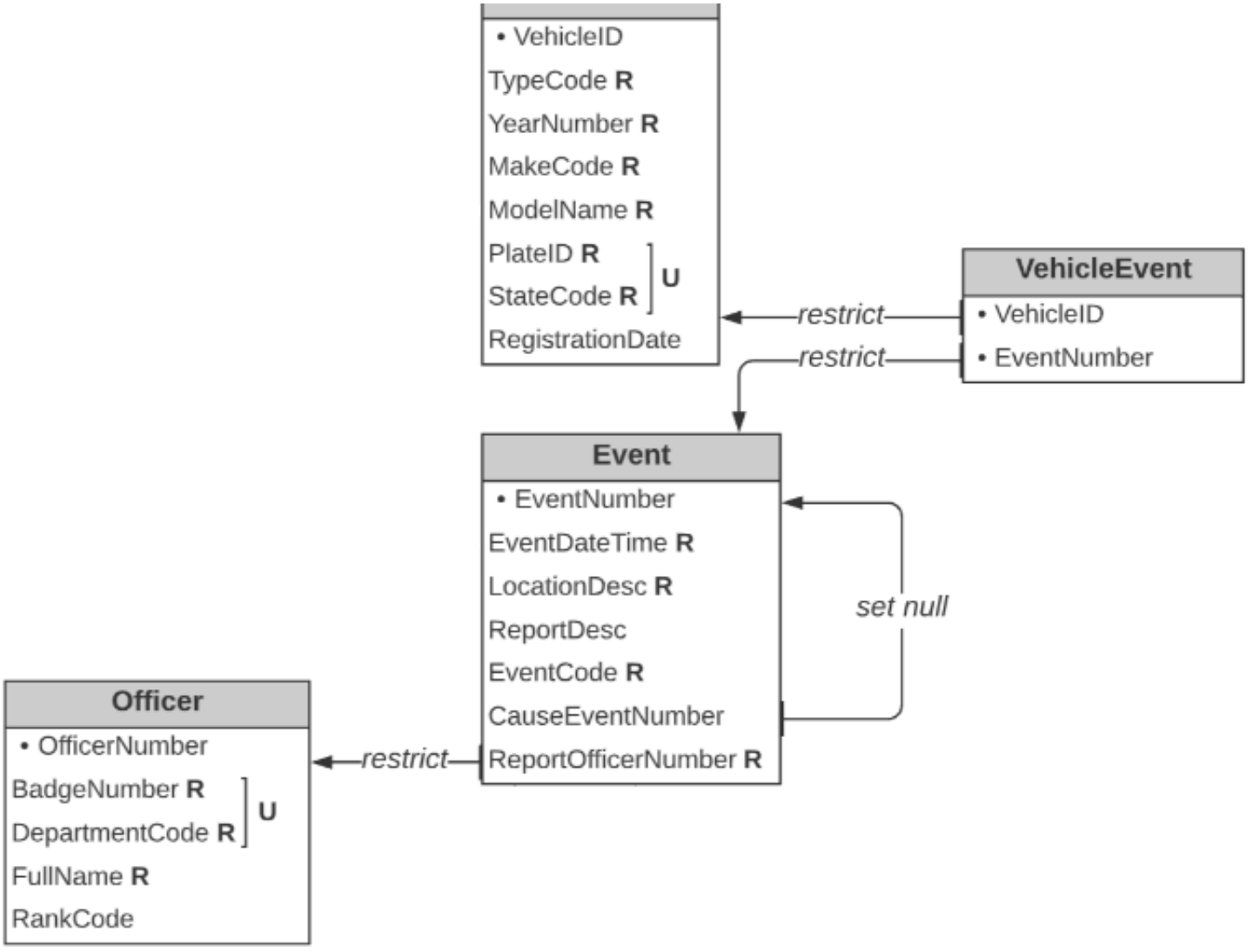
- AddressType column in the Address table
- ResidentialFlag, CommercialFlag, and GovernmentFlag columns in the Address table
- AddressType column, along with a foreign key referencing Address, in a new AddressType table

3) The classification has nine additional values, such as education and transportation, for a total of twelve. Some addresses have many of these values.

- AddressType column in the Address table
- Twelve columns of type Flag in the Address table
- AddressType column, along with a foreign key referencing Address, in a new AddressType table

24.7 LAB - Create Vehicle and EventVehicle tables

The California DMV's completed table diagram is presented in the zyBook. The following is a portion of the table diagram:



The SQL below creates the **Officer** and **Event** tables.

Create the **Vehicle** and **VehicleEvent** tables with columns matching the table diagram above and with the appropriate keys and constraints. Some additional notes:

- All "ID", "Number", and "Code" columns should use unsigned INT data types.
- All primary keys should auto-increment.
- The vehicle's model name should be limited to 20 characters.

544874.3500394.qx3zqy7

LAB ACTIVITY | 24.7.1: LAB - Create Vehicle and EventVehicle tables | 0 / 10

Main.sql — Load default template...

```
1 CREATE TABLE Officer (
2    OfficerNumber INT UNSIGNED AUTO_INCREMENT,
3    BadgeNumber INT NOT NULL,
```

```
    DepartmentName VARCHAR(30) NOT NULL,
    FullName VARCHAR(30) NOT NULL,
    RankCode CHAR(3),
    PRIMARY KEY (OfficerNumber),
    UNIQUE (BadgeNumber, DepartmentName)
);

CREATE TABLE Event (
    EventNumber INT UNSIGNED AUTO_INCREMENT,
    EventDateTime DATETIME NOT NULL,
    LocationDesc TEXT(1000) NOT NULL,
    ReportDesc TEXT(5000),
```

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.