Chapter 3
Data Modeling and Normalization

Introduction

In this third chapter, we will explore effective data modeling. Readers so far have had a big bite of database history and relational theory; things are getting busy. Next up, we will start exploring the complexity of the process known as normalization and gain advanced skills in database design in addition to preparing everyone for advanced analytical skills with SQL.

After observing historical human categorization and classification systems that are hierarchical in design, the previous chapter introduced the relational model and explained how to read a completed logical data model including crow's foot notation. From this foundation, you will now be exposed to the process of creating your own unique database which will add a layer of independence as a superior data analyst free from relying on others to provide relevant data. This will prove to be a valuable skill!

This chapter reinforces the symbols and vernaculars involved with data modeling, normalization, and the inner workings of creating a relational database from initial research through to a physical database schema. We will see two different methods of creating a professional Entity-Relationship Diagram (ERD) that is essentially a blueprint that is ready to code as it is in Third Normal Form (3NF). These skills will not only create fluency in the database design process, but also functional independence from needing any other engineer to aid in your analyses journey.

Remember to keep an open mind as we expand concepts of storing data for machines and not people. Relational design is still considered revolutionary!

Structure

By the end of this chapter, readers will:

- Understand the Systems Development Lifecycle with keen awareness as to where and when developers focus on the database
- Explore the fundamentals of data modeling and explain the range of symbols used in the conceptual and logical design phases
- Explain the high-level process of normalization as well as the characteristics of each normal form
- See two methods of creating a fully normalized data model including a traditional approach and another called 'ironing'

As we are embarking on a formal design process, it is important to remind ourselves of the standard framework introduced earlier in chapter 1, "*Organize or Die*", that will improve the probability that we build something deemed useful by our users.

Systems Development Lifecycle

The Systems Development Lifecycle (SDLC), as we saw earlier, provides a structure by which developers, users, and stakeholders in any system can learn about the requirements and desired outcomes of a project and ultimately better craftspeople. We become more and more efficient at our jobs after each iteration (or "build cycle") of the SDLC because we are documenting each step and measuring the results. This is true whether we are constructing tall buildings or bridges, manufacturing cars, boats, sandwiches, bird feeders, or delivering consulting services. Things are no different when building databases or other software applications.

We embellish the SDLC model slightly by expanding the DESIGN phase to include two subphases of Conceptual and Logical database design:

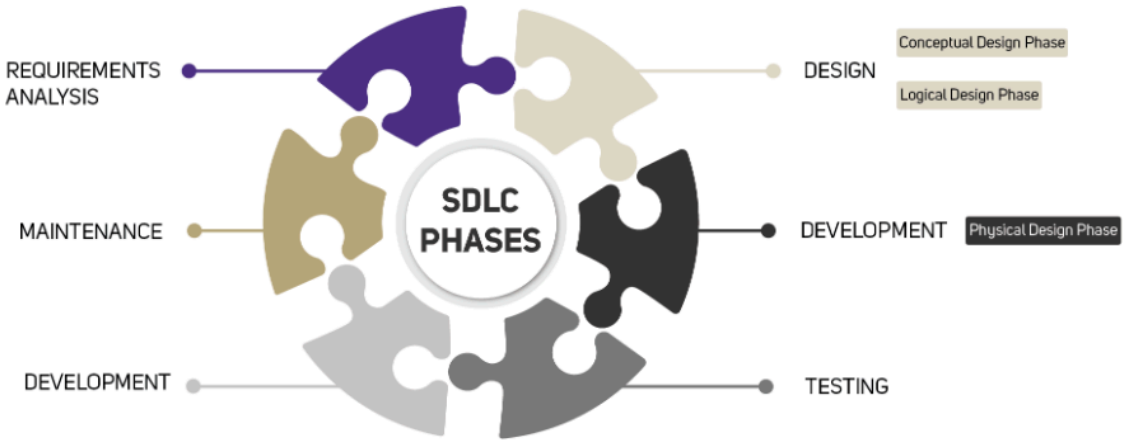


Figure 3.1: Visual image of iterative Phases in Systems Development Life Cycle (SDLC)

In the section that is titled DESIGN, database developers will have two subphases in the Conceptual Design Phase and the Logical Design Phase. In the section titled *Development* from the graphic above, database professionals will have the Physical Design Phase. We will explore these sub-phases in greater detail coming up.

The important part of this section is to be aware of the location where database design and development occurs within the wider scope of software application development. Much of the effort to construct and code databases will be in the DESIGN phase of the SDLC.

As we begin the study of data modeling, it is important to see it as an opportunity to articulate the guts of an organization and decide the elements that end up being recorded perhaps thousands and thousands of times each day. This opportunity can establish the key objects that are critical to learning throughout an organization whereby the entire mission of the business can continuously improve year after year. Missing on this core understanding by cutting corners trying to save time will ultimately limit the specific data that is captured and therefore curtail the scope of data that can be analyzed. Shortcuts during the data modeling phase can literally be the difference between an organization having an exceptional presence in an industry or becoming irrelevant and leading to premature death. Organizations that can learn over time will outperform others that do not; it all starts with being able to analyze the effectiveness of everything the organization produces as well as the resources and processes to deliver them to customers.

One of the best methods to understand these complex processes is to see demonstrations of a real-life database in a step-by-step manner. In this chapter, an incremental

construction of two databases will be shown starting with brainstorming business nouns and finishing with an Entity-Relationship Diagram (ERD) in at least Third Normal Form.

Where do databases come from?

Let's start from the beginning; when and where do relational databases get created? If we think in terms of a project, a relational database is usually 'along for the ride' as a supporting component to an application that is intended to either solve an existing problem or address a new market opportunity. This simply means that many companies start with an application focused on engaging people to address a data problem and afterwards put attention to designing a database to support the application.

Solving an existing problem

Problems are endemic in most organizations; ask most members (manager, executive or employee) and many will answer that their average day could be improved if there was better access to data or higher quality information. This internal data strife is often the driving force that sparks an interest to establish a project that provides a fix. This fix could be relatively simple (such as a new dashboard drawing on existing data) or it could be more complicated (like designing and building out an application to support a new process that captures data that does not currently exist).

New market opportunity

Companies are encouraged to be aware of new opportunities to grow and expand their businesses to generate additional revenue. Frequently these opportunities require new applications and processes to manage people and control the execution of processes or manufacture and delivery of products and services. Much of these efforts mandate brand new applications and the required supporting databases. Rarely are databases built without concern to an elaborate frontend interface or an application of some sort.

To simplify, problems and opportunities are brought to the attention of executives from all departments in a company. This includes operations/manufacturing, sales and marketing, legal, human resources, accounting and even the IT department. Each of these departments need to track events and associated costs with tools beyond pen and paper.

Many times, there are popular commercial products, such as Salesforce for customer relationship management and QuickBooks for accounting that are available as a subscription for relatively very low cost. Buying a ready-made product or service like these effectively reduces the risk (measured in time and money) and technical resource requirements of building new software from the ground-up. Aside from entering data into these commercial systems, they can be ready to go live nearly immediately.

Other times, a company has a very specific problem or opportunity and decides that a commercial product or service is not going to be a good fit or cost-effective in the long term. These companies also usually have the technical staff (or the financial resources to hire an external development company) to build a custom solution. The trade-off of spending more time and money on a custom application is owning the technical asset (as opposed to leasing) that is a better fit for the problem or opportunity being addressed.

Very quickly, departments are usually provided with an annual budget for payroll, facilities, office furniture, computers, and software tools. Common software tools include personal productivity applications like email, spreadsheets, and job specific applications like those that diagram architecture projects or do engineering costing from measurements.
Prior to receiving budget allocation, departments often review priorities, known issues, outstanding or ongoing development projects and make a prioritized list of projects for funding. Not every project gets funding!

Estimates would put the ratio of perhaps 1 out of every four proposals requested end up getting funding. Even the best proposals must complete thorough research into alternatives in addition to feasibility studies, and cost-benefit/risk analyses in support of funding requests to get to the final round of consideration. Again, it is not always the most important project that gets funded (or 'green lit') as it often comes down to which project has the most compelling argument from a high-ranking advocate or sponsor.

Some of the research for making a funding request occurs in the PLANNING or ANALYSIS phases of the SDLC. The following diagram is a very simplified flowchart illustrating the cyclical nature and sequence of decisions that frequently occurs with the creation of software application projects:

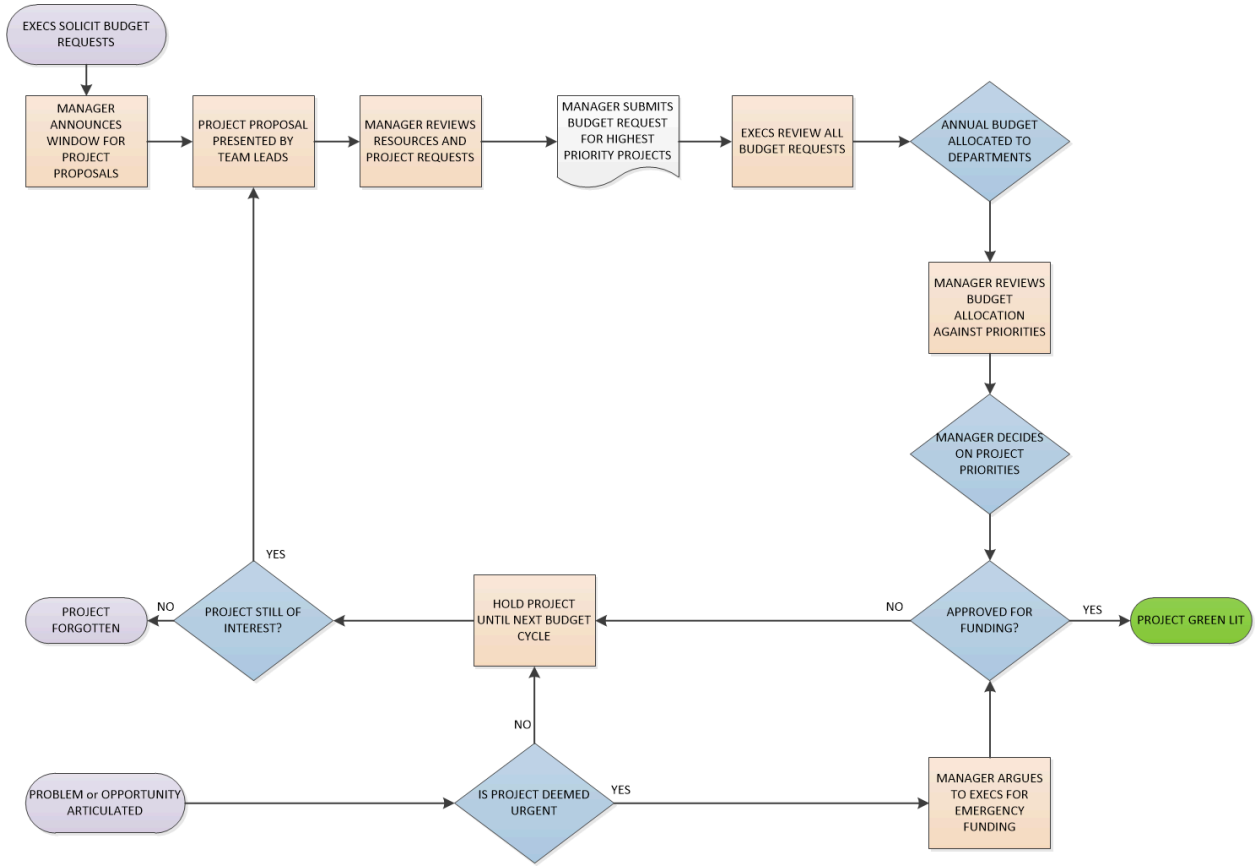


Figure 3.2: Visual image of an example process flow for project funding

Please note the flow chart depicted in Figure 3.2 begins with either a scheduled review of a plethora of budgetary concerns that are all encompassing and not limited to technology applications. The standard budget process often includes opportunities for many different projects to submit funding requests. Individual projects must be researched, documented, and argued for based on ROI and timeline for impact. Again, most requests are denied.

Once a project has been green lit and allocated funding, the real work has just begun. Assuming preliminary research has occurred in developing the material to request money, additional data-centric research must be conducted. We often call this sub-process defining requirements, which entails many forms such as observing how employees conduct business during regular operations of live activity, researching industry-centric best practices, as well as interviewing stakeholders and vendors. Once a body of knowledge has been assembled on the space getting a new application, we are ready to begin designing the database, starting with a robust data model that will address the stakeholder features that are required to be included in the application.

The process of designing a database has conceptual and logical phases; normalization occurs in the logical design phase after we have completed a consolidated conceptual diagram which includes all entities and attributes pieces of a puzzle.

Overview of METRO_TRANSIT

Many cities have an organized, formal, and publicly funded transportation system. This is fantastic and useful for getting people of all incomes and mobile abilities around town in a safe and predictable manner.

The effective ubiquitousness of these systems is why we have chosen METRO_TRANSIT as one of the working database examples for the entire textbook. While the data in METRO_TRANSIT is mostly fabricated, the structure and sophistication of the database allows learners to have a familiar backdrop while engaging in lessons and practice.

Let us consider a fictitious METRO_TRANSIT system that is struggling to manage operations with data held in a spreadsheet. They need a new relational database design:

DateTime	Passenger	RouteName	Type	Driver	StopName	Type	Destination	Type	Fare	Neighborhood
2/13/2025 7:36:00 AM (delayed)	Ivey Hazekamp	32 Capitol Hill-Downtown	Regular	Jimi Hendrix, 206555-7661	Broadway Ave Cherry St	Covered	Convention Center	Articulated	$ 4.50	Capitol Hill
2/13/2025 9:36	Darcel Eustache	78 Fremont-Waterfront-Downt	Special	Meryl Streep	Hwy 99-N 36th	Covered	Fremont, Space Needle, P70	Extra-long	$ 2.75	Fremont
2/14/2025 6:36		78 Fremont-Waterfront-Downt	XP	Bruce Lee	Elliott Avenue and Mercer St	Covered	Downtown	Extra-long	$ 2.75	Interbay
2/16/2025 6:32	Kenyetta Terron	42 Sodo-Downtown Express	Express	Meryl Streep	First Avenue and Terry Street	Covered	Fremont, Space Needle, P70	Doub	$ 2.75	SODO
2/19/2025 15:39	Kenny Terron	42 - E Sodo-Downtown Express	Espress	Bruce Lee	First Avenue and Terry Street	Uncovered	SoDo, DT	Doubled	$ 4.50	South Downtown
2/21/2025 6:13	Darcel Eustache	78-S Fremont-Waterfront-Dow	Special	Jimmy Hendricks, 206 5557661	Elliott Avenue and Mercer St	Cvd	Fremont, Space Needle, P70	Extra-long	$ 2.75	Interbay
2/21/2025 7:36	Kenyetta Terron	Sodo-Downtown Express	Express	Bruce Lee	First Avenue and Terry Street	UC	Downtown	Doubled	$ 4.50	South Downtown
2/22/2025 8:32	Ivey Hazekamp	32 Capitol Hill-Downtown	Regular	Jim Morrison	Broadway Ave and Cherry St	Covered	CCntr,	Articulated	$ 4.50	Capital Hill
3/1/2025 6:33			Regular	Bruce Lee, 425 6109225	Broadway Ave and Cherry St	Covered	Downtown	Articulated	$ 4.50	Capitol Hill
3/1/2025 6:36:00 AM (Blocked)			Regular	Mary Streep (mstre@mtransit.org)	Broadway Ave and Cherry St	Covered	Downtown	Articulated	$ 4.50	Capitol Hill
3/3/2025 6:42	Darcel Eustache	78 Fremont-Downtown Comm	Commute	Jim Morrison	Sixth Avenue and Battery	Regular	Fremont, Space Needle, P70	Doubled	$ 4.50	Downtown
3/5/2025 6:32			XP	Meryl Streep	Elliott Avenue and Mercer St	Covered	Downtown	Extra-long	$ 2.75	Interbay
3/9/2025 7:52	Kenyetta Terron	42 Sodo-Downtown Express	Express	Jimi Hendrix	First Avenue and Terry Street	UC	Downtown	Doubled	$ 4.50	South Downtown
3/10/2025 6:33	Darcel Eustache	78 Fremont-Downtown Comm	Commute	jim	Elliott Avenue and Mercer St	Covered	Downtown	Extra-long	$ 2.75	Interbay
3/11/2025 6:32	Kenyetta Terron	Sodo-Downtown Express	Express	Bruce Lee; blee@metrotran.org	First Avenue and Terry Street	Uncovered	Downtown	Doubled	$ 4.50	South Downtown
3/13/2025 8:32:00 AM (Late)	Ivey Hazekamp	32 Capitol Hill-Downtown	Regular	Bruce Lee	Broadway Ave and Cherry St	Covered	Convention Center	Articulated	$ 4.50	Capitol Hill
3/14/2025 6:32			Express	Bruce Lee	Broadway Ave and Cherry St	Covered	Downtown	Articulated	$ 4.50	C Hill
3/15/2025 1:36	Janey Lundgren	32 Capitol Hill-Downtown	Reg	Jimmy Hendricks	Broadway Ave and Cherry St	Covered	Convention Center	2 Decks	$ 2.75	Capitol Hill
3/16/2025 6:32	Darcel Eustache	78 Fremont-Waterfront-Downt	Commute	Jim Morrison	Sixth Ave and Battery Street	Regular	Fremont, Space Needle	Doubled	$ 4.50	Downtown
3/17/2025 11:33	Ivey Hazekamp	32 Capitol Hill-Downtown	Regular	Jimi Hendrix	Fourth Avenue and Seneca	Covered	Capitol Hill	Doubled	$ 2.75	Downtown
3/18/2025 16:13	Jane Lundgran	32 Capitol Hill-Downtown	Reg	Meryl Streep	Fourth Avenue and Seneca St	Covered	CH	2 Decks	$ 2.75	DT
3/19/2025 6:32	Janey Lundgren	32 Capitol Hill-Downtown	Reg	Meryl Streep	Broadway and Cherry St	Covered	Downtown	2 Decks	$ 2.75	Capitol Hill
3/21/2025 6:32	Darcel Eustache	78 Fremont-Waterfront-Downt	Commute	Meryl Streep	Sixth Avenue and Battery	Regular	Fremont, Space Needle, P70	Doubled	$ 4.50	Downtown
3/26/2025 9:04	Janie Lundgren	32 Capitol Hill-Downtown	Regular	Jimmy Hendricks, 206 5557661	Broadway and Cherry St	Covered	Downtown	2 Decks	$ 2.75	CHill
4/18/2025 5:23	Darcie Eustache	78 Fremont-Waterfront-Downt	Commute	Jimi Hendrix	Fourth Avenue and Seneca	Regular	Fremont, Pier 70, Needle	Doubled	$ 4.50	Downtown

Figure 3.3: Example of METRO_TRANSIT data held in a spreadsheet

While this data is fabricated and fictitious, the design it represents is an amalgamation of several very common design errors we encounter frequently. Please review the data in Figure 3.3 and determine which flaws are present. How and why are these data issues not controlled? As we embark on a trek to build an original database design from scratch, we may return to Figure 3.3 as a comparison of what it was like as a raw and untrained view. We will see two approaches to designing a new relational database, both of which start with gaining a full understanding of stakeholder requirements.

Method 1: Focus on each Object with a Vertical Design Approach

The first method is a more traditional academic process that has a focus on identifying objects involved in the business as well as the core characteristics and attributes of each object that is critical for the business. After these objects and their attributes are listed, then we will evaluate their relationships and patch together a puzzle where all are connected in a finished design. When this method is introduced, students are instructed to visualize this 'vertical' approach with emphasis on an exhaustive exploration of each database object. After each object has been fully explored, we move on to other objects before eventually putting the puzzle pieces together.

Method 2: Focus on the Puzzle with a Horizontal Design Approach

The alternative method has a simple standard set of characteristics (ID, Name, and Description) that will almost by default become the beginning columns. These are assigned to each business 'noun' that are identified during brainstorming efforts. The initial focus of this method is getting each object placed in their own entity and immediately defining relationships with other objects. Students are often instructed to visualize a 'horizontal' approach with an initial shallow definition of each database object and instead focus on how the puzzle of objects fit together. After the relationships are defined with a frame of the puzzle clear, designers then circle back and build out the characteristics of each object.

This second alternative method of building an original relational database model will be explored in greater depth in Chapter 4 *More Normalization*.

Stakeholder Requirements

Any build process, whether it is a database, bunny hutch, or sandwich, should begin with an understanding of purpose that details the user needs. For a database, it is critical to answer the following questions BEFORE brainstorming, diagramming or writing a lick of code:

1. Who is going to use the database?

2. What decisions are the database users going to be making?

3. What data elements need to be stored in the database?

For the basis of METRO_TRANSIT, we can make very broad assumptions from a simple view learned from using transit systems in North America, Europe, and Asia. The points of view and interests will include both management operations and consumers that we would have elicited from our research efforts during the early phases of the SDLC. Each of these needs listed below should be validated with the project sponsor and primary stakeholders.

- MANAGEMENT/OPERATIONS
    - Which routes are the busiest?
    - Which destinations are most popular?
    - Which routes have the most incidents?
    - Which neighborhoods need coverage?
    - Which drivers are most reliable?
    - When are riders taking trips?
    - Is capacity sufficient?

- CONSUMER
    - Which routes are best suited for my need to get to a specific destination?
    - What times are best to travel?
    - What concerns should I have regarding safety?
    - What are the costs for traveling to my destination?

Once the data needs are validated, we will engage in brainstorming WITH our primary stakeholders present and actively participating. Stakeholders are critical to creating a well-designed database by helping clear up any ambiguities we might encounter as well as identify terms and objects considered out of scope.

BRAINSTORMING

The act of brainstorming occurs during the Conceptual Design Phase with the intention of determining an exhaustive list of business objects required to meet the objectives listed in the data needs provided by stakeholders. These are frequently ‘nouns’ as opposed to verbs. This meeting to define objects for the database includes members from across the company, including members of the technology side (web developers, a project manager, infrastructure engineers as well as database designers) in addition to people from the

business side of the company. Collectively we have a broad understanding to address the application problem space.

The meeting starts with one person holding a pen writing ALL suggestions on a whiteboard. Quick note, there are no 'bad' suggestions at this point. We write everything down (including duplicate suggestions!) to get a full picture of the range of thoughts. We will filter and eliminate some objects during a specific clean-up process, but for now the goal is to get a feel for what everyone's thoughts are regarding the problem space.

VEHICLE	DRIVER	CUSTOMER
DESTINATION	OIL CHANGE	TRAFFIC
FARE	ROUTE	CANCELED
STREET	STOP	LATE
TIRE ROTATION	SEAT	INCIDENT
FIGHT	SECURITY CAMERA	GARBAGE SPILL
TRANSFER	ELDERLY	DISABLED
SHUTTLE	EXPRESS	CAPACITY
WI FI	AIR CONDITIONING	WHEELCHAIR LIFT
DELAYED	NEIGHBORHOOD	PASSENGER
YOUTH	TRAIN	BUS
SECURITY OFFICER	INCLEMENT WEATHER	ENGINEER

Table 3.1: Example of initial brainstorming session for METRO_TRANSIT

In Table 3.1 above, there are a range of objects related to the management and use of a common metropolitan transportation system. While this list is relatively simple and was constructed in about 20 minutes, it is a fair representation of what might be produced in a single session. In a more realistic situation at a company, the list may be closer to 100 objects created over 90 minutes. This is good enough for our purposes as we see the evolution from this raw list to a finished database design.

The next step of our data model evolution is to review the brainstorming output and do a bit of editing and clean-up. Our goal here is to validate strong objects (those that everyone agrees is critical as a core entity) as well as perhaps find others that may survive as columns or rows as opposed to being an entity.

During this clean-up process, we do not want to throw away any values; while we may cross-off more than half of the suggestions, we can still see where our ideas started and end, perhaps returning later to resurrect a suggestion upon further review.

We review the list of suggestions and highlight the 'keepers' by making them bold red. Also, we strike through the suggestions that are not strong candidates as entities:

VEHICLE	~~DRIVER~~	~~CUSTOMER~~
DESTINATION	~~OIL CHANGE~~	~~TRAFFIC~~
~~FARE~~	**ROUTE**	~~CANCELED~~
~~STREET~~	**STOP**	~~LATE~~
~~TIRE ROTATION~~	~~SEAT~~	**INCIDENT**
~~FIGHT~~	~~SECURITY CAMERA~~	~~GARBAGE SPILL~~
~~TRANSFER~~	~~ELDERLY~~	~~DISABLED~~
~~SHUTTLE~~	~~EXPRESS~~	~~CAPACITY~~
~~WI FI~~	~~AIR CONDITIONING~~	~~WHEELCHAIR LIFT~~
~~DELAYED~~	**NEIGHBORHOOD**	**PASSENGER**
~~YOUTH~~	~~TRAIN~~	~~BUS~~
~~SECURITY OFFICER~~	~~INCLEMENT WEATHER~~	~~ENGINEER~~

Table 3.2: Clean-up of initial brainstorming

Our initial clean-up has yielded seven candidate entities, including VEHICLE, DESTINATION, ROUTE, STOP, INCIDENT, NEIGHBORHOOD, and PASSENGER. Next, let's take a closer look at the reasons other suggestions got crossed out.

Common Parent

Nearly every database brainstorming session ends up with several very similar suggestions that can be consolidated as rows underneath a more abstract title. For example, consider the suggestions AIR CONDITIONING, WIFI, WHEELCHAIR LIFT and SECURITY CAMERA; these are possibly better labeled as instances or values of a different noun not mentioned is called AMENITY. There may even be a category above AMENITY with another entity named AMENITY_TYPE, which we can explore later.

Consider other combinations of suggestions that might have a common parent:

- STATUS
    - Late
    - Delayed
    - Canceled

- EMPLOYEE_TYPE
    - Driver
    - Security Officer
    - Engineer

- VEHICLE_TYPE
    - Heavy Rail Train
    - Light Rail Train
    - Shuttle
    - Double-Decker Bus
    - Regular/Standard Bus

- ROUTE_TYPE
    - Express
    - Inclement Weather

- MAINTENANCE
    - Oil Change
    - Tire Rotation

- PASSENGER_TYPE
    - Elderly
    - Youth
    - Disabled

- INCIDENT_TYPE
    - Fight
    - Garbage Spill

Now we have established several new entities above that were not present on our brainstorming, including STATUS, EMPLOYEE_TYPE, VEHICLE_TYPE, ROUTE_TYPE, MAINTENANCE, PASSENGER_TYPE, INCIDENT_TYPE, and AMENITY_TYPE.
The next step is to explore these new entities for any values beyond the ones we have already. The objects that have more values are going to be better candidates for entities.

We want to quickly continue brainstorming and explore these entities for any additional data values that may exist (again, with brainstorming, there are no wrong answers)::

- INCIDENT_TYPE
    - Fight, Garbage Spill already included
    - Medical?
    - Collision
    - Fare Evasion
    - Drug Use
    - Verbal Abuse

- MAINTENANCE_TYPE
    - Oil Change, Tire Rotation already included
    - Vacuum
    - Steam Clean
    - Garbage Collection

- AMENITY_TYPE
    - Security
    - Safety
    - Comfort
    - Legal Mobility Accommodation

Additional entities might be found by exploring potential 'TYPE' values for the remaining base entities we thought were very strong (the ones that were highlighted as bold and red).

Consider the following brainstorming around various 'types' of values:

- DESTINATION_TYPE
    - Tourist
    - Employment
    - Shopping

    - Educational
    - Social HUB
    - Arts
- FARE_TYPE
    - Unlimited Monthly/Daily Pass
    - Youth
    - Anonymous
    - Disabled/Elderly

Not bad! We have taken an initial 20-minute brainstorming session that started with 7 entities, and after a few tweaks, we have come up with another 15. We are still in the conceptual design phase and are not stuck with keeping any of these objects. We will now take each of our candidate entities and explore the attributes of each. These attributes will eventually be called 'columns' when the code for the database is built.

The following list represents a vetted beginning point to constructing a data model. There are two methods that are popular among database designers, including a formal method with symbols that includes an exhaustive definition of each object in isolation, followed by defining relationships before consolidating everything in a single diagram. This method will be demonstrated first, followed by a second step-by-step example of the 'ironing' method. Hopefully, one of the two methods will be more comfortable for the way your brain works!

Method 1: Focus on each Object with a Vertical Design Approach
Again, method 1 has a more traditional, academic approach with many benefits; we gain a significant depth of understanding of the conditions of business objects that generate data and how that data will need to be used. As there will most likely be formal documentation required for a larger audience, this method will take longer. This is probably the preferred method when the business processes are being invented or designed at the same time as the database, like when our organization is new to an industry or market.

Common Data Modeling Symbols: Boxes, Orbs, and Diamonds

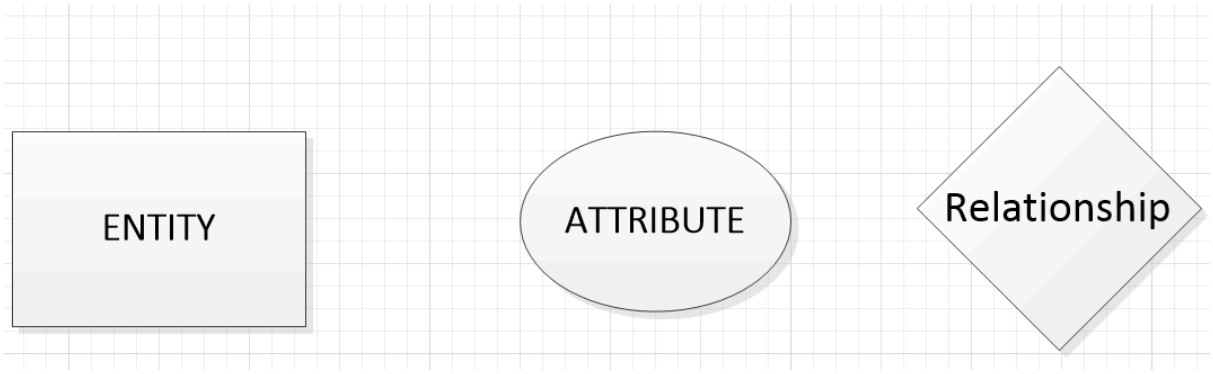


Figure 3.4: Common data modeling symbols

The next steps to defining the relational database METRO_TRANSIT following method 1 involve using the standard set of symbols to begin visually representing the entities (soon to be called 'table') and the various attributes of each (again, soon to be called 'columns'). This process involves more brainstorming as we again engage our stakeholders, review the application requirements, and possibly conduct additional research.

We start out this process by evaluating the critical characteristics for each potential entity by itself, without consideration to any relationship with other entities.

Starting with the most obvious objects of EMPLOYEE, VEHICLE, and PASSENGER, explore the characteristics (formally named 'attributes'). These are the actual elements of data that will be critical for users of the database to leverage when writing queries or viewing a dashboard. This should still be considered brainstorming as we can always add or drop elements as we progress through the design process. Please observe Figure 3.5 below:

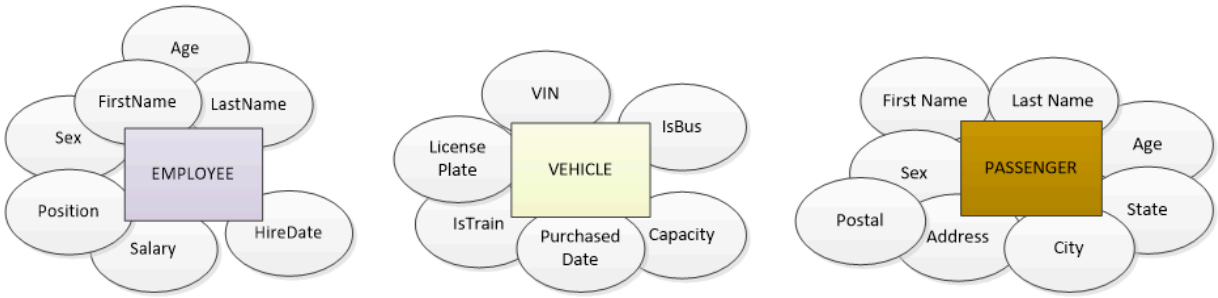


Figure 3.5: Brainstorming attributes for EMPLOYEE, VEHICLE, and PASSENGER

Next, the brainstorming of attributes continues by vetting the details for objects INCIDENT, DESTINATION, and NEIGHBORHOOD as seen in Figure 3.6:

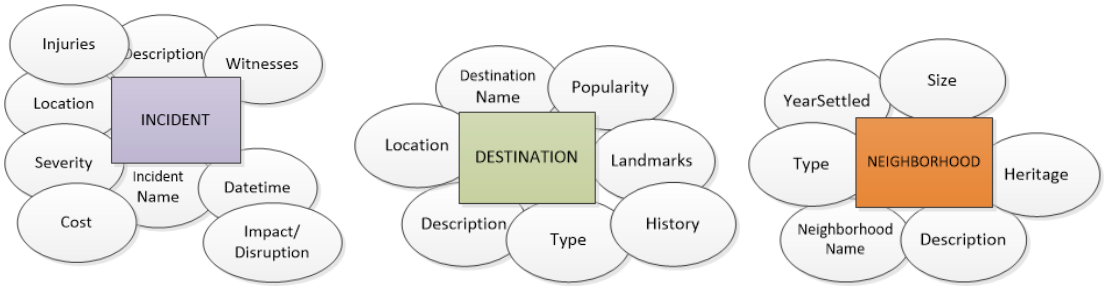
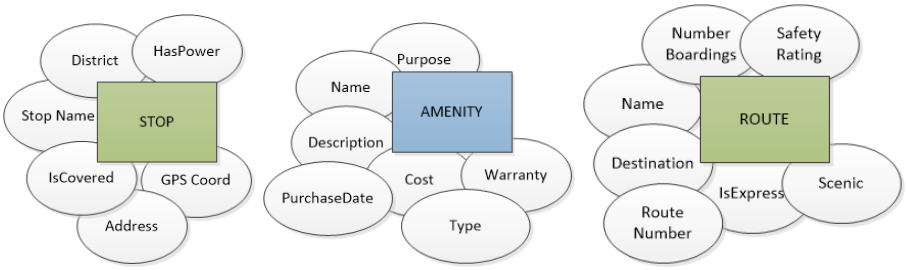


Figure 3.6: Brainstorming attributes of INCIDENT, DESTINATION and NEIGHBORHOOD

Further brainstorming includes STOP, AMENITY, and ROUTE in Figure 3.7:



3.7: Brainstorming attributes for STOP, AMENITY and ROUTE

The process of coming up with the various attributes for each entity should be based on all knowledge and research conducted from before the project was approved for funding. Our goal is to identify every descriptor, element, or characteristic of the entity object that will be relevant to the organization when presenting reports, measuring performance or otherwise learning about how well the company is doing. Again, it is encouraged to be exhaustive as we can always trim attributes later.

Defining Relationships

Next, we can begin toward articulating the relationships these have with each other entity. Recognize that while exploring relationships between entities, we may discover attributes (or even new entities) that were missed previously.

Defining relationships between entities might be more difficult than coming up with their attributes as it requires having a deeper understanding of the business. Many times,

database designers are dependent on other employees within the company with a stronger awareness of the operations of the organization. This means interviewing people from different departments like marketing, accounting, and human resources among others.

Defining relationships between entities can be described as putting a puzzle together, not only because there are dozens of pieces that need to be placed in a physical position, but we also need to decrypt the relationship in two directions. We will probably be a bit confused about these for the first couple of tries, however, most students get a good handle on defining relationships after a few examples. A quick reminder of the object names that will be evaluated for initial relationships:

EMPLOYEE	NEIGHBORHOOD	INCIDENT	STATUS
VEHICLE	AMENITY	DESTINATION	FARE
PASSENGER	MAINTENANCE	STOP	ROUTE

Table 3.3: List of initial base entity objects to be evaluated for direct relationships

AMENITY_TYPE	FARE_TYPE
INCIDENT_TYPE	EMPLOYEE_TYPE
STOP_TYPE	PASSENGER_TYPE
MAINTENANCE_TYPE	ROUTE_TYPE
DESTINATION_TYPE	VEHICLE_TYPE

Table 3.4: List of initial type entities to be evaluated for direct relationships

The step of deciding which entities have a direct relationship with others is not easy; it truly is like putting a puzzle together. A *direct* relationship can be assigned if there is clear demand for data each time the two objects interact while *indirect* if the answer is 'maybe'. If there is no clear or obvious data connection between the entities, then we should label the relationship as *incidental*.

Base Entity Name	Relationship With EMPLOYEE
NEIGHBORHOOD	This relationship is defined as ***Incidental*** unless the company needs to track where an employee lives.
INCIDENT	At this time, we will say there is a ***Direct*** relationship between INCIDENT and EMPLOYEE. There most likely will be a need to track which employee was involved in certain incidents. We will see later that other entities will be discovered during the normalization process to remove the need for this direct relationship to exist.
STATUS	This relationship is defined as ***Incidental*** unless the company needs to track the employment status (such as good standing, suspended, on leave, or perhaps retired).
VEHICLE	At this time, we will say there is a ***Direct*** relationship between VEHICLE and EMPLOYEE as there will need to be a record of which person was the driver or maintenance person who provided service. Normalization may change our definition of this relationship.
AMENITY	This relationship is defined as ***Incidental*** unless the company needs to track which amenities a particular employee needs as accommodation (unlikely).
DESTINATION	This relationship is defined as ***Incidental*** as no clear or relationship exists between DESTINATION and EMPLOYEE.
FARE	This relationship is defined as ***Incidental*** as no clear or relationship exists between FARE and EMPLOYEE.
PASSENGER	At this time, we will say there is an ***Indirect*** relationship between PASSENGER and EMPLOYEE as there is most likely a need to know when drivers and passengers were on the same vehicle. Normalization will change this relationship!
MAINTENANCE	At this time, we will say there is a ***Direct*** relationship between MAINTENANCE and EMPLOYEE. Normalization may change this relationship if we determine this later as M:M.

STOP	This relationship is defined as ***Indirect***. There is probably critical data required to know when the EMPLOYEE_TYPE 'driver' was at a STOP, it can be seen as M:M. Normalization will resolve this relationship.
ROUTE	This relationship is defined as ***Indirect***. Again, there is probably data required for each time a driver is assigned a ROUTE. We will learn that this is a M:M relationship to be resolved during normalization.

Multiplicity

In addition to having words in the middle of the diamond describing the relationship between two entities, we have numbers that identify the range of permissible occurrences. This is known as multiplicity with the following choices:

0..* Read 'zero-to-many', this is a very common relationship definition

1..1 Read 'one-to-one', this is very common with enhanced data modeling

1..* Read 'one-to-many'

.. Read 'many-to-many' this appears in early conceptual diagrams before we have been able to resolve the relationship into two 1 : * relationships.

Let's see what a first draft of including relationships looks like by evaluating many combinations of entities. Please recognize that not every entity will have a direct relationship with every other entity. Our job is to check out each combination of identity-relationships (perhaps initially just in our head) to determine the handful that have direct relationships. It does not really matter which entities are evaluated first from the working list of entities, but we can start at the top with EMPLOYEE.

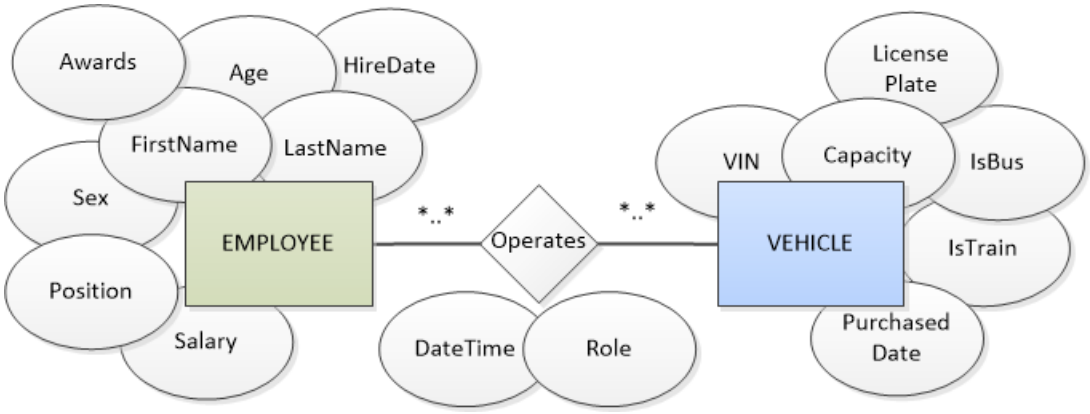


Figure 3.8: Relationship EMPLOYEE : VEHICLE

The relationship in Figure 3.8 is read many-to-many in both directions as one employee over time can operate many vehicles. One vehicle, again over time, can be operated by many employees. As relational theory does not allow M:M relationships to be coded, the relationship of EMPLOYEE : VEHICLE must be resolved to multiple 1:M relationships.

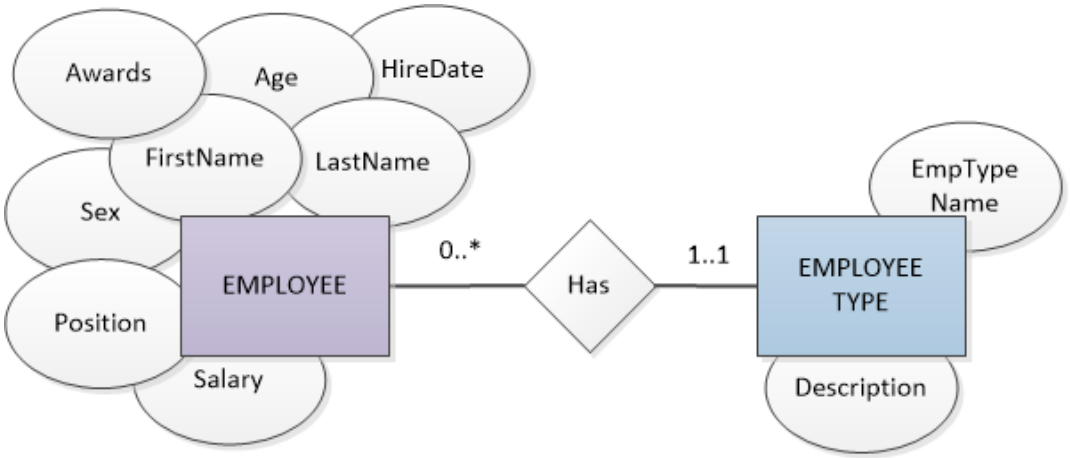


Figure 3.9: Relationship EMPLOYEE : EMPLOYEE_TYPE

The relationship shown in Figure 3.9 is an example of a 'type' entity and the base table, which is perhaps the most common we encounter as database designers. There may be as many as 50% of all relationships of this kind in an average relational database. The relationship in Figure 3.9 is read 'one employee can have one and only one employee type and an employee type is defined as between zero and many employees. The multiplicity is included in the diagram while the cardinality is spoken as 'one to many' and written as 1:M.

There most likely is a little bit of anxiety around the difference between multiplicity and cardinality. Multiplicity defines the range of possible occurrences while cardinality

describes the relationship that a single row in one entity has to the second entity. We will get good at this by the end of this chapter!

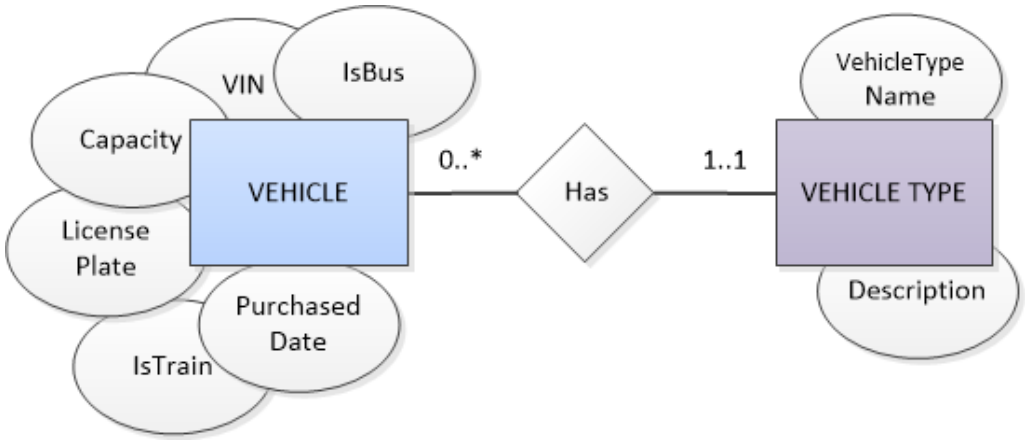


Figure 3.10: Relationship VEHICLE : VEHICLE_TYPE

The relationship in Figure 3.10 between VEHICLE and VEHICLE_TYPE is a classic 1:M relationship (again, as are most base tables to a 'type'). The way to configure the relationship is for the base table (which in Figure 3.10 is VEHICLE) to receive the primary key from the type of entity as a foreign key. This means the primary key in VEHICLE_TYPE (VehicleTypeID) will become the foreign key of the same name in VEHICLE.

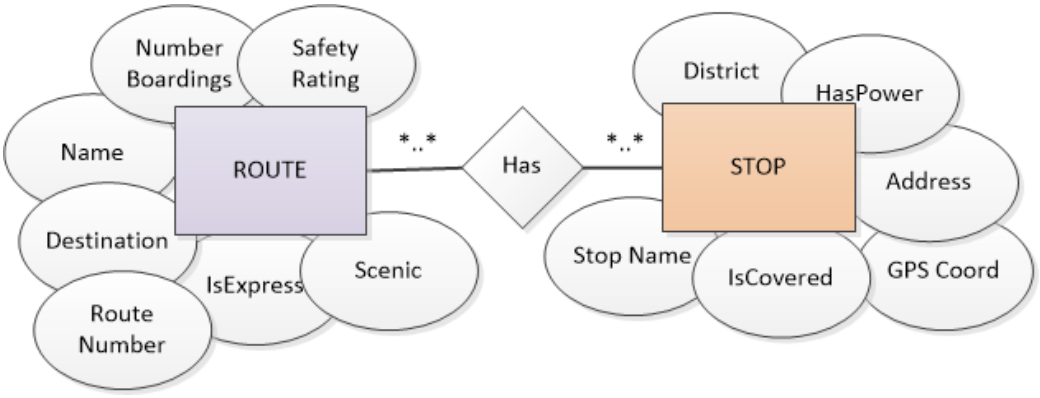


Figure 3.11: Relationship ROUTE : STOP

The relationship above between ROUTE and STOP is M:M and must be resolved with an associative entity before coding. Most students need to talk this one through. It is read as

'one route has between zero and many stops while the same stop can have between zero and many routes that use it'. The associative entity can be temporarily called ROUTE_STOP before eventually being re-titled something more consistent with the popular corporate vernacular, like TIME_SCHEDULE or just SCHEDULE.

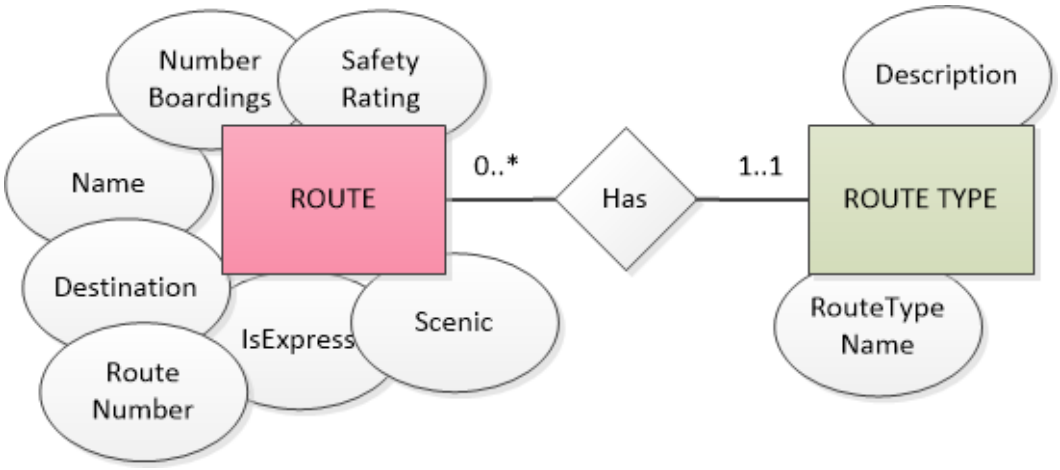


Figure 3.12: Relationship ROUTE : ROUTE_TYPE

This relationship and the next 8 examples are all 1:M and very typical of how a base table relates to its 'type'. Recognize this pattern as it is very common!

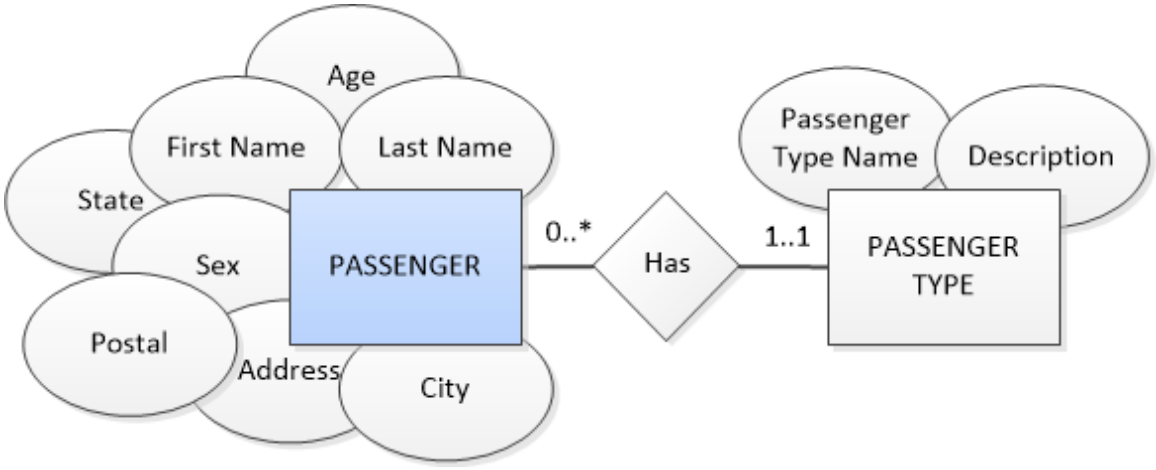


Figure 3.13: Relationship PASSENGER : PASSENGER_TYPE

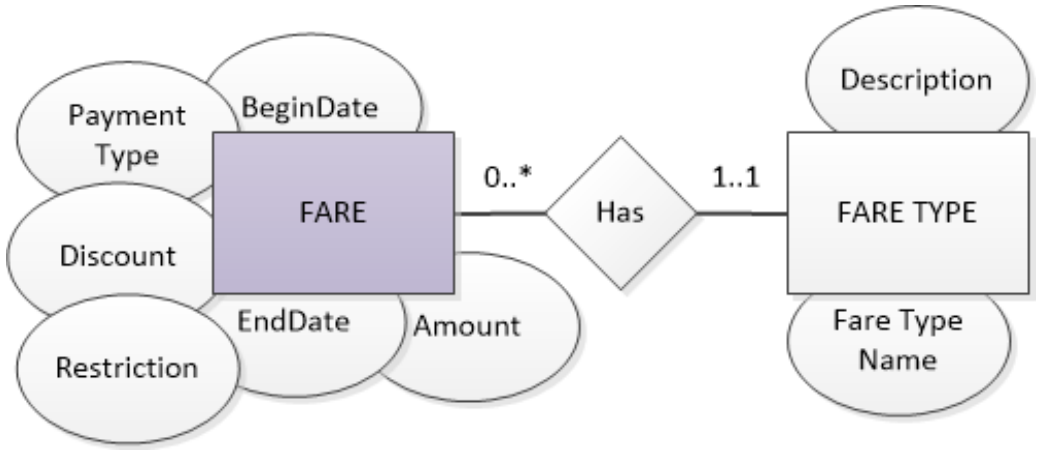


Figure 3.14: Relationship FARE : FARE_TYPE

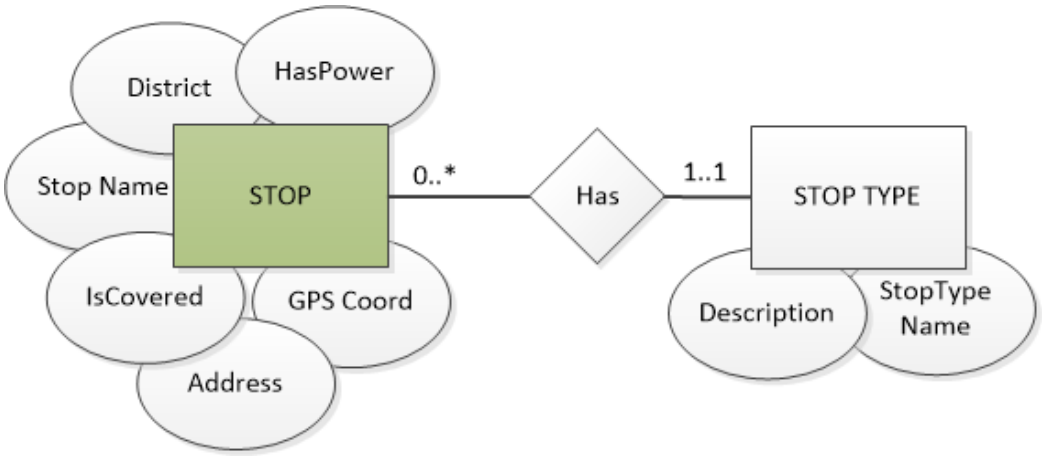


Figure 3.15: Relationship STOP : STOP_TYPE

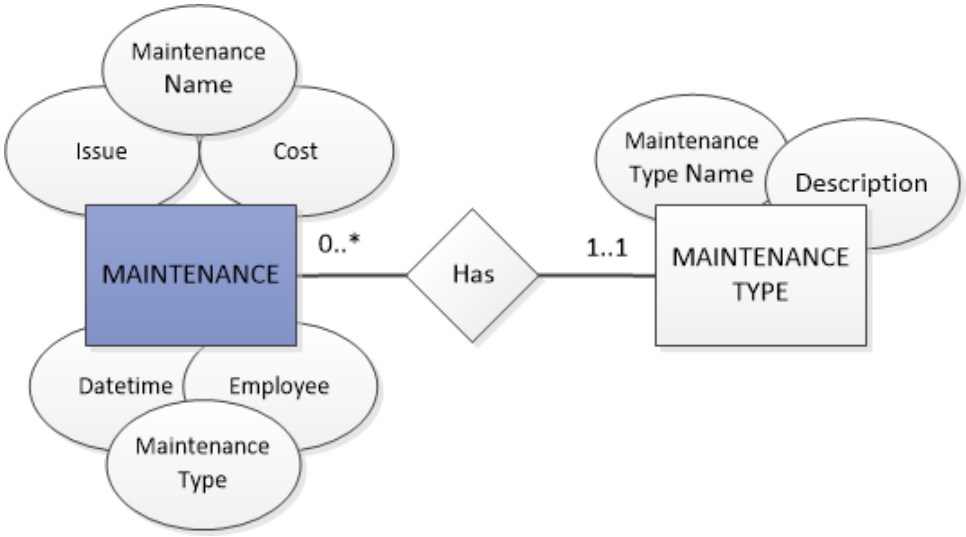


Figure 3.16: Relationship MAINTENANCE : MAINTENANCE_TYPE

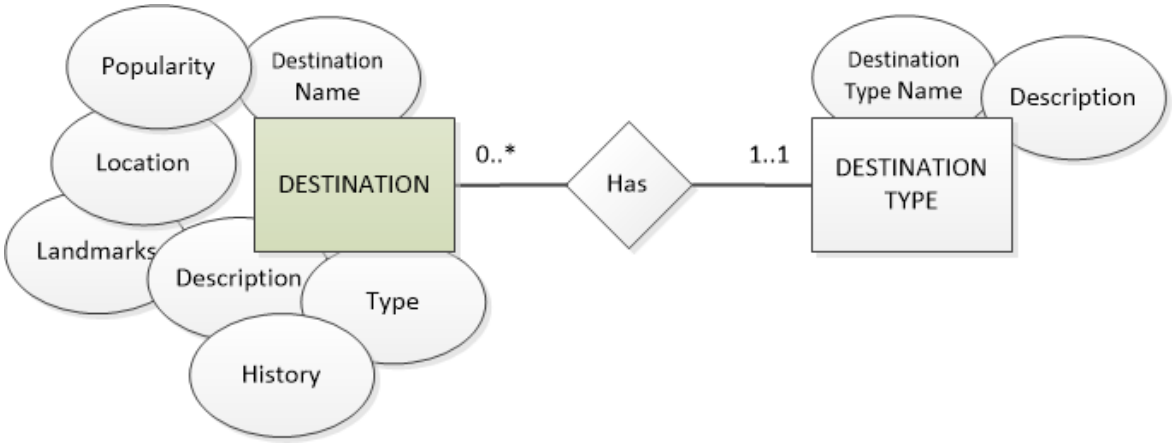


Figure 3.17: Relationship DESTINATION : DESTINATION_TYPE

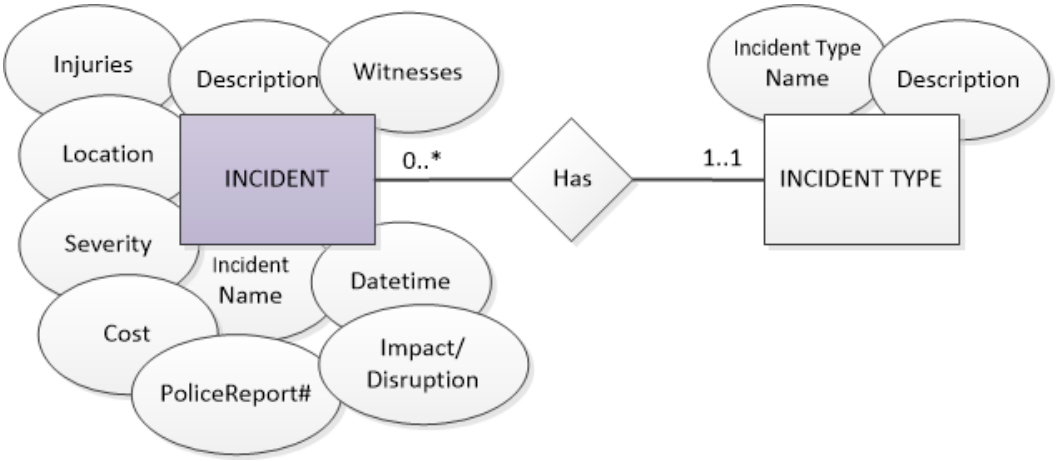


Figure 3.18: Relationship INCIDENT : INCIDENT_TYPE

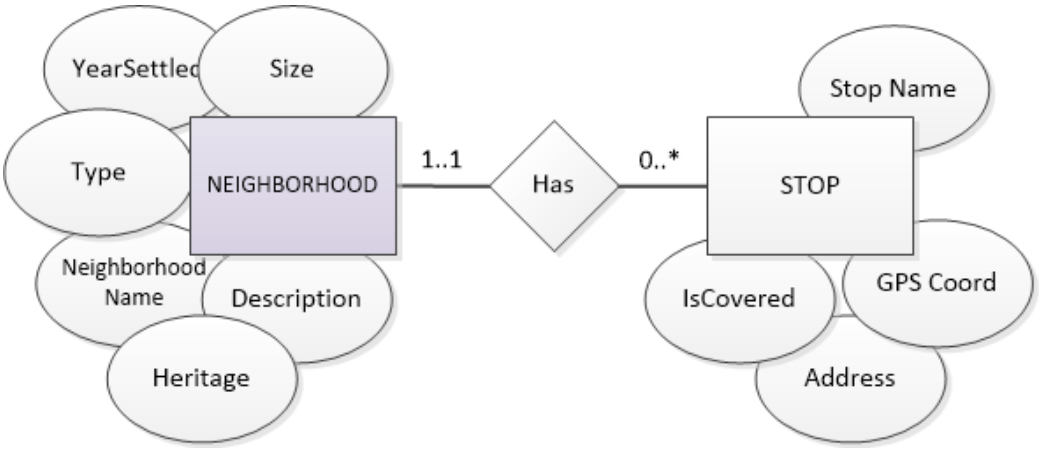


Figure 3.19: Relationship NEIGHBORHOOD : STOP

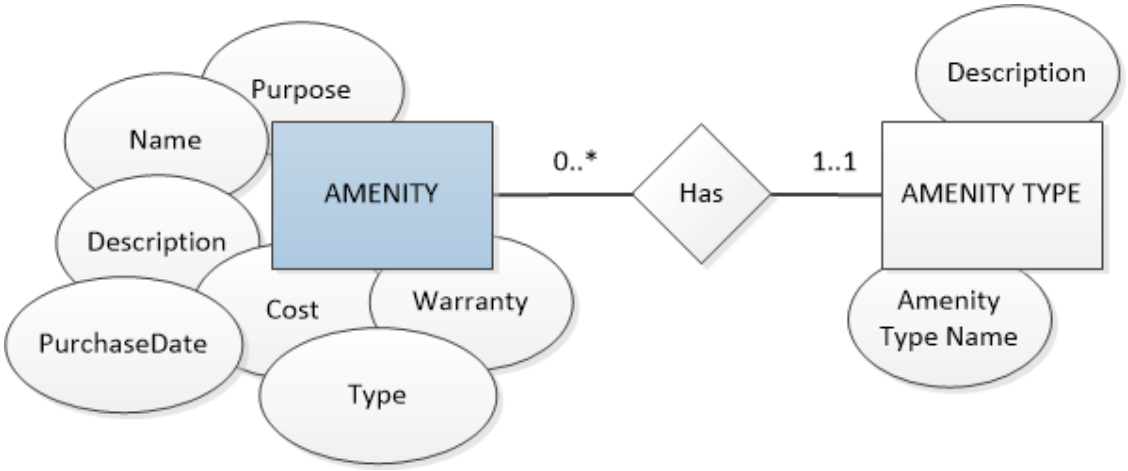


Figure 3.20: Relationship AMENITY : AMENITY_TYPE

The previous 8 relationships all had a cardinality of 1:M and similar multiplicity. We want to review these somewhat slowly to recognize the pattern. Be able to quickly see the relationship and know immediately how to construct the tables involved including assigning the foreign key correctly. We will see this in greater detail coming up because it is important to understand thoroughly.

Consolidate All relationships into one large Conceptual Diagram

When we consolidate relationships into a single, large diagram, it is important to review each entity and relationship to reaffirm each is contributing to the purpose of the application and database. It is never too late to drop or add entities based on a shift in priorities or market conditions. Let's see an example of dropping an entity or two.

In the METRO_TRANSIT database example, it makes sense to DROP the entity MAINTENANCE as well as the M:M resolution entity VEHICLE_MAINTENANCE in the consolidated design to illustrate this point. In real life, there are many reasons to drop a previously selected and validated entity; redundant systems is perhaps the most common reason in my experience. Let's imagine that VEHICLE_MAINTENANCE is already being tracked in a database specifically tied to manufacturer's warranties and union payroll systems. For the sake of simplicity, it has been removed from the diagram.

As we will see shortly, there are several M:M relationships that must be resolved with an associative entity. The M:M relationships are indicated in the consolidated conceptual diagram below with a YELLOW highlight as well as a designation of 'MANY' in the relationship diamond. These include the following relationships:

- ROUTE : STOP
- ROUTE : STATUS
- VEHICLE : ROUTE
- PASSENGER : VEHICLE : ROUTE : FARE

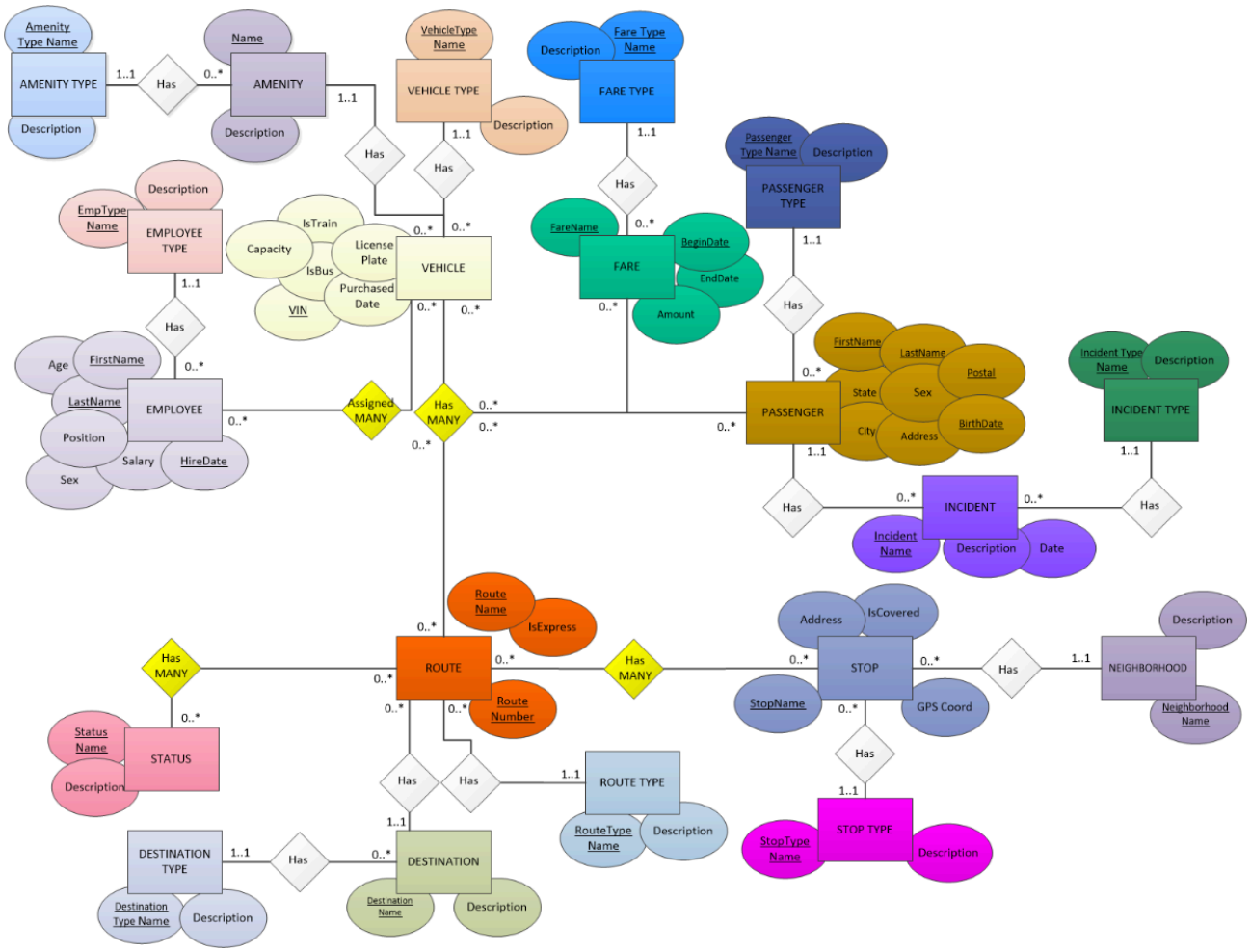


Figure 3.21: Consolidated Conceptual Diagram with four M:M relationships

The next steps are to work closer with the M:M relationships we have initially identified. There needs to be redefinition and diagramming of these relationships (as well as a possible re-naming) because relational theory does not allow the coding of a M:M relationship inside of a finished database.

In each case of a M:M relationship, we will create a new entity in between the entities participating in the relationship. See below for TRIP, BOARDING, TRIP_STATUS and ROUTE_STOP as new associative entities:

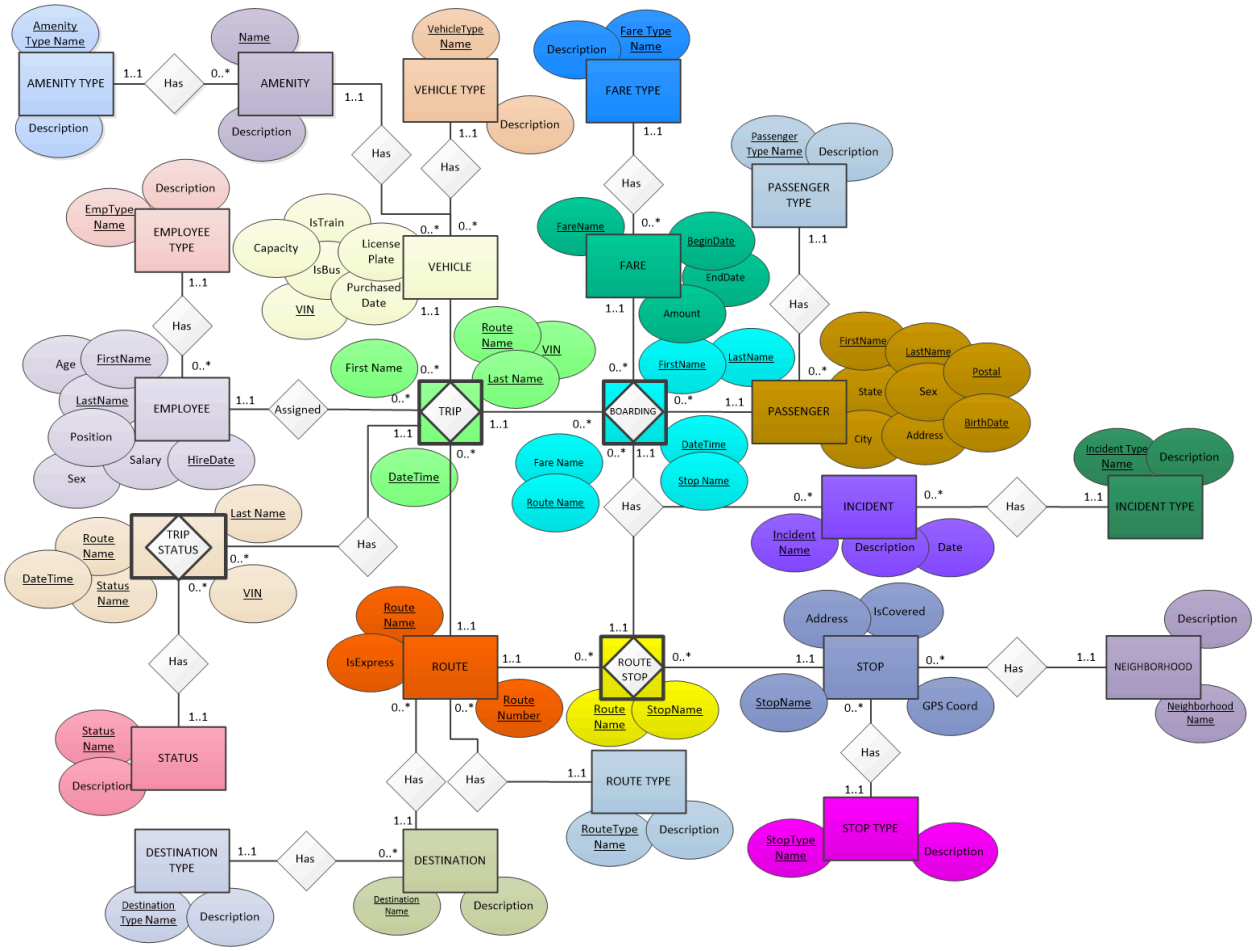


Figure 3.22: Consolidated Conceptual Diagram with four associative entities

The image in Figure 3.22 is a great step towards building a relational database in Third Normal Form (3NF). There have been four M:M relationships resolved with associative entities. These are given new names and placed in an entity with a 'diamond'.

- TRIP
- BOARDING
- TRIP_STATUS
- ROUTE_STOP

TRIP seems like a decent name to record the occurrence of an EMPLOYEE, assigned to drive a VEHICLE on a specific ROUTE on a specific date and time.

Additionally, BOARDING seems like an accurate descriptive name for the resolution of the M:M relationships between PASSENGER, FARE, STOP and then ROUTE and VEHICLE (which has been renamed to TRIP). Hopefully we can see how this consolidation was made. Both TRIP_STATUS and ROUTE_STOP have been created to resolve two simple M:M and are named literally.

While Figure 3.22 represents a significant amount of work, we must acknowledge there are several fundamental flaws in its current design. These flaws include some rather awkward primary keys for most entities in addition to having several yes/no questions as columns.

Awkward Primary Keys

Many database designs have a collection of columns that are combined to create a primary key; this is known as a 'natural' key. As mentioned previously, this is fine for many database designs but creates some awkward situations in managing foreign keys (which we have learned are the primary keys of one entity being placed in another entity to cement the relationship). Addressing this flaw, let's first assign a surrogate primary key for each entity.

With surrogate keys, designers are better able to define relationships with other entities and bring over the PK as an FK as warranted. For a quick example of a clogged mess for a primary key, look at TRIP and BOARDING entities; these have four and five attributes that are required to be assembled just to find uniqueness.

Yes/No Questions as Column Names

Second, we want to take those entities with 'yes/no questions' as column headers and replace them with a TYPE table. When we remove the 'yes/no' columns, we are reducing the burden on the base table to account for every possible 'type' value. In some cases, there could be a dozen potential 'yes/no' options to choose from, which means 12 columns are required! Additionally, the query writing burden is significantly reduced as no longer does a query writer need to be aware of all permutations before writing any code.

EMPLOYEE had an attribute of Position in Figure 3.22. Upon further evaluation, a single EMPLOYEE can have many positions during their career at METRO_TRANSIT. Also, the same POSITION, such as 'Senior Coach Operator' may describe many employees, each with a different BeginDate, EndDate, and Salary.

Upon re-evaluation, this relationship is better defined as M:M and must be resolved with an associative entity. The new entity can be called EMPLOYEE_POSITION or perhaps JOB. For simplicity, it is probably easier to use the title of JOB.

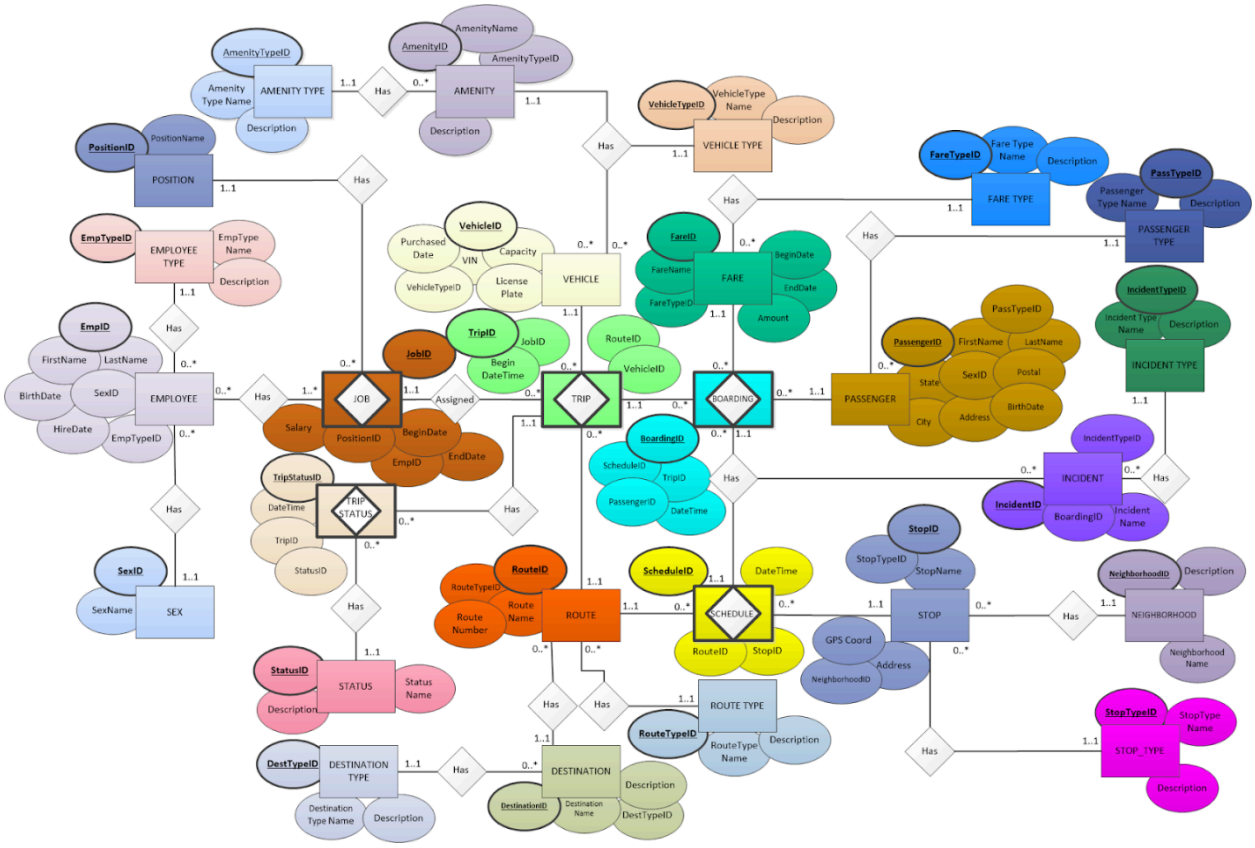


Figure 3.23: Consolidated Conceptual Diagram with PK/FK and five associative entities

Looking a bit closer at the design in Figure 3.23, please notice that each entity has a proper primary key (unique identifier). These are referred to as *surrogate* keys since they are not directly a part of the entity except to provide uniqueness. These are auto incremented in a sequential manner by the computer system, thus eliminating the need for people to get involved in determining any values (imagine trying to find 'the next value' when there are millions of new records each day).

While there are arguments in academic circles to find a combination of existing columns in a table to provide an enforceable uniqueness (and be used as a primary key), many experienced database designers advocate for using surrogate keys whenever possible.

Figure 3.23 also shows the evolution of the design to include foreign keys, as well as substituting out difficult columns, including those that are asking of 'yes/no' questions, and those with a fixed 'age' with a birthdate.

One more tricky update! ROUTE_STOP had a composite primary key of RouteID and StopID which has been changed to a surrogate key. Having a composite primary key is fine in most cases, however when BOARDING needed a foreign key value from ROUTE_STOP, it became awkward to send two columns (again, RouteID and StopID make up a single primary key) over as a foreign key. Better to establish a single surrogate primary key of the newly re-named entity of SCHEDULE and be better able to manage the surrogate key ScheduleID as a foreign key in BOARDING.

Final Steps: Re-evaluating relationships looking for M:M

Lastly, we want to review all relationships now that a proper PK has been assigned to each entity. Specifically, we are looking for entities that are a M:M over a period that may not have been recognized as such just yet. This is due to having re-defined four relationships already which may have unwittingly impacted other relationships.

Upon doing so, we will confer with our stakeholders to validate the interpretation we have and their urgency for capturing outliers or other low-probability occurrences. We end up with the following relationships that are in fact M:M:

- AMENITY : VEHICLE
- ROUTE : DESTINATION
- BOARDING : INCIDENT

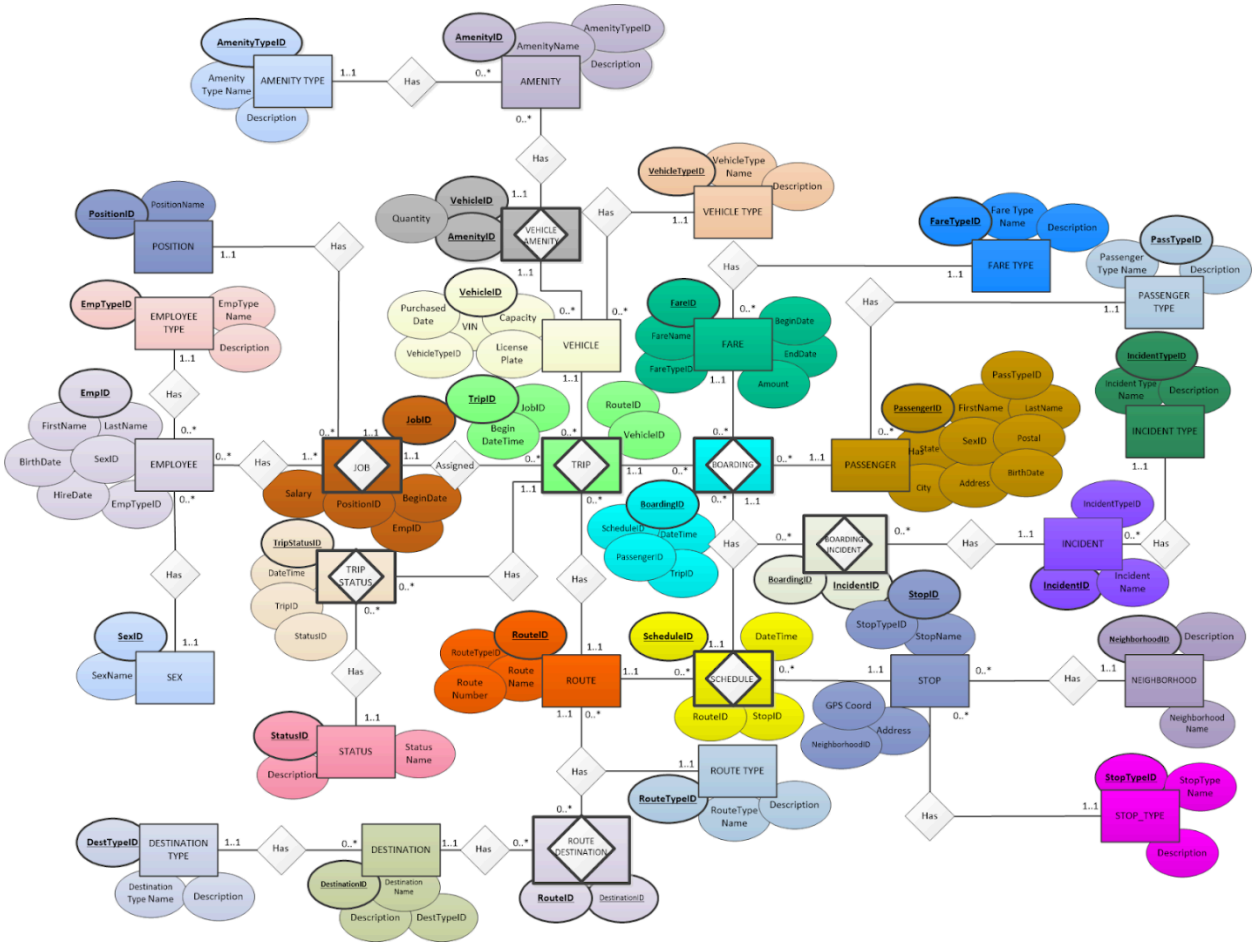


Figure 3.24: Near Final ERD with 3 additional M:M relationships resolved

This latest rendering might be as accurate as we will get. It represents 'one-way' to get a decent data model completed from scratch without having to worry about the details of the normalization process.

Before we meet one final review with the business folks and primary stakeholders, we may want to at least UNDERSTAND the different normalization forms and be prepared to defend many of our decisions. Additionally, if we have a better-than-nothing comprehension of the different forms, we may discover some of our design logic is flawed or perhaps missing other attributes or even entities.

Quick Review of how we got here

Summarizing our steps so far include the following:

1. Conducted research: this includes requirements of the company, industry best practices, sought clarity in the problem space, interviewed stakeholders, and observed production operations

2. Validated findings and interpretations with project sponsor

3. Conducted brainstorming with active participation from stakeholders

4. Cleaned-up brainstorming results in a filtered set of objects aligned as entities

5. Further explored characteristics of each entity

6. Defined relationships between the entities including cardinality and multiplicity

7. Assigned surrogate primary key values for each entity

8. Resolved four many-to-many (M:M) relationships into several one-to-many (1:M)

9. Reviewed priorities of data and determined that MAINTENANCE was outside the scope of the data needs and dropped the entity from the diagram

10. Re-evaluated relationships and determined there were a total of 8 M:M relationships that needed to be resolved instead of the just the four we thought

11. Created the consolidation diagram in Figure 3.24, which reflects the current state of the design using an exhaustive entity-by-entity approach

Database design can be complex and tedious, often beyond the interest and learning objectives of most people. Hang in there! One of the goals of this textbook is to prepare every reader with the skills to quickly construct a well-formed relational database.

The final touches of understanding data modeling are having basic comprehension of normalization. Normalization can be very complex and confusing; it seems very few people (including this author) can explain the detailed engineering or academic math behind how and why certain decisions are made during the normalization process. What can be explained is a simpler approach that will enable readers to recognize better design choices without having to explain the math.

Post-Chapter Challenges
Based on your objectives and intentions on learnings from this book, please approach the following challenges as appropriate.

Track 1 (THINK): Data Tourist Seeking Ancillary Awareness
Please spend a total of 10 minutes reviewing the following questions, exploring your thoughts. These questions and your responses cut to the essence of this chapter:

- How well do you grasp data modeling concepts and processes now that you have seen a step-by-step review of a database build?

- Which of the two methods of data modeling explained in this chapter aligns better with your way of thinking and brainstorming? Why do you think one method aligns more easily? Reminder: the first method is the academic 'vertical' approach and the second method is a 'horizontal' approach which has standard column names (ID, Name, and Description) to start.

- What other topics or business scenarios might you find interesting in creating a data model?

- How has exposure to the data layer design process (even if just briefly) changed your interest and desires to add more 'full stack' development and design skills?

Track 2 (WRITE): Dedicated Student or Recent Graduate
Based on your completion of this chapter, WRITE several paragraphs in response to each question; explore these as if you are being asked a similar question during a job interview!

- Reflect on the presence of misspellings in the spreadsheet of data in Figure 3.3; what are the challenges presented in allowing typos and abbreviations in any collection of data? How might it be best to eliminate or manage the typos and/or abbreviations out of any data set?

- Write out in a brief brainstorming session the basic core nouns for one of the following business topics:

    - A retail store that sells second-hand or used clothing

    - A 5-person operation that offers private and guided tours of a major city including historic landmarks, museums, and major scenic outlooks

    - A house -repair contractor who has a seasonal team of 6 workers

Track 3 (BUILD): Full Speed Learner Seeking Job
This track targets readers of this book who want to develop professional skills working with data to launch a career or obtain a more satisfying job. Let's begin with a challenge to structure data collection to keep track of a large-scale metropolitan transit system:

- Copy the column headers (as well as some of the data from figure 1.3) into an Excel spreadsheet that allows for greater tracking of a metropolitan transit system.

- Assume the users of your data set want to learn about route safety; begin tracking data on incidents of injuries, assaults, or aggressive panhandling. What data needs to be captured? Why? Don't worry about proper data modeling---yet. Just get your ideas down in a manner that makes sense to you.

Conclusion

Data modeling is a refined skill that is critical to building robust relational transactional database systems. A well-defined model not only provides a repository of critical organizational data that is trustworthy as well as readily available and secure. A well-designed database also allows organized learning from assembling a series of reports drawn from a database (we call this read activity) without intruding on or slowing down the process of creating or collecting new data (we call this write activity).

Ultimately, a database is intended to provide organizational learning that enhances the capabilities and competitive strength of an organization as it evolves. Proper analytics with truthful data allow for high-speed learning with the results of companies becoming nimbler and more responsive to customer and market demands. Their investments in people and technologies become deliberate and intentional with greater long-term impact. Those people who know how to design these structures are effectively independent and capable of conducting their own systems without relying on others for infrastructure or transactional data.

Next up is more exploration of the specifics of the normalization process. This is a key component of building data layer structures that scale and provide data that is secure, reliable, and trustworthy. Keep it going!