Dallas Area Rapid Transit Database Project

Abigail Burns

The University of Texas at Dallas

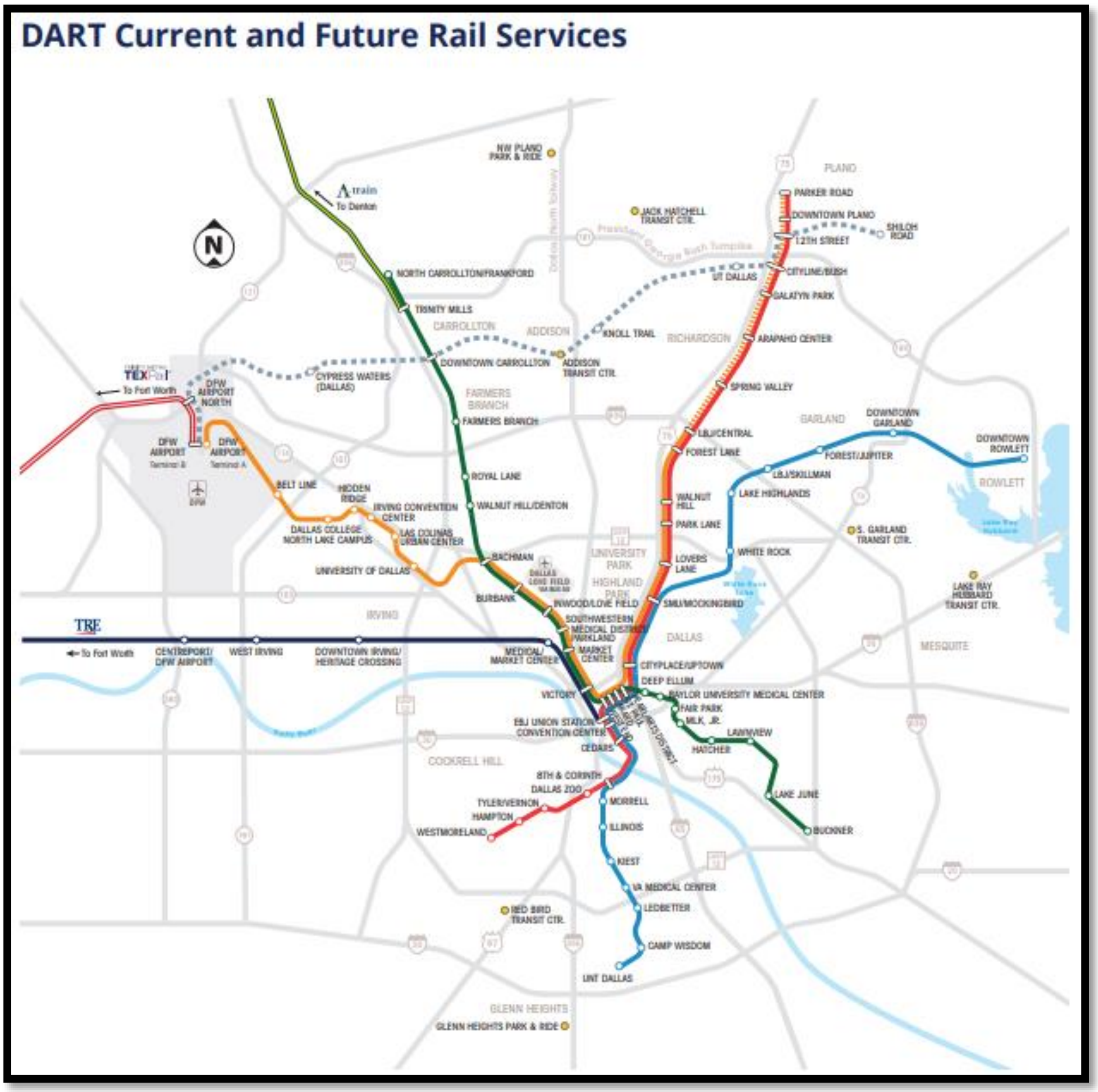
EPPS 6354 Information Management

Spring 2023

8 May 2023

## Abstract

The Dallas Area Rapid Transit (DART) train and bus system transports over 220,000 passengers daily, operating nearly around the clock to connect them across a 700-square-mile service area (About DART). As an avid transit supporter, I chose the DART rail network as my database topic so that I could create interactive web applications for searching arrival times at each station for each of the four lines (Red, Blue, Green, and Orange) and their routes (North-South and East-West). Inclusion of the available connection modalities at each station (train, bus, shuttle and/or parking) allows for easier trip planning.



## Literature Review

Transportation providers commonly use Oracle cloud-based systems to manage operations and route their fleet networks. These massive databases provide the backend for rail network and other logistical transportation network modeling (What Is Oracle Transportation Management).

DART hosts a GoPass mobile application for ticket purchase and route planning. The app allows the user to search for a destination and pulls up the soonest-arriving public transit modalities in the user's location, offering a list of modality combinations which could be used to reach the destination. Unless the user is currently at one of DART's sixty-five station, these combinations always require some portion of walking, biking, or driving. This exemplifies the "last mile problem" a logistical dilemma in which the first and final portions of transport are the most inefficient. This dilemma is "one of the most pressing challenges facing public transportation today" (How Cities Can Solve).

## Data and Methods

### Methodology

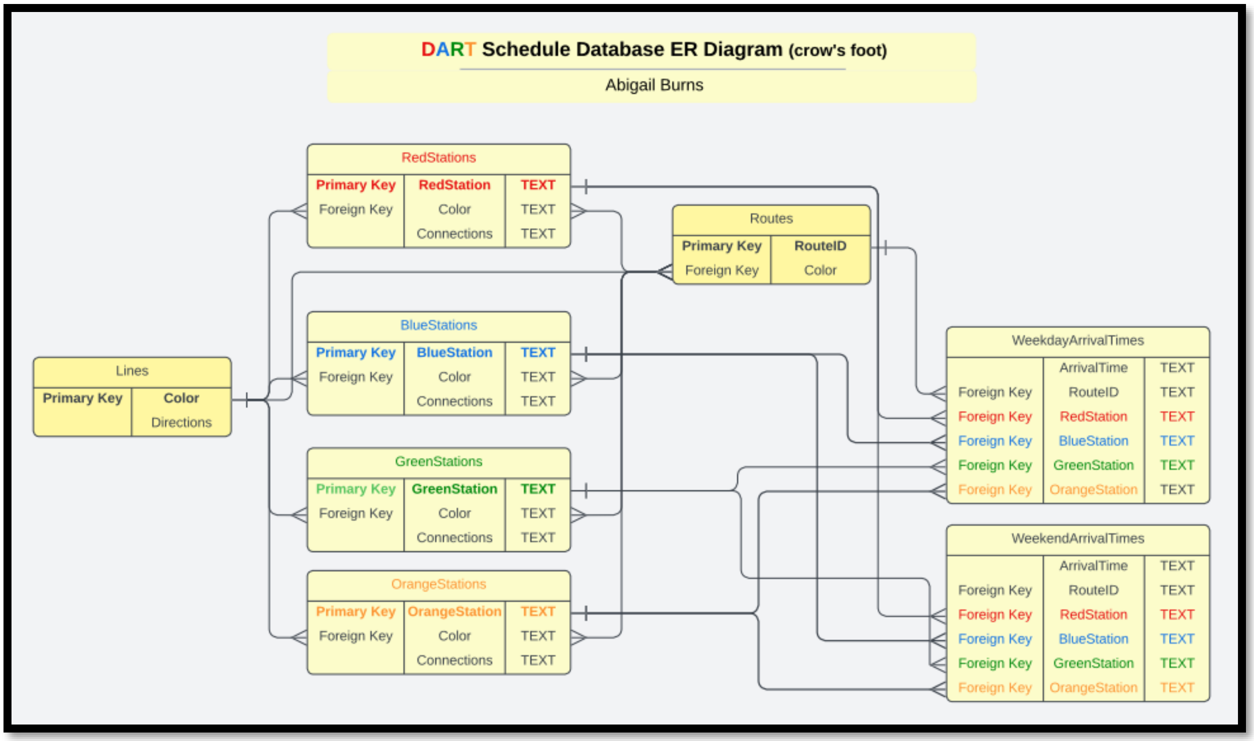
I began my project by gathering data from the authoritative DART website. For each of the four lines, there were four tables for arrival times. Of these four, there were two for each of the route directions. Among these two, one contained weekday schedule times and one contained weekend schedule times for each station along the route.

I then converted these tables from PDF files to Excel files using the Smallpdf extension for Chrome browser. The conversion tool did not correctly process the small images symbolizing the connection types available at each station. Although I intended to remove these images from the Excel file, this glitch meant that I had to manually enter the connection types with reference to the original PDF files. Additionally, I removed rows for times before 7:00 AM and after 11:00 PM to reduce the amount of SQL needed to populate my tables.

I began constructing my database by creating the eight tables and designing a schema that would link the tables together. There would be a table listing the four colors of lines that make up the DART rail system. There would be four tables containing the names of stations along each color line along with the accessible connections at each station. There would be a table containing the eight routes (four lines with two directions per line). Lastly, there would be two tables containing the weekday and weekend arrival times for all routes.

For each cell in my Excel tables containing an arrival time, portions of the SQL import query to populate an ArrivalTime record were added before and after the time using the Find & Replace editing tool. Because most arrival times were in regular increments, this was a reasonably efficient method. I then copied and pasted the cells, by column, into the SQL execution tab and generated an SQL file for each of the four Red line tables.

All fields would be of type ‘TEXT’ for simplicity. Only the ArrivalTime field could possibly be considered another data type. Three fields would be included that were neither primary nor foreign keys; ‘Directions’ to indicate both route directions of a given line, ‘Connections’ and ‘ArrivalTime’. Neither ‘Connections’ nor ‘ArrivalTime’ could meet the unique constraint in order to serve as primary keys, as many connection types, connection type combinations, and times of arrival were repeated in the source data.




| Name | Type | Schema |
|---|---|---|
| Tables (8) | | |
| BlueStations | | CREATE TABLE "BlueStations" ( "BlueStatio |
| GreenStations | | CREATE TABLE "GreenStations" ( "GreenSt |
| Lines | | CREATE TABLE "Lines" ( "Color" TEXT NOT |
| Color | TEXT | "Color" TEXT NOT NULL UNIQUE |
| Directions | TEXT | "Directions" TEXT NOT NULL |
| OrangeStations | | CREATE TABLE "OrangeStations" ( "Orang |
| OrangeStation | TEXT | "OrangeStation" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| Connections | TEXT | "Connections" TEXT |
| RedStations | | CREATE TABLE "RedStations" ( "RedStatio |
| RedStation | TEXT | "RedStation" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| Connections | TEXT | "Connections" TEXT |
| Routes | | CREATE TABLE "Routes" ( "RouteID" TEXT |
| RouteID | TEXT | "RouteID" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| WeekdayArrivalTimes | | CREATE TABLE "WeekdayArrivalTimes" ( " |
| ArrivalTime | TEXT | "ArrivalTime" TEXT NOT NULL |
| RouteID | TEXT | "RouteID" TEXT NOT NULL |
| RedStation | TEXT | "RedStation" TEXT |
| BlueStation | TEXT | "BlueStation" TEXT |
| GreenStation | TEXT | "GreenStation" TEXT |
| OrangeStation | TEXT | "OrangeStation" TEXT |
| WeekendArrivalTimes | | CREATE TABLE "WeekendArrivalTimes" ( " |
| ArrivalTime | TEXT | "ArrivalTime" TEXT NOT NULL |
| RouteID | TEXT | "RouteID" TEXT NOT NULL |
| RedStation | TEXT | "RedStation" TEXT |
| BlueStation | TEXT | "BlueStation" TEXT |
| GreenStation | TEXT | "GreenStation" TEXT |
| OrangeStation | TEXT | "OrangeStation" TEXT |

A foreign key constraint was placed on fields RedStation, BlueStation, GreenStation, and OrangeStation to maintain data integrity. However, because many stations are only part of one rail line, a blank record was added to each of the station tables (RedStations, BlueStations, GreenStations, and OrangeStations).

| Foreign Key |
|---|
| |
| "Routes"("RouteID") |
| "RedStations"("RedStation") |
| "BlueStations"("BlueStation") |
| "GreenStations"("GreenStation") |
| "OrangeStations"("OrangeStation") |

| | RedStation | Color | Connections |
|---|---|---|---|
| | Filter | Filter | Filter |
| 1 | | Red | |
| 2 | 8th Corinth | Red | Train,Bus,Shuttle |
| 3 | Akard | Red | Train,Bus |
| 4 | Arapaho Center | Red | Train,Bus,Shuttle,Parking |
| 5 | Cedars | Red | Train,Bus,Shuttle |

| | | |
|---|---|---|
| Tables (8) | | |
| BlueStations | | CREATE TABLE "BlueStations" ( "BlueSt |
| BlueStation | TEXT | "BlueStation" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| Connections | TEXT | "Connections" TEXT |
| GreenStations | | CREATE TABLE "GreenStations" ( "Gree |
| GreenStation | TEXT | "GreenStation" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| Connections | TEXT | "Connections" TEXT |
| Lines | | CREATE TABLE "Lines" ( "Color" TEXT N |
| Color | TEXT | "Color" TEXT NOT NULL UNIQUE |
| Directions | TEXT | "Directions" TEXT NOT NULL |
| OrangeStations | | CREATE TABLE "OrangeStations" ( "Ora |
| RedStations | | CREATE TABLE "RedStations" ( "RedSta |
| Routes | | CREATE TABLE "Routes" ( "RouteID" TE |
| RouteID | TEXT | "RouteID" TEXT NOT NULL UNIQUE |
| Color | TEXT | "Color" TEXT NOT NULL |
| WeekdayArrivalTimes | | CREATE TABLE "WeekdayArrivalTimes" |
| ArrivalTime | TEXT | "ArrivalTime" TEXT NOT NULL |
| RouteID | TEXT | "RouteID" TEXT NOT NULL |
| RedStation | TEXT | "RedStation" TEXT |
| BlueStation | TEXT | "BlueStation" TEXT |
| GreenStation | TEXT | "GreenStation" TEXT |
| OrangeStation | TEXT | "OrangeStation" TEXT |
| WeekendArrivalTimes | | CREATE TABLE "WeekendArrivalTimes" |
| ArrivalTime | TEXT | "ArrivalTime" TEXT NOT NULL |
| RouteID | TEXT | "RouteID" TEXT NOT NULL |
| RedStation | TEXT | "RedStation" TEXT |
| BlueStation | TEXT | "BlueStation" TEXT |
| GreenStation | TEXT | "GreenStation" TEXT |
| OrangeStation | TEXT | "OrangeStation" TEXT |

Next, I began programming interactive web apps in RStudio. I rewrote R programs used in a previous workshop and sourced from Dr. Ho’s GitHub repository to program two Shiny applications which I deployed on my personal Quarto website. I referenced Shiny documentation to customize the program to suit the DART traveler user experience, modifying it to include one slider input, two text inputs, and two table outputs. The first text input reduced the ArrivalTimes table based on the ‘RouteID’ column. The second text input constrained the RedStation column in the SQL selection query for the first table while also selecting for matching records in the RedStations table to show all connection modalities available at the selected rail station.

Debugging consisted mainly of checking that separate variables were assigned for the arguments of the different input controls and table outputs as well as for the different selection query statements.

### Data Analysis

Arrival time data was regularly incremented outside of busiest operating hours. During peak hours occurring around rush hour in the mornings and evenings, rail schedules are less predictable and trains take longer to load passengers between stops.

## Conclusion and Implications

### Limitations

Because of the high volume of records in each of the sixteen Excel tables, even after the removal of early-morning and late-night times, I chose to populate my tables using only the four Excel files comprising the Red line system. I initially attempted to generate SQL import queries using a Visual Basic Application (VBA) macro which integrates ChatGPT with Excel using an

API key from my OpenAI account (Bo, S.). Selection and input of numerous rows as prompts yielded inconsistent results from ChatGPT. Additionally, the prompt input character limit of 4,097 tokens reduces effectiveness. In the future, I could write a program in Python or R which automates the generation of SQL import queries to reduce the time needed to perform this preprocessing task.

## Recommendations for Future Research

I recommend that I improve the data entry process for population of my tables by automating SQL import query generation. This will make it much easier to complete my database with schedule data for the Blue, Green, Orange, and (future) Silver DART rail lines. As well, updates and changes to the table could be made more easily as the DART system continues to advance and expand. Additionally, the Shiny application interface could be aesthetically improved to be more colorful and engaging.

## Conclusion

This database project produced a practical and useful implementation of an interactive DART schedule querier. The results of these queries do not require internet accessibility when the application is run without web deployment in RStudio and can be relied upon with near certainty, all else equal.

References

About DART. (n.d.). Retrieved from https://dart.org/about/about-dart/about-dart

Bo, S. (2023). Retrieved from https://github.com/Sven-Bo/Integrate-ChatGPT-in-Excel-using-VBA

Ho, K. (2022). Retrieved from https://github.com/datageneration/informationmanagement/tree/master/workshop/Shiny

How Cities Can Solve the First and Last Mile Problem. (2021). Retrieved from https://www.remix.com/blog/solving-the-first-and-last-mile-problem

What Is Oracle Transportation Management? (2023). Retrieved from https://oracle.argano.com/oracle-blogs/what-is-oracle-transportation-management/