

IT for Fleet Managers: Part 1

Fleet Information Management:



This is the first in a series of articles on information technology for fleet managers. Understanding how the data elements relate makes using fleet management information easier and more productive.

By Christopher D. Amos, CAFM

No doubt, fleet managers have enough to do without becoming information technology (IT) experts too. If your internal or fleet management company IT support is responsive, and your fleet management software provides standard reports to meet all your needs, count yourself fortunate indeed. For the rest of us, understanding enough about IT issues to convey what we need intelligently or, better still, becoming self-reliant enough to extract and manipulate fleet data to meet unusual needs can be very useful.

The goal of this article series and the upcoming NAFA Fleet Management Information Guide is to provide fleet managers with the tools required to function in a data-rich, information-poor work environment by better using the tech-

nology tools they probably already have at their disposal.

The series' first article covers the structure of fleet data, how various pieces of data are connected, and how that data can be accessed and manipulated using ad-hoc query tools. **WARNING:** This stuff is not very sexy, but a "look under the sheets" of your normal fleet information management software interface is necessary if you ever hope to use anything other than "canned reports." It's better to experiment now than wait for that burning, oddball question from your CEO/elected official that can't be answered with the information summaries provided by your vendor.

RDBMS Software Handles Grunt Work

Virtually every modern fleet information management system (FIMS) is built

atop a relational database management system (RDBMS) product. Common RDBMS products like Oracle, Sybase, SQL Server, or even Access for smaller applications do the grunt data manipulation work behind the scenes. The RDBMS software provides computerized management facilities used to structure and manipulate data, and to ensure data privacy, recovery, and integrity in a multi-user environment. The automated function is similar to the tasks clerical personnel must do in reviewing and storing hard-copy information to be retrieved later — except, of course, the computer-supported process is much quicker, less expensive, and more reliable.

Data within a database is structured into tables, columns, and rows. Using a paper storage model, think: database=file cabinet; tables=file drawers; columns=file folders; and rows=sheets of paper inside folders. Using a spreadsheet model, think: database=Excel workbook (file named on disk); tables=worksheets (tabs at bot-

System Data Architecture

tom of the screen); columns=vertical groupings (labeled across the top); and rows=horizontal groupings (labeled down the side). Data can be stored logically at every intersection of columns and rows within a table. Multiple tables comprise a database. Products designed to track all fleet functions including maintenance garages and refueling operations may have more than 100 tables.

Most tables in a FIMS database serve as logical groupings of data including unique information about vehicles, transactions, and lists of codes to restrict the use of some information in other tables. Some tables serve to simply record information allowing the connection of two or more other tables, as described later in this article.

For example, in Figure 1, "Unit" table contains all the unique information about fleet vehicles and equipment such as unit number, year, make, model, vehi-

cle identification number, license plate, organizational/driver assignment, fuel type, fuel card number, net acquisition cost, date placed into service, expected life, and a category code to group it with similar units. It may also include columns to save current information for easy reference such as the most recent odometer reading and life-to-date operating costs.

"Fuel_Type" table contains unique information about fuels used in the fleet, but its primary purpose is to limit the choices that can be assigned in other tables (e.g., Unit and Fuel_Trans). Were it not for this type of table, multiple codes could be used for the same fuel, making summarizing data a problem. The front-end software interface used by fleet management personnel would not allow a vehicle to be assigned a fuel type code other than those listed in this table.

"Fuel_Trans" table contains refueling transaction data collected by a fuel card

program or fleet's card/key lock internal tanks and uploaded to the database or manually entered from receipts or tanker operations. This data may include fuel card number, fuel type (e.g., regular unleaded, premium unleaded, diesel), refueling station location, date, time, gallons, cost, driver personal identification number, and odometer reading entered by the driver or read electronically from a device on the vehicle.

Program logic in effect when entering or uploading this transaction data accepts data that matches information in the Unit table updating odometer readings and operating cost totals. However, it may reject an odometer reading smaller than the one already in the database or the entire transaction if the fuel type is not that listed for the vehicle. These rejections are typically summarized in an "exceptions report" produced to alert fleet management staff to investigate and resolve the inconsistencies.

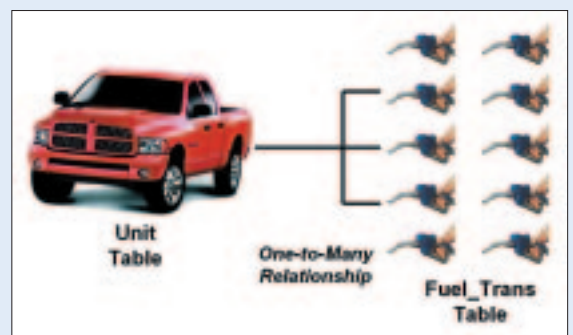
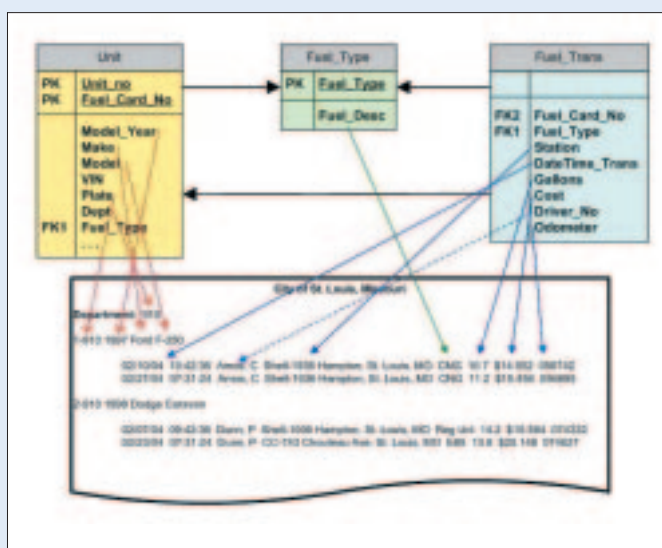


Figure 2. Relationships between tables are defined as one-to-many, many-to-one, or many-to-many. In this example, the relationship would be described as "Unit one-to-many Fuel_Trans" and "Fuel_Trans many-to-one Unit."

Figure 1. Data from tables "Unit," "Fuel_Type," and "Fuel_Trans" are combined into a single report.

Relationships

A primary reason data in a RDBMS is divided into so many tables is to avoid unnecessarily duplicating the storage of large amounts of information. It also simplifies updating information when it only appears in one location within the database. Using our example tables above, there is no reason to store data

nance garage transactions, you'd have a table listing the full inventory of "Parts" and another containing "Work Order" details, plus a supporting cast of other tables. A given work order appearing as a row in the "Work Order" table can be linked to multiple part numbers appearing as rows in the "Parts" table. Likewise, a given part number appearing as a

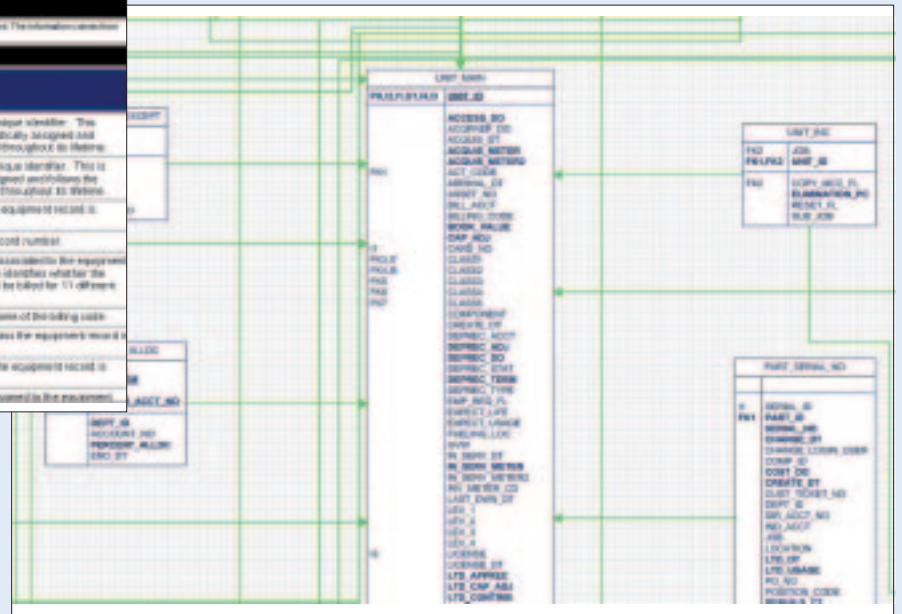
as technical information such as data type (e.g., character, number), length (e.g., 10 characters), whether or not the column can be left null (i.e., blank) as the row is added or updated, and whether or not the column is indexed to speed sorting and retrieval.

Because column names can be somewhat cryptic, the column description

Column	Data Type	Length	Nullable	Indexed	Description
ISSUED	Integer	4	✓	Y/N	The billing date unique identifier. The number is automatically assigned and follows the record throughout its life.
ISSUEID	Integer	4	✓	Y/N	The equipment unique identifier. This is automatically assigned and follows the equipment record throughout its life.
ISSUECOMP	Variable	3		F	The company the equipment record is assigned to.
ISSUEID	Variable	11		F	The equipment record number.
ISSUECODE	Variable	3		F	The billing code associated for equipment record. This code identifies whether the equipment record is billed for 11 different items.
ISSUECODE	Variable	40		F	The descriptive name of the billing code.
ISSUECLASS	Variable	10		F	The equipment class for equipment record is assigned to.
ISSUEID	Variable	30		F	The equipment the equipment record is assigned to.
ISSUEID	Variable	30		F	The equipment the equipment record is assigned to.

Figure 3. A table layout document clarifies how individual tables of data connect with each other.

Figure 4. Table relationships and linking key fields are graphically depicted in a data base map called an entity relationship diagram (ERD).



such as a vehicle's year, make, and model with every fuel transaction ever recorded, even though you would want that identifying information on a report of refueling history. To avoid this waste of computer storage capacity and slowing data processing/transmittals as it is referenced, a common link between the tables permits assembling the pieces of information in various tables as needed. This brings us to a discussion of relationships that you don't need Dr. Phil to sort out.

Relationships between tables are defined as one-to-many, many-to-one, or many-to-many. By placing the column "fuel card number" in both the Unit and Fuel_Trans tables and ensuring that all encoded cards/keys activated have been assigned to a vehicle, all fuel transactions are linked to vehicles. The relationship between these two tables would be described as: "Unit one-to-many Fuel_Trans" and "Fuel_Trans many-to-one Unit." (Figure 2.)

Illustrating a many-to-many relationship takes a slightly more complicated example. In a FIMS that tracks mainte-

row in the "Parts" table can be linked to multiple work orders appearing as rows in the "Work Order" table. Because both tables have a one-to-many relationship to the other, a third table called a "junction" or "foreign key" table is created by the database designers to store all the various combinations of work orders and part number matches in this many-to-many relationship.

Navigating the Tables of a Relational Database

Understanding how tables of data connect as described above is essential to using ad hoc query tools to extract sets of data for special study. However, databases with many tables, particularly when junction tables are involved can have relationships more convoluted than the shifting alliances of players on *Survivor*. That's where a "table layout" document and an "entity relationship diagram" come in handy.

The table layout typically lists column names and a column description as well

can be very useful in matching it to a column in one or more other tables. Database designers who document their work well include table relationship information in this column description. (Figure 3.)

Just as no man is an island, neither is a table in a RDBMS. Tables include a column or combination of columns that constitute a "key" that is linked to one or more other tables. A table designed to have every row unique (e.g., Unit) includes a "primary key" that will link to a non-unique "foreign key" in other tables (e.g., Fuel_Trans) in the one-to-many relationship described earlier.

Table relationships and linking key fields are graphically depicted in a database map called an entity relationship diagram (ERD), illustrated in Figure 4. With ERD and table layout spread on the conference table or windowed on the computer screen in front of you, you can determine in which tables the information you want is stored and how those tables are linked. This is most of the battle.

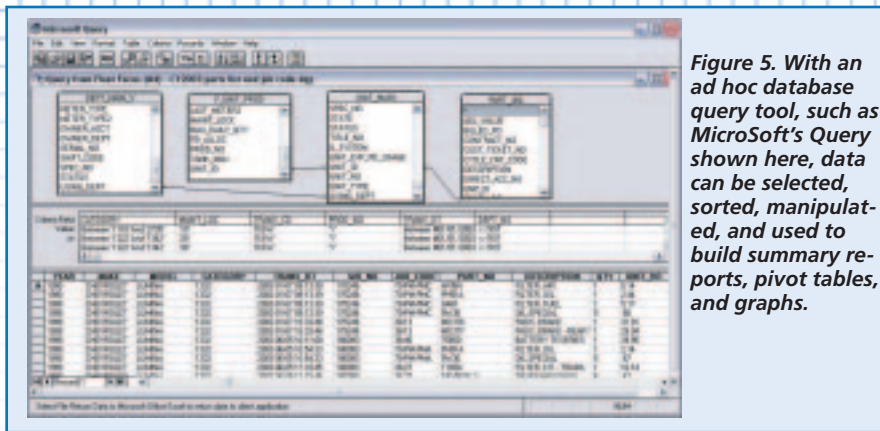
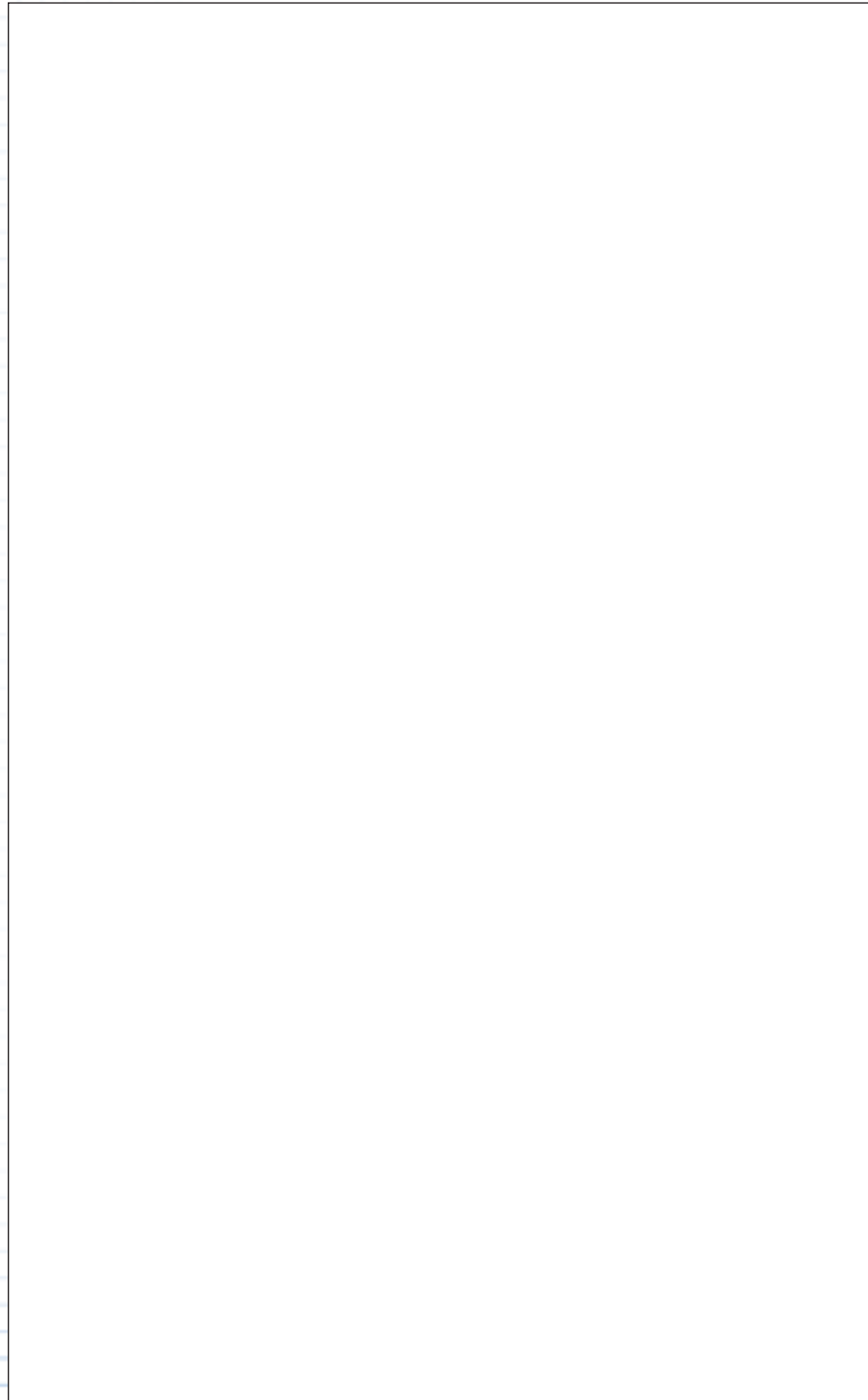


Figure 5. With an ad hoc database query tool, such as Microsoft's Query shown here, data can be selected, sorted, manipulated, and used to build summary reports, pivot tables, and graphs.



Custom Data Query Tools

Graphical data query tools have come a long way in recent years and make it possible to extract data from multiple tables in a RDBMS without the need to understand and write structured query language (SQL) statements. Some FIMS products include an ad hoc query tool eliminating the need to purchase and configure one. A Google search for “ad hoc query tool” also produces literally thousands of hits for products great and small, new and venerable, including the warhorse “Crystal Reports,” now in version 10. Better still, if your database is on a local server and accessed by client software installed on your PC, rather than via a Web browser, why not use something you probably already have — Microsoft Query to populate Excel worksheets. (Figure 5.)

The Microsoft Office suite ships with this handy ad hoc database query tool that uses the open database connectivity (ODBC) standard to connect to and extract data from popular RDBMS products. From inside Excel, access MS Query from the “Data” pull down menu, highlight “Import External Data,” and select “New Database Query...” Your system administrator can add the database connection for MS Query to use on any PC already connected to the database. If necessary, your system administrator can add the MS Query tool to your computer if a full install of Office wasn’t done when your PC was configured.

Once you are configured to access the database, you’ll typically find a graphical user interface (GUI) to select tables and columns of data, a way to link them by dragging a connector from one key to another, and another way to limit the data set returned by matching values,

Data within a relational database is structured into tables, columns and rows. Tables relationships are one-to-many, many-to-one, or many-to-many. The Microsoft Office suite ships with a handy ad hoc database query tool you probably already have called Microsoft Query.

Graphical data query tools have come a long way in recent years and make it possible to extract data from multiple tables in a RDBMS without the need to understand and write structured query language (SQL) statements.

eliminating values, or selecting ranges of values. MS Query allows you to work in “wizard” mode, in which you are either led through building the query one step at a time or able to jump right into the advanced GUI mode.

Query tools allow you to save data selection criteria once developed for future use or modification. The data returned by the query can be manipulated further in formulae, sorted, or color-coded based on value inside Excel or other packages and used to build summary reports, pivot tables, and graphs. Manipulating the raw data can actually produce useful information you can use to discover trends, support policy changes, and present to executive decision-makers.

Summary

Now that you know the basics of relational database design, you can explore your data and experiment with an ad hoc query tool. Talk to your FIMS software vendor’s technical support folks to learn what tool they recommend and secure their assistance with setting it up to integrate correctly with your database.

No worries mate — performing a query simply extracts a copy of your data for use on your local PC, so no harm can come from experimenting to see what is possible. Try combining information in ways other than those available on canned reports. Do a trend analysis, compare performance data for various internal fleet customers, develop some snazzy graphics to copy into a PowerPoint presentation, and see where these new-found skills will take you. You might discover helpful things about your fleet that you never

noticed before. At the very least, you’ll be ready to quickly tackle that next oddball problem or VIP question by providing more help to your IT support staff, or maybe you’ll even be able to handle it all by yourself. **GF**

FOR MORE INFORMATION:

Chris Amos, CAFM is Commissioner of Equipment Services, City of St. Louis, Missouri, and a senior associate with Mercury Associates, Inc. providing fleet management consulting services. He can be reached at camos@mercury-assoc.com