Module_5 – SQL & ER

5.1 - Introduction and Background

In the world of today, record keeping is used amongst all types of people and entities. Whether for individuals, businesses, or governments, record keeping is used to organize, store, and change data. One of the main tools today that allows record keeping to be more efficient and practical is SQL.

SQL, or Standard Query Language, is a type of language used to store, manipulate, and retrieve data in and from a database. SQL has its origins in 1970 when Dr. Edgar F. "Ted" Codd created the relational model of database management. SQL itself would appear 4 years later in 1974 when it was created at the IBM Research Laboratory in San Jose, California based on Codd's idea of a relational database. In 1978, a product called System/R was created by IBM to further develop these concepts. In 1986, the first standard prototype of SQL was created by IBM, and in that same year, SQL was considered the official standard by the American National Standards Institute (SQL – Overview, 2020).

Unlike Java, Python, or C, SQL is not a procedural language. Another contrast is that SQL is a 4th-generation language, meaning it is more similar to regular human language and speech and is able to be understood by an untrained person unlike Java, C, Python, or others considered 3rd generation languages (Williams, 2018).

Since SQL allows for easier use, especially when it comes to some of the large organizations that use it. Benefits include reduced training costs and increased productivity, because workers can focus on a single language, making it easier to train new employees. Workers are able to learn SQL proficiently, allowing them to increase their productivity and maintain existing programs. The use of SQL on multiple machines in a company also allows for application portability. By using SQL, application longevity is also improved, as standard languages do not go through frequent, large changes. This means that rewriting of older applications is not common with the use of SQL. Another benefit of using SQL is cross-system communication, which allows for the managing of data through multiple applications. These benefits mainly pertain to the use of SQL by corporations. Due to the fact that SQL can be used by almost anyone and is efficient in record keeping, it is known as one of the best (if not *the* best) language in its category (Ramesh, 2011).

The first ANSI SQL standards were published in 1986 and have been updated every few years after that. The updated ANSI SQL standards of 1992 are known to have greatly revised its structure. The structure now features three levels: Entry, Intermediate, and Full. These revisions have made SQL what it is today, constantly improving to better perform at the top level (Ramesh, 2011).

SQL has many major components within its process that enable it to work or process data and execute tasks:

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine

With these components, the way a user interacts with SQL is by commands. These commands allow the user to execute a multitude of actions, including Create, Alter, Drop, Select, Insert, Update, Delete, Grant, and Revoke. Commands can be modified by clauses, which set the conditions for a command. These clauses include Where, From, and Using. These commands and clauses are just one example of how SQL can be understood by an average person who has little experience in programming (SQL – Overview, 2020).

The highest ranking and/or most popular database management systems today are Oracle, Microsoft SQL Server, MySQL, MongoDB, and PostgreSQL (2019 Database Trends - SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use., 2019). Oracle is ranked the highest out of all database systems because of its level of functionality, portability, performance, recovery, speed, multiple database support, and reliability (Khamlichi, 2018). Many of these languages are capable of linking or syncing with other types of software and languages like Java. These attributes, among others, help make SQL the leading data organization and collection language in the world.

5.2 Define a Database using SQL data definition language

It is common knowledge to most who are familiar in the coding world that SQL is used universally to define databases and perform certain actions to interact with the content in those databases. Of these, there exist 4 major categories: Data Definition Languages, Data Query Languages, Data Manipulation Languages, and Data Control Languages.

A Data Definition Language, also known as DDL, is what consists of the "SQL commands that are used to define the database schema" (Varshini, D., 2019, August 26). DDL is also used to modify the content within the database: the database objects as well as the schemas. There are many different commands, including CREATE, DROP, ALTER, TRUNCATE, COMMENT, and RENAME. Each of these commands has its own unique utility and function when implemented as a SQL query. For instance, CREATE and DROP are used to simply create and delete database tables. ALTER can be used to add new columns to a table. So for example, if we had the table below:

EmployeeName	**EmployeeID**	**Position**
John	123	Engineer
Jeff	456	Accountant
Jason	789	QA

Employee (Kashefi 2020)

If we wanted to add each employee's salary, we could use the command:

```
ALTER TABLE Employee
ADD Salary
```

Which would give us this table:

EmployeeName	EmployeeID	Position	Salary
John	123	Engineer	
Jeff	456	Accountant	
Jason	789	QA	

Employee (Kashefi 2020)

And because the salary column was not filled, we are left with a new, blank column. If we wanted to clear all the information but leave the table and columns, we could use TRUNCATE:

```
TRUNCATE TABLE Employee
```

Which would leave us with this table:

EmployeeName	EmployeeID	Position	Salary

Employee (Kashefi 2020)

The next category used to define a database with SQL is Data Query Language, or DQL. This category of language is actually used to perform manipulations to the data within schemas rather than

the schemas themselves as mentioned above. The primary example of DQL is the SELECT command which selects a set of data which you want to perform your operation on and does this by retrieving the data from the database (Varshini, D., 2019, August 26). It is also worth noting that the primary purpose of DQL is to get a schema relation based on the query submitted. More information on DQL, its commands, and its uses can be found in Section 5.3.

DML, or Data Manipulation Language, is the next major category, which deals with data present in the database. This is different from DQL, because DQL only takes into consideration data from the schema object. It is also worth noting that this category includes most of the commonly used SQL statements such as INSERT, DELETE, and UPDATE. These allow users to actually modify data in small increments, rather than make sweeping changes that would be seen in DDL. For example, let's look back at this table:

EmployeeName	EmployeeID	Position	Salary

Employee (Kashefi 2020)

Currently this table is empty, but what if we wanted to add a new employee who has a name, ID, position, and salary? We would use this command:

INSERT INTO Employee(EmployeeName,EmployeeID,Position,Salary)

VALUES (“Jonah”, 159, Janitor, 1000000)

Which would leave us with this table:

EmployeeName	EmployeeID	Position	Salary
Jonah	159	Janitor	1000000

Employee (Kashefi 2020)

Next is DCL, which stands for Data Control Language. This category is mostly concerned with the rights and permissions of other users on the network database and has commands which manage these. These commands such as GRANT and REVOKE simply grant or revoke permissions to a specified user on the database system. This is important for both data security and data integrity. It ensures that data can only be seen by people who must see it, and it also ensures data cannot be changed by someone who does not have an understanding of what is happening in the database.

The last category we will discuss is TCL. TCL was not mentioned above, as there is debate on whether or not TCL is actually something that should be considered a major category within the data definition namespace. However, to be thorough we will touch on it briefly here. It essentially deals with transactions of data within the database. It does this through the use of commands such as COMMIT, ROLLBACK, SAVEPOINT, and SET TRANSACTION. All of these perform a certain action by specified keyword on a transaction. This can give users the ability to control versions of their databases. This can become important in the case of incorrect information or even a corrupted database.

5.3 Write a single table query in SQL

In data science, a query is a call for a specific set, group, or combination of data. In order to query a database, we need to use a language the database can understand (Gibbs, n.d.). Tables, on the other hand, are objects within a database that include some, or all, of the data from the database. This data is organized into a grouping made up of rows and columns. The rows each represent a unique item from the database records. The columns each represent different attributes that the item contains (Cai et al., n.d.).

StudentName	StudentID	ClassID	Grade
John	123	987	90
Jeff	456	654	83
Jason	789	321	97

Student (Kashefi 2020)

Above is an example table named Student. The first column represents the student's name. This student name is what we will use to name the unique items in the rows. For example, John is the name of a student. Every other column of that row contains data that describes him. No other row will have the exact same data. This makes his, and every other row unique.

In the example table, the columns represent attributes in the data. Every student has a name, a student ID, a class ID, and a grade.

In order to query from this table, we must use specific SQL commands. We must use the SELECT command to choose the columns in the table we are querying from. To select the correct table, the FROM command must be used. In order to ask for the entire table as a query, therefore:

```
SELECT * FROM Student
```

The asterisk is a shorthand way to ask for every column. This shorthand is also used in other database and programming languages. This command would be read in plain English as "select all columns from the student table." This would return everything from the above table, as the table is named *Student* and all of the information is in one column or another.

Now what if we just wanted to know the name and grade for each student? This is how it would look:

```
SELECT StudentName, Grade FROM Student
```

In this example, we used SELECT to choose columns. Instead of an asterisk used for "all columns," we use the specific column names "StudentName" and "Grade," which are intuitively the name and grade of the student, and the columns are separated by a comma. The FROM command underneath once again chooses the correct table. In plain English, this is "select the columns student name and grade from the student table." This would be the returned table:

StudentName	Grade
John	90
Jeff	83
Jason	97

Student (Kashefi 2020)

This query can be done with any combination of columns in the table, as long as the same formatting is used.

The last staple command of SQL is WHERE. This command will only return a row if the information in one of the specific columns fits a specific condition of the query(“Learn SQL: Queries Reference Guide,” n.d.). To allow a better understanding, here is an example:

```
SELECT StudentName, Grade
FROM Student
WHERE Grade < 95
```

For this example, we have the same queried table as above, except that any row with a grade above 95 is removed. In plain English, this would be “select the columns student name and grade from the student table if the student’s grade is below 95.” This would be the final queried table:

StudentName	**Grade**
John	90
Jeff	83

Student (Kashefi 2020)

This is just a small part of all the possible ways to query in SQL, but they are the main ones that can allow basic table queries to be practiced.

5.4 Establish referential integrity using SQL

Referential integrity is the accuracy and consistency of data in a relationship. Referential integrity is a subset of data integrity, which is concerned with the accuracy and consistency of data as whole in a database. When relationships occur, data is linked between two or more tables. A primary key is the key or specific column in a parent table, and a foreign key is the key in a child table that references the primary key. Referential integrity requires that a foreign key references a primary key (Ian, 2016).

The tables below will show how referential integrity works within a database management system:

Child Table

Department	
Employee ID	Department
6789	Marketing
5632	R&D

Parent Table

Employee		
Employee ID	Age	Salary
6789	25	56000
5632	29	83000

As you can see in the tables above, Employee is the parent table and Department is the child table. The primary key is the Employee ID and is referenced in the child table, therefore making it a foreign key in this table. The relationship is denoted by a curved line connecting the columns in the two different tables. As a result of this relationship a user will be prevented from:

- Adding information in the child table if the same information is not also in the parent table
- Changing data in the primary table that result in parentless keys in the child table (orphaned)
- Deleting records from the parent table if they exist in the child table

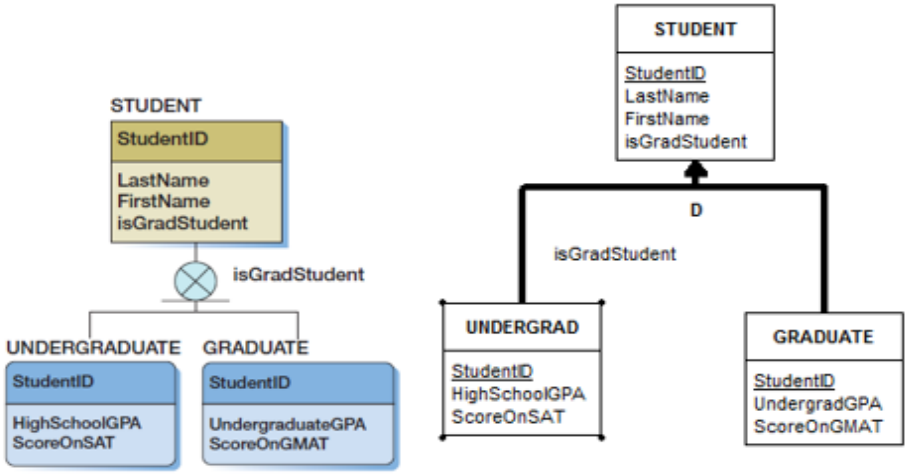
A lack of Referential integrity can result in records being lost and/or inaccurate or confusing. This can have serious and negative effects for entities that make use of database systems (Ian, 2016).

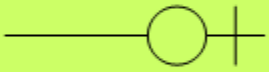
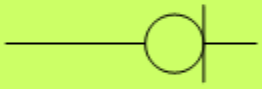
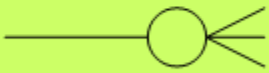
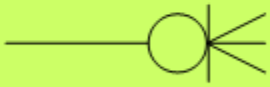
To ensure or establish referential integrity within SQL (Microsoft SQL Servers), it is necessary to establish variables (columns) as a primary key when creating tables. After creating the child table, expand the database in project explorer and look at "table." After finding the table tree, expand it and right click on "keys." After right clicking, it will provide options, one of them being a "new foreign key." Click "new foreign key" and expand "Tables and columns specification." After expanding, click on the 3 dots on the right side of the box. This will allow the user to enter a primary or parent able for foreign keys along with the specific column. After doing this, the tables will be linked to one another. The user will only be able to perform tasks that are within the constraints of referential integrity.

SQL and ER

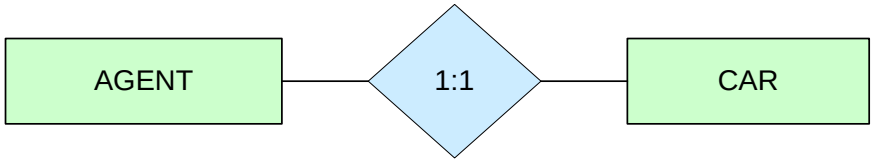
It is very important to remember that a database is a model of a user's view of the world. The only question is "How well does it fit the mental models of the people who are going to use the database system?" It is up to database administrators to create a SQL and Entity-Relationship (E-R) platform and provide the needs to fit user requirements. ER-Assistant provides relationships that are expressed using a different notation. It is easy to use but the entity boxes cannot be resized, leading to text length limitations.

Eriwin uses **solid vs. dashed lines for M:N relationships**; this can only be specified on "**children**" in a relationship. These features mean that nearly any E-R diagram created using erwin will be incorrect for this text. Figure_1 below illustrates the differences between the notation used in the text and the notation used by erwin.



Symbol Used In Database Concepts	ERWin Symbol	Meaning
		One - Mandatory
		Many - Mandatory
		One - Optional
		Many - Optional
		Exclusive Subtype
		Inclusive Subtype

Database Drawing such as Dia and Visio 2016 display names and roles of relationships and do not distinguish between weak and strong entities (no rounded corners). Connecting lines are solid:

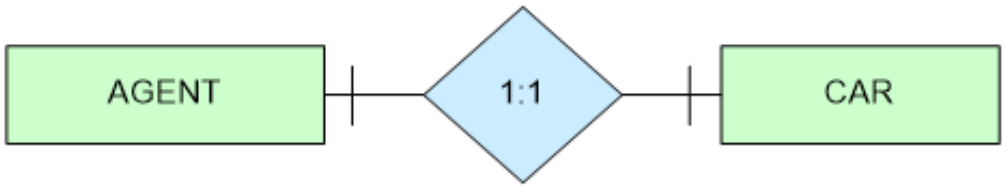


AGENT_VEHICLE

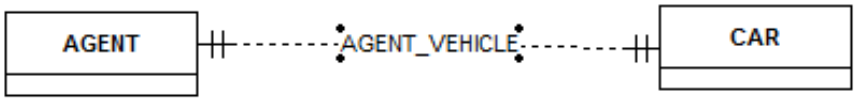
Information Engineering (IE) began with the work of Clive data modeling tools which is one of the most popular notations for database design using with a crow's foot.

Give examples of M-M, M-O, O-M, and O-O relationships (other than those presented in this chapter). Draw two E-R diagrams for each of your examples: one using the traditional diamond notation and one using IE Crow's Foot notation.

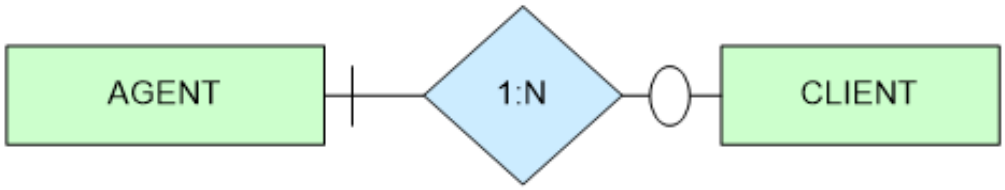
In the Real Estate Agency example in question 5.5, each AGENT must use an agency car when on agency business. Further, to keep costs down the agency keeps exactly enough cars for the agents. Therefore, each AGENT must have a CAR, and each CAR must be assigned to an AGENT. This is an M-M relationship.

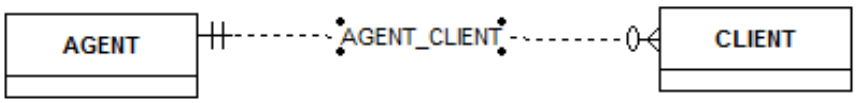


AGENT_VEHICLE



In the Real Estate Agency example in question 5.5, each CLIENT must be assigned to an AGENT, but there may be AGENTs who currently have no CLIENTs. This is an M-O (same as O-M, but seen reversed) relationship.





The E-R Crow's Foot model above is based on the model in question C but adds the entity ACTOR. Since there are no additional attributes needed, this can be modeled as an N:M relationship. The data for the new parts of the model are contained in the following table:

RELATIONSHIP			CARDINALITY [Blue = Inferable]	
PARENT	CHILD	TYPE	MAX	MIN
ACTOR	MOVIE	Strong	N:M	O-O

5.5 - Section Concise Summary

SQL, a non-procedural query language, is a major database management system that helps organize and log data. Commands and clauses like CREATE, SELECT, etc. are what makes SQL work and gives it functionality. Using these commands and clauses, it is possible to create databases, create tables, edit data within tables, add columns and rows, create keys and so on.

The intuitive words and language that make up these tables allow people who may not be skilled in computer science to be able to use the system. Some of the commands are powerful in another sense as well: they allow the user to quickly scan through all of the given information and return only a specific set based on any number of parameters that the user decides are necessary.

Using other commands, data can be linked between two tables, using characteristics such as Referential Integrity, primary keys, and foreign keys. Primary keys uniquely identify records in tables, while the foreign keys reference the primary key. The linking that results from this is called a relationship. Referential integrity helps prevent the duplication of already existing data and it helps make sure that necessary records aren't accidently deleted. Referential integrity gives SQL a way to make sure that the data that is in the table is correct and updated due to constraints on what can be changed, and who it can be changed by. The many features of SQL enable it to be the most popular query language in the world.

Furthermore, this module discusses the process of transforming a data model into a database design as various aspects of data models and designs, how they relate to each other, and how they relate to the systems analysis and design process in general and to the systems development life cycle (SDLC) in particular. Transforming a data model into a database design requires three major tasks: replacing each entity with a table and each attribute with a column, representing relationships and maximum cardinality by placing foreign keys, and representing minimum cardinality by defining actions to constrain activities on values of primary and foreign keys.

Four uses for ID-dependent entities are N:M relationships, association relationships, multivalued attributes, and archetype/instance relationships. An association relationship differs from an intersection table because the ID-dependent entity has no key data. In all ID-dependent entities, the key of the parent is already in the child.

5.6 Extended Resources

1. This is Microsoft's SQL documentation. This resource helped us gain information on different commands and queries and when they should be used. The information is detailed and in depth, albeit dense to read in some parts. https://docs.microsoft.com/en-us/sql/?view=sq l-server-ver15

---

2. This Khan Academy video gives in depth basics to SQL and querying tables. It explains what querying is and gives some detail on how to do it.
   https://www.khanacademy.org/computing/com puter-programming/sql/sql-basics/v/welcome-t o-sql

---

3. This W3Schools resource explains what referential integrity is. It is also a good source in showing examples of joining with referential integrity.
   https://www.w3resource.com/sql/joins/joiningtables-through-referential-integrity.php

---

4. This Geeks for Geeks resource gives explanations on SQL, specifically what it is and what it can do. It most specifically gives information on querying tables in the database.
   https://www.geeksforgeeks.org/structured-quer y-language/

---

5. This YouTube video shows how to complete early steps of using a SQL database, including defining a database and using the database to query information.
   https://www.youtube.com/watch?v=Z-vsvu9Y H7M

---

6. This Code Academy page lists every query command that exists in SQL. It also gives examples of the use of each one.
   https://www.codecademy.com/learn/learn-sql/ modules/learn-sql-queries/cheatsheet

---

7. This W3Schools resource gives examples of the usage of multiple queries and operations to clear any confusion that may come from a specific one or even just the syntax.
   https://www.w3schools.com/sql/sql_where.asp

---

8. This website allows for users to practice many SQL functions online. It also has videos for users to look at to help them with their exercises. https://sqlzoo.net/wiki/SQL_Tutorial

---

9. This website allows users to learn about SQL and how it works. It also serves as a site where users can practice SQL and look at tutorials to help with their practice. https://sqlbolt.com/

10. This website allows users to learn the fundamentals of SQL and database concepts. https://www.essentialsql.com/

11. This source lists 18 different sites that can help users learn about and practice SQL. https://academy.vertabelo.com/blog/18-best-online-resources-for-learning-sql-and-database-c oncepts/

5.8 - References

SQL - Overview. (n.d.).(2020) Retrieved February 22, 2020, from https://www.tutorialspoint.com/sql/sql-overview.htm

SEQUEL (SQL) is Developed1974. (2020). Retrieved February 22, 2020, from http://www.historyofinformation.com/detail.php?id=910

SQL clauses. (n.d.). Retrieved February 22, 2020, from https://docs.oracle.com/javadb/10.8.3.0/ref/rrefclauses.html

Williams, H. (2018, August 6). What Is SQL. Retrieved February 22, 2020, from https://www.computerworld.com/article/3427818/what-is-sql--all-you-need-to-know-aboutstructured-query-language.html

2019 Database Trends - SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use. (2019, March 4). Retrieved February 23, 2020, from https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple -database-use/

Khamlichi, M. (2018, February 6). Why oracle database is the best DBMS solution. Retrieved February 23, 2020, from https://www.osradar.com/oracle-database-best-dbms-solution /

Ian. (2016. May 28). What is Referential Integrity? Retrieved February 24, 2020, from https://database.guide/what-is-referential-integrity/

Borsecnik, J., Guyer, C., Miliner, G., Hamilton, B., Masha, Macauley, E., … Woodard, J. (n.d.). Lesson 1: Create and query database objects. Retrieved February 23, 2020, from https://github.com/MicrosoftDocs/sql-docs/blob/live/docs/t-sql/lesson-1-creating-database -objects.md

Guyer, C., Miliner, G., Ray, M., Hamilton, B., Cai, S., Kumar, S., & Stein, S. (n.d.). Tables. Retrieved February 23, 2020, from https://github.com/MicrosoftDocs/sql-docs/blob/live/docs/relational-databases/tables/table s.md

What is a Query in SQL? (n.d.). Retrieved February 22, 2020, from https://study.com/academy/lesson/what-is-query-in-sql.html

Varshini, D., & GeeksForGeeks. (2019, August 26). SQL: DDL, DQL, DML, DCL and TCL

Commands. Retrieved from https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

Learn SQL: Queries Reference Guide. (n.d.). Retrieved March 23, 2020, from https://www.codecademy.com/learn/learn-sql/modules/learn-sql-queries/cheatsheet

Hoffer, J. A., Ramesh, V., & Topi, H. (2011). *Modern database management*. Upper Saddle River, NJ: Prentice Hall.