# Module_5 – NoSQL, MongoDB, & SQL

## 5.1 - Introduction and Background

At the heart of many large-scale applications and services lies a high-performance data storage solution. The backend data store plays a crucial role in storing vital information such as user accounts, product data, accounting details, and blogs. Effective applications require a data storage mechanism that can handle data storage and retrieval with accuracy, speed, and reliability. It is essential to select a data storage solution that can meet the demands of your application.

Various data storage solutions are available to cater to the data requirements of your applications. The three most used options are direct file system storage in files, relational databases, and NoSQL databases. In this book, MongoDB is chosen as the NoSQL data store due to its widespread usage and versatility.

The upcoming sections provide an overview of NoSQL and MongoDB, along with discussions on important design considerations to consider before deciding on the data structure and database configuration implementation. These sections will address the relevant questions to ask and explore the built-in mechanisms in MongoDB that effectively fulfill the resulting requirements.

### NoSQL

NoSQL stands for "Not only SQL," highlighting the fact that NoSQL databases serve as an alternative to SQL and can incorporate SQL-like query concepts. NoSQL encompasses databases that differ from traditional relational database management systems (RDBMS). The driving forces behind NoSQL are simplified design, horizontal scalability, and enhanced control over data availability. NoSQL databases are specialized for specific data types, making them more efficient and performant compared to RDBMS servers in many scenarios. NoSQL aims to break free from the conventional structure of relational databases and allows developers to model their data flow according to the specific needs of their systems. This grants the flexibility to implement NoSQL databases in ways that would not be feasible with traditional relational databases.

Numerous NoSQL technologies exist, including HBase with its column structure, Redis with its key/value structure, and Virtuoso with its graph structure. However, this book focuses on MongoDB and the document model due to its exceptional flexibility and scalability for implementing backend storage in web applications and services. Moreover, MongoDB is currently the most popular and well-supported NoSQL language available. The following sections will delve into various types of NoSQL databases.

## 5.2 Define a Database using SQL data definition language

In the realm of coding, it is widely known that SQL serves as the universal language for defining databases and executing actions to interact with their contents. These actions can be broadly categorized into four major groups: Data Definition Languages (DDL), Data Query Languages (DQL), Data Manipulation Languages (DML), and Data Control Languages (DCL).

Data Definition Language (DDL), also referred to as DDL, encompasses SQL commands used to define the structure of a database schema. It is also employed for modifying the content within the database, including the database objects and schemas. DDL comprises various commands such as CREATE, DROP, ALTER, TRUNCATE, COMMENT, and RENAME, each with its own unique purpose and functionality when executed as SQL queries. For instance, CREATE and DROP are used for creating and deleting database tables, while ALTER allows the addition of new columns to a table. To illustrate, consider the following table:

```
sql
Copy code
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Age INT,
    Salary FLOAT
);
```

| ***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:

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)
```

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.).

| *1. **StudentName*** | *2. **StudentID*** | *3. **ClassID*** | *4. **Grade*** |
|---|---|---|---|
| *5. **John*** | *6. 123* | *7. 987* | *8. 90* |
| *9. **Jeff*** | *10. 456* | *11. 654* | *12. 83* |
| *13. **Jason*** | *14. 789* | *15. 321* | *16. 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 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).

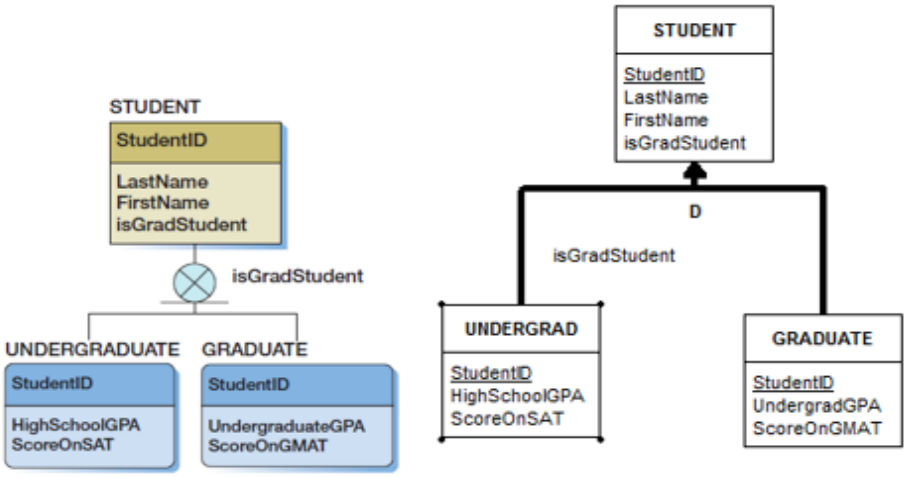
To ensure or establish referential integrity within SQL (Microsoft SQL Servers), it is necessary to establish variables (columns) as a primary key when creating tables. After creating the child table, expand the database in project explorer and look at “table.” After finding the table tree, expand it 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.

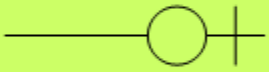
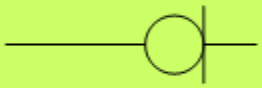
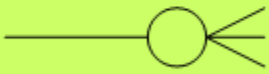
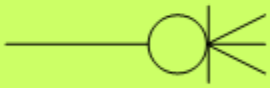
#### SQL and ER

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

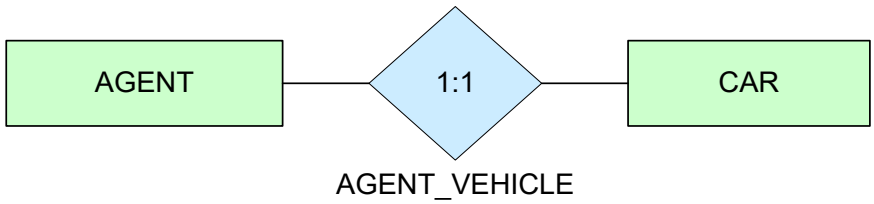
system?" It is up to database administrators to create a SQL and Entity-Relationship (E-R) platform and provide the needs to fit user requirements. ER-Assistant provides relationships that are expressed using a different notation. It is easy to use but the entity boxes cannot be resized, leading to text 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 |

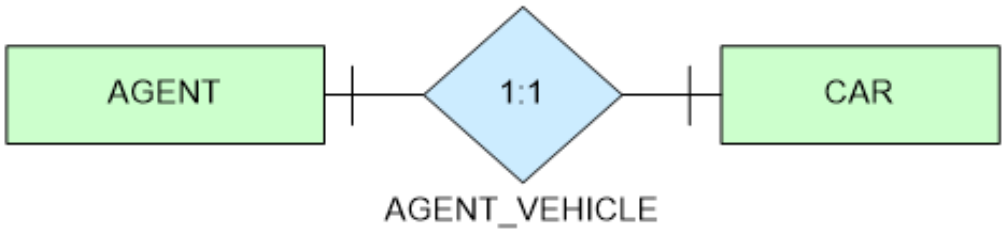
Database Drawing such as Dia and Visio 2016 display names and roles of relationships and do not distinguish between weak and strong entities (no rounded corners). Connecting lines are solid:

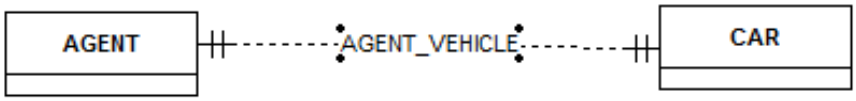



**Information Engineering (IE)** began with the work of Clive data modeling tools which is one of the most popular notations for database design using with a crow's foot.

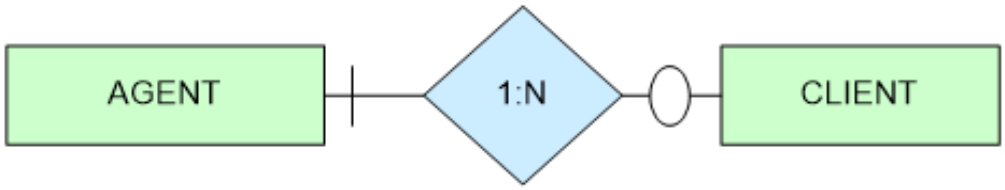
*Give examples of M-M, M-O, O-M, and O-O relationships (other than those presented in this chapter). Draw two E-R diagrams for each of your examples: one using the traditional diamond notation and one using IE Crow's Foot notation.*

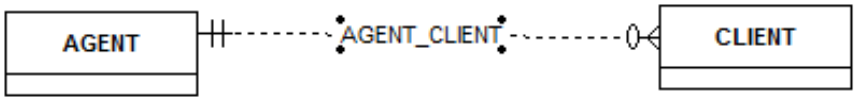
In the Real Estate Agency example in question 5.5, each AGENT must use an agency car when on agency business. Further, to keep costs down the agency keeps exactly enough cars for the agents. Therefore, each AGENT must have a CAR, and each CAR must be assigned to an AGENT. This is an M-M relationship.




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 – Data Dictionary**

A data dictionary, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". Oracle defines it as a collection of tables with metadata. Example is:

**A data dictionary in database** is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data. Typically, only database administrators interact with the data dictionary.

A **data dictionary** is a centralized repository of metadata. Metadata is **data** about **data**. Some **examples** of what might be contained in an organization's **data dictionary** include: The **data** types, e.g., integer, real, character, and image of all fields in the organization's databases.

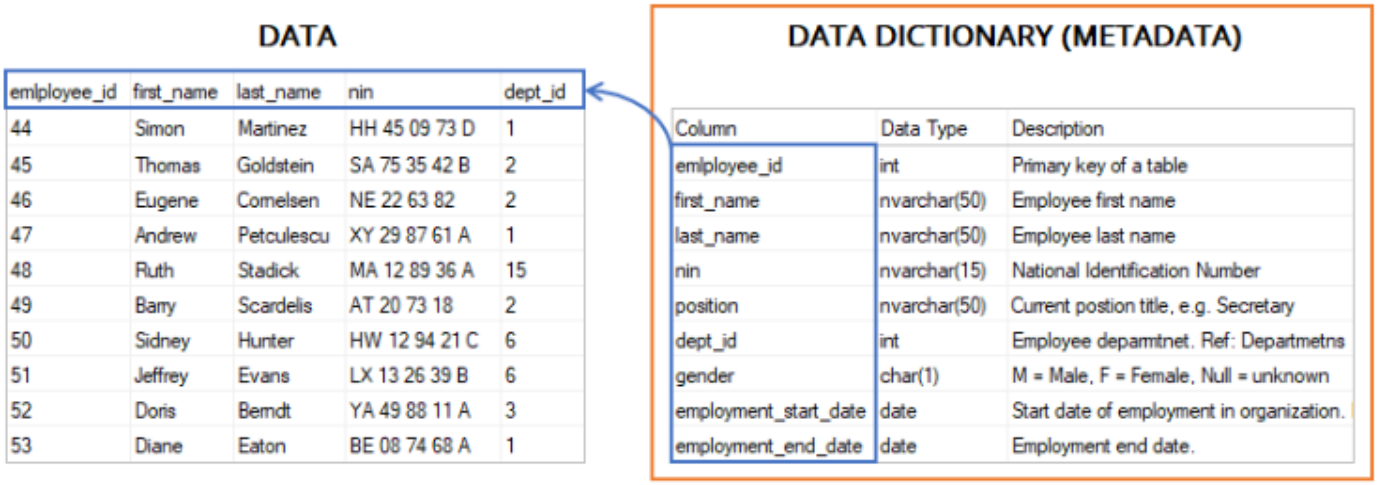
The Purpose of Data dictionaries are used to provide detailed information about the contents of a dataset or database, such as the names of measured variables, their data types or formats, and text descriptions. A data dictionary provides a concise guide to understanding and using the data.

In SQL Server the data dictionary is a set of database tables used to store information about a database's definition. The dictionary contains information about database objects such as tables, indexes, columns, datatypes, and views. The data dictionary is used by SQL Server to execute queries and is automatically updated whenever objects are added, removed, or changed within the database.
All the examples for this article are based on Microsoft SQL Server Management Studio and the AdventureWorks2012 database. You can get started using these free tools using my Guide *Getting Started Using SQL Server*.

```
SELECT   name,
         type_desc
FROM     AdventureWorks2012_Data.sys.tables
ORDER BY Name
```

Data Dictionary is a list of data elements (entity/table and attribute/column) with their attributes and descriptions. It has a form of a set of tables. In **SQL** Server the data dictionary is a set of database tables used to store information about a database's definition. The data dictionary is used by **SQL** Server to execute queries and is automatically updated whenever objects are added, removed, or changed within the **database**.

Source from Dataedo; https://dataedo.com/kb/data-glossary/what-is-data-dictionary



DATA

| emlployee_id | first_name | last_name | nin | dept_id |
|---|---|---|---|---|
| 44 | Simon | Martinez | HH 45 09 73 D | 1 |
| 45 | Thomas | Goldstein | SA 75 35 42 B | 2 |
| 46 | Eugene | Cornelsen | NE 22 63 82 | 2 |
| 47 | Andrew | Petculescu | XY 29 87 61 A | 1 |
| 48 | Ruth | Stadick | MA 12 89 36 A | 15 |
| 49 | Barry | Scardelis | AT 20 73 18 | 2 |
| 50 | Sidney | Hunter | HW 12 94 21 C | 6 |
| 51 | Jeffrey | Evans | LX 13 26 39 B | 6 |
| 52 | Doris | Berndt | YA 49 88 11 A | 3 |
| 53 | Diane | Eaton | BE 08 74 68 A | 1 |

DATA DICTIONARY (METADATA)

| Column | Data Type | Description |
|---|---|---|
| emlployee_id | int | Primary key of a table |
| first_name | nvarchar(50) | Employee first name |
| last_name | nvarchar(50) | Employee last name |
| nin | nvarchar(15) | National Identification Number |
| position | nvarchar(50) | Current postion title, e.g. Secretary |
| dept_id | int | Employee deparmtnet. Ref: Departmetns |
| gender | char(1) | M = Male, F = Female, Null = unknown |
| employment_start_date | date | Start date of employment in organization. |
| employment_end_date | date | Employment end date. |

**Data dictionary** is an inventory of data elements in a database or data model with detailed description of its format, relationships, meaning, source and usage.

A data dictionary is used to catalog and communicate the structure and content of data, and provides meaningful descriptions for individually named data objects.https://www.usgs.gov/products/data-and-tools/data-management/data-dictionaries

## 5.6 – SQL DML – Refer to CRUD

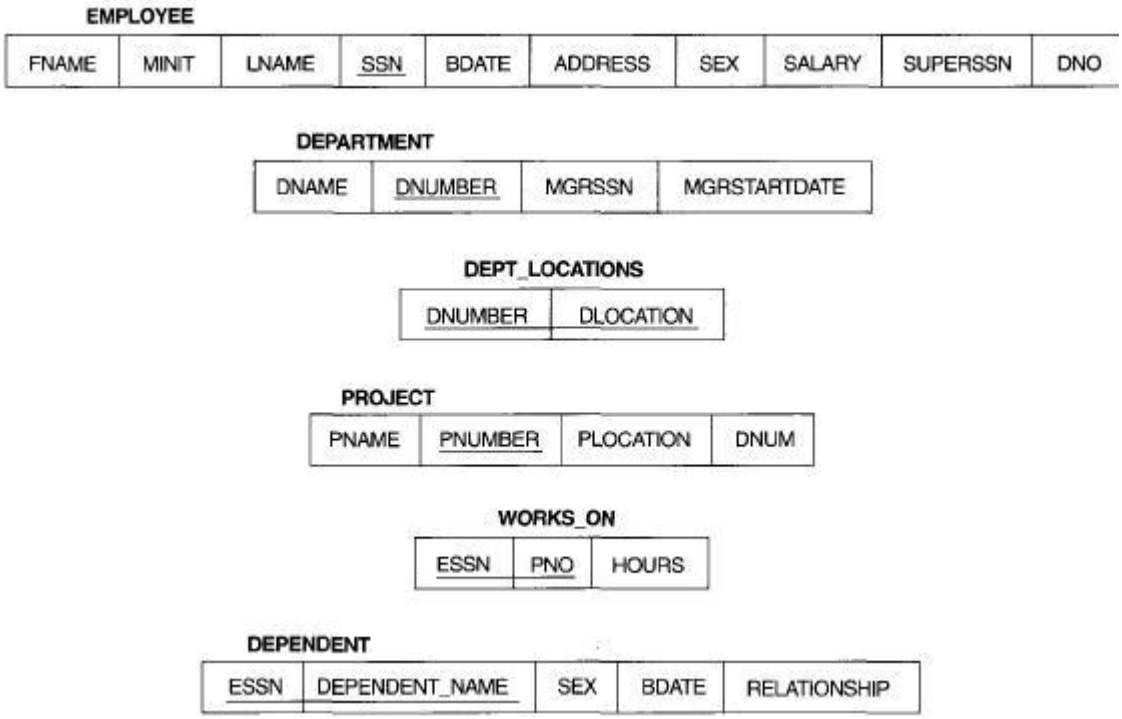
**CRUD is a**n acronym for **create, read, update, and delete**. It is used to describe the four actions done to data by a DBMS. The four actions listed for SQL DML are sometimes referred to as **CRUD**: create, read, update, and delete. We do *not* use this term in this book, but now you know what it means. Video Shows the CURD Matrix Generation for SQL server Database.
https://www.youtube.com/watch?v=K_T3AptuDKA

## 5.7 – Rational Schema model

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and to uniquely identify any record or row of data inside a table. A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Questions is that, Can a relationship have a primary Key? At least, the **primary key** of the connected entities appears in the **relationship** as foreign **keys**. We **can**'t use the AuthorId attribute as the **primary key** because its value is not unique.

Questions is that, you can use a **relational schema** to describe self-referencing relationships. This is the same as a unary 1:1 **schema**. For **example**, an Employee who is the manager of a Subordinate may have manager as his or her foreign key. In this case, the Employee ID would be the primary key in the same table. Primary and foreign key relationships are used in relational databases to define many-to-one relationships between tables. A primary key is a column or a set of columns in a table whose values uniquely identify a row in the table.




## 5.8 - Section Concise Summary

Deploying a MongoDB solution in your environment necessitates the implementation of access control, which is a crucial aspect of the process. Access control involves the creation of user accounts with distinct roles and the requirement for users to authenticate themselves in order to access MongoDB database servers.

Users can be assigned various roles, such as read, read/write, or administration, which define the extent of their permissions when connected to the database. By implementing access control, you can establish separate user accounts with specific privileges: one user dedicated to administering user accounts and another user with exclusive rights to administer the database. This approach ensures tighter access control measures. In the previous chapter, you had the opportunity to create, view, and delete users. Additionally, an example was provided that illustrated the setup of a user administrator and a database administrator, along with configuring the MongoDB server to enforce authentication.

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.

**5.9 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/

---

12. Performance evaluation of SQL and MongoDB databases for big e-commerce data, Seyyed Hamid, Mehdi Rqzapour, Nasser Ghadir, IEEE Explore, https://www.ieee.org/

---

13. A review on various aspects of MongoDB Databases, Anjail Chauhan, International Journal of Engineering Research & Technology (IJERT), http://www.ijert.org

---

14. A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages, Mayur Patil, Akkamahadevi Hanni, etal.
https://ieeexplore.ieee.org/abstract/document/8058365 published at https://www.ieee.org/

---

**5.10 - References**


Data Management – Data Dictionaries: US Department of Interior, 2021, https://www.usgs.gov/products/data-and-tools/data-management/data-dictionaries

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.

Ashis Kumar Samanta, Bidut Biman Sarkar, Nabendu Chaki, "Query Performance Analysis of NoSQL and Big Data", *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pp.237-241, 2018.

Kopal Chaudhary, Mohini Gupta, Parmeet kaur, "Analyzing IPL dataset with MongoDB", *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp.212-216, 2019.

Yanyan Wang, Chunfang Li, Jiangnan Sun, Min Li, Jintian Yang, "An Athlete's Foot Data Platform with 3D Point Cloud Processing and Management Technology", *2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall)*, pp.192-197, 2021.