# Give Me Data

# or

# Give Me Death!

## Why Business Waits for Data and What You Can Do About It

***Essential Data Access Skills for the Business User***

*by* ***D. Crypt***

For more information contact:
d.crypt@dataselections.com

FIRST EDITION



DataSelections.com

# Table of Contents

# About the Author

**Devin Crypt** *is a technologist with decades of experience in both architecting and managing enterprise level software systems. Often called the master of metaphors or the sultan of similes, Devin has a unique way of simplifying complex subjects. Often accused of standing on a soapbox to clarify what really matters when designing software, his focus is always driven by a true empathy for the end user.*

# About the Title

In 1775 Patrick Henry gave a speech at the Virginia convention. Patrick spoke of liberty and freedom from the tyranny of British rule. He ended his speech with "Give me liberty, or give me death!"

---

*Liberty: The condition of being free from oppressive restriction.*

---

His call for action is based on the strong desire for colonies to become independent by breaking off from British rule which condemned the colonies.

His speech points out that everybody can have a different opinion on how to handle a situation, however, many problems are not resolved due to fear from being different than everyone else.

For 10 years the colonies stood idle and hoped for change. Patrick describes the past and shows evidence that their current situation will continue unless they do something to change it.

He states that there is no longer any room for hope and calls the Virginians to fight for their liberty.

Still, after decades of hope for change, business users are still waiting for data from IT. *Give Me Data, or Give Me Death* is meant to help you fight for your own data without fear. History has shown us that your current situation will not change unless you do something about it.

Your journey begins now …

# Preface

Business data from a software program, whether purchased from a vendor or built internally, is primarily stored in a relational database.

For what seems like an eternity, business users have been complaining about getting access to their data for reporting. Whatever IT offers, is simply never enough.

What are users complaining about, why are they complaining, and how might they help themselves solve the age-old issue of data access?

To quickly dispel any technical naysayers or experts who will inevitably attempt to discredit anything said in this document, this book was not written for you (although you'll probably learn something about your users). Nobody knows everything about data access regardless of what they may want to believe. I'm nothing more than a skilled practitioner (one who acquires knowledge from actual practice), sharing knowledge I have acquired over the years from my own experiences. What the reader chooses to do with this information is up to them.

I don't go into deep technical detail or try to cover 100% of all data access reporting scenarios. I have decades of technical knowledge and understand fully that corporate data can be spread across multiple databases be in different formats, and that reporting means something different to everybody. Consolidating all corporate data or handling all ETL[1] (Extract Transform Load) scenarios, is way beyond the scope of this or any other single document.

I am quite clear in stating that I believe in the 80/20 rule (Pareto principle[2]). For those of you not familiar, Vilfredo Pareto was an Italian economist. The 80/20 rule simply means that 80% of your results comes from 20% of your effort.

Pareto is known for the 80/20 rule which he discovered while observing pea yield in his garden. He noticed that not all pea plants were yielding the same amount. In his estimation 20% of pea pods produced 80% of the yield. Pareto applied the same logic to the land distribution in Italy and found that 80% of the land was owned by 20% of the population. This gave birth to the Pareto principle. (pareto-chart.com)

20% of your reporting efforts should satisfy 80% of your reporting needs. Unfortunately, for most users, it's the other way around. Most users spend 80% of their time to satisfy only 20% of their reporting needs.

---

[1] ETL, in computing, extract, transform, load is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source(s) or in a different context than the source(s). (wikipedia.org, en.wikipedia.org/wiki/Extract,_transform,_load)

[2] Pareto principle, is a principle, named after economist Vilfredo Pareto, which specifies an unequal relationship between inputs and outputs. The principle states that, for many phenomena, 20% of invested input is responsible for 80% of the results obtained. Put another way, 80% of consequences stem from 20% of the causes. Also referred to as the "Pareto rule" or the "80/20 rule". (Investopedia.com)

*Many users can only access 20% of their data for reporting.*

If I can help remove 80% of your data requests to IT then they have more time to work on the other 20%. Regardless, having more knowledge about data and reducing some burden from IT can't hurt.

In short, I'm not trying to solve every reporting problem. I just want help explain some of the issues with reporting and hopefully clarify some ways to have better access to your data. Essentially providing you limitless possibilities to report on your data in a manageable way.

*Data access is not as complicated as everyone makes it out to be.*

Your database was populated with some form of computer software system. accounting, medical, HR, sales, inventory, etc. The type of system doesn't matter, all the data access rules for reporting on the database still apply. The only thing that matters is that the data is accessible using SQL[3] "sequel" (Structured Query Language). Which covers millions of systems around the world.

Some people pronounce the Structured Query Language as individual letters "S Q L" [ˈɛs kjuː ˈɛl] others as "sequel" [ˈsiːkwəl]. Why the confusion? The original query language was named SEQUEL (Structured English Query Language) but had to be changed to SQL due to a trademark infringement with the Hawker Siddeley Aircraft Company that claimed the rights to the word SEQUEL.

*Knowledge is power.*

The more data about your business that's readily available to you, the faster and more accurate your business decisions can be. The results are real tangible dollars, either made or saved.

*The sad news is that most business users settle for less reporting. Not because they can't benefit from more information, but because it's too time consuming or costly to obtain.*

Having fast and easy access to your data, in business terms you understand, is the single most important factor to making smart decisions.

[3] SQL (Structured Query Language), is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. (wikipedia.org, en.wikipedia.org/wiki/SQL)

## Full Transparency

I work for DataSelections.com and we sell a product called Data Selections Framework (DSF), which helps you easily access and understand your data dynamically the way it was intended. We build SQL queries for you.

---

*DSF is NOT a reporting tool to create pretty charts.*
*It's a data access helper.*

---

DSF does NOT replace any reporting or Business Intelligence (BI[4]) tools that you've already invested in. DSF works with your existing tools to enhance their abilities for *you* to access *your* data. Our mission statement is simple.

---

*"Our goal is to make business data readily available, so that business users can make great decisions in a timely manner."*

---

Again, the product's main purpose is to provide *intelligent* and *easy* access to your data in *business terms* you understand. Not all of it, but most of it. Some data fields in databases may be useless to the business user.

Even without DSF you can still apply any knowledge learned in this book on your own data queries.

## Moving On …

If, for a single moment you believe that you can access 80% of your data from your database on your own with your current BI or reporting tool, think again. These tools require "*data miners*" (IT experts) every time, you, the business user, need some new data. More importantly, they have no *data access dynamicity*.

You won't find the word *dynamicity* in the dictionary.

---

*Dynamicity – The condition of being dynamic.*

---

In other words, the ability to access data in a flexible manner. When users can't access data in a flexible manner for themselves, they get aggravated. This aggravation leads to frustration which ultimately leads to giving up. They find another way to do their job without the data.

I plan to unconventionally arm you with knowledge to make it easier for you to access your data from your database yourself or at least let you understand why you have to wait for IT to do it for you.

---

[4] BI (Business Intelligence), is a technology-driven process for analyzing data and delivering actionable information that helps executives, managers and workers make informed business decisions. (Rouse)

I'm not going to bloat the pages of this book with a section on *Audience for This Book*, *How to Use This Book*, *Structure of This Book*, *Explaining the Examples in This Book*, *Typographical Conventions*, *Coding Conventions*, or 3 pages of *Acknowledgements*. It's a book. Read it and maybe learn something new.

If you already know what I'm covering then skip ahead. Nobody's holding you back.

Although I cover many SQL commands and features, this book is not intended to teach you the SQL language. It's designed to give a business user an understanding on how data access works and tips on why a report query may be slow. Some sections may get a little technical. I apologize in advance.

# Data Access by the Numbers

---

**50+ Years**

*Databases have existed and still end-users complain about their reporting capabilities.*

**Billions**

*Of lost opportunity dollars due to lack of access to business data in a timely manner.*

**80%**

*Of Business Intelligence initiatives fail.*

**4 OUT OF 5**

*Managers state that their current reporting abilities fall short of their needs.*

**224**

*Data professionals surveyed, many responded that they were 'unsatisfied' with the current time to value of their BI project. …most BI teams are just too slow in meeting the business needs.*

**100%**

*Of all business managers believe that fast access to the right data is crucial in making good business decisions.*

---

# CHAPTER 1

## Introduction

This book was written to help the business user take the mystery out of data access for the purpose of reporting. Having a strong understanding of why reporting is difficult or slow in today's world will give you the power to change it.

I could have titled this book *My Business is Dying Without Access to My Data*. Because that's exactly what's happening. Your business may not actually be dying, but it is painful trying to run a business without access to all your data. The business data is locked in your database.

After learning some basic data access principles and techniques, you'll no longer be at the mercy of your database that's currently holding your data hostage. I focus on data availability as it pertains to you, the business user. Isn't it time to get the reports you want, when you want them?

Business users have been asking to break the chains from IT for many years. Money was spent. Time has passed. After billions of dollars and decades of time, business users still have problems reporting on their data.

---

*Reporting is data mining for any purpose. The search and extraction of data from a database in business terms that is understandable to the business user. The actual display medium of the information, whether it is on a screen as a chart, a piece of paper, a dashboard, a list in a spreadsheet or an extract to another system, is immaterial.*

---

I believe that having access to your business data shouldn't be a privilege granted to you by your IT department. Why do you have to wait in line until they have time to help you?

After all, it's your data not theirs. I'm not talking about access to secure data that you legitimately shouldn't have access to, that's a separate topic altogether. I'm also not talking about modifying any data that could harm your data integrity.

I'm simply talking about reading data, to enhance your reporting abilities, which computer systems have stored in a Relational Database Management System (RDBMS[5]) or simply, a database.

[5] When we refer to an RDBMS, we mean any database where data can be accessed using SQL (Structured Query Language). Some newer database structures are not relational but still support SQL for data access.

Data that *you* provided to the IT staff to store for *your* business. You entered data into some software application and then it was locked away on a database server in a cryptic format you can't understand.

---

*You know your data is in the database somewhere, you just can't get it back out without IT's help. Why?*

---

There are thousands of great reporting tools on the market today and still after decades of dealing with databases, business users still have reporting problems. If everything is so great then business users shouldn't be complaining. But they are.

This book will help you understand why reports can be a struggle and how you may be able to make them load faster with some simple changes to your way of thinking.

The lessons found here will attempt to de-mystify your data access troubles in simple business terms. Once you understand how data access works, small changes may help your tools perform better.

## About Your Existing Reporting Tools …

The following paragraph was found on the internet described as a benefit for a commercially sold BI tool:

> *"With ad-hoc reporting, all the technical user does is set up the BI solution, connect it to the data-sources, establish security parameters and determine which fields end-users can see. From that point on, the actual reports are created by business end-users." (logianalytics.com)*

Think about this statement for a moment. The IT technical user will *"… determine which fields end-users can see."*

After IT gives you a limited set of data fields, you can do whatever you want with them. Thanks, but isn't IT still limiting you to a small set of data fields? When you need more data, you must go back to them and ask for it. They spoon feed you your own data. Your frustration ensues. They just don't get it.

---

*Why the frustration? Besides the obvious, humans have a basic need to direct their own lives. Accessing your own data isn't any different.*

---

Your current reporting tools will claim that they can connect to a multitude of database types (Microsoft SQL Server, Oracle, MySQL, Etc.) and get any data you want.

I'm quite sure they can "*connect*[6]" to a database, but then what?

---

*If the computer industry has done such a great job for everyone with such things as BI tools, custom report generators or even the Structured Query Language (SQL), then why are end-users still struggling to get data out of the database in a timely fashion?*

---

Many of you have tried this already. After connecting to the database, the business user must figure out the rest by themselves. The data model[7] and the context[8] of that data. This is why IT needs to get involved to write you an SQL query. You may even have learned some SQL on your own out of necessity to help solve this problem.

---

*"According to Gartner[9], 70 to 80 percent of business intelligence projects fail. The solution could be a more agile development process — and organization."*
*Why Most Business Intelligence Projects Fail*
*(enterpriseappstoday.com)*

---

If you do quick internet search for "*business intelligence failure rate*", the results are astonishing! The experts will primarily state that failures are caused by the developers and data miners not working closely enough with corporate end-users to understand their needs.

---

[6] Connect, a database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not. (wikipedia.org, en.wikipedia.org/wiki/Database_connection)

[7] Data Model, defines how data is connected to each other and how they are processed and stored inside the system. (tutorialspoint.com, tutorialspoint.com/dbms/dbms_data_models.htm)

[8] Context, something that clarifies meaning or purpose of an object as its own related group. (merriam-webster.com, merriam-webster.com/dictionary/context)

[9] Gartner, Inc. (NYSE: IT) is the world's leading information technology research and advisory company. They deliver the technology-related insight necessary for their clients to make the right decisions, every day.

*Gartner warns, "Organizations often develop and deploy hindsight-oriented reports and/or query applications focusing on metrics that users may find interesting, but they do not represent the operational or strategic controls used to facilitate business performance." (computerweekly.com)*

Really? *"... they don't represent the operational or strategic controls used to facilitate business performance."* IT needs to ask the end-users what data they need to run their business? Genius! Because IT doesn't know your business like you do.

I believe that users know what data they want, but also may not always know what they need up front. BI solutions require the business to determine all possibilities ahead of time. Unfortunately, businesses evolve and reporting needs change frequently.

*Data access needs to be flexible.*

So, here's a thought; how about letting the business users pick whatever data they want, when they needed it, in terms they understand?

After all, the end-users are running the business. IT needs to realize that their entire existence depends on them meeting the needs of the business.

*I believe the power of most data access can and should be in the hands of the business users with little or no IT involvement.*

This book is all about business users freely and quickly getting access to the data elements they need to make those all-important business decisions as soon as they needed it. Not when IT has time to get around to it.

*"No way! Business users don't know what they're doing!"*

Really Mr. IT? Business users are running the business that pays your salary. I think having some read access to the business data isn't going to kill anybody.

# CHAPTER 2

## What's the Problem?

The problem isn't users asking for data from the database. Every user wants real Ad-hoc (dynamic or flexible) reporting and data extracts from their database. Yes? Which means they want *limitless data access* to their business information! Access to more data results in better decision making.

Asking IT departments for data is not an uncommon request. In fact, it happens quite frequently. Which is where the problem starts. They become the bottleneck to the flow of the business.

---

*IT departments are stretched thin. Users have no choice but to wait their turn.*

---

Users change their minds as to what data they need daily. That's because businesses are fluid. Changes happen often.

There is no way the IT staff can keep up with everything the user wants. When IT doesn't respond quickly, users inevitably look for their own alternatives to reporting.

How do I know this? Because there are hundreds of successful companies that do nothing but try to give users a better alternative to reporting. (Crystal Reports, Tableau, Microsoft's SSRS just to name a few.)

Decades of experience in the computer software business has scarred me well. You can never keep up with the user's crazy requests for data.

---

*Everyone knows that getting useful data from a database is a difficult technical task best left to the IT experts. Is it?*

---

If you purchase a fancy reporting tool you can get to your data (*physical data connection*) but you still need to know how to make sense of the data in your database. It's context. That's where IT comes back into the picture and you're back waiting in line or learning SQL yourself.

*There seems to be a direct correlation to the cost of a system and its ability to provide good reporting.*

A million-dollar system is harder to report on than a one-hundred-dollar system. Why is that? Larger, more expensive systems, were built years ago and have evolved over time. As they evolve, they're data model gets more and more complex.

*One of the last things that software developers think of, when adding new features to a system, is reporting on the data they just created.*

Developers like working on the "*cool stuff*". Reports aren't cool.

Developers don't use the system for the business purpose it was designed. Creating reports is not that important to them. They barely even think about reports until an actual user asks for one.

Ask an IT person to add additional data to an existing report. Go ahead, ask.

The first words out of their mouth will be something like: "Why do you need it?"

You'll proceed to explain your reasons to them, as if they even care. They don't care!

What they're really asking is: "How bad do you need this data? Cuz I'm kinda busy doing other things and I really don't feel like writing you a query."

## Frustrations

---

*The reports that came with my system are missing a couple of fields I need to run my business. Vendor wants money to change them.*

*My software vendor does not have an Ad-hoc Reporting tool but says I can just buy a third party tool like Crystal Reports or Tableau, connect to the database and extract what I need.*

*IT made me a view in my database that has 20 fields but today I need a couple of new fields that are not in the view. I can't get anybody from IT to change it for me.*

*My BI vendor said they can extract data from any database. I can contract them to work with me to create the reports I need.*

*IT doesn't have time to create my new business report now. They will put it on the schedule for a later date. Low priority unless I can get executive approval to override their other priorities.*

*I was finally able to create the reports I needed with my third party reporting tool. I just received a new update from my software vendor and now all of my reports are failing.*

*My reports ran fine when I first got them but now when they run, it takes them forever to load.*

---

If any of this sounds like you, you're not alone. Everybody has a different reason of why reporting from their database has issues. This list could go on forever.

I recognize your frustrations. I can't solve all your issues, but I may be able to help explain why some exist.

Databases are not as complicated as people would want you to believe. Your business may be complicated, but you know your business. So, you must understand your business data. Right?

## Understanding a Little Bit About Computer Software

All computer software programs are built with 3 basic principles:

- Input - can be any process that takes in information.
- Process - is what the program does with that information.
- Output - are the results of the process.

A computer game takes *input* from a joystick. The *process* could be calculating how far to move an image on the screen. The *output* could be as simple as displaying the image at the new location on the output device (the screen).

When you buy something at a store, the clerk scans the bar code of the item (*input*) or types in the price (also *input*). The *process* looks up the item and/or calculates the amount owed. *Output* is the act of displaying on the register how much you need to pay or even the printed receipt.

In both examples, what would be the point of *input* and *process* if there was no *output*? You move the joystick but see nothing happen on the screen or your items are scanned for purchase but you don't know how much you have to pay. Sounds stupid right?

Relate that to a business computer system where users spend thousands of hours *inputting* data elements into a database and then IT only provides a small percentage of those items for reporting (*output*). Yet, with this scenario it's not considered stupid, but normal! Why is this scenario acceptable? For years users have been conditioned to expect less from their database.

Don't feel bad, you're not alone. Almost every business professional has the same problem. IT professionals took the data in a format that the business owners understood and converted it into a format that can only be understood by IT. The relational database.

## De-Mystifying the Database

This is a general overview of the database design process. Although sections may get a little technical, this is not meant to be a detailed technical reference document.

When building a computer system for a business or industry, whether a custom internal system or a commercial product sold by a vendor, the basic design process is the same.

The data that you have in your business industry (before it was *relationalized* in your database) is analyzed by a Software Engineer or Architect. With a Subject Matter Expert (SME), which is someone with experience in the business, the architect designs a way to capture and organize your data for you. This is usually, but not limited to, a series of input forms on the screen.

The architect breaks the business data down into meaningful data elements and categories (with a lot of help from the SME) to create the data model of the database. The data model is nothing more than an idea on how to organize your data in a relational database, which consists of a series of tables or lists, that may or may not have a relation to one another.

## Relational

Although we all understand the basic definition of the word *relational*, what does it really mean when we apply it to your data?

Let’s assume for a minute that your business has Customers, Employees and Vendors.

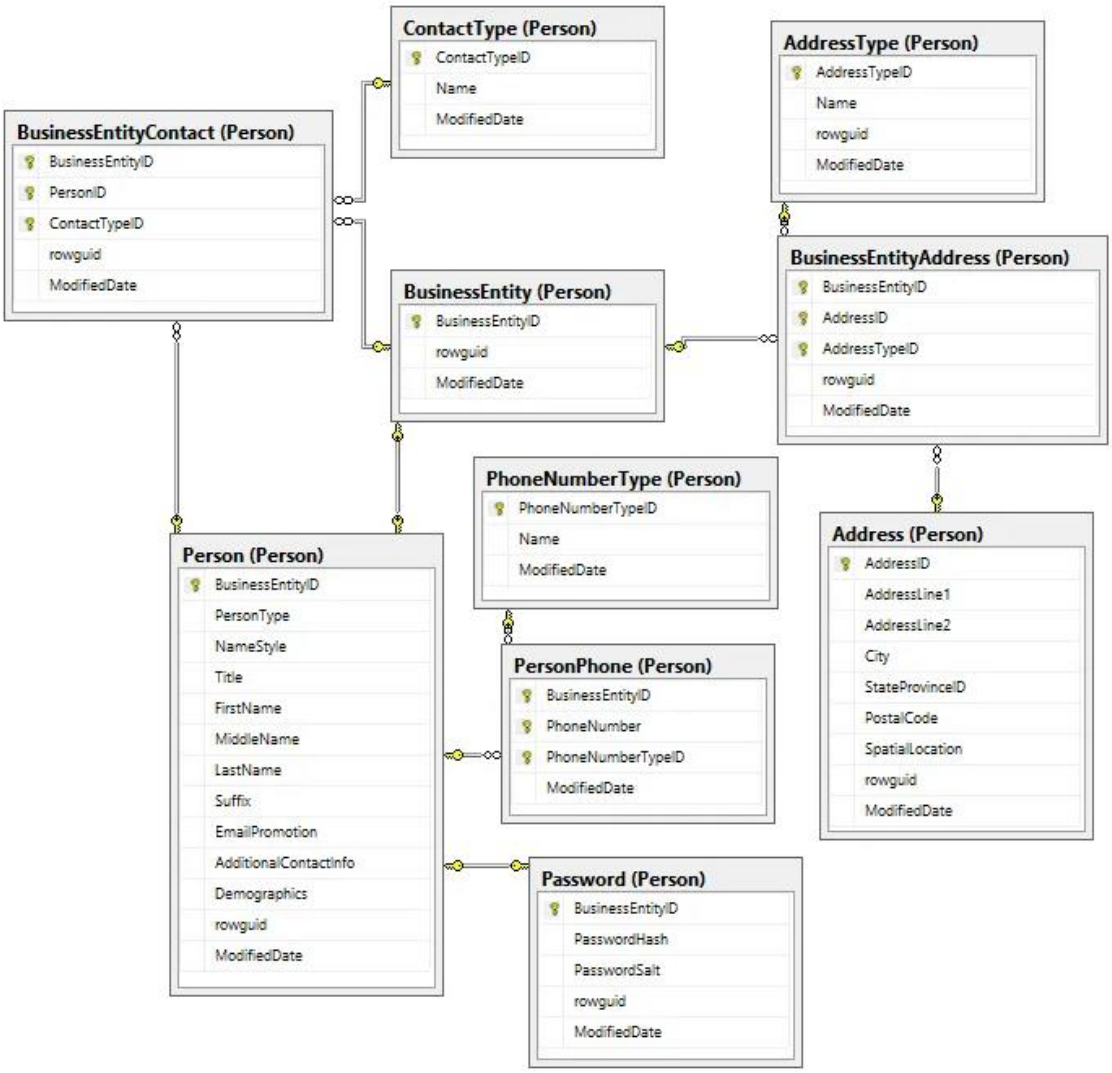
Each one of these has Contact Information like; Name, Address, Phone Number, Etc. As a normal human you would probably get an address book and write this information down in alphabetical order. You may even make a note to determine whether the person is a Customer, an Employee or a Vendor.

Not a Software Architect! They figure that a Customer is a Person and an Employee is a Person and Vendors have people too. They also realize that each of these People could have more than one Phone Number or Address.

They skillfully dissect your business information into series of data categories which get created as Tables and Columns in the database.

Your database is nothing more than a bunch of tables with column names.

Usually, table relationships follow closely to the business purpose the database was designed for. For example, if it’s an Accounting system, then the tables and columns should be structured to some likeness of an accounting model but not always.




The architect decides on a link or an *implied* relationship between the tables that are related. Hence the name Relational Database.

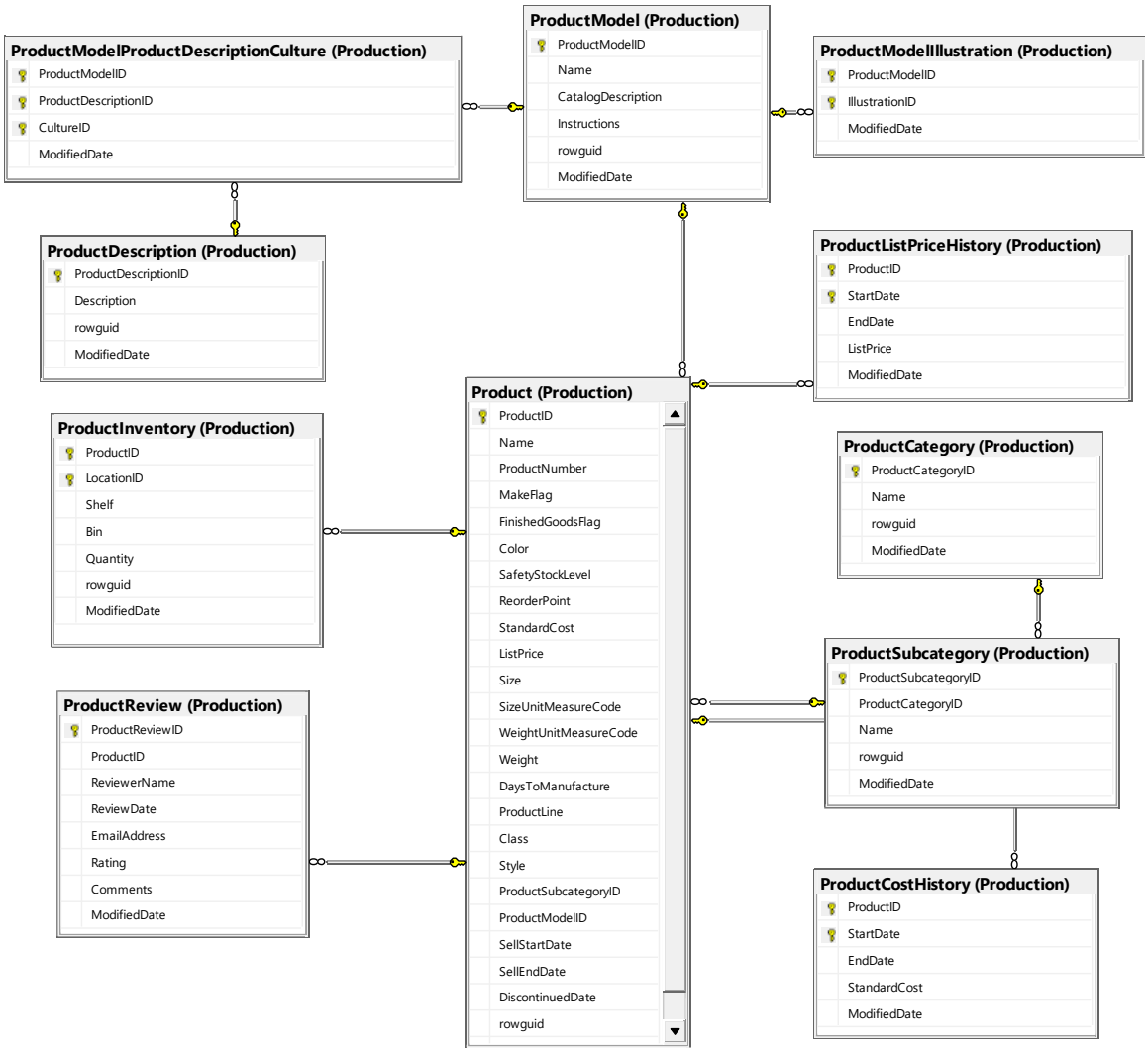
If the diagram above seems confusing don't worry because at first glance it is. Even a skilled Software Developer will have difficulty until they become familiar with the architect's data model. Only the architect knows what was going on in his head at design time.

---

*Most normal people don't view data in this relational form. That's why you need a query to pull it all back together for reporting.*

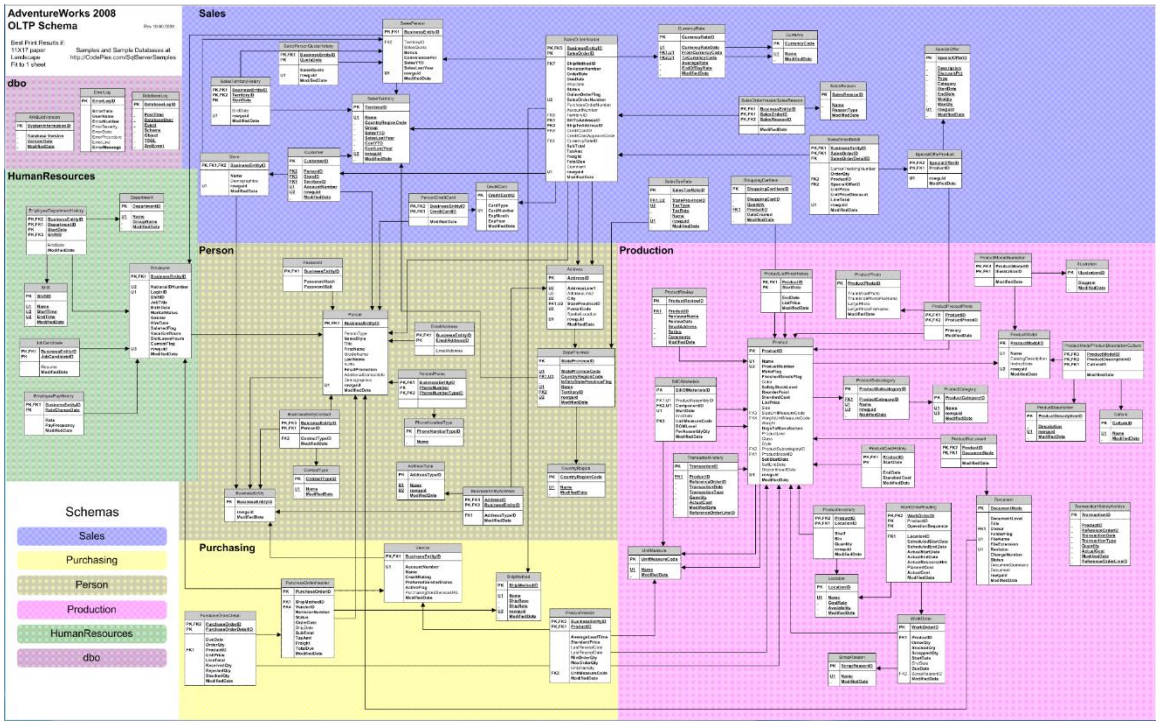
---

You may also have Products that you sell.




As you can see the data model can easily become very confusing.

## Entity Relationship Diagram




In the examples above the architect created a firm relationship called a (Foreign Key Constraint) between the tables for the relationships. The Foreign Key Constraints (relationship links) are the lines between the boxes in the diagram. Which means, you can use a software utility program that will create a picture of a data model for you.

The image above is called an Entity-Relationship Diagram[10] (ERD) or Entity-Relationship Model. It's like a roadmap of your database. That's the good news and very helpful in understanding the structure of your database.

The ERD above depicts the table relationships from the Microsoft SQL sample database, AdventureWorks. Sample=Simple. Your enterprise database is probably much larger and much more complex.

Then of course there's the bad news;

[10] Entity-Relationship Diagram, is an abstract and conceptual representation of data. (dictionary.sensagent.com)

*An architect does NOT have to create a firm relationship between the tables in a database and many don't.*

That's why I describe the links (lines from box to box) between tables as *implied* relationships.

Without the firm relationships defined in the database an ERD tool is useless (no lines will be drawn between the boxes).

*It may only be tacitly understood that specific tables are related in the architect's design.*

To add insult to injury, a database can have no Foreign Key Constraints in the database and yet still be a valid relational database containing thousands of relationships between the tables.

*Your Entity Relationship Diagram may have no lines connecting the tables but it's still a relational database model.*

Confused yet? Let me quickly summarize what was just covered.

All data in a database is simply a bunch of lists of information called tables. Tables have columns just like an address book. Your address book would be a table (list) and Name, Address, City, State, Zip Code and Phone Number would be the column names in that table.

Each table may or may not be somehow related to another table (list). If tables are related, envision a line drawn from one table to the next. Hundreds of tables with hundreds of relations are your relational database, depicted like the images shown above.

*However, just because there's no line drawn between the tables DOESN'T mean there's no relationship.*

If a roadmap was missing the street you live on, that doesn't mean the road is non-existent. The road is there, it just wasn't mapped. That's the same with relationships between tables. They can exist even if there is no line (Foreign Key Constraint) on the ERD.

I am not trying to make excuses for the IT industry but there are many reasons a database can end up with missing or incorrect Foreign Key Constraints. Even though you may consider a database like this to be bad, it's not. I have decades of experience with relational databases and other than simple databases, I have yet to find a single enterprise level database that had all of the necessary Foreign Key Constraints.

Databases have had everything from completely missing Foreign Key Constraints, to having incorrect ones, and even a mix of both. This does NOT alter the software system's ability to access the database relationally for reporting.

---

*It's very simple for a developer or query writer to create table relationship without a Foreign Key Constraint.*

---

Just like you know you can drive down the street you live on even if the street's not on a roadmap.

Developers, data miners and report writers can create their own relationships between tables as needed simply by adding a JOIN statement to a query. The JOIN creates the *implied* relationship between tables as I mentioned before. It's actually more like a forced relationship than an implied one. Implied indicates it's a suggestion. However, in an SQL query a JOIN is a forced relationship whether it's valid or not.

---

*Frequent users of a database know the relationships between the tables even if they're not documented in the database with a Foreign Key Constraint.*

---

**Data Integrity**

Foreign Key Constraints are not only used to determine relationships for reporting. They are also used to maintain Referential Integrity[11] when data records are added or removed from the tables. Primarily used by Database Administrators[12] (DBAs) to maintain data integrity, constraints on a large referential integrity tree are difficult to maintain.

---

[11] Referential Integrity, requires that if a value of one attribute (column) of a relation (table) references a value of another attribute (either in the same or a different relation), then the referenced value must exist. (en.wikipedia.org, en.wikipedia.org/wiki/Referential_integrity#Declarative_referential_integrity)

[12] Database Administrator (DBA), is a role usually within the Information Technology department, charged with the creation, maintenance, backups, querying, tuning, user rights assignment and security of an organization's databases. (techopedia.com, techopedia.com/definition/1187/database-administrator-dba)

*Referential Integrity is not important for someone just reporting on data because reporters only read data.*

As shown previously in the ERD, the architects have converted your data into a relational database form, using terms like Normalized, De-Normalized, Reference Data, Foreign Key Constraints, Etc.

They have even taken some of your data elements and changed their names. Something you may have called *Quantity* could be changed to *QAmt*. Why? Could be anything from shorter to type, to, fits on the screen better or just easier for them to remember.

*Sometimes database table and column names are so cryptic it's impossible to figure out what they even mean.*

There are no common reasons as to why architects do anything when designing a database.

## Normalized Data

When structuring a database, the architects can use Normalization[13] or Denormalization[14]. Which data structure to use with what data depends on many factors beyond the scope of this document. Databases can also combine both structures and many do.

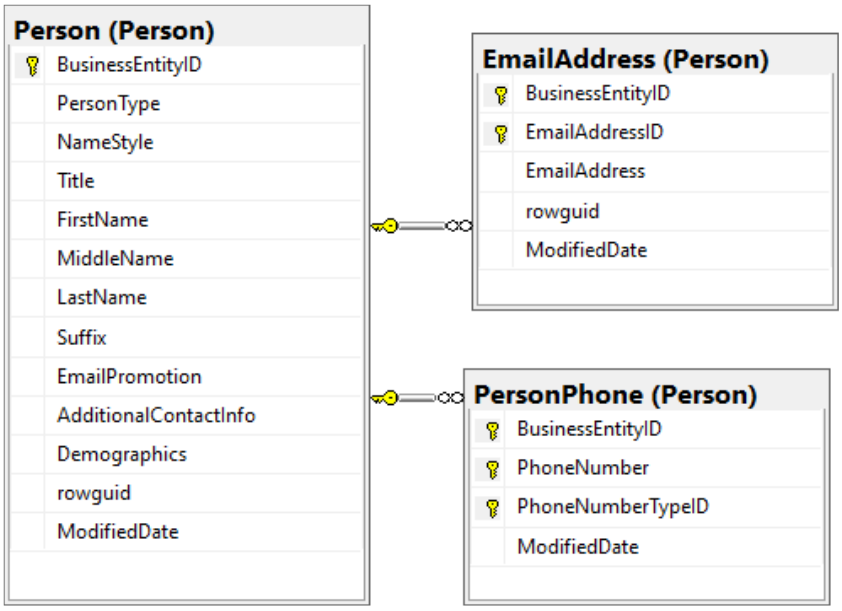
Normalization is a fancy way of saying that your data has been broken up into reusable pieces to reduce the duplication of values. For example, rather than having the same State or Providence field repeated over and over again in a contact list, the architect would create a State Table in the database. This table would only have one record per State and link to a contact list by the zip code.

Which method is better? It depends on who you speak to, there's no consensus for the best style. On the internet the debates on this topic are endless. Each have their pros and cons. However, if your data is stored in a relational database then its original design was mostly commonly created using a Normalized structure.

[13] Normalization, is the process of reducing redundancies of data in a database. (en.wikipedia.org, en.wikipedia.org/wiki/Database_normalization)

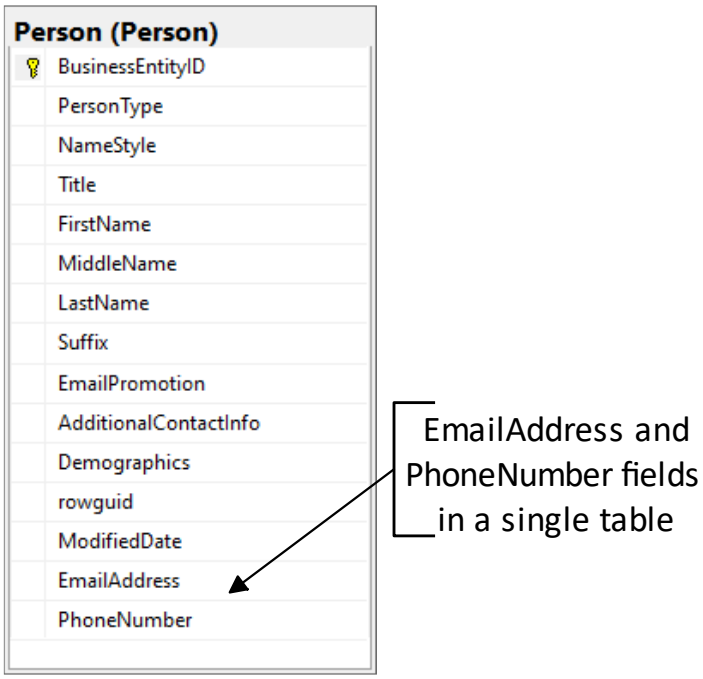
[14] Denormalization, is the process of trying to improve the readability and/or performance of a database. (en.wikipedia.org, en.wikipedia.org/wiki/Denormalization)

**Normalized:**




This structure would have only one row for a Person's name and many rows for multiple Email Addresses or Phone Numbers.

**Denormalized:**




This structure would have multiple Person records for every Email Address and Phone Number. Thereby, duplicating the Name fields over and over again.

The following shows some of the differences:

| Reason | Normalization | Denormalization |
|---|---|---|
| Creation | Used to remove redundant data from the database and to keep data consistent. | Used to combine multiple table data into one so that it can be queried quickly. |
| Purpose | Mainly focuses on reducing data redundancy and clearing unused data. | Used to achieve faster execution of the queries through less database reads, also used to make reporting easier. |
| Number of Tables | Generally, more in number. | Like data is commonly integrated into more tables. Data is duplicated. |
| Memory consumption | Possibly uses better optimized memory, indexes and has a smaller internal data payload hence using less memory. | Denormalization may not be able to benefit from indexes and almost always introduces larger internal payloads introducing some waste of memory. |
| Data integrity | Always maintains data integrity. Adding or deleting data from the table will not create any mismatch in the relationship of the tables. | Does not maintain any data integrity. |
| Why | Is generally used where number of insert/update/delete operations are performed and joins of those tables are not expensive. | Denormalization is used where joins are believed to cause performance issues and frequent queries are executed on the tables. |

The most common reasons to use a normalized structure, are data integrity and to reduce data redundancy. The most common reasons to use a denormalized structure, are a perceived faster query times and convenience.

An architect's main concern when creating a database is its data integrity. The last thing on their mind is reporting. Reporting is something you do after the system is running and data has been stored properly in the database.

**How normalization creates reporting problems**

Normalization is good. It maintains the data integrity of your database and that's important.

Yes. Data integrity is particularly important.

A database is designed in a normalized fashion to maintain data integrity and then queries are created later to denormalize the data for reporting purposes. Sometimes complete denormalized databases are created called data-warehouses[15] or reporting databases.

---

***The simple fact that your data is normalized in your database, is the main reason you need an IT person to write a query for you to get your data on a report.***

---

Let me re-phrase that; The fact that your data is *relationalized* (scattered between many related tables) is the reason you need an IT person to write a query for you to get your data out on a report.

This sounds backwards but when data is denormalized for reporting it's also de-relationalized (combined from many tables) to make it normal for a human to make sense of.

I know, denormalized means normal. Weird.

## Strange Data

By now, I hope you're starting to see a pattern.

[15] Data Warehouse, is constructed by integrating data from multiple heterogeneous sources that support analytical reporting, structured and/or ad hoc queries, and decision making. (tutorialspoint.com, tutorialspoint.com/dwh/dwh_data_warehousing.htm)

*Your original business data was transformed into some strange relational database form.*

Strange for you maybe, but not for the IT professionals. This format is perfectly normal for them. It does, however, explain why you need them to get your data back out of the database in a form that you can report on.

Many IT professionals don't understand why there's a reporting problem at all because you can just ask them to write a query for you.

*They have yet to figure out that IT is a bottleneck to the business.*

Which, by the way, is another part of the problem. IT doesn't recognize this as a problem, so they've never seen a need to create a better solution. Sort of like a gas company spending time on perfecting solar panels to reduce the dependency on gas. Why would they?

The problem isn't that the data is in a strange format in the database. The problem is that the IT industry has not given users an easy way to get the data back out in the same business form that it was understood before it was saved in the database. To them writing an SQL query is the easy way.

*"Just connect to the database, write an SQL query and get whatever data you want."*

Maybe some power users[16] can write queries but most business users are not capable, nor do they want to be. IT departments write queries because they know how all this relational stuff works. That's why you need them to help you access your data. (Plug: DSF does this for the business user without IT.)

**Data Warehouse**

One solution to the strange data format is the creation of a denormalized data warehouse. This can be called many things, but no matter what it's called, it's nothing more than a second database that holds a sub-set of data elements from your original database using a series of SQL queries.

[16] Power user, may not be capable of computer programming or system administration, but is rather characterized by the competence or desire to make the most intensive use of computer programs or systems. (en.wikipedia.org, en.wikipedia.org/wiki/Power_user)

The data warehouse will have a new data model structure that IT believes will be easier for you to report from. Did I say, "sub-set?" Yes. IT decides which data is important for you to report on, but you will still need an SQL query to retrieve it for reporting.

There are many excuses as to why IT needs to create a special secondary database for reporting. Depending on the organization the reason and the process varies greatly.

A few excuses in no particular order, security, speed, load on production server, merge of multiple databases, load on IT, etc.

I use the word excuses because if you ask IT why you must report from a secondary database (many times stale or a day old) they will start to ramble on with reasons which mostly sound like excuses.

There are many valid reasons for creating a separate reporting database like consolidation of data from multiple locations, data transformations, etc. (I would even argue that many of these aren't valid reasons either, but that's for another time.)

Unfortunately, once a business goes down the road of having a special database to report from, they make it the only place users are allowed to get data from even if it would be faster to just get the data from its original location.

The majority of the reasons I have run into are unfounded. Remember the 80/20 rule? One poorly written query that is suspected to slow a production database can send an entire company down that road to creating a reporting database that EVERYONE MUST now use.

80% of the simple, easy, or non-disruptive reporting queries users may want to run on the production database, now have to point to a stale reporting database or worse. The processes that move the data from production to the secondary database have not been updated with the needed data so users have to wait until IT updates them. In some organizations this can take weeks to accomplish.

**Quick story:**

I encountered a large company that had so many requests for new data columns to IT, that they made a company policy to only release new columns into the reporting database every 2 weeks. Period. If users need any new data elements they must wait. Good for IT but bad for business.

When I inquired as to why users were not allowed to report from production databases, I was told it was for security reasons. Security? Yes. We have sensitive data that some users are not allowed to have access too. Okay, so how do users who <u>are</u> allowed access, report on

the data? It's moved to the secondary database. One big database? Yes. How do users connect to the secondary database? Wait for it…All users use the same single read only userid and password regardless of who they are! No security separation at all! Anybody in the company can report on any data in the secondary database.

This company created a delay for users wanting to report on 80% of the data under the guise of security, when in fact they were only looking to maintain data integrity. Since everyone had the same access to the secondary database the data was by no means secure! They could have just as easily locked down a production database for the 20% of the data they felt was sensitive and let the rest be reported on with a read only access user. No 2-week releases, no waiting, no maintenance, etc.

Other than the waste of IT resources, nothing would matter in this story except for the fact that the users were actively complaining about not having timely access to data elements they needed to run the business! IT was the bottleneck to the business under false pretense.

## Relationships

Understanding the table relationships in your database is important in your goal of getting access to your data. Just like knowing what streets to use to get back home. When a user like yourself needs to access data for a report, not only do you need to know the relationships between the tables but you will also need to know how to write an SQL query to make it all happen!

---

*As the business expert, you have a good chance of knowing the proper table relationships in your database because they should normally follow closely to the structure of your business context.*

---

Architects know all the technical jargon, but you know the proper construct of the core data elements. Remember, it was your data in the first place based off your business model.

You've heard this before:

*"Just learn SQL. It's in simple plain English. Buy a custom report generator like Crystal Reports or Tableau, connect to your database and you're good to go. Create whatever reports you need."*

Here's the query that's needed to access the Microsoft sample database to get some basic Vendor information:

```
SELECT Vendor.Name AS VendorName, Addresses.AddressLine1, Addresses.AddressLine2,
       Addresses.City, Addresses.PostalCode AS Zipcode,
       StateProvinces.StateProvinceCode AS State,
       CountryRegions.Name AS Country,AddressTypes.Name AS AddressType, People.Lastname,
       People.Firstname, PeoplesPhones.PhoneNumber,
       PhoneNumberTypes.Name AS PhoneNumberType
FROM   Purchasing.Vendor Vendor
INNER JOIN Person.BusinessEntityAddress BusinessEntityAddresses ON
           BusinessEntityAddresses.BusinessEntityID=Vendor.BusinessEntityID
INNER JOIN Person.Address Addresses ON
           Addresses.AddressID=BusinessEntityAddresses.AddressID
INNER JOIN Person.StateProvince StateProvinces ON
           StateProvinces.StateProvinceID=Addresses.StateProvinceID
INNER JOIN Person.CountryRegion CountryRegions ON
           CountryRegions.CountryRegionCode=StateProvinces.CountryRegionCode
INNER JOIN Person.AddressType AddressTypes ON
           AddressTypes.AddressTypeID=BusinessEntityAddresses.AddressTypeID
INNER JOIN Person.BusinessEntityContact BusinessEntityContacts ON
           BusinessEntityContacts.BusinessEntityID=Vendor.BusinessEntityID
INNER JOIN Person.Person People ON
           People.BusinessEntityID=BusinessEntityContacts.PersonID
LEFT JOIN  Person.PersonPhone PeoplesPhones ON
           PeoplesPhones.BusinessEntityID=People.BusinessEntityID
LEFT JOIN  Person.PhoneNumberType PhoneNumberTypes ON
           PhoneNumberTypes.PhoneNumberTypeID=PeoplesPhones.PhoneNumberTypeID
```

Power users can write this once they learn the database model and the SQL Query Language, but this looks complicated. The words are in English but it's a language all to itself. It's definitely not simple English. Most systems require you write an SQL query to get to the data for reporting.

As you can see, SQL is not that easy, so you're back again requesting that IT build reports for you. Don't give up yet. SQL is not easy, but once you learn a few basics it's not that hard either.

As you will see, having IT write your queries for you may also not be the best solution. Sometimes it's better to just do it yourself.

## Conclusion

The take away from this section is simple; the problem with accessing your data easily for reporting, is that IT professionals have taken your business data that you used to understand and wrapped into a world of technical jargon only they understand. The relational model. It wasn't done maliciously. They did it to maintain data integrity and good database structure.

Now you just need to learn a few SQL tricks and understand some of the relationships in your database to get the data you want on your report.

*Time to turn all of this computer jargon into something useful!*

# CHAPTER 3

## The Basics

What's happening?

We now know that data in a database can be problematic. Software vendors can change the structure, we may not be able to detect the table relationships and it can be very difficult to write an SQL query to get the data we need (even though it's in plain English).

The only way to get data from a relational database is to use SQL. Sooner or later someone somewhere will write an SQL query to pull the data from your database. There's no getting around it.

> *Okay! Calm down. Yes. Debatably there are other sciences that access a database without SQL. Since you probably don't have a PhD in computer science and you're most likely running your reports against a relational database, let's stick to what 99% of the world is doing. Writing SQL queries.*

When your database holds tens of thousands of data elements there are billions of possible SQL query permutations to access your data.

---

*There's no way that IT can create every view of the data ahead of time that you may need to run your business.*

---

As you need new data elements for reporting, you're forced to contract an IT professional to get the data for you. You're spoon fed your data as needed. Usually having to pay for it with time and/or money.

---

*Users have learned that waiting for IT is not always the best solution for the business.*

---

The SQL query that IT is creating for you may be imbedded in a program, created as view in your database or written directly in your reporting tool. Regardless of the method, an SQL query is ultimately sent to the database to extract the data.

Whether you wrote the query or IT did it for you, it ultimately creates the business context that you will use on your report. It gives your data purpose.

## Speed

Once the query is sent to the database the waiting starts.

Let's face it, we've all been there. Waiting for our report to load data. The proverbial, never ending hourglass or progress bar.


Developers have gotten pretty fancy over the years showing users better ways to wait.

Even as far as to wait forever! This called an infinite progress bar.


This process will take a while and we have no idea how long that will be. Please be patient.

Like watching toast brown or waiting for water to boil, it always seems to take longer than you think it should.

---

*My report used to load faster what happened?*

---

So, what's actually happening? A lot!

First, let me reiterate what I said above:

> *"an SQL query is ultimately sent to the database to extract the data."*

This document is not meant to teach you everything about the SQL Query Language. There are many places to learn SQL online for free. [learnsqlonline.org](learnsqlonline.org)

However, simply learning SQL syntax[17] is only the beginning. There are hundreds of idiosyncrasies to think about if you want your query to perform well. Sometimes they differ from database vendor to database vendor (Microsoft SQL vs Oracle).

The same query can run differently, even on the same system, under different circumstances.

---

*The performance of a query can be unpredictable.*

---

SQL has a syntax standard but over the years database vendors have also created their own specific rules and features.

It's easy to write simple queries, but most reports require complex JOINs that only an SQL expert will know how to do. Even most programmers don't understand the intricacies of SQL query performance tuning as they are not database experts. Many Database Administrators (DBAs) may know how to maintain and tune your database but they too may struggle with SQL query performance tuning.

Also, many report queries are not written by SQL experts. Because of this, they may initially perform well but as your data grows in size, over time, they become slower and slower. Inexperienced query writers no nothing about making them efficient or scalable[18].

---

*Many users running reports seem to live with the fact that it takes a long time to load.*

---

A DBA can "tune" the database to make a specific report run better or your organization can create a reporting database (separate database specifically used for reporting that is a subset of the production database) to make the report "appear" faster.

I say "appear" faster because the data in a reporting database is usually stale. The slow query that is used to move the data from the production database to the reporting database is frequently run at night. This removes the load on the production database during the day when you want your report, but it's yesterday's data. This is fine for some organizations but in today's world that may not be the best solution.

---

[17] Syntax, the rules or structure as to how a phrase is understood. (merriam-webster.com, merriam-webster.com/dictionary/syntax)

[18] Scalable, is an attribute that describes the ability of a process, network, software or organization to grow and manage increased demand. (techopedia.com, techopedia.com/definition/9269/scalability)

The separate database also creates other issues like, security, licenses, etc.

Why doesn't the DBA just "tune" all the reports?

They can, as long as they're not random or ad-hoc reports, which are the ones you're probably looking for.

## Some History

Years ago, a report from a database was a planned process. A "Daily Journal" from an accounting system was a planned report at the onset of the project. The database was structured to be conducive to executing the report and the report probably ran very fast (in relation to its era). A computer system had a hand full of "canned" or predetermined reports. If a user needed a new report then it was a development project to create it like any other program change. This could take months to complete.

As organizations evolved, businesses wanted access to more and more data, especially for reporting. The randomness of an SQL query fell right into that role. Months of development waiting for a report could now be done in weeks or even a few days. However, you still needed to be a technically skilled expert to write an "efficient" query.

---

*Organizations created whole departments to write queries specifically to handle the reporting demands of the business.*

---

As user's became more tech savvy and the demands for reports continued to increase, weeks or even days was still too long for the business to wait. Some business didn't have the budget or man power to just have people writing queries. So evolved, the power user.

These power users can write their own SQL queries to pull data from the database. Many do this to populate Excel so that they can do further analysis on data or just by using a third-party tool like Crystal Reports for reporting. However, a power user is not a data access expert. They can write queries that "work" but their queries may also not be efficient or scalable. Most power users have a title like Business Analyst.

## Business Intelligence

Somewhere in the middle of all this, Business Intelligence (BI) was born.

*Business intelligence is a technology-driven process for analyzing data and delivering actionable information that helps executives, managers and workers make informed business decisions.*

Initially, BI tools were used by IT professionals to produced visual dashboards and reports for business users primarily for analytics. Why IT professionals? Because BI tools still require the creation of an SQL query to access the data.

BI claims that data is self-service and enables the business user to query BI data. Once BI is setup by an IT professional (renamed to a Universe Designer), who creates a series of Business Objects, the user can easily select data elements from the (Business Object Universe[19]). They give your data its business context.

*BI requires users to know up front what data they think is relevant to the business, so that their data universe can be created.*

Big business can afford to implement a full-scale BI solution and even then, as mentioned before, not always successfully. More than 80% of businesses simply have a relational database they write SQL queries against for reporting, with no need for sophisticated analytics.

BI was partially invented because of the problems you learned in the previous section, but there is more too BI than meets the eye. Joining multiple databases, analyzing data, some ETL, graphical representations, drag-drop features, drilldown, etc.

The IT industry as blurred the idea of a BI solution and a self-service dashboard tool. A BI solution is a big deal that will include a dashboard tool as part of its solution.

*Your organization would have invested a lot of time and effort to create their Business Object Universe to give context to the business data for you.*

They also would have made sure that there was an entire support organization surrounding the BI solution to help you succeed.

[19] Business Object Universe, is the semantic layer that resides between an organization's database and the end user but more importantly, it is a business representation of your data warehouse or transactional database. (alteksolutions.com)

Purchasing a self-service dashboard tool and connecting to a database for reporting allows power users to create some analytic dashboards, but it's not a BI solution.

---

*A self-service dashboard tool will have the same issues accessing the data from a database as any other reporting tool.*

---

There's a place for a BI solution in this world. People pay a lot of money for it. But it can be complicated. If that's what you need, great. There's no "right way" of doing anything with data. If it works for you then it's the "right way".

However, I'm guessing you're here because you are in the 80 percent group. You're reporting from your database without the need for a large-scale BI solution. If you're here because your BI reports are running slow, I suggest you contact your Business Objects Developer or the BI support group within your organization. They are the people that wrote the SQL queries to access the data from your database.

If you're here because your self-service dashboard tool is expecting you to know how to write SQL or is running slow, then you're in the right place.

## Thinking About Data

This is not meant to be an exhaustive technical document on SQL but I will cover what I have found to be common issues with report performance.

You now know that an SQL query gets the data from the database for your report. I also said that a DBA can tune the database to make the report run faster. I further said that the same query can run differently even on the same system under different circumstances (Regardless of the data provider).

A slow report query can be slow for thousands of reasons. There is no single silver bullet or magic solution. Every scenario is different. Your data needs to travel from your server to some display. You could be running on a virtual machine, a web server, a desktop computer or any number of methods. Each one has its pros and cons.

---

*Just remember, distance or network speed can matter.*

---

Your database could be performing super-fast and your network is slow. Your query could be fine but your server is overloaded or undersized.

## Database Tuning

Tuning a database can involve hundreds of different processes and are different between database vendors.

Talk to your DBA about:

- the physical database server hardware (CPU, Memory, etc.)
- server load when you're running your reports
- maintenance plans to mitigate Index Fragmentation (See Appendix)
  This can have a huge impact on report queries that use indexes

Also speak to your Network Administrator if you think your network access is lagging.

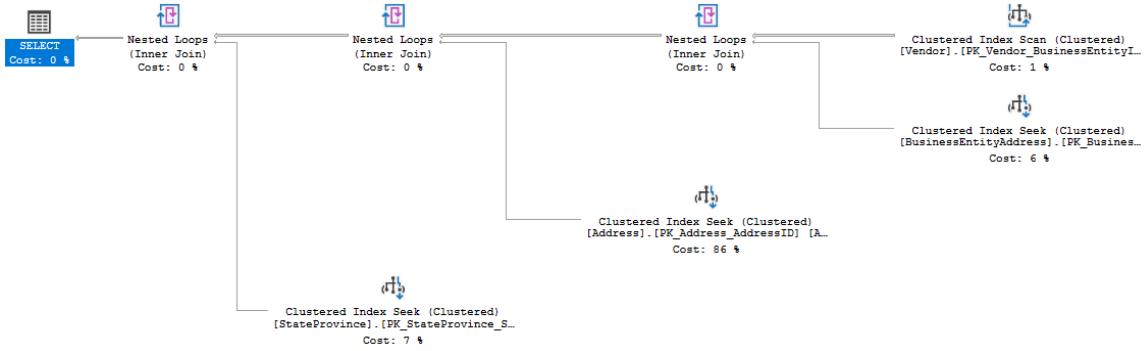
If all this is good, then let's talk about queries.

## The Query Optimizer

Every database provider has their own Query Optimizer[20]. The optimizer takes your query and tries to determine the best execution plan[21] for the server run to retrieve your data. The execution plan are the exact steps the server will take to most efficiently return your data.

If you needed to get something off the roof of your house, a good plan could be to get a ladder and climb up to get it. Another plan could be to climb a nearby tree and jump over to the roof. Maybe not the best plan, but a plan none the less. You get the idea. Some plans may be better than others. If you don't have a ladder then maybe the tree method is your only plan.

There are graphical tools available to view a queries execution plan to determine what actions the server has taken or will take to retrieve your data.




The optimizers generally do a good job at determining the most efficient way to access the required data. I would leave reading and evaluating an execution plan like the one pictured above to your DBA.

Optimizers are not perfect and can sometimes get confused. This makes them not totally reliable.

---

[20] Query Optimizer, analyzes a number of candidate execution plans for a given query, estimates the cost of each of these plans and selects the plan with the lowest cost of the choices considered. (Nevarez)

[21] Execution Plan, is the list of steps your database server will take to retrieve your data. (microsoft.com, docs.microsoft.com/en-us/sql/relational-databases/performance/execution-plans)

*Optimizers are finicky and can use a poor execution plan for no apparent reason.*

I have direct experience in upgrading a database from one version to another that brought a system to its knees. The new version had a new optimizer which decided to change the execution plans for the same queries that were running fine in the previous version.

*"Some queries may be negatively affected by the new optimizer. We suggest you re-evaluate and modify your queries to perform correctly with the new optimizer." – Database Vendor*

Here's the big question Mr. Database Vendor. Which queries? I have hundreds of queries! I had to re-test the entire system to see which queries needed to be tweaked to run faster with the new optimizer!

Unfortunately, there's not much you can do about the query optimizer's execution plan. There are things call HINTS that you can use to help steer the optimizer, however my experience suggests to just leave it alone unless you really know what you're doing.

*The query optimizer is more likely to randomly decide to change the execution plan on a complex query than it is on a simple one.*

Without going into too much detail, the optimizer changes its mind based on what it believes to be the best fit at the time. Something called data statistics[22] are what optimizers use to make this determination. If the statistics on the data changes then the optimizer may change the execution plan.

Statistics are a sampling of data values that the optimizer uses to mathematically determine the best course of action for the execution plan. As data changes in the database tables, the optimizer may re-sample the values and make a different determination. This is why is seems like and execution plan changes randomly.

Just let the experts worry about all this.

[22] Statistics, information about the distribution of values in one or more columns of a table. (microsoft.com, docs.microsoft.com/en-us/sql/relational-databases/statistics)

*Keep your queries as simple as possible and it's highly unlikely you'll have any issues with the optimizer.*

## Query Tuning

Many times, reports start out fast and then slow up over time because the queried data grows in size. The more data the longer it takes a query to run.

If you started out with a few thousand records and now your requesting a few million, it's going to be slower.

*If IT wrote the query for you and you have no visibility or ability to change it then you're at the mercy of IT to fix it for you.*

Let's break down an example query.

If a report has 2 columns of vendor data. The query could look something like this:

```
SELECT Name, AddressLine1
FROM Vendor
```

Simple. The report sends the query to the server and it returns the resultset to your report. Think of a resultset as a block of rows and columns like you see on an Excel spreadsheet. It's nothing other than a list of values. The SELECT clause chooses the data columns you want FROM the TABLE named Vendor.

As you can see SQL is in plain English, as long as you're speaking SQL English!

| Resultset | |
|---|---|
| Name | AddressLine1 |
| Bob | 1 Main St |
| Bill | 57 Oak St |
| Diane | 8976 Alpine Rd |
| Frank | 437 Park St |
| Grace | 98 Wayne Ave |
| Harry | 12 Parson Blvd |
| Jim | 789 Warren Ter |
| Joan | 36 Barker Blvd |
| Kathy | 27896 Maple Pl |
| Larry | 11 New England Ct |
| Max. 25 Chars | Max. 50 Characters |

### Query Tools

Before I continue, some of you may not even know how to run a query on your database. If you have a login to your database and know a

few things like the server's name, you can use a utility program to send queries to your database. Microsoft SQL Server utilizes SQL Server Management Studio (SMSS). It's a free download from Microsoft and it's the same tool your DBA uses to manage the database. Your database provider may have a different tool. There are also third-party tools like Toad that are available for almost any database provider.

These tools allow you to type in an SQL query and then send it to the server for execution which will return the resultset of that query.

They also may color code your query phrase to make it easier to read and maintain the proper syntax.

When you first ran your report 2 years ago, using the simple query above, you had 10 vendors and it ran fast. It was moving 10 rows by 2 columns or maximum of 750 bytes of data.

---

*Every character is a byte of data.*

---

Bytes add up to kilobytes which add up to megabytes and so on. (Ignoring compression and such for the moment, this example is only moving 171 bytes) Theoretically, what would happen, if you had 100 million vendors now?

Yes. This is an absurd example and you will never have 100 million vendors. That's not the point. The point is what would actually happen?

The database server, your network and your reporting tool would need to find, move and load 100 million data rows. That's a long list! 100 million rows of 75 max. characters or possibly 7.5 gigabytes of data. (Average example data values above would be 1.7GBs) Which of course takes a lot longer than a few hundred bytes.

Envision a little man carrying rocks in a wheelbarrow back and forth between your server and your report.

He can only carry so many rocks in the wheelbarrow on each trip. Determining how many rocks he can carry on each trip is feasible but technically way past this discussion. If you're asking for a lot of rocks it's going to take a lot of trips. It's simple logic.

Meanwhile, think of the database server as the quarry where the little man is getting the rocks. The quarry needs to find which rocks the little man is asking for and is also busy handling other requests. Sometimes the little man may have to wait to get his wheelbarrow filled up.

There are many other factors that determine how long it will take to transfer 7.5GBs of data from your server to your report. You only need to recognize that more takes longer. Every process along the way has more work to do.

---

*Visualizing what is actually happening when your report makes a data request with a query, will help you understand why some queries run faster than others.*

---

If you had the same number of vendors but asked for more data columns, you're asking for more rocks also:

```
SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor
```

This query is asking for 3 times as many columns of data which could be twice as many bytes of data. More rocks, more work for the little man. Even though you still may only have a few hundred vendors. Shorter list but wider in size. This query will run slower than the first because it's asking for 177 more bytes of data. 10 rows of 162 characters or maximum of 1,620 bytes of data.

| Resultset | | | | | |
|---|---|---|---|---|---|
| Name | AddressLine1 | AddressLine2 | City | State | PostalCode |
| Bob | 1 Main St | Apt #4 | Roanoke | VA | 24102-3452 |
| Bill | 57 Oak St | | Bumpkin | MS | 38602 |
| Diane | 8976 Alpine Rd | Suite 78 | Warren | NV | 89010 |
| Frank | 437 Park St | P.O. Box 99 | Bishop | WA | 98002 |
| Grace | 98 Wayne Ave | | Florida | MA | 01002 |
| Harry | 12 Parson Blvd | Second Floor | Candy | MI | 48004 |
| Jim | 789 Warren Ter | | Cave | IN | 46001-2836 |
| Joan | 36 Barker Blvd | P.O. Box 576 | Pleasant | CO | 80011 |
| Kathy | 27896 Maple Pl | | Vernon | VA | 24103 |
| Larry | 11 New England Ct | Attn: Larry | Small | NY | 10114 |
| Max. 25 Chars | Max. 50 Characters | Max. 50 Characters | Max. 25 Chars | Max. 2 Chars | Max. 50 Chars |

You can argue that the second query is not any slower than the first (even though it's requesting twice as much data), and it's probably not any slower. The data sizes in this example are insignificant. Not for a human to determine anyway. A human can't tell the difference between nanoseconds and there are other considerations like data caching, compression, etc. However, you can't argue that the database server, the network and your reporting tools are doing "more" work. The question is; When does "more" become a problem.

---

*I can guarantee you, that querying a terabyte of data will be slower than a gigabyte of data all day long.*

---

So, "more" does have an impact on speed. But, when does it matter?

Many software developers forget this. They write a query that they think is fast. It only takes a tenth a second to run. No big deal. Then it somehow works its way into a routine that gets called multiple times in some loop. Runs great in test. Runs great in production. Then about a year later clients are calling with complaints that the system is slow. Over time more data is accumulated. That small tenth of a second starts totaling up to minutes or even hours!

*Don't dismiss the "more".*

More matters when you feel it. Your report runs slower. Since you can't determine when more will create an issue, if ever, it's best to only request the data you need. Nothing more.

## Filtering Data

If you had 100 million vendors would you really need to see them all on your report? Probably not. Say for example, you only need to see the vendors located in Virginia. Which would hypothetically limit your list to 100 records instead of 100 million.

You could go into your reporting tool and filter them out. Like Excel, most reporting tools support filtering display results. You save the filter with your report so the next time it loads, it will be faster. Most likely not!

Did the reporting tool change the query that you got from IT? Are you running from a database view[23] as the source to your data? If your reporting tool is not changing the source query and it's only applying a display filter, it won't be any faster retrieving data from the server.

If the query didn't change then what's the little man doing? He's still moving all of those rocks! Just because you aren't using them doesn't mean he doesn't have to still move them. If the query was not changed, then you never told the little man to move less rocks.

*The only way to have the little man move less rocks, is to request less data.*

An SQL WHERE CLAUSE does just that. A WHERE CLAUSE tells the server to only give the records you want to the little man. Limiting the number of data rows returned. He's moving less rocks.

```
SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor WHERE State='VA'
```

This filter is executed on the server-side not the client-side (your reporting tool).

Server-side simply means the action is executed on your database server. Client-side means the action is executed on your display rendering device. If you are running Excel on your desktop then your desktop is the client machine. If you are running

[23] Database View, is a virtual table whose contents are defined by a query. (microsoft.com, docs.microsoft.com/en-us/sql/relational-databases/views)

some web reporting tool then it could be the machine your tool is hosted on. Filtering data on the client-side does not help speed up your query on the server.

| Filtered Resultset | | | | | |
|---|---|---|---|---|---|
| Name | AddressLine1 | AddressLine2 | City | State | PostalCode |
| Bob | 1 Main St | Apt #4 | Roanoke | VA | 24102-3452 |
| Kathy | 27896 Maple Pl | | Vernon | VA | 24103 |
| Max. 25 Chars | Max. 50 Characters | Max. 50 Characters | Max. 25 Chars | Max. 2 Chars | Max. 50 Chars |

As you can see, this query with the WHERE CLAUSE only returns 2 rows. Filtered queries move less data.

---

*Less to move means faster reporting.*

---

**Keep the WHERE clause simple**

I've seen situations where one small change to a complex WHERE clause increased a query by more than 50%. Here's the example:

```
SELECT BusinessDate, TransactionID, Amount
FROM TableA
WHERE BusinessDate < TodateFromyyyymmddhh24miss('20200502000000')
```

Putting the WHERE check value in the same format as the table data instead of converting it with the function, made a dramatic difference to the optimizer.

```
SELECT BusinessDate, TransactionID, Amount
FROM TableA
WHERE BusinessDate < '2020-05-02'
```

There's overhead for the server to execute the date conversion function. I was stumped as to why this would be slower since it only needed to convert the date once.

It made sense to me that I would get a performance hit if the statement were reversed.

```
WHERE TodateFromyyyymmddhh24miss(BusinessDate) < '2020-05-02'
```

In this case the date conversion would need to run for every record.

Apparently, the version of the query optimizer from the data provider was not smart enough no know that it only needed to convert the date once. It created a worse execution plan and made the query run dramatically slower.

Again, the only way to know how well a query works is to run your own test. Even if you think it should be good.

*Keep WHERE clauses as simple as possible, whenever possible.*

## Indexing (or lack of)

If you add a filter using a WHERE CLAUSE the little man will move less data. That's good. But it can take some time for the server to figure out which records to send and which ones to skip. This is where an index can speed the process.

I previously stated that a DBA can "tune" the database to make your report run faster. Adding an index to the data you are querying is one way they can do that.

*Indexes are like medication. A little can do a little good but a lot can be really bad. It's a balancing act.*

A DBA will not (or should not) create tons of indexes just because you want to add different filters to your report. You need to have a discussion with your DBA to determine what's the best balance. The same way a doctor determines how much medication you should be taking.

Like everything else related to databases, indexing can get quite confusing. (Clustered, Non-Clustered, Binary, B-Tree, R-Tree, etc.) There are many types of indexes. For now, all you need to know is that when filtering data; Index=Faster, No Index=Slower.

If there's no index on the data field(s) you are filtering then the server needs to do an INDEX SCAN or TABLE SCAN.

*Scans are bad on large amounts of data.*

Simply put, without an index, your data has no order to it. Therefore, the server has to read each record (SCAN) to determine if it's the one you want. One record at a time, until it reads every record. The more records it has to sift through the longer it takes. This can explain why sometimes a report runs fine initially and then gets slower over time.

If you went to the library to look for a book on "Virginia" and there was no order to the books, you would have to start at the beginning and go through every book until you found one. If you only needed one book you could stop there. You may be lucky and it will be in the first few books you look at. But it could also be the last book.

If you were looking for any book on "Virginia" you could stop as soon as you found the first one. If you wanted all records where State='VA', you would have to look at every book in the building to make sure you had them all!

If your library only has a few hundred books it will take a lot less time than if there were thousands or hundreds of thousands. Scanning takes time, which is the same way your database server will have to find your data when it has no index.

**Sargability**

Just because you add an index to your data does not always mean the server will use it either. A query can be non-sargable.

---

*Sargable derived from **S**earch **ARG**ument **ABLE***

---

Trust me, I don't make this stuff up!

If a WHERE CLAUSE (filter) is not constructed properly it can be deemed non-sargable by the server's query optimizer and the index is ignored. It will fall back to an INDEX SCAN or TABLE SCAN. Looking at every record.

A good analysis of sargability and its effect on query performance can be found at www.sqlshack.com. (Erkec) There are many scenarios that determine if a WHERE CLAUSE will be sargable, however the basic guideline is a sargable statement will usually have field values on the left of the operator by themselves, and expressions on the right side of the operator.

Sargable: (Index can be used)

```
SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor WHERE City LIKE 'Roa%'

SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor WHERE AddressLine2 IS NULL
```

Non-Sargable: (No index will be used)

```
SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor WHERE LEFT(City,3)='Roa'

SELECT Name, AddressLine1, AddressLine2, City, State, PostalCode
FROM Vendor WHERE ISNULL(AddressLine2,'EMPTY')='EMPTY'
```

Unfortunately, query optimizers from different database vendors will have different results even when utilizing the same query.

If you're unsure, just try to keep basic field names to the left side of your WHERE CLAUSE without any parenthesis wrapping it up in some function.

There are also graphical tools your DBA can use to view a queries execution plan to determine what actions the server has taken to retrieve your data.

The easiest way I've found, is trial and error.

---

*If one query runs faster than another, then use the faster one.*

---

## Time for JOINs

Simple queries are just that. Simple. What happens when a query gets more complex?

When queries need to get related data from multiple tables it's done with a JOIN clause.

---

*JOINs are what separate you from query experts and power users.*

---

You should know by now, that your data was scattered between many related tables in your database. Query experts know your database model and know how to put the data back together for reporting. So, can you. With just a little bit of knowledge you can get access to a lot of data.

There are 4 different types of JOINs:

- LEFT JOIN: Returns all records from the left table, and the matched records from the right table
- INNER JOIN: Returns records that have matching values in both tables
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
- FULL JOIN: Returns all records when there is a match in either left or right table

Most basic reporting queries only need a LEFT JOIN, so I will focus on that. A LEFT JOIN is the easiest to visualize hierarchically. Visually it's nothing more than an outline or a multilevel list that we all learned how to do in school.

1) Major
    a) Minor A
        i) Sub Minor1
    b) Minor B

i) Sub Minor1
ii) Sub Minor2

Windows Explorer displays the structure of your files stored on your computer in a similar way.

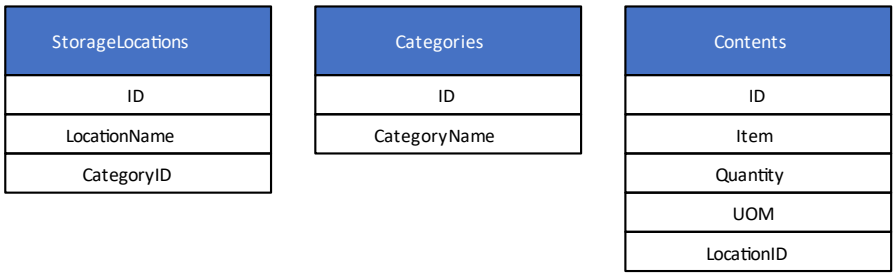
Let's use an example that everyone can relate to; Stuff in your house.

House Outline:

1) Storage Locations
   a) Garage
      i) Car
      ii) Hammer
      iii) Drill
      iv) Lawn Mower
   b) Refrigerator
      i) Beef
      ii) Chicken
      iii) Jalapeno
      iv) Minced Garlic
      v) Broccoli
      vi) Lettuce
      vii) Ketchup
      viii) Beer
   c) Spice Cabinet
      i) Salt
      ii) Pepper
      iii) Cumin
      iv) Cloves
      v) Chili Powder
   d) Pantry
      i) Tomato Paste
      ii) Peas
      iii) Chicken Soup
      iv) Onions
      v) Kidney Beans
      vi) Crushed Tomatoes
      vii) Diced Tomatoes
      viii) Beef Broth
      ix) Sugar
      x) Brown Sugar

Your database has a series of tables. A set of tables JOINed together can create a category or group of data elements. It creates data context. It gives the data a purpose.

I've converted the example House Outline above into a database. The database has 3 tables; StorageLocations, Categories, and Contents.




We want to make a meal. We have a chili recipe and we need to see if we have all of the needed ingredients.

Chili Recipe Ingredients:

- ground beef
- onion
- jalapeno
- minced garlic
- chili powder
- cumin
- green bell pepper
- crushed tomatoes
- kidney beans
- diced tomatoes
- beef broth
- beer
- tomato paste
- brown sugar
- salt
- pepper

We need a report to list all of the food items from our food storage locations.

What storage locations do we have in the StorageLocations table? We can check with a simple investigative query:

```
SELECT *
FROM StorageLocations
```

| StorageLocations | | |
|---|---|---|
| ID | LocationName | CategoryID |
| 1 | Garage | 1 |
| 2 | Refrigerator | 2 |
| 3 | Spice Cabinet | 2 |
| 4 | Pantry | 2 |

An SQL SELECT query requires a FROM clause which determines the main TABLE in which your data is to be sourced. From there, any extended or related data you want on your report will require a JOIN clause.

The '*' in the SELECT clause returns all columns in a table. Using a SELECT * is bad in many ways and should not be used in your final query.

---

*You should always list the names of the columns to be returned by your query.*

---

I use SELECT * here just as an investigation tool.

Even though there are many reasons why SELECT * is bad, the biggest for me are the future unknowns that will crush a query's performance.

Remember, the little man carrying rocks? SELECT * gets all columns from a table whether you need them or not. As time goes on columns in tables can and commonly will be added. Your query may be fast now, but if someone adds a couple of Image columns to store pictures to one of your queried tables it's game over!

Above is a simple query on a simple table.

---

*Many tables in a database will have a column that uniquely identifies their rows.*

---

During normalization of the data in the database unique identifiers (IDs) are frequently used to link the record to other tables.

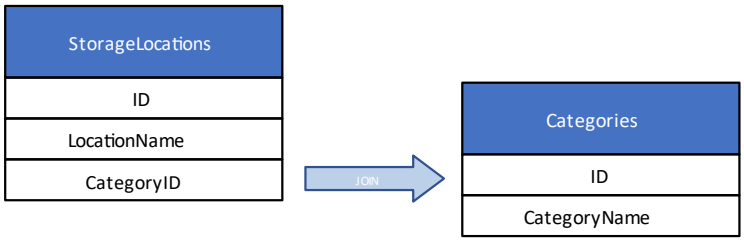
We need to determine which storage locations are food. In the StorageLocations table there is a column called 'CategoryID' which is the link to the Categories table.

Let's investigate the Categories table to see what's in it:

```
SELECT *
FROM Categories
```

| Categories | |
|---|---|
| ID | CategoryName |
| 1 | Junk |
| 2 | Food |

Based on the data values we see; we can safely presume that the Categories table is linked to the StorageLocations table with the 'CategoryID' column. We can add a JOIN clause to our query to reflect that link.




## Foreign Key Constraints

Since we are working with a relational database, the data is related between tables. Regardless how your database was originally designed, a JOIN creates a forced relationship between tables whether it's valid or not. No Foreign Key Constraint is required.

---

*Foreign Key Constraints are NOT required to create links between tables.*

---

```
SELECT SL.LocationName, SL.CategoryID, Categories.Name AS CategoryName, Categories.ID
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```

| JOINed Results | | | |
|---|---|---|---|
| LocationName | CategoryID | CategoryName | ID |
| Garage | 1 | Junk | 1 |
| Refrigerator | 2 | Food | 2 |
| Spice Cabinet | 2 | Food | 2 |
| Pantry | 2 | Food | 2 |

I did a lot in this query, so let's break the query down a bit.

```
SELECT SL.LocationName, SL.CategoryID, Categories.Name AS CategoryName, Categories.ID
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```

**Alias Names**

I introduced what's called an alias name for the StorageLocations table. This was done with "**StorageLocations AS SL**" the AS can be omitted and it will still work. An alias just makes the table name shorter when using the table name elsewhere in the query.

**Qualifiers**

I used the shorter table name when I was listing the columns to return in the SELECT clause. Prefixing the column name with the table name and a period "**SL.LocationName**" qualifies the name with an identifier.

---

*Qualifying names prevents collisions between table column names when joining tables together.*

---

```
SELECT SL.LocationName, SL.CategoryID, Categories.Name AS CategoryName, Categories.ID
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```

I also aliased a column name "(**Categories.Name AS CategoryName**)" I converted the Name column in the Categories table to CategoryName.

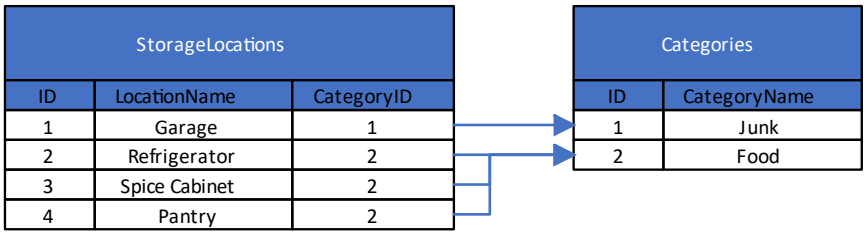
If there were 2 columns with the same name in multiple tables the query optimizer wouldn't know which column you were referring too. You need to identify or specifically state which one you want to use.

```
SELECT SL.LocationName, SL.CategoryID, Categories.Name AS CategoryName, Categories.ID
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```

I linked the StorageLocations table to the Categories table with a LEFT JOIN clause.

In the joined resultset you can see that the CategoryID and ID columns match. Since they match, a Categories record is found for every StorageLocation record which returns the CategoryName column.

In English speak the query says: Return every record from the StorageLocations table and match it to the one found in the Categories table where the StorageLocations CategoryID column matches the Categories ID column. This is referred to as a Many-To-One relationship.



| StorageLocations | | |
|---|---|---|
| ID | LocationName | CategoryID |
| 1 | Garage | 1 |
| 2 | Refrigerator | 2 |
| 3 | Spice Cabinet | 2 |
| 4 | Pantry | 2 |

| Categories | |
|---|---|
| ID | CategoryName |
| 1 | Junk |
| 2 | Food |

## Implicit Join

The ON clause explicitly determines how the tables are joined.

```
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```

This line in the query tells the server to join the StorageLocations table (aliased as SL) with the Categories table, ON the column named ID in the Categories table with the CategoryID column in the StorageLocations Table.

Some legacy queries will have something called an implicit join. This happens within the FROM and WHERE clauses. Do not write queries with an implicit join in a WHERE clause. If you want to maintain your sanity convert them to an explicit join using the ON clause. There is no performance difference between them when the query executes.

**Implicit:**

```
SELECT TableB.numID
FROM TableA, TableB
WHERE TableA.linkNumber = TableB.numID
```

Joined tables are listed in the FROM clause and the joining criteria is in the WHERE clause. If you have many tables to join listed in the FROM clause but neglect to include a corresponding matching link in the WHERE clause you can inadvertently end up with an unwanted CROSS JOIN.

The CROSS JOIN or Cartesian[24] join produces a resultset which is the number of rows in the first table, multiplied by the number of rows in the second table, possibly multiplied by the number of rows in a third table, etc. This kind of result is rarely wanted. Be careful, as this mistake can return millions or even hundreds of millions of records.

[24] Cartesian Join, is a join of every row of one table to every row of another table. (tutorialspoint.com, tutorialspoint.com/sql/sql-cartesian-joins.htm)

*Always use an explicit join!*

### Filtering Rows

We now know which storage locations are food related. Adding a WHERE CLAUSE to filter out anything not related to food should be easy.

```
SELECT SL.LocationName, Categories.Name AS CategoryName
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
WHERE Categories.Name = 'Food'
```

| JOINed Results ONLY Food | |
|---|---|
| LocationName | CategoryName |
| Refrigerator | Food |
| Spice Cabinet | Food |
| Pantry | Food |

The 'Junk' category has been removed because now only 'Food' categories are included. I've also removed the ID's from the SELECT clause because we don't need them on our report.

Now that we know in which locations to find food, let see what items we have at those locations.

We can pretty much guess that the contents of the locations are found in the Contents table. But how is it linked?

```
SELECT TOP 3 *
FROM Contents
```

| Contents | | | | |
|---|---|---|---|---|
| ID | Item | Quantity | UOM | LocationID |
| 1 | Ground Beef | 2 | Pound | 2 |
| 2 | Chicken | 4 | Pound | 2 |
| 3 | Jalapeno | 3 | Pepper | 2 |

### Limiting Records

Since this a just an investigative query, I limited the number of returned records to 3 with the TOP clause.

*There's no reason to return hundreds or thousands of records if you're just trying to get a preview of what's in a table.*

The TOP clause is a Microsoft SQL Server command that may be different for you if your database is managed by another database provider.

In the Contents table there is a LocationID which links to the Locations table.

```
SELECT Item, Quantity, UOM AS UnitOfMeasure
FROM Contents
LEFT JOIN StorageLocations SL ON Contents.LocationID=SL.ID
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
WHERE Categories.Name = 'Food'
```

Hold on! Look closely at this query. The FROM clause this whole time was the StorageLocations table and now it's the Contents table. Why?

I can explain; As stated above, the FROM clause determines the main table in which your data is to be sourced (the LEFT most table). Our original goal was to create a report to list all of the *food items* from the food storage locations. This type of data on our report can determine if we have the necessary ingredients for the chili recipe.

In English speak, we want a query that returns every record from the Contents table regardless of where it's stored, as long as it's food.

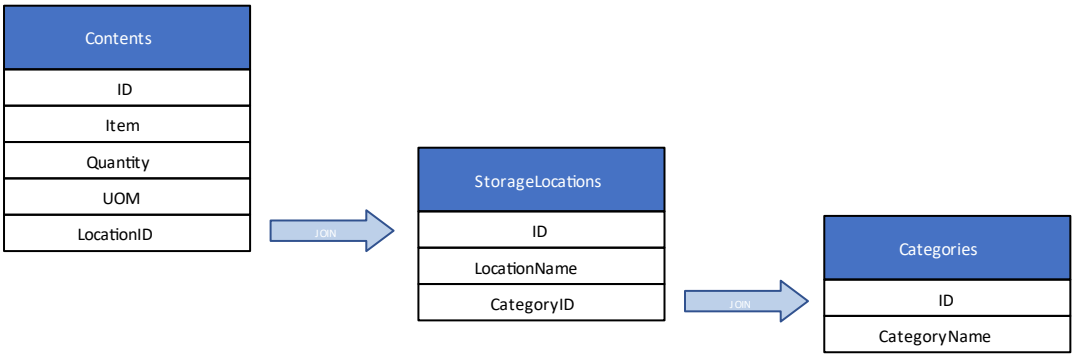
*A LEFT JOIN returns all records from the left table, and the matched records from the right table.*

Since the primary source of your report are the food items found in the Contents table, it's good practice to make that the FROM or LEFT table. We JOIN with the other 2 tables just to make sure the food filter in the WHERE clause will get resolved.

The Contents table links to the StorageLocations table, which links to Categories table to acquire the Category name for the WHERE clause.

Moving the Contents table as the LEFT most table, will visually keep the structure in a basic outline form and maintains a many-to-one relationship.

Since a picture is worth a thousand words:




```
SELECT Item, Quantity, UOM AS UnitOfMeasure
FROM Contents
LEFT JOIN StorageLocations SL ON Contents.LocationID=SL.ID
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
WHERE Categories.Name = 'Food'
```


Having the Contents table as the LEFT table is by no means an SQL rule or requirement. The query can also be written with the StorageLocations table in the FROM clause and return the same results.

```
SELECT Item, Quantity, UOM AS UnitOfMeasure
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
LEFT JOIN Contents ON Contents.LocationID=SL.ID
WHERE Categories.Name = 'Food'
```

However, this version is harder to visualize.

Select every record FROM the StorageLocations table, match it to the Categories table (many-to-one) then duplicate this record for every record found in the Contents table (many-to-one) and only include any of these duplicates where the Category.Name is Food. This is harder to visualize than a simple outline.

**INNER JOIN**

You can also use a RIGHT JOIN and incorporate INNER JOINs if you'd like. Many professionals debate that INNER JOINs execute faster than LEFT JOINs. I say test your own query for performance to see how it affects you.

However, always be cautious with an INNER JOIN!

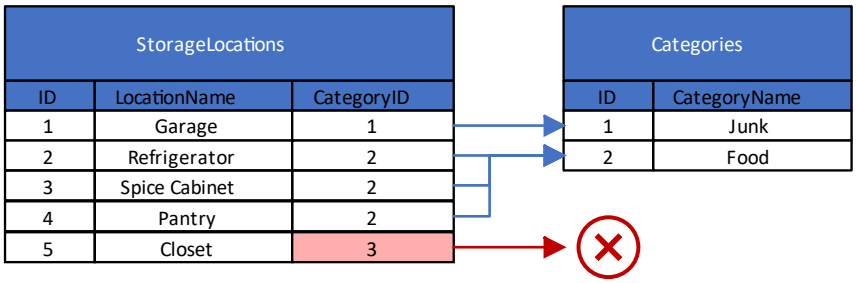
---

*INNER JOINs will toss a record away when the records don't match between tables and you may not know it!*

---

When using a LEFT JOIN, you will visually see NULLs or blanks in your resultset. This will give you an indication that something is wrong with the linked data.

```
SELECT SL.ID, SL.LocationName, Categories.CategoryName
FROM StorageLocations AS SL
LEFT JOIN Categories ON Categories.ID=SL.CategoryID
```



StorageLocations

| ID | LocationName | CategoryID |
|---|---|---|
| 1 | Garage | 1 |
| 2 | Refrigerator | 2 |
| 3 | Spice Cabinet | 2 |
| 4 | Pantry | 2 |
| 5 | Closet | 3 |

Categories

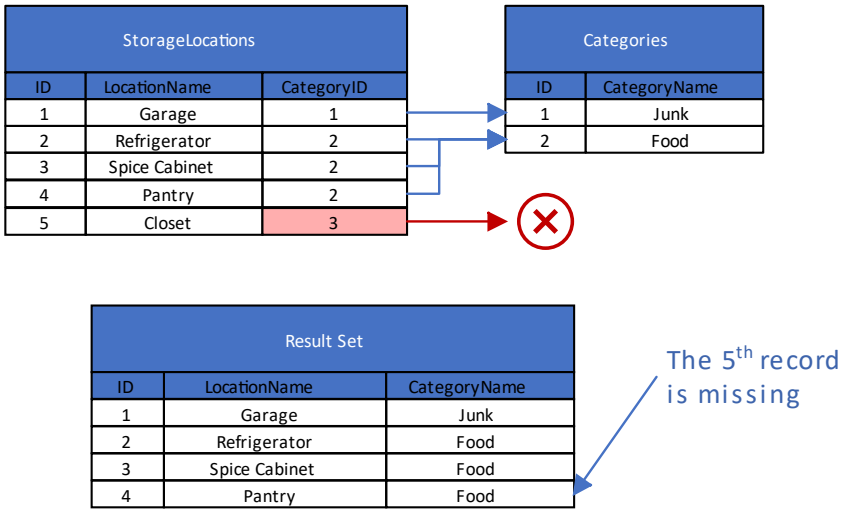
| ID | CategoryName |
|---|---|
| 1 | Junk |
| 2 | Food |

Result Set

| ID | LocationName | CategoryName |
|---|---|---|
| 1 | Garage | Junk |
| 2 | Refrigerator | Food |
| 3 | Spice Cabinet | Food |
| 4 | Pantry | Food |
| 5 | Closet | NULL |

When using an INNER JOIN, you will just be missing records. But how do you know what you don't know?

```
SELECT SL.ID, SL.LocationName, Categories.CategoryName
FROM StorageLocations AS SL
INNER JOIN Categories ON Categories.ID=SL.CategoryID
```




The LEFT JOIN returned 5 records but the INNER JOIN only returned 4!

If your query returns 100 records when using an INNER JOIN should it have returned 101? How would you know? If you were summing up values from the JOINed table how would you know that you were missing a few records?

I never use an INNER JOIN unless I know it gives me a performance boost. Even then, only after I have tested with a LEFT JOIN first.

---

*Don't think, know! How do you know? Because you tested and have real results.*

---

Don't ever let anyone tell you that they think an INNER JOIN is better than a LEFT JOIN or any other crap you may read about queries from the self-proclaimed experts. Even my crap. Always test a query on your data in your own environment and evaluate the results. Every system and query optimizer can give different results. There are certain guidelines and recommendations you can follow, but that's all they are; Guidelines.

*To think[25] is an opinion, to know[26] is a fact!*

Once you've learned how a table JOINs with another it's the same for every query you need within that same business context. They are re-usable.

Note: There are rare occurrences where the same table pairs can join on different columns to create different business context.

If you are not familiar with your data model you may need some assistance from someone that is. Most users familiar with a database have an arsenal of SQL queries or views that they use to access the database. These queries will include the JOIN clauses that you can use to make sense of your data.

Acquiring copies of SQL queries that other people are already using to access your data is the best way to create your own queries for reporting. They've done all the hard work for you.

You don't need to care too much about the data fields that they are selecting in their query. You're looking for the JOIN relationship columns. There are only 2 questions you need answered; What are the main categories of data in the database (FROM Tables) and how are they JOINing with the related tables.

*If you maintain a library of all the join pairs you find, you can mix and match for different reports when needed.*

Looking back at the Contents table and the Food query, you can see that there is a column called "UOM" which was aliases to return "UnitOfMeasure" in the resultset. What makes this column interesting is that it's a denormalized value. I talked about this earlier.

Multiple item records in the contents table have the same UOM. The value "Pound" is repeated for Beef and Chicken. You can guess that Sugar is also measured by the "Pound" and there may be other items that would repeat the same UOM values. Tomato Paste is by the "Can" and Soup is by the "Can", etc.

If you were to normalize this column in the database the Contents table would have a UnitOfMeasureID column that would point to a UnitOfMeasure table that would only contain one "Pound" record.

[25] Think, to have as an opinion. (merriam-webster.com, merriam-webster.com/dictionary/think)

[26] Know, to be aware of the truth or factuality of. (merriam-webster.com, merriam-webster.com/dictionary/know)

You would then have to include another JOIN clause in your query to join in the UnitOfMeasure table.

As discussed, architects can normalize and denormalize data in the same database.

## JOINs Create Work

How do JOINs affect the speed of your report?

JOIN clauses can slow down your report because you're making your server do more work. To make sure it can perform at its best when you have a JOIN, make sure your query has covering indexes[27].

A covering index just means that any fields used in your ON clause of your JOINs or your WHERE clause have corresponding indexes.

We talked about how bad INDEX and TABLE SCANS are when using WHERE clauses. ON clauses without covering indexes can have the same negative effect on your report speed. But don't forget, having too many indexes can sometimes be bad. Not for your report, but for other people using the same tables.

There are many types of indexing terminologies and methods beyond the scope of this document (B-tree, Binary Search, hash table, etc.). For now, let's just assume that there is only one type and it's nothing more than a list of names in alphabetical order.

| Index | |
|---|---|
| LastName | ID |
| Alfieri | 5 |
| Dema | 3 |
| Gordon | 6 |
| Johnston | 2 |
| Kingston | 10 |
| Presher | 4 |
| Rikki | 7 |
| Simpson | 9 |
| Smith | 1 |
| Stock | 8 |

| Data Table | | | | | | | |
|---|---|---|---|---|---|---|---|
| ID | LastName | FirstName | AddressLine1 | AddressLine2 | City | State | PostalCode |
| 1 | Smith | Bob | 1 Main St | Apt #4 | Roanoke | VA | 24102-3452 |
| 2 | Johnston | Bill | 57 Oak St | | Bumpkin | MS | 38602 |
| 3 | Dema | Diane | 8976 Alpine Rd | Suite 78 | Warren | NV | 89010 |
| 4 | Presher | Frank | 437 Park St | P.O. Box 99 | Bishop | WA | 98002 |
| 5 | Alfieri | Grace | 98 Wayne Ave | | Florida | MA | 01002 |
| 6 | Gordon | Harry | 12 Parson Blvd | Second Floor | Candy | MI | 48004 |
| 7 | Rikki | Jim | 789 Warren Ter | | Cave | IN | 46001-2836 |
| 8 | Stock | Joan | 36 Barker Blvd | P.O. Box 576 | Pleasant | CO | 80011 |
| 9 | Simpson | Kathy | 27896 Maple Pl | | Vernon | VA | 24103 |
| 10 | Kingston | Larry | 11 New England Ct | Attn: Larry | Small | NY | 10114 |

The index holds the data value of the column being indexed and a pointer or reference to the actual data record in the table for that indexed entry. Usually, the Unique Identifier or primary key of the table.

Because indexes are sorted, the server can quickly find the index record and retrieve the ID number that references the exact location of the data record. Rather than scanning all of the data records to find the right one, it can directly read the record at the reference ID location.

Go to my house and get a can of Tomato Paste will take longer than; Go to my house, enter the front door, turn left, make 10 steps and you will see a door on the

[27] Covering Index, is an index that contains all required information to resolve the query. (Behara)

left. That's the pantry. In the pantry on the righthand side, top shelf, you will find the Tomato Paste.

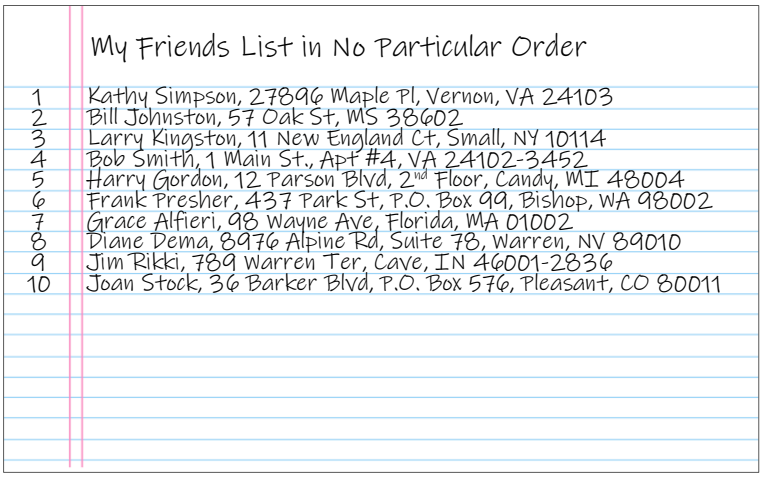
Without the specific instructions you may have to search my entire house to find the Tomato Paste.

---

*Indexes tell the server exactly where to go to get the data.*

---

Your data tables and indexes are stored in a database on a physical hard drive. Think of the hard drive as a 200-page lined notebook. Imagine, taking the notebook and on page one, start writing down all of your friends' names and addresses. One per line. Numbering each line as you go, 1,2,3…




Great! 10 friends.

When you wrote down the names did you do it in alphabetical order? By first name or last name? Maybe you just wrote them down as they popped into your head.

Either way is fine for this exercise. Now that your done let's assume we are new friends and you need to add me to the list. My name is "Devin Crypt, 1 Main St. Anytown, NJ 07998".

If you had no order to your list you can just add me to the bottom at line 11.

My Friends List in No Particular Order

1 Kathy Simpson, 27896 Maple Pl, Vernon, VA 24103
2 Bill Johnston, 57 Oak St, MS 38602
3 Larry Kingston, 11 New England Ct, Small, NY 10114
4 Bob Smith, 1 Main St., Apt #4, VA 24102-3452
5 Harry Gordon, 12 Parson Blvd, 2nd Floor, Candy, MI 48004
6 Frank Presher, 437 Park St, P.O. Box 99, Bishop, WA 98002
7 Grace Alfieri, 98 Wayne Ave, Florida, MA 01002
8 Diane Dema, 8976 Alpine Rd, Suite 78, Warren, NV 89010
9 Jim Rikki, 789 Warren Ter, Cave, IN 46001-2836
10 Joan Stock, 36 Barker Blvd, P.O. Box 576, Pleasant, CO 80011
11 Devin Crypt, 1 Main St, Anytown, NJ 07998

If you have no order to your list, it's like NOT having an index. Which is fast when you add new friends to the list. You just keep adding them to the bottom.

But what happens when you need to look up a friend's address? You need to SCAN the whole list until you find the right names. Pretty fast with one page of friends. Not so fast if the 200-page notebook was full!

If you had previously alphabetized your list by last name then what?

My Friends List Alphabetized by Last Name

1 Alfieri, Grace, 98 Wayne Ave, Florida, MA 01002
11 Crypt, Devin, 1 Main St, Anytown, NJ 07998
2 Dema, Dema, 8976 Alpine Rd, Suite 78, Warren, NV 89010
3 Gordon, Harry, 12 Parson Blvd, 2nd Floor, Candy, MI 48004
4 Johnston, Bill, 57 Oak St, MS 38602
5 Kingston, Larry, 11 New England Ct, Small, NY 10114
6 Presher, Frank, 437 Park St, P.O. Box 99, Bishop, WA 98002
7 Rikki, Jim, 789 Warren Ter, Cave, IN 46001-2836
8 Simpson, Kathy, 27896 Maple Pl, Vernon, VA 24103
9 Smith, Bob, 1 Main St., Apt #4, VA 24102-3452
10 Stock, Joan, 36 Barker Blvd, P.O. Box 576, Pleasant, CO 80011

My name fits alphabetically between records 1 and 2. You'd need to re-write the list from scratch on a new page and insert me in at the appropriate line. Then throw away the old list.

If you wanted to maintain the notebook with an alphabetical list by first name and another by last name and yet another by state, you would need to maintain 3 different lists!

Which means you would need to re-write all 3 lists every time you made a new friend.

I little time consuming yes? That's the same with indexes. The more indexes you have the longer it takes to add data because the server needs to re-organize the indexes.

As a read only user reporting on data you will not feel the slow down because you're never adding data. However, any user using the system that populates the data will. That's why your DBA shouldn't just add a ton of indexes for you. This may cause issues with the production system.

## What Am I Selecting?

JOINs can be costly from a performance perspective. Some more than others. If you don't need the JOIN then don't include it.

```
SELECT TableA.Name, TableA.Address, TableA.City, TABLE_A.State, TableB.Country
FROM TableA
LEFT JOIN TableB ON TableA.linkNumber = TableB.numID
```

If you're not going to use the TableB.Country field on your report from the joined TableB why are you including the JOIN or the field in the query?

Furthermore, if you only needed the "Name" field from TableA, why include the other fields from TableA?

---

*Don't make the server do any more than it has too.*

---

The server has to find, lock, cache and transmit all of this data. *Data access dynamicity*. If your queries are dynamic (only ask for what you need) then they will run faster.

Remember, it's not just you making the server do extra work. If everyone in your organization had dynamic queries think about how much faster everyone's reports would run!

## Subqueries (Sub-selects)

A subquery is nothing more than a query wrapped in parentheses "()" nested inside of another query.

I'm only referring to subqueries that SELECT data, although subqueries can also be used to UPDATE, DELETE and INSERT data.

As a SELECT it can be found as a data source in a FROM and JOIN or as a column in a SELECT or WHERE clause.

Because a subquery can return multiple rows and multiple columns it must return the appropriate structure for the location it's being used.

When used as a data source like a FROM clause it can have multiple rows and columns in its resultset.

```
SELECT SubqueryResults.FullName, TableB.Country
FROM (SELECT TableA.FirstName + ' ' + TableA.LastName AS FullName, TableA.CountryID
      FROM TableA) SubqueryResults
LEFT JOIN TableB ON SubqueryResults.CountryID = TableB.ID
```

Reviewing the query above, it uses a subquery in the FROM clause in place of a table name. The inner subquery will execute first and the resultset will be used as the source table for the outer query. I aliased the inner subquery as SubQueryResults for clarity.

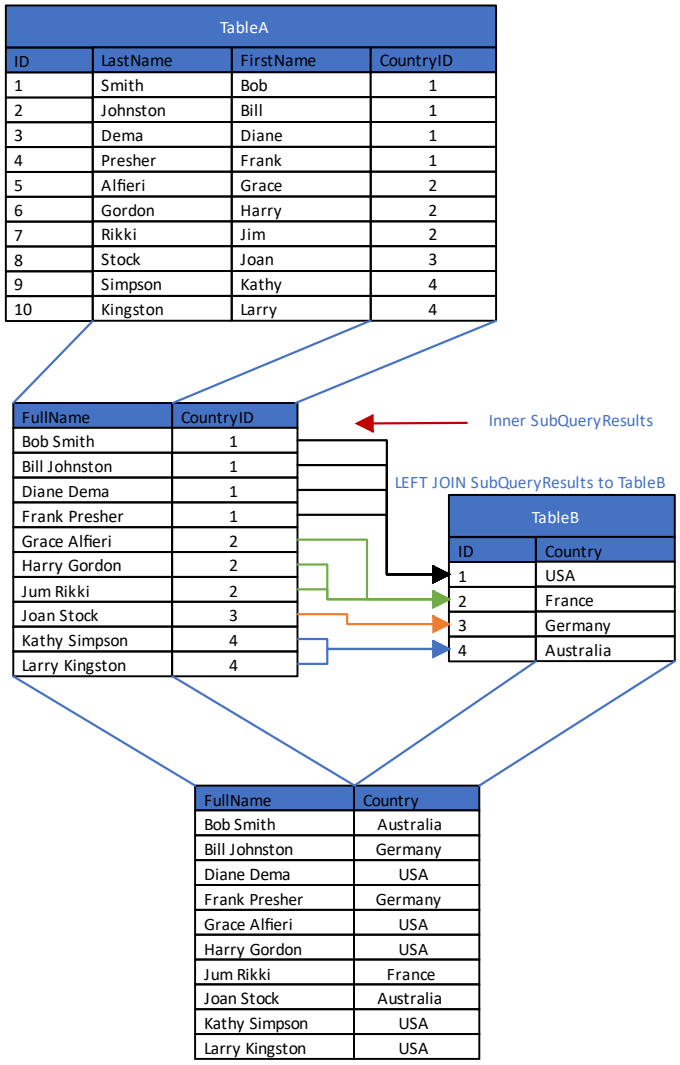
**Derived Table**

Giving a subquery data source an alias is called a Derived Table (mssqltips.com).

It's nothing more than a temporary table that gets created internally when the query is executed.

Some data providers (like Microsoft), require you to add an alias as a Derived Table name even if you are not using it as a qualifier in your query.

The image below is just to visualize how the server is handling the subquery:




Final Query Resultset

The server will store the inner query resultset as a temporary table then execute the outer query. Notice that the subquery created a derived column called FullName which is the combined values of TableA.FirstName and TableA.LastName.

A derived column is known as an expression. An expression is the result of a formula. More than one value combined to create a result. The concatenation of strings or the addition of 2 numbers are simple examples of an expression.

The outer query recognizes the subquery as a table and JOINs with TableB accordingly to get the Country column.

Query optimizers do very well with sophisticated subqueries before they begin to get serious performance problems. However, if you suspect a query is running slow and

it has subqueries, break the query up into smaller parts and slowly diagnosis which piece is causing you grief through trial and error.

Even with a smart optimizer, stop to think about what happens when a subquery is used to return a single column in a SELECT clause.

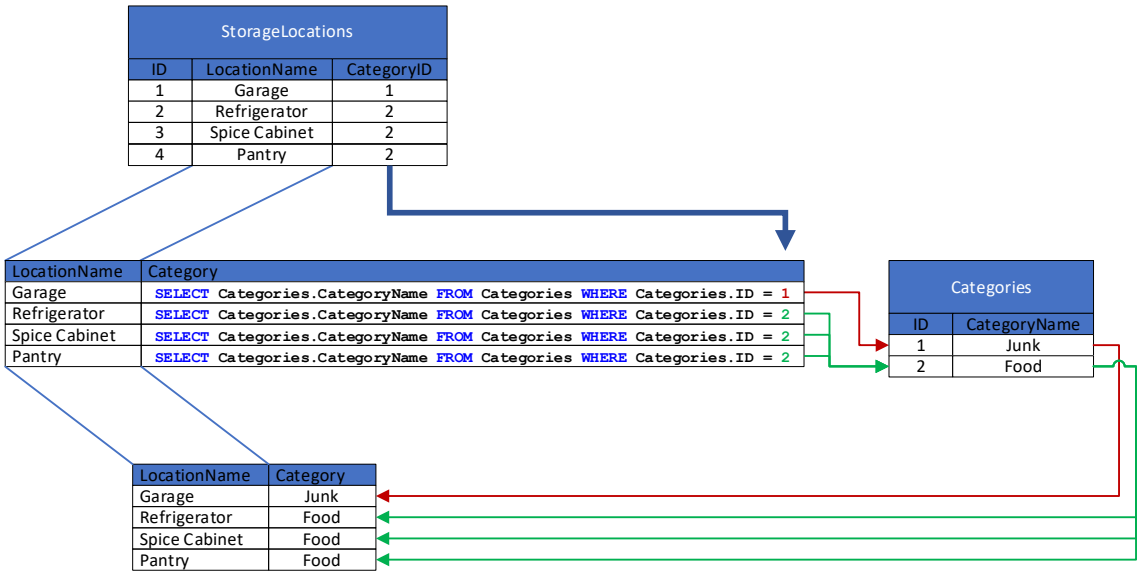
When used as a column selection a subquery can only return a single value. Single column and single row.

```
SELECT StorageLocations.LocationName,
       (SELECT Categories.CategoryName
        FROM Categories
        WHERE Categories.ID = StorageLocations.CategoryID) AS Category
FROM StorageLocations
```

Although this is yet another dumb scenario, I'm using it simply to depict how the server handles this type of subquery.

Unlike the previous example the outer query must resolve the inner subquery for every record returned in the outer query. As each outer query record is resolved the server inserts the StorageLocations.CategoryID value in the WHERE clause then executes the subquery.

You can envision it something like this:




Final Query Resultset

*The inner select is executed for every row found in the outer query.*

Every row! Since the subquery is executed for every row, you can expect it to take a while if your outer query is returning thousands of records.

In the example above, if there is no index on the ID field in the Categories table, then every row of the outer query would technically need to do a table scan of the Categories table to find the right record. Scans are bad under normal circumstances; can you imagine doing one for every record?

Knowing this, some newer query optimizers may try to create an index for you. I wouldn't count on it. There is still a tremendous amount of overhead to do this and the indexes can't be shared between query executions. Therefore, the server would be creating and dropping temporary indexes over and over again. To index the data, every record would have to be sorted before your query even started. Add nested subqueries to this mess and the server has very little chance to help you with performance.

Like the example used above, I've seen situations where a subquery has been used where a JOIN would have been 100's of times more efficient.

The query above should have been written like this:

```
SELECT StorageLocations.LocationName, Categories.CategoryName AS Category
FROM StorageLocations
LEFT JOIN Categories ON StorageLocations.CategoryID = Categories.ID
```

Even without indexes this query would only do a single table scan, not one for every record found in StorageLocations!

---

*It's possible for you to write horrid code inside of a subquery that seriously impacts performance.*

---

But it's also possible that it will not affect performance at all.

Just be careful when using subqueries.

## The OR Clause

An 'OR' clause in a WHERE or even a JOIN clause could be making your reports run slower but not always.

```
SELECT StorageLocations.LocationName, Categories.CategoryName AS Category
FROM StorageLocations
LEFT JOIN Categories ON StorageLocations.CategoryID = Categories.ID
WHERE Categories.CategoryName = 'Junk' OR Categories.CategoryName = 'Food'
```

Everything was working fine until you added the OR clause to your query. What happened?

Again, the query optimizers can suddenly decide that lookups become too expensive for the server and thinks it'll be faster to just scan the entire clustered index.

Converting to a UNION ALL looks uglier but may give you a major performance boost:

```
SELECT StorageLocations.LocationName, Categories.CategoryName AS Category
FROM StorageLocations
LEFT JOIN Categories ON StorageLocations.CategoryID = Categories.ID
WHERE Categories.CategoryName = 'Junk'
UNION ALL
SELECT StorageLocations.LocationName, Categories.CategoryName AS Category
FROM StorageLocations
LEFT JOIN Categories ON StorageLocations.CategoryID = Categories.ID
WHERE Categories.CategoryName = 'Food'
```

OR vs UNION ALL[28] you decide.

As with any performance tip, always try it first to see if it helps you. I've had some great successes, but not always.

## Database Views as a Data Source

An SQL query data source does not necessarily have to be a table. Anywhere in a query that normally sources data from a table (FROM, JOIN, etc.), can be a database view or a subquery.

A database view as your data source could very easily be causing your report to run slow. Since a database view is nothing more than a saved query, what if that query is bad or wasteful. Even worse, what if there are data sources in the view's query that reference yet another view and so on.

---

*Each view in the nested chain of views could be wasteful or slow.*

---

Nested views[29] usually happen out of convenience or time constraints when the IT professionals created them.

In general, nested views are bad. Bad because every query optimizer will give you intermittent performance hits without warning.

Review your query and verify that each data source is pointing directly to a table or that the data source it's using is not wasteful.

I've discussed many topics on how a query can be wasteful. The single most important one is; Don't request data from the server that you'll never use.

---

[28] OR vs UNION ALL, is one better for performance? (Wagner)

[29] Nested Views, are database views that call other database views. (West)

*Unless a view was specifically created for you, it's most likely requesting data you don't need.*

**Cargo Cult Programming**

Cargo Cult Programming[30] is basically what happens when an inexperienced programmer simply copies code and uses it without understanding what it really does or if it's really needed.

Database views get abused the same way. I call them *Cargo Cult Queries*. Users blindly use views without understanding what they're really doing. Over time systems get dependent on them and nobody really knows their original purpose.

*Wasteful views can easily get propagated throughout an entire system, secretly using server resources for no reason.*

A tree of nested views can get pretty deep over time. Sometimes JOINing the same table over and over again. Views can be JOINing tables totally unrelated to your report and selecting or converting data you don't even need. They can be summing thousands of records, calling slow Table Valued Functions or any number of other things you don't care about.

A view is also a virtual table. It gets created when executed. This means the resulting virtual table is a table without indexes.

We learned that not having indexes can be bad. If a view is used as a data source or values from its results are part of a WHERE clause to filter the data, then the server must do a slow table scan of the virtual table. Which may be fine if you're not filtering the data with a WHERE clause and are using every record of the virtual table.

If you're JOINing the view with other tables then the JOINs can't be optimized either.

Sometimes, a nested view will internally completely resolve its resultset before continuing to the next level of view. A sub level resultset could end up returning hundreds of thousands of rows internally only to throw most of them away in your final resultset.

Don't forget that each database vendor has a different query optimizer that may or may not be smart with your nested views.

[30] Cult Programming, is a style of computer programming characterized by the ritual inclusion of code or program structures that serve no real purpose. (en.wikipedia.org, en.wikipedia.org/wiki/Cargo_cult_programming)

*Most of the time nested views are bad.*

Use the knowledge you've gained here to determine if using the view is the best option for your report.

In the case of nested views, things can change later even if your query is running fine when you first tested it. Someone can easily change one of the views in the nested tree without your knowledge, which can cause your report to slow down for no apparent reason.

Debugging a series of nested views can be a nightmare. My suggestion is to avoid them if at all possible.

## The ORDER BY Clause

An ORDER BY clause sorts your resultset. Sorts are slow.

*Never use an ORDER BY clause unless you absolutely need your resultset in a sorted order.*

ONLY use an ORDER BY clause in your last, outer or final query.

```
SELECT SubqueryResults.FullName, TableB.Country
FROM (SELECT TableA.FirstName + ' ' + TableA.LastName AS FullName, TableA.CountryID
      FROM TableA) SubqueryResults
LEFT JOIN TableB ON SubqueryResults.CountryID = TableB.ID
ORDER BY TableB.Country
```

NEVER use an ORDER BY clause on inner queries like below:

```
SELECT SubqueryResults.FullName, TableB.Country
FROM (SELECT TableA.FirstName + ' ' + TableA.LastName AS FullName, TableA.CountryID
      FROM TableA
      ORDER BY TableA.LastName) SubqueryResults
LEFT JOIN TableB ON SubqueryResults.CountryID = TableB.ID
```

Microsoft SQL Server does not allow this but other data providers may. Furthermore, a database view can have an ORDER BY clause even if you don't have one in your final query.

## Say "NO" to NOLOCK (Microsoft SQL Server)

This is more of a pet peeve than a query performance note.

Many query writers add a NOLOCK (READUNCOMMITTED) hint to all of their queries because it seems like a magical command that makes all queries run faster. (Because they haven't read this book to know why their query may be slow in the first place.)

Don't do it!

Let's start with one true fact. If it was magical, Microsoft would have it as the default without having to add the hint.

Adding the NOLOCK means you agree that;

- you do not care if committed rows are missed.
- its ok that uncommitted rows are included.
- in rare cases the same row can be returned more than once.
- in very rare cases rows are returned which do not match the original query.

---

- *You're basically okay with the possibility of getting bad data results.*

---

**Committed vs Uncommitted Rows**

> Think of a word to write down on a piece of paper. The word is uncommitted until you write it on the paper. Once written, it is considered committed.
>
> You can change the word in your mind many times, from the time you think of the word until the time you write it down.
>
> A read of uncommitted data can return the first, or all of the words you thought of and skip the one you ultimately decided to write down.

I don't know about you, but I'm expecting my data to be correct! Why would you want to take the chance of reading uncommitted data?

Others may add the NOLOCK hint because they are getting deadlocks.

**Deadlock**

> a state of inaction or neutralization resulting from the opposition of equally powerful uncompromising persons or factions. (merriam-webster.com, merriam-webster.com/dictionary/deadlock)
>
> When process A cannot complete because it's waiting for a resource that is locked by process B, while at the same time process B is waiting for a resource that is locked by process A.
>
> Things come to a standstill.
>
> Sometimes a data provider will recognize this situation and kill off one of the processes so that the other can continue. Great if yours is the one saved. Not so great if your process was the one killed.

Deadlocks are NOT caused by a SELECT query.

*Deadlocks are causes by bad programming practices.*

NOLOCK works in deadlock situations because the hint allows SQL to read data from tables by ignoring any locks and therefore not being blocked by other processes.

The root of the problem is the other processes holding on and not releasing locks correctly, not that your query wants to read the data.

*No SELECT query will deadlock or block any other process.*

## Conclusion

This section discussed quite a few topics in understanding how querying a database works. Writing a query to access your database is not that hard.

Writing an efficiently query is not an exact science. I covered some of the major topics in regards to the performance of queries. Some are obvious others are a little more unnoticeable unless you look really hard.

Regardless of what you read or are told, the only way to know that your query is working efficiently for you, is to test it. On your hardware, with your data. Nothing else matters.

# CHAPTER 4

## Other Things to Consider

### Data Types

A quick note on data types. Don't worry about them!

As a user reporting on data, there is no reason to stress over the difference between a *Float* and *Integer* or a *varchar* and *nvarchar*. What the…? Exactly!

Ask a human what "34.785" is. They will tell you it's a Number. Not a *Float*.
Ask a human what "Hello" is. They will tell you it's a word or text. Not a *varchar*.

Only a Database Architect, DBA or Software Engineer needs to know this jargon.

When you write an SQL query that selects a *Float* value from a database it will display in the resultset as a human readable number. As will all of the other number data types in a database regardless of the provider (Microsoft SQL Server, Oracle, MYSQL, etc.)

99% of all reports display only 4 or 5 types of data to a human:

- **Numbers**
  bigint, decimal, float, int, money, numeric, real, smallint, smallmoney, tinyint, etc.
- **Text**
  char, nchar, ntext, nvarchar, text, etc.
- **Dates/Times**
  date, datetime, time, timestamp, etc.
- **Yes/No** (True/False, On/Off)
  bit, boolean, etc.
- **Images**
  blob, lob, clob, image, binary, etc.

These are just a few. Each data provider may have different names for their data types.

### Formatting Data

There are many ways to format data using a query. Sometimes it makes sense or is convenient. However, every time you tell a server to do more, your query can take longer to run.

If you're formatting your data results in your reporting tool, then there's no reason to spend extra time formatting it in your query. Sometimes your database server can format data faster than your reporting tool. If you're using the query with multiple reports then it may also make sense to format it in your query so that the formatting doesn't need to be repeated in every report.

Only format the data that you need to. Keeping things as simple as possible in your query has other benefits also. It's easier to fix when things go wrong.

## NULL Data

What is a NULL anyway?

A NULL value in a database column indicates there is no data for the field. Let's take the example of a column like, DATE_OF_DEATH. If the person is alive then there is no DATE_OF_DEATH so the value could be NULL (the absence of an answer).

| TableA | | | |
|---|---|---|---|
| ID | LastName | FirstName | DATE_OF_DEATH |
| 1 | Smith | Bob | NULL |
| 2 | Johnston | Bill | NULL |
| 3 | Dema | Diane | 20200918 |
| 4 | Presher | Frank | NULL |
| 5 | Alfieri | Grace | NULL |
| 6 | Gordon | Harry | 20190223 |
| 7 | Rikki | Jim | NULL |
| 8 | Stock | Joan | NULL |
| 9 | Simpson | Kathy | NULL |
| 10 | Kingston | Larry | NULL |

Some databases default to allowing NULLs when a column is created in a table. NULLs in a database can cause chaos when writing queries to report on data. For this reason alone, I don't like NULLs in databases. Like the example above with DATE_OF_DEATH, it's not practical to assume that data should never be NULL. There are ways to avoid NULLs, but in this case a NULL makes sense.

Unfortunately, you will run into databases that are riddled with NULL data where a default value could have been applied.

---

*You need to be careful when querying NULL data.*

---

You need to know that you could be dealing with NULL data and prepare your query accordingly.

This sounds like a broken record but the 80/20 rule really applies here. When saving a record to a database the developer could have stored a default value instead of a NULL or when the table was created the column could have been set with a default value. How often is a record saved versus how many times will it be queried in the future?

Everyone that ever needs to query NULL data will need to add special considerations in their query to deal with that NULL. If they don't, then their results could be skewed.

It makes more sense to just add a default value once, rather than worry about every query ever written against that data in the future! But that's just me.

NULL data requires the use of the ISNULL and IS NOT NULL clauses in a query.

Interestingly enough, IS NOT NULL will use an index and IS NULL will NOT use an index. (Resulting in the hated TABLE SCAN.)

---

*ANY aggregate expression that is used in any query where a NULL value is included will always return a NULL. Which is never the expected result.*

---

Having NULLs where a value can have a valid default makes no sense at all. A column called EMPLOYEE_DISCOUNT is a perfect example. If the employee doesn't get a discount or no value was ever specified then it should default to 1. Not a NULL.

Here's what happens with a NULL with this example:

| TableB | | | |
|---|---|---|---|
| ID | Item | Cost | EMPLOYEE_DISCOUNT |
| 1 | Jeans | 20 | .20 |
| 2 | Shirt | 10 | NULL |
| 3 | Socks | 5 | .10 |

What are the costs of items to employees?

```
SELECT Item, Cost - (Cost * EMPLOYEE_DISCOUNT) AS EmployeeCost
FROM TableB
```

| Result Set | |
|---|---|
| Item | EmployeeCost |
| Jeans | 16 |
| Shirt | NULL |
| Socks | 4.50 |

Jeans originally $20 with a 20% discount is $16
Socks originally $5 with a 10% discount is $4.50

How much do we charge an employee for a shirt? Nothing?

To fix this, every query writer must wrap the EMPLOYEE_DISCOUNT column with an ISNULL shown below:

```
SELECT Item, Cost - (Cost * ISNULL (EMPLOYEE_DISCOUNT,1)) AS EmployeeCost
FROM TableB
```

| Result Set | |
|---|---|
| Item | EmployeeCost |
| Jeans | 16 |
| Shirt | 10 |
| Socks | 4.50 |

Now the results make more sense. The ISNULL states; If EMPLOYEE_DISCOUNT is a NULL then use the number 1 instead (no discount). Because the NULL was allowed, the query writer has to account for it every time. Easily avoided with a default value when inserted.

Another example with a simple text field FirstName; Allowing NULLs here and not having a blank string as a default value is just stupid (A default of a blank string in MSSQL is ‘’).

| TableA | | |
|---|---|---|
| ID | LastName | FirstName |
| 1 | Smith | Bob |
| 2 | Johnston | NULL |
| 3 | Dema | Diane |
| 4 | Presher | Frank |

When a record is inserted into a table and a field has no value specified, allows NULLs and no default value, it will result in a NULL value.

Getting a list of names is now a problem.

```
SELECT FirstName + ' ' + LastName AS FullName
FROM TableA
```

| Result Set |
|---|
| FullName |
| Bob Smith |
| NULL |
| Diane Dema |
| Frank Presher |

What happened to Mr. Johnston? Again, we have to add a provision for the NULL in FirstName.

```
SELECT ISNULL (FirstName,'') + ' ' + LastName AS FullName
FROM TableA
```

NULLs are not what query writers expect.

Remember, ANY aggregate expression that contains a NULL in ANY of its parts results in the result of that expression to be NULL!

A query writer will have to write a lot of ISNULL, NVL, COALESCE or equivalent for every column of every table that allows NULLs. Simply because someone thought that it was too difficult to add a default value!

I don't mind adding provisions to a query for NULL data where it's needed. (DATE_OF_DEATH) But not where a default value could have been just fine.

Special considerations also have to be added to any NULLable columns in a WHERE clause or JOIN.

---

*Nothing is ever equal to NULL, and nothing is ever not equal to, greater than or less than NULL.*

---

The main issue with NULLs is that they have special semantics that can produce unexpected results with comparisons, aggregates and JOINs that are likely to cause subtle bugs.

By the way; wrapping every column with special ISNULL clauses will just make your query more complicated. You'll be making the server do more work which, if you haven't guessed already, is not a good thing.

You should be very careful on how you approach NULLs. There's a lot of nonobvious pitfalls to it and most people can easily be caught off-guard.

# CHAPTER 5

## Summary

I covered a lot of topics and maybe made some enemies in the process. Frankly, I'm tired of the empty rhetoric I hear from the industry about how complicated it is to access data from a relational database for reporting.

This document embodies the idea that direct data access needs to be flexible and available to the business user. Again, not all data just most of it.

IT needs to realize that with thousands of data fields, there are billions of permutations that the business user could want. Restricting access to data elements, that the users know exist in the database, but just can't have, has caused decades of frustration.

I'm a technologist that has spent years honing my skill-sets in many different areas. Nobody is an expert in all technologies and I'm not claiming to know everything either. I've had the unique opportunity to work on both sides of the camp. Business and technology. IT has very little empathy for users' reporting frustrations and that needs to change.

Once adored as a key business enabler, IT has become nothing more than an expensive budget item to many companies.

---

*IT is viewed as a cost center that needs to be reduced because it's not really helping the business achieve its greater goals.*

---

A company directs money to IT budgets because they have too, not because they want to.

Not a believer? Why do you think offshoring, outsourcing, and cloud computing has become the favorite solutions for businesses?

From my business perspective, they reduce costs and make IT predictable to measure.

My technology experience has shown that offshoring, outsourcing, and even the cloud can often slow down business processes and reduce quality. Hidden dollars lost that the business can't easily see or measure.

Remember what I said: When the business asks for a new piece of data from IT, the most common reply is: "Why do you need it?" What they really mean is: "How important is this to you?"

They want to know if they have to stop what they're currently working on to help you get data. They really don't want to write the query. The request is a pain in the ass.

IT has very little empathy for the business, which leads to quick fixes that are not the most efficient solutions.

You now realize that getting data from your database is not as impossible as everyone would want you to believe. Your newly obtained knowledge is your best weapon.

In the 1980's there was a clothing retailer that ran a television commercial over and over again. Their slogan was "An educated consumer is our best customer." The meaning of the phrase was that the more knowledge the consumer had, the more they could see that the retailer provided the best service and value. This concept is an important one.

After learning everything you need to do to access your data with SQL, I hope you can appreciate the value of something that can do this for you.

Check it out at DataSelections.com

# APPENDIX

## Pet Peeves

In no particular order, this section is dedicated to just shit that bothers me.

### The Fallacy of Change (System Evolution)

Developers and system architects tell you that when moving to a new version of a computer system the data model could change and all of your report queries may no longer work. They want to control the queries to make sure you don't have any issues. This is a major fallacy.

It's possible that a major version change might modify the data model, however, it's a very rare occurrence. When a new version or revision comes out, developers are simply adding new features or modules.

New features add new data elements to a database. Adding additional fields to some tables or adding some totally new tables, will not break the existing data access logic that you set up in your queries. Only the removal or renaming of existing data elements would give you a problem. Something developers shy away from doing at all costs.

If a developer were to make a major change to the data model, they would have to make major changes to their base code logic to keep their own system working. No one wants to do that unless they absolutely have to. It's just too costly.

Don't worry too much about taking on new versions of a system.

### You Must Use Special API's to Access Data

There are many systems where this is true. Cloud based systems are notorious for this. They don't allow direct access to their database for commercial reasons. Some will hold your data hostage so that you will continue to use their services. However, if your database is in-house (your company controls the database server), then this isn't true.

**API (Application Programming Interface)**

> API's are used for many things but in a nutshell, they simplify programming by abstracting underlying commands or componentizing reusable actions.
>
> For the purposes of this document, think of an API as a glorified database view. A series of repeatable instructions.

You don't "have to" use an API to access the database. API's access the data the same way you do, with a query. The only difference is that API's can do more to the

data after it's pulled from the database or there may be security restrictions tied to it.

As an example; If the API were to query a raw value from the database and that value was .90, the API could manipulate that value and return 90 instead of the raw data value.

This is a very simplistic example; however, some API's can do some very sophisticated things. They can be complete program routines that query the database multiple times and calculate major resultsets. Just like views, they can include complicated JOINs to return contextual data results.

Sticking to my 80/20 theme, 80% of the data you need could just be sitting in the database. You don't need an API to get it. A simple query may just be fine.

## Calling SQL a Programming Language.

There are always 2 sides to every story. However;

---

*SQL is NOT a programming language.*

---

There are people on the internet posting articles to convince you that SQL is a programming language. This article (https://www.techrepublic.com/article/is-sql-a-legitimate-programming-language/), believes SQL is a programing language because you can have loops.

They're idiots. If they didn't want to be called idiots then he shouldn't write stupid shit and post it on the internet. Again, don't believe everything you read on the internet.

The article is titled "Is SQL a legitimate programming language?" and then emphasizes the fact that SQL IS a language because; "It even has the word language in its name" (Structure Query Language).

Apparently, they don't understand the difference between a programming language and a scripting language.

The article further states there is "Bigotry within the ranks of the development community…".

---

*Bigotry: intolerant devotion to one's own opinions.*

---

Some things in the development community are opinions and we do have some intolerant devotion to certain subjects. However, the development community lives by a few unfaltering sets of rules.

You will get laughed off the internet by real Software Developers if you call yourself a Software Developer (programmer) simply because you know SQL.

---

*Real developers know that SQL is NOT a programming language.*

---

SQL is a language. A query language. More accurately a scripting language.

Software Developers know that you can't develop a software application with SQL.

A Software Developer knows that the SQL engine that executes SQL scripts was written with a real programming language.

No matter how good you are with SQL, you would never be able to create a brand new real programming language with it.

Programming languages are referenced by levels. 1GL, 2GL, 3GL etc. "GL" stands for Generation Language". The further your code is from the actual machine (CPU) the higher the number. Anything above a 3 is considered a scripting language by real developers. SQL is a 4GL. 1GL is used to write 2GL, 2GL is used to write 3GL, etc.

Hard to believe but an SQL query to a computer gets stepped down through the GL's and is nothing more than a series of zeros and ones when it gets there. "0100111001010011"

One basic difference is that most programming languages are compiled whereas scripting languages are interpreted, and programming languages run independently but scripts do not. An SQL Query requires the SQL Server Engine for it be executed.

Some developers, like myself, understand how to convert this binary machine language into English.

01011001 01101111 01110101 00100000 01100001 01110010 01100101 00100000 01100001 00100000 01110000 01110010 01101111 01100111 01110010 01100001 01101101 01101101 01100101 01110010 00100001

Without using the internet!

Developers don't do this by hand anymore, not even me, but I am a strong believer that all developers should know how each GL level works.

---

*If you know that 1 + 1 = 10 in binary, then you could quite possibly be a real programmer.*

---

If you want to know more about the differences between programming and script languages, this site is a good reference:

https://www.thecrazyprogrammer.com/2020/03/difference-between-programming-language-and-scripting-language.html

Information on the GL levels can be found here:
https://www.techbaz.org/Blog/Generation-of-programming-languages.php

## Chasing "The Cloud"

It seems like every business on earth is trying to get their systems running on the cloud. I'm not so sure it's the best thing. But first, what is "The Cloud" anyway?

I was interviewing a guy for a programming job and he tried to explain to me that the cloud was a "window to the universe". Yea! For real. You can guess I didn't hire him.

---

*"The Cloud" is nothing more than someone else's computer.*

---

There's nothing magical about it. However, how people refer to it in conversation gets fuzzy really quick.

"I remote into my office from home, I'm on the cloud."

"I use Google Docs on the cloud."

"I bought food online in the cloud."

"My SQL database server is in the cloud."

"My company is moving everything to the cloud."

Although referencing the cloud means different things to different people, there are basically three types of clouds:

- Public cloud - shares resources and offers services to the public over the internet.
- Private cloud - isn't shared and offers services over a private internal network commonly hosted on-premises.
- Hybrid cloud - shares services between public and private clouds depending on their purpose, and a community cloud that shares resources only between organizations, such as with government institutions.

Unfortunately, the confusion doesn't stop there. There are cloud service organizations and cloud services.

Microsoft Azure, Amazon Web Services (AWS) and Goggle Cloud are cloud service organizations. Which are services that rent you hardware to host software applications and such.

Dropbox, Google Drive, OneDrive, Box etc. are storage cloud services specializing in data storage.

Vimeo, YouTube, etc. are storage cloud services specializing in video storage.

Snapfish, Shutterfly, Zazzle are storage cloud services specializing in photography storage.

CrashPlan, Carbonite, IDrive, etc. are backup storage cloud services.

These are just a few. There are many companies that offer other cloud services.

Is the cloud the internet, a hosting company or just a website? Yes. If you're on the internet you're using someone else's computer. If you're hosting your company's software with a hosting service, you're using someone else's computer. If you're on a website you're using someone else's computer.

You can debate who coined the phrase "The Cloud" for hours or why it even caught on in the first place, but we can agree, that it means some computer system that is running somewhere else other than where you are physically located. Usually, outside of your building.

Now that we know what the cloud is, I want to explain the difference between a company that offers a cloud service like Dropbox or YouTube, from a cloud service organization, like Azure or AWS.

You can subscribe to many cloud services and say; "My photos are stored on the cloud." or "My local hard drives are backed up to the cloud."

However, companies chasing "The Cloud" I put into two categories. Software companies that want to profit from having their systems available on the internet as a service and companies that want to reduce the IT costs related to having their own computer room in their building filled with computer hardware.

Software companies want to reach a wider customer base at a faster pace. SaaS (Software as a Service) has been a growing trend for many years. Easier sale if their prospective clients don't have to buy hardware to get up and running. The vendor sets them up on their cloud and the client gets some login information. Faster implementation and the client's IT department doesn't even need to get involved. A SaaS solution is great for Software Vendors because it removes the prospect's IT bottleneck from the sales cycle. Too many rules and hurdles to jump.

If you have a Dropbox account or similar you've done the same thing on a smaller scale. Your credit card gets charged once a month or year etc. and you're up and running in no time.

Corporations can purchase a SaaS solution on a larger scale with manufacturing software, accounting, payroll etc. SalesForce is a classic example.

Under this model software updates and hardware incompatibilities become drastically reduced. Some vendors have you login remotely to their systems from a small piece of software installed on your local computer, others are known as "browser based". Meaning you run their system through an internet browser like Chrome, Edge, Internet Explorer, etc. This method would be referred to as a web-based system.

Many SaaS vendors utilize the services of a cloud service organization like Azure or AWS to host their systems rather than having their own private cloud (their own hardware).

Which brings us to companies that want to reduce or remove their own in-house computer room. They move their systems to the cloud by subscribing to a cloud service organization that allows them to buy "virtual" hardware at-will. They can build virtual machines where they install their core business applications. If they need a bigger machine with more power or disk space, they can just request it online and it's magically available.

---

*"Virtual" hardware is simulated through software.*

---

Mindboggling for some but as it turns out, pretty easy to explain.

Hardware is something that you can physically touch with your hands. Software you can't touch. If you have a computer program (software) on a jump drive, you can touch the hardware (jump drive) but not the software (the program).

When a computer is built (hardware) it's constructed with specific parts. CPU (Control Processing Unit, the brains), memory chips (DDR RAM), hard drives (diskspace) etc.

The computer is loaded with an OS (Operating System) like Windows 10 which knows how to communicate with the hardware and your software application to make it all work.

Like your laptop computer. Computers have specifications:
8th Generation Intel® Core™ i3-8100 4-Core Processor (CPU), 8GB DDR4 RAM, 1TB of hard disk space.

As we all know, faster CPU, more Gigabytes of RAM or a larger hard disk means it costs more money. But it also means the faster our programs will run and the more photos we can save.

Multi-core processors can do multiple tasks at the same time. Cloud service organizations have racks and racks of very large computers with huge multi-core processors and tons of RAM. They also have a gigantic amount of hard disk space called a SAN (Storage Area Network). They install one piece of software on these

machines called VMS (Virtual Machine Software). There are multiple companies that make VM Software but VMWare is a popular industry standard.

This software allows the service company to split up the big machines into many smaller ones. They can make a "virtual" machine and sell its services as if it were a tangible piece of hardware. Because it's virtual, they can delete it, move it to other hardware, clone it, etc. If the virtual machine is too small and the client wants it bigger, they can just add virtual CPU cores or RAM at the press of a button (and a bigger bill to the client). No need to physically modify any of the hardware.

---

*Virtual machines can be configured and used for any purpose, the same way a physical piece of hardware can.*

---

A hosted database server on a virtual machine can affect the speed of your queries. Where a physical server has 100% of the available resources dedicated to the Operating System, a virtual machine doesn't. Troubleshooting this type of performance issue can be tricky.

The VMWare software is another layer of code between you and the CPU. More layers mean more code, which means things take longer to execute.

Companies move to this model for many reasons but I will venture to guess it's because of money. It's always because of money.

Computers have a shelf life of about 5 years. No 4 years. 3 tops! This changes by the minute. Then you have to upgrade or replace it. Anyone who has ever purchased a computer or even a cell phone, can relate to this.

CFO's don't like to get hit with a multimillion-dollar budget item every time the IT director tells them they need to upgrade the hardware. Especially when IT is a cost center and not a profit center.

An easy way to keep the budget relatively flat is to move to the cloud and just pay a monthly fee for services. As a bonus they get a lower electric bill, need less IT staff and free up some space for a gym where the computer room was located!

Now that you understand what all the hubbub's about, why do I think that the cloud may not be the best thing for all companies? Mine is one of them.

There are pros and cons to everything, including the cloud. It depends on who you are and what situation you're in. The cloud is not good for every system and of course there are systems that it's perfect for.

*Putting everything in the cloud because "it's the thing to do" is a bad model.*

Renting versus buying a car. Like the cloud, renting a car is using someone else's equipment. There are rules you must follow when renting and you have less control, but you can just pay a higher payment and let someone else worry about maintenance, insurance and breakdowns. If you want to go 4 wheeling in the woods, add a winch, lift kit and bigger tires, then you're not renting, you're buying.

If you're rental car stops working, you need to wait for the car's owner (rental company) to tell you what to do. Maybe they have a replacement available, maybe they don't. Even if they do, you could be hundreds of miles away from one of their office locations. You can't take the car to the local repair guy because it's not your car. You have to wait until the rental company is ready to help you. They may be helping other customers and will get to you tomorrow. Maybe you can drive out of state with your rental, maybe not. Like I said, rules, pros and cons. The choice is yours.

One thing's for certain. You're in a pool of customers and your need to get somewhere in the car today is not their priority. Regardless of your circumstances. That's the nature of renting.

If you bought your car, you can get it fixed anywhere or even work on it yourself. You can drive anywhere you want and add any fancy options you'd like. Paint it bright yellow, nobody cares.

*When a business puts their core systems on the cloud, they're relinquishing control.*

If you use a cloud service to store your photos, you're relinquishing control. If you post a video on TikTok, you're relinquishing control.

As long as you're okay with waiting in line while your business is down and being at the mercy of someone else to fix your problem, then you should have no problem with the cloud.

As long as you're okay with someone else having access to your data, that you don't know, then you should have no problem with the cloud.

As long as you're okay that a cloud service can turn off access to your system for literally any reason, then you should have no problem with the cloud.

As long as you're okay with an outage to your business that you did not cause, then you should have no problem with the cloud.

I'm a control freak when it comes to my systems and my data. That's just me. I limit what services I use in the cloud. How did I get here? I've experienced burn by cloud vendors multiple times. As the saying goes; Fool me once shame on you. Fool me twice shame on me!

**Burn #1 – (Cloud based service)**
Let's start with a simple on-line CRM (Customer Relationship Management) system.

I was evaluating a CRM system to see if it would fit my needs. The vendor offered a free subscription with limited features. Over the course of a few months, I manually entered the information of about 200 prospects. It seemed to fit my needs but had a few things I didn't like. I decided to hold off purchasing the subscription so that I could evaluate 3 other systems. I actually found another system that fit my needs better than the first. All I needed to do was export the 200 prospect records from the first and import it into the one I chose.

I'm no dummy (so I thought), I made sure that the first system had an export option before I entered the prospect data. The data is safe in the cloud.

I login to the first CRM to export my data and I get this message;

---

*"Your account has been deleted due to inactivity."*

---

Contacting the CRM vendor, they inform me that they no longer have my data because their policy for free accounts is to delete the data if there is no activity for 60 days! All of that work gone! I was burned by the vendor. Which I call "Vendor Burn".

I should have exported the data immediately and saved it locally on my own machine. Who knew? I'm sure it was somewhere in the fine print.

**Burn #2 – (Cloud Service Organization)**
I worked for a large software company that offered a hosting service solution to its customers for its premier CTRM (Commodity Trading and Risk Management) system. We're talking large, multimillion-dollar customers. Although they had an IT department and a computer room (private cloud), someone in their infinite wisdom, decided to use a cloud service organization instead to.

I was a Software Director of a different division which offered a SaaS (Software as a Solution) system to our customers. Slightly different than a hosting service solution but everything was in our own computer room with our own hardware.

The hardware for the CTRM system was owned and managed by an external hosting company like an Azure or AWS. An outage would occur and our clients would call our support line. "My trading system is down! When will it be back up and running!" Our

support staff would log a ticket with the hosting vendor and wait. Excuses were invented and threats were thrown. As you can guess a trading system is time critical.

Our support staff of the CTRM system had no choice but to wait until the hosting vendor fixed the issue and got back to them. They were on the list with many other customers.

This went on for a while until one day I get a call from our CCO (Customer Care Officer).

CCO: You offer a SaaS solution for your system, don't you?
Me: Yes.
CCO: Do you have these problems?
Me: No.
CCO: Can you help me solve this outage problem? Our customers are threating lawsuits.
Me: Yes and No. But you're not going to like the answer.
CCO: Why?
Me: Because the only way to solve your problem is to move the systems to your own data center and not use the hosting service.
CCO: What! That will cost a lot of money!
Me: Yes. But someone decided a long time ago to relinquish control and use a hosting service. I can only assume it was to save money in the short term, not realizing it would cost us customers and our reputation. A non-hosted system allows you to control all aspects of the hardware, how long it will take to fix, and who's working on it.

In the long run, I was tasked by the CEO to prepare a hardware budget, hire staff, find a location and move ALL hosted systems to our own private cloud. One that we controlled. There was no other choice. No amount of begging or pleading with the existing cloud vendor could solve the fact that our systems were not that big of a priority to them.

Critical productions systems are, well, critical.

You may think it can't happen to Microsoft (Azure) or Amazon (AWS). I hate to be the bearer of bad news but these systems have outages more often than you would believe.

As recently as November 2020 AWS had a major outage that affected thousands of systems. Everything from banking to video games to robot vacuums and doorbells.

Big deal. So, my vacuum didn't work for a little while. This is true, your vacuum isn't critical. But what if your company's payroll was in the cloud and you didn't get paid or you're bank card would no longer work or patient history wasn't available for a doctor to diagnose a critical patient. What if your business relied on systems to process transactions? How much money can be lost in 15 minutes in the stock

market if a trader can't access the system? Depending on the business this could be a lot of money! What's on the cloud can matter.

It's a lot harder for mistakes to happen at your own data center then it is at a hosted cloud service. As basic as this sounds, you're in control of your own data center and you care. You always know what's happening, who's doing it and when. Some companies know this and refuse to have their systems at the mercy of the cloud. Surprisingly, there are many that don't.

Choose wisely how you use the cloud.

**Burn #3 – (Nickeled and dimed to death)**
At one point I thought that I would jump with the rest, get an Azure account, and build a simple SQL database server on the cloud for testing purposes. What a mistake that was!

First, it's complicated. Now keep in mind I've built SQL servers from scratch in the past. Hell, I built a whole enterprise level private cloud at a data center utilizing the same VMWare software hosting companies do. But this was different. You need to learn how to navigate through their web portal to get anything done. The options are in the gazillions! Then there are profiles, security options, service tiers, etc. You need a whole training regimen just to figure it out. Okay. This is to be expected, I guess. But that's not what made me give it up.

For my scenario, which was just an SQL Server (maybe a couple of hits a day), it was just too costly. Somewhere around $350 per month. If I wanted more of anything, diskspace, memory, CPU it was another nickel here and a dime there.

---

*Every time I tried to do anything it was costing me more.*

---

In their defense I'm used to having a lot of power on my servers. Hardware is basically pretty cheap (relative to what I had to pay in Azure). Instead, I bought a server (hardware) and the SQL Server licenses to run it for a total of $1,500. With more CPU, memory and disk space than the Azure configuration. Granted, I don't have the hardware redundancy abilities that Azure has. If my hardware goes bad, I need to fix it. But I didn't need the level of redundancy that Azure was giving me anyway. If I really wanted hardware redundancy, I could purchase a little more equipment and had that also.

I do have disk redundancy, data backups and a UPS (Uninterruptable Power Supply). Other than my time to set it up, it paid for itself in 5 months. For more power and full control to do as I wish. It's been functioning fine for over 2 years and still going strong so I'm happy.

By the way, if you're planning on moving to the cloud, you'll still need IT staff to manage it, they will still constantly ask your CFO for more budget, hosted services

are not cheap, and it won't make any of your systems run faster. In fact, they will probably run slower. You can of course get them to run fast by paying more money, which brings you back full circle to where you started now doesn't it? You moved to the cloud to save money. However, this time, you've lost control and end up paying more in the long run.

Remember, the cloud is great until it isn't. Can your company afford an outage? Because it will happen.

## Why I Hate Stored Procedures

During my career as a technologist, I've had the unique opportunity to be in charge of Support, Development and Quality Assurance (QA) departments all at the same time. If it does anything to someone with a developer background, it humbles you.

As strictly a developer or development manager you don't really worry or care how many calls come into support. It's the support managers job to manage the customer. You have very little empathy for the end user. Okay, you may think about it a little, but not really.

Once you start dealing with customer complaints for things that can be fixed in an hour by a developer, you start to change your focus in development and QA. Instead of just new features, you begin to harden the ones you've already delivered that are causing grief in support.

The idea of delivering software that is not fully tested just makes you sick. Customer issues and the work surrounding them is not something you want to deal with all day long.

Which brings us to Stored Procedures. Ask a developer and they'll tell you they're great. They can encapsulate a bunch of data instructions that will run on the database server really fast. Because they are stored in the database and can be nested together, if something is wrong, you can just fix it in one place and it fixes it everywhere.

---

*You can also break it in one place and it breaks it everywhere.*
*Yikes!*

---

Yup. They do run faster. But how much faster? As it turns out, not a significant amount more than just writing the commands in a program and sending the instructions to the server.

If you're a DBA and your building routines only using the database then you have no choice but to use stored procedures. But as a software development company building software you should not make stored procedures part of your core system.

The delivery and maintenance of them at the customer site is unmanageable! You wouldn't know this if you're a developer, because you don't feel the pain.

To prove this to a developer, who had issues with my no stored procedures policy, I assigned them to the support department to handle some client calls. In less than 2 days they said: "This is ridiculous. I get it. Can I go back to development please? I won't create any more stored procedures!"

Since the commands in a store procedure can be replicated in a software program and there is no significant performance difference, and they are very difficult to manage at the client site for support, then why have them?

When you send a program to a client it has a version and it's easily tracked. When you send a stored procedure that gets inserted into a client's database it's not. When you send a program to QA they know what to test. When you send a stored procedure that can be used from 100 other stored procedures, QA doesn't know what logic path to test. They don't know what part of the system could be negatively affected by the new changes.

Why all the support calls? Because every time someone fixed a stored procedure (that wasn't fully tested by QA) it inevitably would negatively affect other areas of the system. Requiring another fix (that wasn't fully test) which again …you get the idea. It's a viscous circle of bad.

Even when we finally got it right, the client would restore a backup of their database (which did not have all the new store procedures) and the problems would start all over again. If they switched from a production database to a test database, same bad results. They would end up with a mix of good and bad stored procedures in multiple databases.

If QA asked the developer what areas of the system the stored procedure was affecting, so that they could focus their testing efforts, the answer was always; "I don't know." There were just too many logic paths it could affect.

To stop the support calls I needed QA to properly test. To do that we had to eliminate store procedures.

There are other ways to share common routines without the use of stored procedures but that's for another day.

How does all of this pertain to reporting? If your report relies on a store procedure and it just stops working correctly one day don't be surprised.

In addition, many stored procedures are created with CURSORS which is what a friend of mine calls "Loopy Code". This type of script is slower than a basic set-based query.

CURSORs utilize a row-by-row execution (row-based) process whereby queries utilize a batch (set-based) process.

CURSORs can be converted to set-based queries which can yield tremendous performance benefits but that will have to be for another time.

## Fragmented Indexes

How do indexes get fragmented and why do I care?

When an index gets fragmented, accessing the data with a query slows down. A fragmented index can slow query times exponentially!

---

*I've seen hundreds of databases where the indexes were fragmented and the DBA wasn't even aware.*

---

Implementing a simple maintenance plan can prevent the problem.

Since you went through all of the effort to make sure your query was using an index to avoid the slow TABLE SCAN, then the least you can do is make sure it's working efficiently.

In some circles I'm famous for explaining what fragmentation is with an example we can all relate too. Since it seems to work, I'll do the same here.

---

*How long does it take you to get dressed in the morning?*

---

5 minutes? 10 minutes? What if I told you that it could take you over an hour or more, if your clothes were fragmented?

The only reason you can get dressed in 5 minutes is because your clothes are indexed and it's not a fragmented index. They're in a specific order. When you want your socks, you go directly to the sock drawer and get them. When you want your underwear, you go directly to your underwear drawer and get those. Shirts in the closet, etc.

When you do the laundry or buy new clothes you put them in the proper places so that every day you get dressed it's efficient. You're putting your clothes in an order that you know you will access later. An index works the same way with your data.

An index is ordered in a specific way so that when your query runs it can find your data fast. When an index is fragmented, your database server has to hunt around to find the index information used to get your data.

If your clothes were in one big pile it would take you longer to get dressed because you would have to sift through them all to find what you needed. That's your clothes with no index. (TABLE SCAN) A drastically fragmented index can be just as bad or worse.

What if some socks were in the attic, some pants were in the garage, others in the basement? Maybe a couple of shirts in the shed and your underwear was in a box in the living room. Other articles of clothing you had at a friend's house. How long would it take you to get dressed then? That's a fragmented index.

When I tell this story everyone asks; "Why would the clothes not be where they're supposed to be? How did they get fragmented?"

---

*There's nothing you can do to avoid fragmentation, but you can fix it and keep from getting out of hand.*

---

When you buy new socks where would you put them? In your sock drawer of course. If the drawer is completely full where would you put them? In another drawer maybe? What if all of the drawers were full? You may have no choice but put them in the next available space. Which just happens to be the attic.

That's how fragmentation starts. Combine this with the fact that all of your clothes are different sizes so even if you had room in your sock drawer new pants won't fit. They need to be stored somewhere else.

As data gets added and removed from your database, index entries are removed and added over and over again. Indexes in a contiguous location in your database are great, but they can't stay that way for long. The server starts putting things where ever it can. (Coincidentally, saving files on your hard disk works the same way.)

---

*Since you can't stop fragmentation you need to mitigate it.*

---

This is done by periodically rebuilding your index. Which starts from scratch, like emptying out all of your drawers, finding all of your clothes and putting them back in order.

Your DBA can do this as a maintenance plan weekly or even nightly depending on the size of your database.

There's also a query that can be run to report how fragmented your indexes already are:

```
SELECT t.name AS TableName, ISNULL (s.name,'HEAP') AS IndexName,
       avg_fragmentation_in_percent AS FragmentationPercent
FROM sys.dm_db_index_physical_stats(DB_ID('AdventureWorks'),NULL,NULL,NULL,'DETAILED') st
LEFT JOIN AdventureWorks.sys.indexes s ON st.object_id = s.object_id
            AND st.index_id = s.index_id
LEFT JOIN AdventureWorks.sys.tables t ON s.object_id = t.object_id
WHERE avg_fragmentation_in_percent > 40
ORDER BY avg_fragmentation_in_percent DESC, t.name
```

This query run on a Microsoft SQL Server will return a list of all indexes where the percentage of fragmentation is greater than 40.

To execute this query on your database you must replace all 3 highlighted areas where AdventureWorks is used with the name of your database.

A sample resultset of this query is shown below:

Results | Messages

| | TableName | IndexName | FragmentationPercent |
|---|---|---|---|
| 1 | SalesOrderDetail | AK_SalesOrderDetail_rowguid | 100 |
| 2 | TransactionHistory | PK_TransactionHistory_TransactionID | 100 |
| 3 | WorkOrderRouting | PK_WorkOrderRouting_WorkOrderID_ProductID_Operat... | 100 |
| 4 | Store | PXML_Store_Demographics | 98.4375 |
| 5 | Employee | AK_Employee_LoginID | 66.6666666666667 |
| 6 | Product | AK_Product_Name | 66.6666666666667 |
| 7 | ProductCostHistory | PK_ProductCostHistory_ProductID_StartDate | 66.6666666666667 |
| 8 | ProductDescription | AK_ProductDescription_rowguid | 66.6666666666667 |
| 9 | ProductListPriceHistory | PK_ProductListPriceHistory_ProductID_StartDate | 66.6666666666667 |
| 10 | ProductReview | IX_ProductReview_ProductID_Name | 66.6666666666667 |
| 11 | SpecialOfferProduct | PK_SpecialOfferProduct_SpecialOfferID_ProductID | 66.6666666666667 |
| 12 | BusinessEntityContact | AK_BusinessEntityContact_rowguid | 50 |
| 13 | BusinessEntityContact | IX_BusinessEntityContact_PersonID | 50 |
| 14 | BusinessEntityContact | IX_BusinessEntityContact_ContactTypeID | 50 |
| 15 | CountryRegion | PK_CountryRegion_CountryRegionCode | 50 |
| 16 | Employee | AK_Employee_NationalIDNumber | 50 |
| 17 | EmployeeDepartmentHistory | PK_EmployeeDepartmentHistory_BusinessEntityID_Start... | 50 |
| 18 | EmployeePayHistory | PK_EmployeePayHistory_BusinessEntityID_RateChange... | 50 |
| 19 | Product | AK_Product_ProductNumber | 50 |
| 20 | Product | AK_Product_rowguid | 50 |
| 21 | ProductModelProductDescriptionCulture | PK_ProductModelProductDescriptionCulture_ProductMo... | 50 |
| 22 | ProductProductPhoto | PK_ProductProductPhoto_ProductID_ProductPhotoID | 50 |

How to rebuild and reorganize indexes can be found here:
https://www.sqlmvp.org/how-to-rebuild-and-reorganize-index-in-sql-server/
(Sharma)

Simple example script to rebuild the indexes for a specific table:

```
ALTER INDEX ALL ON Sales.SalesOrderDetail REBUILD
```

After rebuild of all indexes the sample resultset now looks like this:

Results | Messages

| | TableName | IndexName | FragmentationPercent |
|---|---|---|---|
| 1 | Person | PK_Person_BusinessEntityID | 87.5 |
| 2 | Person | PXML_Person_Demographics | 87.5 |
| 3 | ProductModel | PXML_ProductModel_CatalogDescription | 83.3333333333333 |
| 4 | TransactionHistory | PK_TransactionHistory_TransactionID | 75 |
| 5 | SalesOrderDetail | PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID | 71.4285714285714 |
| 6 | WorkOrderRouting | PK_WorkOrderRouting_WorkOrderID_ProductID_Operat... | 66.6666666666667 |
| 7 | CountryRegion | PK_CountryRegion_CountryRegionCode | 50 |
| 8 | CountryRegion | AK_CountryRegion_Name | 50 |
| 9 | Employee | AK_Employee_NationalIDNumber | 50 |
| 10 | Employee | AK_Employee_rowguid | 50 |
| 11 | EmployeeDepartmentHistory | PK_EmployeeDepartmentHistory_BusinessEntityID_Start... | 50 |
| 12 | EmployeePayHistory | PK_EmployeePayHistory_BusinessEntityID_RateChange... | 50 |
| 13 | ProductProductPhoto | PK_ProductProductPhoto_ProductID_ProductPhotoID | 50 |
| 14 | ProductReview | PK_ProductReview_ProductReviewID | 50 |
| 15 | ProductVendor | IX_ProductVendor_UnitMeasureCode | 50 |
| 16 | SalesOrderHeader | PK_SalesOrderHeader_SalesOrderID | 50 |
| 17 | SalesPersonQuotaHistory | PK_SalesPersonQuotaHistory_BusinessEntityID_QuotaD... | 50 |
| 18 | SpecialOfferProduct | IX_SpecialOfferProduct_ProductID | 50 |
| 19 | Store | IX_Store_SalesPersonID | 50 |
| 20 | Vendor | PK_Vendor_BusinessEntityID | 50 |
| 21 | Person | XMLPROPERTY_Person_Demographics | 43.75 |
| 22 | Person | XMLPATH_Person_Demographics | 42.1052631578947 |

On the second resultset the top 3 indexes that were fragmented at 100% have moved.

The index named AK_SalesOrderDetail_rowguid which was at 100% now didn't even make it on the list. That's great!

PK_TransactionHistory_TransactionID went down to 75%
PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence is now 66.66%

Even after a rebuild or a reorganization, you can still have fragmented index percentages that seem high. Index Page size, number of pages, the size of the index, statistics, etc. are all technical reasons, beyond the scope of this document, which result in some indexes to remain fragmented.

*Having some fragmented indexes is normal.*

Rebuilding indexes periodically will keep things running smoothly. Talk to your DBA about this.

After the indexes are rebuilt, run your report queries again. You may be surprised and get a huge performance increase.

# REFERENCES


alteksolutions.com. *alteksolutions.com/blog/what-is-a-business-objects-universe*. 2020.

Behara, Samir. *c-sharpcorner.com/UploadFile/b075e6/improving-sql-performance-using-covering-indexes*. February 2016.

computerweekly.com. *computerweekly.com/news/2240113585/Almost-a-third-of-BI-projects-fail-to-deliver-on-business-objectives*. 2012.

dictionary.sensagent.com. *dictionary.sensagent.com/Entity-relationship_model/en-en*. 2020.

en.wikipedia.org. *en.wikipedia.org/wiki/Cargo_cult_programming*. 2020.

—. *en.wikipedia.org/wiki/Power_user*. 2020.

—. *en.wikipedia.org/wiki/Referential_integrity#Declarative_referential_integrity*. 2020.

—. *en.wikipedia.org/wiki/View_(SQL)*. 2020.

enterpriseappstoday.com. *enterpriseappstoday.com/business-intelligence/why-most-business-intelligence-projects-fail-1.html*. 2020.

Erkec, Esat. *sqlshack.com/how-to-use-sargable-expressions-in-t-sql-queries-performance-advantages-and-examples*. November 2017.

Investopedia.com. *investopedia.com*. 19 August 2019. Website.

logianalytics.com. *logianalytics.com/resources/bi-encyclopedia/ad-hoc-reporting*. 2020.

merriam-webster.com. *merriam-webster.com/dictionary/context*. 2020.

—. *merriam-webster.com/dictionary/deadlock*. 2020.

—. *merriam-webster.com/dictionary/know*. 2020.

—. *merriam-webster.com/dictionary/syntax*. 2020.

—. *merriam-webster.com/dictionary/think*. 2020.

microsoft.com. *docs.microsoft.com/en-us/sql/relational-databases/performance/execution-plans*. 2020.

—. *docs.microsoft.com/en-us/sql/relational-databases/statistics*. 2020.

—. *docs.microsoft.com/en-us/sql/relational-databases/views*. 2020.

mssqltips.com. *mssqltips.com/sqlservertip/6038/sql-server-derived-table-example*. 2020.

Nevarez, Benjamin. *red-gate.com/simple-talk/sql/sql-training/the-sql-server-query-optimizer*. 3 August 2011.

pareto-chart.com. *pareto-chart.com/about-vilfredo-pareto.html*. 2020.

Rouse, Margaret. *searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI#*. 2020.

Sharma, Naveen. *sqlmvp.org/how-to-rebuild-and-reorganize-index-in-sql-server*. July 10 2018. 2020.

techopedia.com. *techopedia.com/definition/1187/database-administrator-dba*. 2020.

—. *techopedia.com/definition/9269/scalability*. 2020.

tutorialspoint.com. *tutorialspoint.com/dbms/dbms_data_models.htm*. 2020.

—. *tutorialspoint.com/dwh/dwh_data_warehousing.htm*. 2020.

—. *tutorialspoint.com/sql/sql-cartesian-joins.htm*. 2020.

Wagner, Bert. *bertwagner.com/posts/or-vs-union-all-is-one-better-for-performance*. 20 February 2018. 2020.

West, Randolph. *bornsql.ca/blog/nested-views-bad*. 14 February 2018.

wikipedia.org. *en.wikipedia.org/wiki/Database_connection*. 2020.

—. *en.wikipedia.org/wiki/Extract,_transform,_load*. 2020.

—. *en.wikipedia.org/wiki/SQL*. 2020.

# INDEX

## E



## F



## G



## H



## I

## J



## L



## M



## N



## O



## P

## Q



## R



## S



## T

## U



## V



## W