SQL

Simple and Effective Strategies to learn SQL Programming

Daniel Jones

SQL Strategies:

Table of Contents

The follow eBook is reproduced below with the goal of providing information that is as accurate and reliable as possible. Regardless, purchasing this eBook can be seen as consent to the fact that both the publisher and the author of this book are in no way experts on the topics discussed within and that any recommendations or suggestions that are made herein are for entertainment purposes only. Professionals should be consulted as needed prior to undertaking any of the action endorsed herein.

This declaration is deemed fair and valid by both the American Bar Association and the Committee of Publishers Association and is legally binding throughout the United States.

Furthermore, the transmission, duplication or reproduction of any of the following work including specific information will be considered an illegal act irrespective of if it is done electronically or in print. This extends to creating a secondary or tertiary copy of the work or a recorded copy and is only allowed with express written consent from the Publisher. All additional right reserved.

The information in the following pages is broadly considered to be a truthful and accurate account of facts and as such any inattention, use or misuse of the information in question by the reader will render any resulting actions solely under their purview. There are no scenarios in which the publisher or the original author of this work can be in any fashion deemed liable for any hardship or damages that may befall them after undertaking information described herein.

Additionally, the information in the following pages is intended only for informational purposes and should thus be thought of as universal. As befitting

its nature, it is presented without assurance regarding its prolonged validity or interim quality. Trademarks that are mentioned are done without written consent and can in no way be considered an endorsement from the trademark holder.

Introduction

Congratulations on downloading *SQL Strategies* and thank you for doing so.

The following chapters will discuss the various strategies that you can use while you work with SQL so that working with SQL becomes even easier than it was before.

There are plenty of books on this subject on the market, thanks again for choosing this one! Every effort was made to ensure it is full of as much useful information as possible, please enjoy!

Chapter one: Versioned Data

When you are working with versioned data, you are not going to be doing updates to your files, instead you are going to insert any new data that you need in that file. This makes it to where you can find the latest version of the file you want to work with. While it would seem like this is too complicated, what is actually done is that a new file is created and saved with the date of the change. The change to the table can be one row or several, it just depends on

what you are doing.

In order to gather data from your code is going to look very similar to the code that you have already been using.

Example

Select * from odb file s

Inside join odb version l

On s file = l file

And l version = (select bottom (5)

V4.version number

From odb version 4

Where v4.file number = 1 file

Order by v4 file number inc

V4. File inc

);

As you are going about your search, you have the option to use the max or even the row number so that you can use a different function and you do not have to rely on the joining function. Each function that we discuss is going to give you the data that you are searching for in one way or another.

Data and the data base

When you are working with versioned data, you can create a data base that is just going to be for those files so that you can see the different versions that you have as well as note any of the primary keys that you need to know about before you work with the data.

Simple testing

A test can be run in order to make sure that your search is starting where you are wanting it to that way you do not have to bother with subqueries, joins, or anything else that can complicate your code.

Every search that you do is going to have an execution plan, as well as execution time. You may also want to run a freesystemcache or a dbcc freeproccahe code before each search so that you can ensure that your system is not cluttered. Clearing up your system is going to make sure you are getting the proper comparison.

Example:

Select bottom (5)

From dob.version l

Where l file number = 587

Order by l file number inc,

L version number inc ;

Once you have fun the test, you are going to get a single scan with three different results. After you have gotten that you are going to run a max query with a sub select.

Example

Select l. *

From obd. Version l

Where l file number = 587

And l.version number = (select max (v4. Version number)

From odb [version] v4

Where v4. File number = l. file number) ;

Once again you are going to get three different results.

The last search that you are going to do is the row number version search so that you can see how many rows are in each number.

Select l. *

From (select l. *,

Row_number () under (order by l. version number inc) as

Rownum

From odb. [version] l

Where l. file number = 487

) as l

Where l. rownum =6;

With your results from this search, you are still going to get three results, however it is going to be more complex

than the other results. What you will notice is that this search is going to be a little slower than the other searches because there are going to be more to the results that you get than the other ones.

The plans of execution that you followed when you were doing the top and max searches are going to be identical to what is known as the clustered index seek. But, in looking at the row number search, your plan will be completely different. Your costs for your row number are going to be buried inside of the clustered index seek so that your search is not adding extra scans or reads that you do not need to have.

Join

The join function being used on a version table is going to end up giving you a single result that comes from one row. The first thing you need to do is your yop function.

Example:

Select l [file title]

L [file number]

T [description for that version]

T [file number]

```
From dob [file] l

Join dob [version] t

On l [file number] = t [file number]
And t [file number] = t [file number]

And t [ version number] = select top (4)

V4. Version number

From dob [version] v4

Where v4. File number = t. file number

Order by v4 file number,

V4. Version number inc
```

Where l [file number] = 9685;

Your results are going to be returned in thirty seven milliseconds. Even though you only did one scan, you are going to get multiple results for every table that you scanned.

The next step is to run a max search.

Example

Select l [file title]

L [file number]

T [description for that version]

T [file number]

From dob [file] l

Join dob [version] t

On l [file number] = t [file number]

And t [version number] = (select max (v4. Version number)

From dob [version] v4

Where v4. File number = t file number

Where l [file number] = 9685;

It is going to take around thirty two milliseconds to get your results back and there will be five reads against each table you ran.

In doing the max search, you will realize that your optimizer is going to work the same way that it did in our example for the testing section. However, your optimizer for the top function is going to create a nested loop with the data instead of giving you each result separately.

The top function ends up running the scan twice giving you six different reads per table as well as joining all the rows that are occupying your table. So, what

is going to need to happen is that the version number does not need to be referenced, the file number needs to be referenced instead. Therefore, you are going to need to change your code to reflect this change.

Example:

Where v4. File number = l file number

Now whenever the search is run, there is only going to be a single scan with six different reads based on the table that you referenced in the search.

So, when you run the top function, you are going to telling the system that the

join has to be forced rather than to create a loop with the records that are inside of that particular version. So, what is going to happen if you end up changing your search once more? Before you run your search this time, you are going to change it to where the apply statement is now going to be a part of the join statement.

Example

Select l [file name],

L [file name],

T [description for the version],

T[version number]

```
From odb [ file] l

Cross apply (select bottom (5)

V4. Version number,

V4 description of the version

From dob [version] v4
Where v4 file number = l. file number

Order by v4 file number,

V4 version number inc

) t
```

Where l [file number] = 9685;

With the scan that you run now, you are going to get five different reads that are going to be for each table that you ran. The execution plan looks very similar to the plans that you have seen before.

But, what does the apply method actually do? This method is going to make it to where one row is going to be taken from the table and find the top matches for that row inside of the table so that the joins are not sorted again and multiple scans do not have to take place.

Row number

Now, how does the join affect the row number?

Example:

Select f. *

From (select l [file name],

L [file number]

T [description of the version],

T [version number],

Row_number () over (order by t. version number inc) as rownum

From dob [file] l

On l [file number] = t [file number]

Where l [file number] = 9685

) as f

Where f. rownum = 2;

The results that you get are going to be five reads that are returned within forty eight milliseconds and the execution plan that was followed by the system is different than any other plan that you have seen before.

While the search is only going to run

once on every table, the clustered index operation is going to be performed as well. Just like the others, the joins are going to go through an operation that creates nested loops. But, what happens next is that the top operator will get segmented through the operator based on the expression that is inside of the system and pulled from inside of the search most likely by the order by statement.

It is going to look complicated, but you do not have to worry about that because the system is going to take care of it all for you so that you are only getting one par per operation.

Full data set

Now is the point in time that you are going to want to join all the data together. ultimately you will want a list of all of the publications in the table that demonstrate what the max version is. You are going to determine what all the points are that match inside of all the tables are.

Example

L [file name],

L [file number],

Up [version number],

T [name of the author],

Up [date of publication],

Up [number of publication]

From dob [file] l

Cross apply (select bottom (5)

V4 version number

V4 file number

V4 description of the version

From dob [version] v4

Where v4 file number = l file number

Order by v4 file number

V4 version number inc

) as l

Join dob . [published] up

On up [file number = l [file number]

And up [version number] = (select bottom (5)

Up5. Version number

From dob publication up5

Where up5 file number = l file number

And up5 id for publication = up id for publisher

Order by up5 file number

Up4 version number inc

)

Join dob [publisher] n

On up [number for publisher] = n [number for publisher]

Where n [file number] = 56897

And n [publisher number] = 3687;

Your search is going to be completed in fifty three milliseconds and you are going to get four different results for your table.

The execution plan is going to be complicated to read, however, twenty percent of the batch is going to go through the nested look join therefore your plan becomes clean and simple for you to read.

Here is where you will see the max version with the from clause.

Example

From dob [file] l

Join dob [version] s

On l [file number = s [file number]

And s [file number] = (select max (v4 version number)

From dob [version] v4

Where v4 file number = s file number

)

Join dob [publication up

On s [file number] up [file number]

And up [version number] = (select max (up4 version number)

From dob publication up4

Where up4. File number = l file number

And up4 version number >= d [version number]

And up4 publiser number = up publisher number

Join dob [publisher] e

On up [publisher number = e [publisher number]

Where l [file number] = 68945

And e [publisher number] = 3567;

After forty six milliseconds, the scan will break down the publication, version, publisher, and documents in the table and give you what the reads are.

Once again, your execution plan is going to look complicated and it is going to have nothing but the nested loop operators that work with the top function along with the clustered index seek function. While most of the time there is

an easier way to come up with something simpler, you are not going to find anything simpler than this.

What you may find interesting is that your top and max functions are going to act as if your top function has been resupplied. The biggest difference that you will find is that the tables are accessed in a different order than they have been before even though your search is the exact same as it has been the entire time. Should your searches be run at the same time and the results put side by side, you will notice that there really are not any differences that can be found without going through the entire code line by line.

At the point in time that your search changes so that you can find the row number you are going to find a few differences.

Example

Select l [file name]

L [file number]

S [description of the version being used]

Up [version number]

E [name of the author]

Up [date file was published]

Up [number of publication]

From dob [file] l

Inner join (select row_number() over (division by v4 file number order by v4 version number inc) nr

V4 version number

V4 file number

V4 description

From dob [version] v4

) as l

Left outer join (select row_number()
under (division b up4 file number,

Number for author

Order by up4 version number inc) nr

Up4 version number
Up4 file number

Up4 date of publication

Up4 number of publication

Up4 publication number

From dob publication up4

) up

On up [file number] = s [file number]

Join dob [author] d

On up [author number] = s [author number]

Where s [file number] = 85964

And a [author number] = 5698

And s. nr = 5

And up nr = 9;

At forty four milliseconds you are going to get some scans that you have never had before. However, the search is going to give you the same data that you were seeing before but it is going to give it to you in fewer scans and reads. There are some ways that this search method is going to be more complex than others but it is going to read easier than the others.

Your execution plan is obviously going to be more complex because the number of performances will be increased. Rather than having five cluster index seeks, you are going to have four. You are going to have to do more work by

moving the data into different rows, however there will be fewer nested loop joins.

However, what if our results are going to change? Doing the same exact search that you did before, you are going to get even more data returned from just one part. For this example, we will remove the author number from the where clause so that when the search is run, you are only going to get forty nine percent from top and fifty from max.

What is the difference though?

Well, the difference is that you are going to be using part of the aggregate operator

so that your data sets are larger and will cost more. It does not matter how many times you execute this plan, you are going to get the same results and the same reads. When you add in the row number search, your results are going to get interesting. When you look at the execution plan, it is going to be ranked as one of the most costly plans that you can run.

This is because you are going to get up to forty eight reads for one scan on the table. That is a lot of reads! Therefore, that is a lot of data off of one scan. But, when you put them side by side, you are going to notice that the data is going to be identical to what you got before. Your

execution plan did not change nor did the results that you were getting.

Any differences that are there, are going to be so small that you are not going to really notice them.

Chapter two: Archiving Your Data onto the SQL Server

Sometimes data is just not needed anymore. When you do not need data anymore you are going to want to archive it onto a server so that you not only have access to it at a later date, but it is not in the way of you performing the tasks that need to be completed.

When there is data that you do not need any more cluttering up your data base, you will not be able to do any maintenance on the table due to the

performance issue you will be facing. As a developer, you know that data cannot just be deleted because it is not being used anymore. That is why you are going to want to put the data on the server for later use just in case.

In trying to figure out which method that is going to be best for you, you need to look at your data base closely and address a few issues that may arise first.

- Look at the legal requirements for the company you are working for. You may come to realize that there are no requirements for retaining data which means you could delete it, but this is not recommended. But, other

companies require data to be kept for up to seven years. Once you understand the policy that your company has, you will be able to better make your plan for archiving the data.

- Look at the data that is in the file. There may be some data that you may still need. Despite that, you need to take a look at the amount of data that you have stored so that you can understand just how much data you are archiving on a quarterly basis.
- Make sure you are archiving the data properly. If you do not know how to archiving data properly and are afraid you may want to be sure and meet with your IT department so

that you can figure out the correct steps to archive the data properly. It is also a good idea to keep up with any updates that may come out in the process that is being used to archive data.

- Have a feasible reason behind why you are wanting to archive the data. Just because you can does not mean that you are going to be able to archive the data. Some of the better reasons behind archiving data that IT and your manager are going to appreciate are:

- Better experience for the user
- Back ups that take up less space.
- Performing maintenance that is going to up the performance of

the system.

- Faster times for restoring the data
- The data base is easier to manage.

- If you work with a lot of data, you are going to want to set it up to where a specific file is archived either daily, weekly, or monthly so that your system is running to its full potential.

At the point in time that you can fully understand how it is that data is going to be archived on your system, you are going to be able to make a better decision as to how you are going to want to archive that data.

The data has to be on the same data base that you use all the time then:

- Modify the ending code so that you can see the data in the table without actually pulling up the table.
- Insert and delete the data that you are archiving into different tables and make sure that you save the file in a way that is going to make sure that you know that is an archived table.
- Merge the old data with new data if the users are still needing to see that data in order to make sure the system runs as it should.
- Make new talbes on a separate disk so that you can improve the performance of the data base.

If you can move the data to a new data base because it is not needed on that data base then:

- Make sure that your users can still access the data or request access to the data so that applications can be used as they are supposed to be.
- Insert and delete the table into a different data base or server so that it is not taking up the room on your current data base.

If the data that you are trying to archive is not needed anymore:

- If the data is going to be needed

because of some reason that you cannot predict, then make back ups.

- Be sure that the data can be restored from the back up and then delete the data off the server.

Automatic archiving options

- If there are large amounts of data that have to be changed, then you will want to do maintenance on the server as part of the archiving process. This is going to make it to where your server runs faster.
- Should you not be able to make it to where data is automatically archived, then you will want to build a process that is going to move the

data from the system to the archiving system.

Once you have made sure that you are following all company policy and that you are archiving the data correctly so that it is not having to be deleted, now you are ready to archive the data.
Make sure that you test your process to make sure that there is enough time in between making sure that your other work is being completed that you are going to be able to archive your data properly. If your users are just going to delete the data because they do not find it useful, then you may find it easier for you to move the data to a different data base all together for a few months so that

if it is needed you will have access to it. If after about three months the data is not needed, then you are going to be able to just delete the data and clear up the disk space for something else.

Chapter three: Security on the SQL Server

Safety is important to everyone. The more secure your server is, the better your users are going to feel about going through your server and make sure that the data that is on your server is going to be protected from those who may want to hack into your company's server.

- Get rid of the log in for guests. Doing this will make it to where people you do not want having access to your data base will be removed and if they want to have

access, you will have to give it to them.

- Common work groups are going to be the ones that are most likely going to have access to the data base. To give the whole group access you are going to need to create a profile for the entire group so that they are able to log into the data base and gain access to the data that they are needing to work with.
- An anonymous long in is going to work best if you are working off of a stand alone server and therefore all other resources that come with the data base are found on this specific IIS machine. Making an anonymous logon will make it to where you can

get onto the domain with the account but you will be required to provide some authentication to ensure that someone is not trying to hack the account and get into the data base that should not be there. Another great thing about the anonymous account is that with SQL it will satisfy the pipes authentication process.

- IIS applications need to have security too. When using the IIS on SQL, it is best you use a standard security mode so that you have flexibility when sending your applications out to various work stations.
- Some data needs to have who can view it. Restricting views will make

it to where only certain people can modify the data let alone who can see it. You can restrict access down as far as you want. If you only want certain columns out of the table to be seen, then you can.

- If the data that you are working on data that needs to be secured more than normal data, you may want to think about used stored procedures so that you can ensure only users who have the proper authentication can view and modify the data. Or, you can fix it to where no one can modify the data but the creator of the table.
- Not all users are going to use the same tools and there are times that this is going to be a problem. To fix

this issue, you can assign one or even two different tools to each user account. Having two tools will allow for the user to have a choice of which is easier for them to use, but ensure that they are getting the results that you want. Or, will make it to where you get more uniformed results.

- Audits are necessary whether you are working alone or with others. The data that you want to be audited will be found by having triggers created that audit be created automatically each time that the table is modified. You are not going to have to have permission from the user to do this. This is going to be a permission that only the

administrator has.

Chapter four: Azure SQL Data Bases

The Azure data base is going to work off Windows and is going to probably be a little different than just working with the SQL server on a Linux system. One of the things that you need to make sure you remember when you are working with the Azure data base is that it is not just another version of SQL. While it does use SQL, it is going to be a completely different platform for SQL.

Here are a few strategies that you may want to keep in mind when it comes to

using the Azure data base.

- Make sure that you have a disaster recovery plan. The Windows Azure SQL Data base (WASD) does not have any way to back up the data that you are working with. So, before you start working with WASD, make sure that you have a way to back up and recover anything that you may end up losing due to system crashes or freezes. It is best that you have this plan in place before you even begin to use WASD.
- Have the proper resources that you are going to need while working with WASD. You are going to learn effective ways to use WASD as you

go through using it if you have never used it before. However, you should do the research that is necessary to make sure that your resources that you are currently using are going to be right for what you are trying to use WASD for. Just because you are using WASD does not mean that you have to get someone completely new to work with the system. The skills that someone else has may work with WASD, you are just going to need to make sure that their skills with SQL are going to cross over to WASD.

- Test the tools that you are using. The tools you are using on other programs may very well work with the WASD. If you do not have to

spend the money, then, why do it? Before jumping onto server take the time to discover what it is that you are going to need to work with the data base. This could even include hiring someone to teach you how to use the tool or to use the tool for you. Also, you need to make sure that the tool works before you go on and spend more money that you may not need to spend. If it does not work after a trial period, then is the time that you are going to want to switch tools or people using the tools if it is in your budget to do so.

- Break the changes. Since WASD is a different platform than what you are probably used to working with,

you are going to need to look at it as the fact that it is going to behave differently than the version that you are going to be. If you expect it to react the same way as a regular SQL platform, then you are going to be disappointed. There is a list of features that WASD supports and you should not assume that what you have been doing this entire time is going to continue to work for you. Once again, you will need to ensure that you do the research that is required to make sure that you understand WASD so that when you go to use it, you are not moving backwards.

- Do not let mistakes that you make on the platform to stop you. You are

going to make mistakes, this cannot be avoided. However, you need to make sure that you learn from these mistakes so that you do not keep making these mistakes and thus setting yourself back on learning about the platform and the way that you use it.

- You are going to learn a lot from troubleshooting. If you already have trouble shooting skills, then you are going to be able to put these skills to use when you are working with WASD.

Chapter five: Compressing Data Inside of SQL

There are functions that are built into SQL that are there in order to assist you in decompressing or compressing the data that is on the server.

- Compression is when the data is made smaller thus returning a binary string of data.
- Decompression will take the binary data that you got from the compression and put the data back into its original text.

It does not matter which function you are working with you are going to be using an algorithm that works with the standard GZip in order to allow you to compress the data and then send that data to the SQL server. If you are wanting to decompress the data, you are going to do it on the client side.

Example of compressed data

Create table books (

_title nit main key identity,

Title nvarchar (full)

Subtitle nvarchar (full)

Info varbinary (full)

)

From there you are going to be able to load the compressed information into your column.

Insert into book (author, subtitle, info)

Select booktitle subtitle, compress (any other info) from. Book. Book

Or if you are wanting to, you can send that data right back to the client who can then decompress it should they want it decompressed.

Select title, sub title, decompress (info) as primary

From books

Lastly, you can put a computed column in that is going to decompress the data for you.

Alter table books

Add info_characters as cast (decompress (info) as nvarchar (full))

What if you want to know what the compression rate is? You can figure that out by creating text and then compressing

it in order to see the ratio.

Declare @ characters nvarhar (full) = (select bottom five * from sys. any_item for json path)

Select length of data (@ characters) as primary length of data (compress (@ characters)) as compressed, 5.9 * length of data (@ characters) / length of data (compress (@ characters)) as ratio.

The number of rows that are formatted can be changed so that it works with the text that you are currently working with. Either of these functions are going to assist you in choosing what kind of data has to be compressed in the table.

Using compression may even aid you should you find that you are working with text data but it has already been compressed however has not been compressed to a good ratio. If that is the case, you are going to want to spend CPU cycles to fix this issue.

Chapter six: Population Stability Index (PSI) Inside of the SQL Data Base

SQL is one of the best programs out there that you can work with when you are trying to work with probability. As you are trying to build up a data warehouse, there are going to be cubes that you are going to use that is going to aid you when it comes to comparing values that are in a report versus the value that you are going to get form systems such as SAP or any other transactional system that you may be working with.

There may be times that you find that you are going to be working with reports that are going to compare data that is going to change throughout the use of that table. Even if it does not change, you are going to have to work with figures such as financial figures or population figures. Anything that may change.

For this example, you are going to look at the number of people who have paid their car payment that month and how it is going to affect their credit. What you are going to be doing is working with the Population Stability Index otherwise known as PSI.

Your syntax is going to look a little something like this:

$$PSI = \sum (a-b) * \log [A/ B]$$

The formula is just the fancy and more complicated way to tell you what it is that the system is going to be working with. However, when you break it down to where you can understand exactly what the system is doing, you are going to need to change the sigma sign to sum() and therefore you are going to understand the formula!

Now, the formula actually looks like this.

PSI = sum() (a-b) * log [a/b]

That is a lot easier to understand isn't it? And now you are going to be able to do the equation if you so desire to.

Example

Declare @ car table (number nit identity (2, 6), present month nit, prior month nit, population number float population prior float)

Insert into @car table amount (325164, 1236589, 987, 456)

Test data

Select * from @ car table

Calculating PSI

Select sum(population number, population prior) * log (population / population prior) as PSI from @ car table

Group by population, population prior

Now that you can see the equation and how it is going to work into SQL code, it seems a lot less intimidating and you are not going to have to worry about not putting the formula in correctly.

Your PSI is going to show the shift from

one month to the next if there is any shift. You are going to be able to create alerts that are going to be based off any values that are going to be returned to you.

Chapter seven: Hyper Threading- How it Works and When You Should Use it

What is hyper threading and how does it work with SQL? The first t hing that you should have some knowledge on is the structure of a CPU. The CPU if you do not know is what makes the computer run for example an intel core i5.

The most common thing that people assume is that it has to be bigger to be better. But, when working with SQL, the

software will make a huge difference and can end up saving you and your company a lot of money, but only if you can configure your software properly.

The first thing you are going to want to do is download the CPU-Z so that you can get the proper information on your processor that you are working with. You can go to www.cpuid.com/softwares/cpu-z.html in order to get the software. The link that you are wanting to go to will be located on the right hand side of the page.

In the event that you are running off of two cores, thanks to hyperthreading, you are going to be able to gain threads to

work off of. This will be consideredworking with logical CPUs. But, if hyperthreading has been disable on your system, you are going to only be using two threads.

With hyperthreading enabled, you are allowing it to increase your threads before assigning some cache to L3 where the threads are located. Therefore, if you are working off of 4mb of cache then it is going to send out one mb to each thread. But, working with hyperthreading off, you will have 2 mb per thread.

Hyperthreading should be enabled so that you have more for your data to go

through. Larger threads of data can be proccessed with less of a problem than if you were working with hyperthreading off.

However, hyperthreading is a personal choice based on what it is that you are doing with SQL. For some projects you are going to want to have your hyperthreading turned off because it is just going to get in the way of what you are doing. But, for other projects, you will want it on because it is going to make dealing with the data that much easier.

Chapter eight: T-SQL Analytic Functions

There are some SQL problems that you are going to discover that will end up being too complicated to know how to write the code for. There are ways around this though, you just have ot know what it is that you are doing before you can do it. For any one problem, there may be up to around eleven different ways that you are going to be able to solve that problem with SQL.

Example

Set A

A004 b004 c004 5

A005 b005 c005 10

A006 b006 c006 15

Set b

A007 c007 d007 20

A007 c007 d008 20

A009 c009 d009 30

The outcome that you are wanting to have is that each set be divided by

adding the similar ones together and then multiplying them by your other ones. There are going to be some that are going to come out as null. This is going to see somewhat complicated, so let's look at an example.

Example

Declare @set a table (a varcar(5), b

Varchar (5) c varchar (5) value nit)

Insert into @set a amount (a004, b004 c004, 5)

Insert into @set a amount (a005 b005 c005 10)

Inset into @seta amount (a006 b006 c006 15)

Declare @set b table (a varchar 20 c varchar20, d varchar20 amount nit)

Insert @set b amount (a007 c007 d007 20)
Insert @ set b amount (a007 c007 d008 20)

Insert @ set b amount (a009 c009 d009 30)

Select * from set a

Select * from set b

From here you are going to check the results and use an inner join as well as a left join so that you can locate which one of them is going to be clsoely matched and give you the results that you are wanting. When you test it more, you are going to discover that there are other ways that you can find a solution to this problem.

Select * from @set a inner join @ set b on a =b = c

Select * from @ set a left join @ set b on a =b =c = c

Those are just two ways to find a

solution to the same problem. As stated before, you may discover that there are eleven diiferent ways to solve the same exact problem and get the solutions that you are wanting. However, you do not have to go through every possible way to find them all. You can just stop at the one that gives you the results that you are looking for.

Chapter nine: Do Not Copy, Clone It!

There may be times where you find that you are going to need to have a copy of your table but using the create table and entering all the data into the table is not going to be the best option for you because you need everything to be identical.

When you use MySQL, you will have the best solution

1. Use the create table command that makes the table exactly like the table

you are wanting to copy.

2. Change the code so that the table title to the table that your cloning so that you know that the table will be the same across the entire board.
3. Should you need the entire table be copied, you will use the insert into and select statement.

Example

Still going with the family table that we have been using for our examples.

Step 1: you need to get the entire structure of the table. This means that you need to get the code for the table so that it can be cloned.

Step 2: the table will need to be retitled and another table created.

Step 3: once step 2 has been completed, you can now clone the table inside of the data base. You can copy the data out of the old table and use the insert into and select statements to get the data into your new table

And congratulations! You have now cloned your table!

Chapter ten: Sequencing with SQL

Sequences are pretty easy to understand. Most likely, you use them everyday and do not even realize it. A sequence is going to be something that comes in an order whenever it is called upon. But, a sequence is going to be used inside of a data base because there applications that are going to be required that every row of data that is inside of the table is going to have a unique value and the sequence is an easy way for you to generate these values.

Auto_increment

One of the easiest ways to use sequences in MySQL is to define your columns with the auto_increment command while allowing MySQL to take care of everything else that needs to be taken care of.

Obtain auto_increment values

There is a function that is going to be used in SQL so that you are able to make sure that any client that you are going to be working with is going to be able to understand how they need to set out the SQL statements. The function that you are going to use is last_insert_id ( ).

PHP and Perl codes are going to be able to give you exclusive functions that you can use in order to get the auto-incremented value that was last recorded by the system.

Perl example:

You will mysql_insertid function when you are trying to get the value that has been auto-incremented and generated by the query that you initiated. This attribute is going to go through the data base or statement based on how you set up your query

PHP example

Once you have issued a query with PHP, then you are going to automatically get the auto_increment value and it will gather up this value using the same function that was used in the Perl example.

Remember a sequence that already exists

There could be some cases that you are going to have to delete a number of different records from the table that you are working with so that you have the ability to put the records in a different sequence. You are going to do this with a single trick, however, you need to be careful whenever you are using this trick

or else you are going to end up with your table joining another table without it being what you really want to happen. If this happens, you are going to end up having to go back and unlink the two tables before you are able to continue.

When you are trying to resequence an auto_increment, it is going to be unavoidable to drop the column and then add it into the table once more.

Example

Alter table inspect drop age;

Alter table insect

Add age int unsigned not null
auto_increment second

Add primary numbers (age) ;

Sequences starting with certain values

MySQL is going to default a sequence so that it starts with the number one. However, if you need to or want to, you can correct your code so that it will start with another number whenever your table is created.

Example

Create table insect

(

Age int unsigned not null auto_increment = 50;

Primary number (age),

Title varchar(20) not null

Date date not null,

Origin varchar(20) not null

If you so choose, you can also create a table and set the sequence value by using the alter table function.

Chapter eleven: Your Data Base- How to Tune it

Becoming an expert on data bases or even a data base expert is not going to happen overnight. In fact, it is going to take a lot of time and experience dealing with a variety of data bases doing a little bit of everything so that you have a well rounded education when it comes to the data base.

When dealing with a data base, there are a few things that you may find helpful so that you can become better with the data base.

1. The design that you should be using is 3BNF for when you are working with a data base.
2. Try and avoid the number to character changes since numbers are going to be compared to character differently and you may end up noticing a downgrade in the performance of your data base.
3. In using the select statement, you are only going to want to have the data that is absolutely neccesary to be displayed returned and try to avoid using an asterisks in these searches so that your system is not loaded when it does not need to be.
4. All the indexes that you make need to

be made very carefully for the tables that you will most. When you are not going to use the table much, try not to create an index because the more indexes that you have, the less space on the disk you are going to have and the more update operations you are going to have to deal with.

5. Full table scans are going to occur whenever there are columns inside of a where caluse but there is no index that is listed with that table. A full table scan can be avoided by simply constructan index on the columns that are going to work with that clause.
6. Equality operators need to be used with caution whenever real numbers, times, and dates are being used.

There is a chance that there are going to be some differences that you may not notice right away, but it is going to make getting an exact match almost impossible therfore your searches are not going to return any rows.

7. Pattern matching should be used judiciously. A valid where condition would be ilke col%, but to reduce the return of these records with this particular string is not going to reduce the rturn of your results being that you are not going to get them to be evaluated effectively. Trying to do an evaluation is going to be too much of a hassle and you should use a different operation so that you can get your results.

8. Turn you searches by looking at the structure of your search for the code that you are using. You may come to realize that you are going to be able to design your tables so that they can handle a faster manipulation of the data that is inside of them and write the search to work with the table appropriately.
9. When you are doing searches regularly, you need to try and stick to the proceedures that are going to work with large groups of SQL statements. These procedures are going to be put together by the data base before they are executed. SQL statements are not like the engine inside of the data base because the

data base is not going to need to optimize the procedure that it is about to carry out.

0. Try to not use the or operator when you are working with your searches if you can. This operator is going to slow do your query when you are using it on any table, no matter how big the table is.
1. Bulk data can be optimized by getting rid of the indexes. Try and think of your history of your table containing over a thousand different rows and this table is most likely going to have an index but possibly more than one. Indexes are going to give you faster access to the data that is inside of the table, but when a batch is loaded,

you are going to be better off not having these indexes because it is going to slow down the system that you are working with.

2. Batch transactions need to be performed with the commit function and it needs to be executed after each record is created.
3. The data base will need to be defragmented on a regular bases, and maitenance needs to be performed on a weekly basis.

Tuning tools that are built in

Oracle contains several different tools that you are going to be able to use when you are dealing with a SQL statement

and how it performance. Two of the more popular tools are:

1. Tkprof: this tool is going to measure the performance that is done over a period of time during every statement that is processed by SQL.
2. Explain plain: this tool will show you the path you need to follow so that the statement is executed properly.

Should you want to measure the time that passes from each search that is done in Oracle, the SQL*plus command can keep track for you.

Chapter twelve: Duplications Happen but, How to Fix Them

Situations may be found that there are going to be duplicate records that are inside of your table. While looking at these records, it is only going to make sense to find records that are unique rather than to gather the duplicate records.

The keyword you are going to use is distinct while working with a select statement so that you can get rid of any duplicate records and only find the

records that have unique data.

Code

```
SELECT DISTINCT column 1, column 2
, ….. columnN

FROM table_title

WHERE [condition]
```

Example

If you are looking at the age of your customers, then you are bound to have several customers that are going to be the same age.

First, you will select query so that you are able to see all of the duplicate records that are in your table.

If you wanted to see the ages of your customers in ascending order, you may come to find that you have three customers that are all 19 years of age.

By using the distinct keyword along with the select query, you are going to get rid of all the duplicate files and only see the single file.

Therefore, instead of seeing that there are three customers that are 19, you are only going to see that one is 19.

Chapter thirteen: Constraints are Everywhere!

When you want to be able to limit what data can be entered into a table, you can make much use out of constraints. You can set up a constraint whenever a table is created. This can also be done by using the CREATE TABLE or ALTER TABLE commands. Listed below are some of the common constraints that can be used.

CHECK- This specifies a constraint that will limit the value range that can be

placed in a column.

DEFAULT- This specifies a constraint that is used in order to insert a default value into a column.

FOREIGN KEY- This specifies a constraint that prevents invalid data from being inserted into the foreign key column because it has to be one of the values contained in the table that it points to.

NOT NULL- This specifies a constraint that enforces a column to not accept values that are null.

PRIMARY KEY- This specifies a

constraint that uniquely identifies each record in a database table. It is important to note that each table should have a primary key and each table can only have one primary key.

UNIQUE- This specifies a constraint that uniquely identifies each record in a database table. It is important to note that a table can have many UNIQUE constraints, but only one PRIMARY KEY constraint.

CHECK

How to use the CHECK constraint on CREATE TABLE:

My SQL:

```
CREATE TABLE Clients
( C_Id int NOT NULL,
Last varchar (255) NOT NULL,
First varchar (255),
Address varchar (255),
```

```
City varchar (255),
CHECK (C_Id 0));
```

SQL Server/Oracle/Access:

```
CREATE TABLE Clients
(C_Id int NOT NULL
CHECK (C_Id 0),
Last varchar (255) NOT NULL,
First varchar (255),
Address varchar (255),
City varchar (255));
```

You can name and define a CHECK constraint on multiple columns:

MySQL/ SQL Server/ Oracle/

Access:

```
CREATE TABLE Clients
(C_Id int NOT NULL,
Last varchar (255) NOT NULL,
First varchar (255),
Address varchar (255),
City varchar (255),
CONSTRAINT chk_Client
CHECK (C_Id 0 AND City=
'LosAngeles'));
```

How to use the CHECK constraint on ALTER TABLE statement:

MySQL/ SQL Server/ Oracle/ Access:

```
ALTER TABLE Clients
ADD CHECK (C_Id 0)
```

MySQL/ SQL Server/ Oracle/ Access:

```
ALTER TABLE Clients
ADD CHECK (C_Id 0)
```

```
ALTER TABLE Clients
ADD CONSTRAINT chk_Clients
CHECK (C_Id 0 AND City=
'LasVegas')
```

How to DROP a CHECK constraint:

SQL Server/ Oracle/ Access:

```
ALTER TABLE Clients
DROP CONSTRAINT chk_Clients
```

DEFAULT

How to use the DEFAULT constraint with CREATE TABLE statement:

```
CREATE TABLE Clients
(Id int NOT NULL,
Last varchar (255) NOT NULL,
First varchar (255),
```

```
Address varchar (255),
City varchar (255)
DEFAULT ‘LasVegas’);
```

This will put a constraint on the “City” column when the “Clients” table is created. The DEFAULT constraint can even be used in order to insert system values by using functions such as GETDATE():

```
CREATE TABLE Orders
( ID int NOT NULL,
OrderNo int NOT NULL,
C_Id int,
OrderDate date
DEFAULT GETDATE());
```

How to use the DEFAULT constraint with ALTER TABLE statement:

MySQL:

```
ALTER TABLE Clients
ALTER City
SET DEFAULT ‘LasVegas’
```

SQL Server/ Oracle/ Access:

```
ALTER TABLE Clients
ALTER COLUMN City
SET DEFAULT ‘LasVegas’
```

How to DROP a DEFAULT constraint:

MySQL:

ALTER TABLE Clients
ALTER City DROP DEFAULT

SQL Server/ Oracle/ Access:

ALTER TABLE Clients
ALTER COLUMN City DROP DEFAULT

Conclusion

Thank for making it through to the end of *SQL Strategies*, let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever it may be.

The next step is to take the strategies that you learned and put them to good use. Some of these strategies may not work for everyone, but they are going to work for some and what does work for you will help to make it to where using SQL is more efficient for you.

Do not think that you cannot keep

learning though! There is always more to learn! Keep up with what is new with SQL and do your research. There is always more out there for you to learn and figure out to make SQL that much better for you to use.

Finally, if you found this book useful in anyway, a review on Amazon is always appreciated!

Do not forget to check out the SQL Beginner's Guide and the SQL Tips and Tricks book on Amazon! They will assist you in making sure that you are getting the most out of using SQL.