What are the use cases for selecting CHAR over VARCHAR in SQL?

Asked 16 years, 3 months ago Modified 1 year, 8 months ago

Viewed 178k times

288

I realize that CHAR is recommended if all my values are fixed-width. But, so what? Why not just pick VARCHAR for all text fields just to be safe.

sql t-sql

Share

Improve this question

Follow

edited Dec 4, 2021 at 23:11

allyourcode

22.6k ● 20 ● 83 ● 108

asked Sep 12, 2008 at 18:15

SkunkSpinner

11.6k ● 7 ● 42 ● 53

19 Answers

Sorted by: Highest score (default)

400

The general rule is to pick **CHAR** if all rows will have close to the *same length*. Pick **VARCHAR** (or *NVARCHAR*) when the *length varies* significantly. CHAR may also be a bit faster because all the rows are of the same length.

It varies by DB implementation, but generally, VARCHAR (or *NVARCHAR*) uses one or two more bytes of storage (for length or termination) in addition to the actual data. So (assuming you are using a one-byte character set) storing the word "FooBar"

- **CHAR(6)** = 6 bytes *(no overhead)*
- **VARCHAR(100)** = 8 bytes *(2 bytes of overhead)*
- **CHAR(10)** = 10 bytes *(4 bytes of waste)*

The bottom line is **CHAR** *can* be *faster* and more *space-efficient* for data of relatively the same length (within two characters length difference).

Note: Microsoft SQL has 2 bytes of overhead for a VARCHAR. This may vary from DB to DB, but generally, there is at least 1 byte of overhead needed to indicate length or EOL on a VARCHAR.

As was pointed out by *Gaven* in the comments: Things change when it comes to multi-byte characters sets, and is a is case where VARCHAR becomes a much better choice.

*A note about the declared length of the **VARCHAR***: Because it stores the length of the actual content, then you don't waste unused length. So storing 6 characters in *VARCHAR(6), VARCHAR(100), or VARCHAR(MAX)* uses the same amount of storage. Read more about the differences when using VARCHAR(MAX). You declare a *maximum* size in VARCHAR to limit how much is stored.

In the comments AlwaysLearning pointed out that the Microsoft Transact-SQL docs seem to say the opposite. I would suggest that is an error or at least the docs are unclear.

Share Improve this answer Follow

edited Aug 25, 2020 at 19:30

answered Sep 12, 2008 at 18:22

Jim McKeeth

38.7k ● 25 ● 124 ● 199

---

20 Another reason is page splits and fragmentation. I had a table with an IDEN PK that got 99% fragmented because of page splits on varchar columns. A very active table and by nature of the application a new row empty row would get created and then populated. Char fixed the fragmentation problem. – paparazzo Sep 4, 2013 at 13:00

12 @Jim McKeeth -- these calculations are only true if you're using latin1 charset. Since most people should be using utf8 these days, your CHAR columns are going to be using 3x the space on average as a VARCHAR that's storing mostly characters in the base multilingual plane. – Gavin Towey Mar 17, 2014 at 17:40

12 @JimMcKeeth yes, that's exactly correct. Since CHAR is fixed length, it must therefore be fixed at the maximum possible space that could be used. In UTF8 that's 3 bytes per character. For varchar, it's free to use 1-3 bytes per character as needed. This is in the MySQL manual: dev.mysql.com/doc/refman/5.0/en/charset-unicode-utf8.html – Gavin Towey Mar 27, 2014 at 2:14

3 What is the difference with the string FooBar and varchar(100) vs char(100)? I'm thinking that demonstrates the difference better, yes? No? – Joel Peltonen May 21, 2014 at 11:22

4 @GavinTowey SQLSERVER uses UCS-2 for its NCHAR and NVARCHAR datatypes. It's always two bytes per character. – 1010 Jun 8, 2015 at 20:12

69

If you're working with me and you're working with Oracle, I would probably make you use `varchar` in almost every circumstance. The assumption that `char` uses less processing power than `varchar` may be true...for now...but database engines get better over time and this sort of general rule has the making of a future "myth".

Another thing: I have never seen a performance problem because someone decided to go with `varchar`. You will make much better use of your time writing good code (fewer calls to the database) and efficient SQL (how do indexes work, how does the optimizer make decisions, why is `exists` faster than `in` usually...).

Final thought: I have seen all sorts of problems with use of `CHAR`, people looking for '' when they should be looking for ' ', or people looking for 'FOO' when they should be looking for 'FOO (bunch of spaces here)', or people not trimming the trailing blanks, or bugs with Powerbuilder adding up to 2000 blanks to the value it returns from an Oracle procedure.

Share Improve this answer

Follow

edited May 12, 2011 at 17:30

Hank Gay
71.8k ● 36 ● 161 ● 222

answered Sep 12, 2008 at 18:42

Ethan Post
3,050 ● 3 ● 28 ● 28

20 I disagree somewhat with your first paragraph, since char may provide a hint that could be useful to optimizers, even future ones, and it may help to communicate the intent of the column. But +1 for your third paragraph. I hate all the extra spaces. A field should just store whatever I put in it without all the [explicative] padding. Basically, I just use char if all of the data is to be exactly the same length, no more and no less, now and forever. This is very rare, of course, and is usually a char(1). – Jeffrey L Whitledge Dec 10, 2009 at 17:34

char also provides a hints to analysts and developers... this thing is x number of chars.... If they are thinking of serializing it in some other format, that might be helpful. (I was forced to store a md5 checksum in a char in mssql that didn't have a uuid type... and i didn't ever want anything < 32 bytes... also put a constraint on the column). – joefromct Dec 1, 2017 at 21:02

32

In addition to performance benefits, `CHAR` can be used to indicate that all values *should* be the same length, e.g., a column for U.S. state abbreviations.

Share Improve this answer

Follow

answered Sep 12, 2008 at 18:57

Hank Gay
71.8k ● 36 ● 161 ● 222

Or country codes - can help distinguish between using a 2 or 3 character country code abbreviation – Dan Field Feb 18, 2016 at 18:00

If it's truly a fixed length, then there should be a constraint enforcing that. Although if you use `CHAR`, you'll have to make sure your constraint discounts padding. – jpmc26 May 2, 2019 at 21:34

20

Char is a little bit faster, so if you have a column that you KNOW will be a certain length, use char. For example, storing (M)ale/(F)emale/(U)nknown for gender, or 2 characters for a US state.

Share Improve this answer Follow

answered Sep 12, 2008 at 18:17

Jarrett Meyer
19.6k ● 6 ● 60 ● 52

5 Not sure that's a GREAT answer, since an ENUM would usually make a lot more sense, although I'm not sure how widely supported that type is (outside of MySQL). – Bobby Jack Sep 14, 2008 at 18:26

Seems to me that the set of states is not necessarily immutable, so char(2) seems much more appropriate than an enum. – Hj David Kearns Sep 18, 2008 at 13:17

1 @Bobby Jack - I don't know the specific details of any particular SQL enum implementation, but keep in mind that an enum stored as a 4 byte integer might require more space than a char(1) or char(2) column with the same data. There is a sense in which enums are more logical in terms of their

interpretation, and that might be compelling, but everything in an RDBMS system is abstract at some level and subject to the predicates defined for the tables. – Jeffrey L Whitledge Dec 10, 2009 at 17:28

4 Bad example, ENUM is best for that case. Better example would be a 3 letter IATA airport code – Andrew G. Johnson Jan 21, 2010 at 19:44

7 @Andrew, not all db's support ENUM data types. MSSQLServer, for example, does not. Also, an ENUM, stored as an int, takes 4 bytes. CHAR(1) takes 1 byte, and NCHAR(1) takes 2 bytes. – Jarrett Meyer Jan 21, 2010 at 20:58

18

Does NChar or Char perform better that their var alternatives?

Great question. The simple answer is yes in certain situations. Let's see if this can be explained.

Obviously we all know that if I create a table with a column of varchar(255) (let's call this column myColumn) and insert a million rows but put only a few characters into myColumn for each row, the table will be much smaller (overall number of data pages needed by the storage engine) than if I had created myColumn as char(255). Anytime I do an operation (DML) on that table and request alot of rows, it will be faster when myColumn is varchar because I don't have to *move* around all those "extra" spaces at the end. Move, as in when SQL Server does internal sorts such as during a distinct or union operation, or if it chooses a merge during it's query plan, etc. Move could also mean the time it takes to get the data from the server to my local pc or to another computer or wherever it is going to be consumed.

But there is some overhead in using varchar. SQL Server has to use a two byte indicator (overhead) to, on each row, to know how many bytes that particular row's myColumn has in it. It's not the extra 2 bytes that presents the problem, it's the having to "decode" the length of the data in myColumn on every row.

In my experiences it makes the most sense to use char instead of varchar on columns that will be joined to in queries. For example the primary key of a table, or some other column that will be indexed. CustomerNumber on a demographic table, or CodeID on a decode table, or perhaps OrderNumber on an order table. By using char, the query engine can more quickly perform the join because it can do straight pointer arithmetic (deterministically) rather than having to move it's pointers a variable amount of bytes as it reads the pages. I know I might have lost you on that last sentence. Joins in SQL Server are based around the idea of "predicates." A predicate is a condition. For example myColumn = 1, or OrderNumber < 500.

So if SQL Server is performing a DML statement, and the predicates, or "keys" being joined on are a fixed length (char), the query engine doesn't have to do as much work to match rows from one table to rows from another table. It won't have to find out how long the data is in the row and then walk down the string to find the end. All that takes time.

Now bear in mind this can easily be poorly implemented. I have seen char used for primary key fields in online systems. The width must be kept small i.e. char(15) or something reasonable. And it works best in online systems because you are usually only retrieving or upserting a small number of rows, so having to "rtrim" those trailing spaces you'll get in the result set is a trivial task as opposed to having to join millions of rows from one table to millions of rows on another table.

Another reason CHAR makes sense over varchar on online systems is that it reduces page splits. By using char, you are essentially "reserving" (and wasting) that space so if a user comes along later and puts more data into that column SQL has already allocated space for it and in it goes.

Another reason to use CHAR is similar to the second reason. If a programmer or user does a "batch" update to millions of rows, adding some sentence to a note field for example, you won't get a call from your DBA in the middle of the night wondering why their drives are full. In other words, it leads to more predictable growth of the size of a database.

So those are 3 ways an online (OLTP) system can benefit from char over varchar. I hardly ever use char in a warehouse/analysis/OLAP scenario because usually you have SO much data that all those char columns can add up to lots of wasted space.

Keep in mind that char can make your database much larger but most backup tools have data compression so your backups tend to be about the same size as if you had used varchar. For example LiteSpeed or RedGate SQL Backup.

Another use is in views created for exporting data to a fixed width file. Let's say I have to export some data to a flat file to be read by a mainframe. It is fixed width (not delimited). I like to store the data in my "staging" table as varchar (thus consuming less space on my database) and then use a view to CAST everything to it's char equivalent, with the length corresponding to the width of the fixed width for that column. For example:

```
create table tblStagingTable (
pkID BIGINT (IDENTITY,1,1),
CustomerFirstName varchar(30),
CustomerLastName varchar(30),
CustomerCityStateZip varchar(100),
CustomerCurrentBalance money )

insert into tblStagingTable
(CustomerFirstName,CustomerLastName, CustomerCityState
Main St Washington, MD 12345', 123.45)

create view vwStagingTable AS
SELECT CustomerFirstName = CAST(CustomerFirstName as C
CustomerLastName = CAST(CustomerLastName as CHAR(30)),
CustomerCityStateZip = CAST(CustomerCityStateZip as CH
CustomerCurrentBalance = CAST(CAST(CustomerCurrentBala
CHAR(10))
```

```
SELECT * from vwStagingTable
```

This is cool because internally my data takes up less space because it's using varchar. But when I use DTS or SSIS or even just a cut and paste from SSMS to Notepad, I can use the view and get the right number of trailing spaces. In DTS we used to have a feature called, damn I forget I think it was called "suggest columns" or something. In SSIS you can't do that anymore, you have to tediously define the flat file connection manager. But since you have your view setup, SSIS can know the width of each column and it can save alot of time when building your data flow tasks.

So bottom line... use varchar. There are a very small number of reasons to use char and it's only for performance reasons. If you have a system with hundrends of millions of rows you will see a noticeable difference if the predicates are deterministic (char) but for most systems using char is simply wasting space.

Hope that helps. Jeff

Share Improve this answer

Follow

edited Sep 11, 2015 at 13:04

Gottfried Lesigang
67.2k ● 9 ● 57 ● 120

answered Mar 4, 2011 at 1:31

Jeff
181 ● 1 ● 3

You're saying fixed chat takes up more space not just when stored, but also when transported or "moved" as you say? From DB Server to my client for example? When do we lose those null bytes? – Nate Anderson Aug 4, 2016 at 6:56

9

There are performance benefits, but here is one that has not been mentioned: row migration.

With CHAR, you reserve the entire space in advance.So let's says you have a CHAR(1000), and you store 10 characters, you will use up all 1000 characters of space. In a VARCHAR2(1000), you will only use 10 characters.

The problem comes when you modify the data. Let's say you update the column to now contain 900 characters. It is possible that the space to expand the varchar is not available in the current block. In that case, the DB engine must migrate the row to another block, and make a pointer in the original block to the new row in the new block. To read this data, the DB engine will now have to read 2 blocks.

No one can equivocally say that varchar or char are better. There is a space for time tradeoff, and consideration of whether the data will be updated, especially if there is a good chance that it will grow.

Share Improve this answer Follow

edited Mar 28, 2023 at 18:48

Giacomo1968
26.1k ● 11 ● 76 ● 105

answered Sep 13, 2008 at 1:28

Tony BenBrahim
7,280 ● 2 ● 38 ● 50

I think you have a typo in your post -- shouldn't varchar2(1000) be CHAR(1000) ? – Matt Rogish Sep 13, 2008 at 2:37

8

There is a difference between early performance optimization and using a best practice type of rule. If you are creating new tables where you will always have a fixed length field, it makes sense to use CHAR, you should be using it in that case. This isn't early optimization, but rather implementing a rule of thumb (or best practice).

i.e. - If you have a 2 letter state field, use CHAR(2). If you have a field with the actual state names, use VARCHAR.

Share Improve this answer Follow

answered Sep 12, 2008 at 18:37

Astra
11.2k ● 3 ● 39 ● 41

8

I would choose varchar unless the column stores fixed value like US state code -- which is always 2 chars long and the list of valid US states code doesn't change often :).

In every other case, even like storing hashed password (which is fixed length), I would choose varchar.

Why -- char type column is always fulfilled with spaces, which makes for column *my_column* defined as char(5) with value 'ABC' inside comparation:

```
my_column = 'ABC' -- my_column stores 'ABC  ' value wh
'ABC'
```

false.

This *feature* could lead to many irritating bugs during development and makes testing harder.

Share Improve this answer

Follow

answered Sep 12, 2008 at 20:11

Grzegorz Gierlik
11.2k ● 4 ● 48 ● 57

1 At least in MSSQL Server, 'abc ' = 'abc'. I've never quite figured out if I like or detest that feature.... – Mark Brackett
Oct 23, 2008 at 16:59

A good read about padding of char here Padding – Edward
Mar 22, 2017 at 18:12

6

CHAR takes up less storage space than VARCHAR if all your data values in that field are the same length. Now perhaps in 2009 a 800GB database is the same for all intents and purposes as a 810GB if you converted the VARCHARs to CHARs, but for short strings (1 or 2 characters), CHAR is still a industry "best practice" I would say.

Now if you look at the wide variety of data types most databases provide even for integers alone (bit, tiny, int, bigint), there ARE reasons to choose one over the other. Simply choosing bigint every time is actually being a bit ignorant of the purposes and uses of the field. If a field simply represents a persons age in years, a bigint is overkill. Now it's not necessarily "wrong", but it's not efficient.

But its an interesting argument, and as databases improve over time, it could be argued CHAR vs VARCHAR does get less relevant.

Share Improve this answer Follow

answered Jan 21, 2009 at 0:07

Scott Duffy
678 ● 7 ● 12

6

I would NEVER use chars. I've had this debate with many people and they always bring up the tired cliché that char is faster. Well I say, how much faster? What are we talking about here, milliseconds, seconds and if so how many? You're telling me because someone claims its a few milliseconds faster, we should introduce tons of hard to fix bugs into the system?

So here are some issues you will run into:

Every field will be padded, so you end up with code forever that has RTRIMS everywhere. This is also a huge disk space waste for the longer fields.

Now let’s say you have the quintessential example of a char field of just one character but the field is optional. If somebody passes an empty string to that field it becomes one space. So when another application/process queries it, they get one single space, if they don’t use rtrim. We’ve had xml documents, files and other programs, display just one space, in optional fields and break things.

So now you have to ensure that you’re passing nulls and not empty string, to the char field. But that’s NOT the correct use of null. Here is the use of null. Lets say you get a file from a vendor

```
Name|Gender|City
Bob||Los Angeles
```

If gender is not specified than you enter Bob, empty string and Los Angeles into the table. Now lets say you get the file and its format changes and gender is no longer included but was in the past.

```
Name|City
Bob|Seattle
```

Well now since gender is not included, I would use NULL. VARCHARs support this without issues.

CHAR on the other hand is different. You always have to send NULL. If you ever send empty string, you will end up with a field that has spaces in it.

I could go on and on with all the bugs I've had to fix from CHARs and in about 20 years of development.

Share Improve this answer Follow

edited Mar 28, 2023 at 18:46
Giacomo1968
26.1k ● 11 ● 76 ● 105

answered Apr 13, 2018 at 18:12
Mauro Torres
751 ● 6 ● 6

1 Agreed. rtrim = yuck – mpalmer78 Sep 27, 2021 at 14:52

4

I stand by Jim McKeeth's comment.

Also, indexing and full table scans are faster if your table has only CHAR columns. Basically the optimizer will be able to predict how big each record is if it only has CHAR columns, while it needs to check the size value of every VARCHAR column.

Besides if you update a VARCHAR column to a size larger than its previous content you may force the database to rebuild its indexes (because you forced the database to physically move the record on disk). While with CHAR columns that'll never happen.

But you probably won't care about the performance hit unless your table is huge.

Remember Djikstra's wise words. Early performance optimization is the root of all evil.

Share Improve this answer Follow

edited Sep 12, 2008 at 18:38

answered Sep 12, 2008 at 18:28

Alvaro Rodriguez
2,848 ● 3 ● 33 ● 40

4 There is a degree of speculation in your comment. I have seen time and time again assumptions like these get tested and the exact opposite turn out to be true. The problem is many engineers will take info like this as the gospel. Please folks, create test cases which reflect your real situations. – Ethan Post Sep 12, 2008 at 18:53

Ethan is totally correct. This so depends on the implementation you're using that without references to actual (Product,Version) it's completely useless. – David Schmitt Sep 13, 2008 at 10:20

When you update a `CHAR` column, indexes need to be updated just as well. There is no difference in updating a VARCHAR or CHAR column in that regard. Think about updating `FOO` to `BAR`. – user330315 Oct 25, 2014 at 20:41

4

Many people have pointed out that if you know the exact length of the value using CHAR has some benefits. But while storing US states as CHAR(2) is great today, when you get the message from sales that 'We have just made our first sale to Australia', you are in a world of pain. I always send to overestimate how long I think fields will need to be rather than making an 'exact' guess to cover for future events. VARCHAR will give me more flexibility in this area.

Share Improve this answer Follow

answered Nov 4, 2008 at 21:38
Craig
36.8k ● 35 ● 121 ● 202

3

I think in your case there is probably no reason to not pick Varchar. It gives you flexibility and as has been mentioned by a number of respondants, performance is such now that except in very specific circumstances us meer mortals (as opposed to Google DBA's) will not notice the difference.

An interesting thing worth noting when it comes to DB Types is the sqlite (a popular mini database with pretty impressive performance) puts everything into the database as a string and types on the fly.

I always use VarChar and usually make it much bigger than I might strickly need. Eg. 50 for Firstname, as you say why not just to be safe.

Share Improve this answer Follow

answered Sep 18, 2008 at 13:01
Toby Allen
11.2k ● 12 ● 79 ● 131

It's the classic space versus performance tradeoff.

2

In MS SQL 2005, Varchar (or NVarchar for lanuagues requiring two bytes per character ie Chinese) are variable length. If you add to the row after it has been written to the hard disk it will locate the data in a non-contigious location to the original row and lead to fragmentation of your data files. This will affect performance.

So, if space is not an issue then Char are better for performance but if you want to keep the database size down then varchars are better.

Share Improve this answer Follow

answered Sep 14, 2008 at 18:21

Leo Moore
2,158 ● 2 ● 19 ● 22

2

Fragmentation. CHAR reserves space and VARCHAR does not. Page split can be required to accommodate update to VARCHAR.

Share Improve this answer Follow

edited Mar 28, 2023 at 18:47

Giacomo1968
26.1k ● 11 ● 76 ● 105

answered Oct 25, 2013 at 12:11

paparazzo
45.1k ● 24 ● 109 ● 179

Because of many other factors, a page split may happen when updating a `CHAR` column. – Rick James Jan 8, 2019 at 6:33

1

There is some small processing overhead in calculating the actual needed size for a column value and allocating the space for a Varchar, so if you are definitely sure how long the value will always be, it is better to use Char and avoid the hit.

Share Improve this answer Follow

answered Sep 12, 2008 at 18:18
Guy Starbuck
21.8k ● 7 ● 54 ● 64

1

when using varchar values SQL Server needs an additional 2 bytes per row to store some info about that column whereas if you use char it doesn't need that so unless you

Share Improve this answer Follow

answered Sep 12, 2008 at 18:22
SQLMenace
135k ● 25 ● 211 ● 225

1

Using CHAR (NCHAR) and VARCHAR (NVARCHAR) brings differences in the ways the database server stores the data. The first one introduces trailing blanks; I have encountered problem when using it with LIKE operator in SQL SERVER functions. So I have to make it safe by using VARCHAR (NVARCHAR) all the times.

For example, if we have a table **TEST(ID INT, Status CHAR(1))**, and you write a function to list all the records with some specific value like the following:

```
CREATE FUNCTION List(@Status AS CHAR(1) = '')
RETURNS TABLE
AS
RETURN
SELECT * FROM TEST
WHERE Status LIKE '%' + @Status '%'
```

In this function we expect that when we put the default parameter the function will return all the rows, but in fact it does not. Change the @Status data type to VARCHAR will fix the issue.

Share Improve this answer Follow

answered Nov 25, 2014 at 3:59

Tuan Le PN

384 ● 1 ● 15

This can be changed as well by ansi_padding How values are retrieved – Edward Mar 22, 2017 at 18:16

0

In some SQL databases, VARCHAR will be padded out to its maximum size in order to optimize the offsets, This is to speed up full table scans and indexes.

Because of this, you do not have any space savings by using a VARCHAR(200) compared to a CHAR(200)

Share Improve this answer Follow

answered Sep 12, 2008 at 18:30

FlySwat

175k ● 75 ● 248 ● 314

4 Which databases implement VARCHAR that way?
– Troels Arvin Sep 12, 2008 at 19:33

6 Seriously, what database implements it that way? What you describe normally applies to CHAR, not VARCHAR.
– Richard Simões Jul 29, 2009 at 18:57

mysql will convert varchar's to chars if there are char's and varchar's in the same table. – Malfist Sep 16, 2009 at 21:26

My interpretation of the MySQL comments is that this doesn't apply to primary table storage, but might possibly be relevant for temp tables eg. for grouping/sorting data.
dev.mysql.com/doc/refman/8.0/en/char.html
stackoverflow.com/questions/262238/… – Thomas W Dec 5, 2018 at 4:35