**Database design best practices**

**Reading time: about 9 min**

Posted by: Lucid Content Team

Designing a database the right way requires some analysis of your data and planning around how you want to structure it. There are many different database types, models, and customizations you can use to achieve your goals.

Here’s how to plan your database and start designing it.

**How will you use your database?**

Today, organizations are using data as part of their business intelligence gathering in end-customer products and services, for forecasting, and to inform real-time business decisions. Databases don’t have to be digital—technically, a notebook counts—but digital databases mean you can work with Big Data and use data analytics much more effectively.

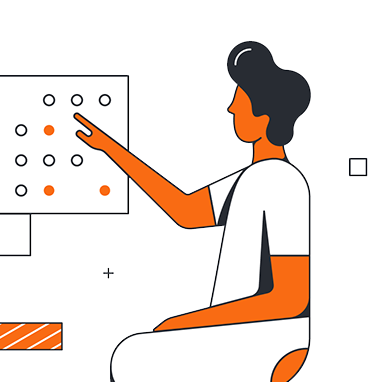
In the past, much of the data that businesses collected was discarded or not used in meaningful ways to drive business decisions. Think, for instance, of all of the retailers who collected purchasing data to process in-store sales at the checkout register and didn’t have an efficient way to keep point-of-sale data or perform useful analysis of it.

Now we know that data can be extraordinarily valuable for organizations, and we have more and more means of leveraging and visualizing data than ever before.

**What does good database design look like?**

How you’re using the data and knowledge your organization collects is one important consideration when you’re developing your goals. Database design is usually dictated by how you’re using your data today and how your organization plans to use it in the future. If you already have a database with existing data, then you have to consider how you’ll migrate as well.

For every database use case, there are different types of databases, database software, and specific designs. The database design you use today may not fit all of your needs tomorrow. This is why databases aren’t chosen randomly but represent a carefully-researched decision at most companies.



Unsure of what database to choose? Have we got the article for you.

[See our tips](https://www.lucidchart.com/blog/choosing-a-database)

Good database design is driven by several core principles:

* Minimize redundancy: To save resources, create an efficient database, and simplify how the database works, data redundancy is minimized and duplication is avoided.
* Protect accuracy: Your database should keep information accurate and reduce the likelihood of accidentally damaging information.
* Be accessible: The business intelligence systems that need reading and writing access should have it. Your database should provide access while also protecting data security.
* Meet expectations: Of course, you keep a database to fulfill a specific purpose—so the database design must successfully support your data processing expectations.

Your database should take into consideration what stakeholders in your organization need from their data. For this reason, it’s a good practice to include them in your database design process.

**Determining your goals for your database**

**Bring in stakeholders**

Who should you invite feedback from on your database design? Think about end-users within your organization, including team members, project managers, devs, and other internal stakeholders, as well as your external stakeholders such as business customers or power users. Before you get too far into mapping out your goals and beginning the design process, think about stakeholders who should be involved and how to involve them.

This stakeholder involvement not only prevents possible backlash by avoiding designs that others in your organization would see as a bad fit. It also brings you more ideas, best practices, and experience to potentially draw from that can save resources and improve the outcome.

**Gather information to help with your decision**

Ask yourself some pointed questions to determine the database you need. First, though, you should start gathering information that will help you with this process and decision.

* Forms: Collect the forms using data that will go in the database.
* Processes: Review each process involved in collecting or processing data for the database. You’ll need to have these processes available for reference as you plan your database.
* Types of data: Any data fields you’d gather and store in your database, such as customer contact information for a database of customers: name, email address, address, city, state, and zip code. Your data should be broken down into basic pieces, removing any complexity.

**SQL vs NoSQL**

Structured Query Language (SQL) allows you to interact with a database and make meaningful use of its data. Often, databases are categorized as SQL or NoSQL (Not Only SQL). NewSQL has properties of both. There are unique pros and cons to these options, so think about how your database’s characteristics enable or restrict how you use them.

**SQL**

Otherwise known as a relational database, SQL databases are made up of tables of data along with the relationships among the data fields. These are traditional databases, and they’re popular for many different database use cases, but they’re also difficult to scale vertically. You can horizontally scale SQL databases, but this isn’t appropriate for every database use.

Today, many types of data need to be stored and managed in a more streamlined way—with databases that don’t have the same requirements and expectations associated with SQL and ACID compliance.

One example of where SQL gets into trouble with large-scale data is with atomicity. A relational database can’t function well without restricting “write” activity and managing it carefully with bookkeeping in the background to ensure data integrity. As you scale, these management activities can be difficult to expand and adapt, which can be a problem for certain Big Data projects.

**NoSQL**

As noted earlier, it’s “not only SQL” rather than “no SQL,” so you can have a NoSQL database with some relational components that are structured with SQL. NoSQL databases run the gamut in terms of how data is stored and structured. With NoSQL, though, you do have some component of your database that’s not managed by SQL.

**Data models**

Aside from choosing SQL or NoSQL, you need to think about the data model you’ll use:

* Relational database: All relationships are already defined in a relational database, connecting together tables with columns and rows of data. With this type of database, you can use your data in many different ways without rearranging it. This is great for many complex use cases involving situations where you need to store data with many different relationships, such as product names along with product information.
* Hierarchical database: A one-to-many, tree-like data structure. For a hierarchy (hence the name), hierarchical databases make a lot of sense. You could create a database with department names, and each department can be associated with a list of employees who work there.
* Network database: Like hierarchical databases, network databases can have a parent record associated with multiple child records. Network databases can also have multiple parents associated with a single child, however, adding flexibility for some uses. If you visualize a network database, it will look something like a net or web of interconnected records.
* Object-oriented database: This last type of database uses objects rather than tables, which relational databases use. With object-oriented databases, object-oriented programmers can purposely build the databases they need.

**Database design best practices**

When you’re ready to design your database, keep these best practices in mind.

**1. Keep it simple**

As you design, think about your users. Put usability at the forefront and ensure that everything is as easy and straightforward as possible for the end-user, even if that means more work for you upfront.

* Use standardization: Stay consistent with naming conventions and avoid abbreviations. You want to create a standard and stick with it throughout your database.
* Consider future modifications: The database is a living thing in the sense that it should be modifiable later.
* Keep technical debt in check: Don’t leave too many potential messes for users to workaround or for future devs to resolve.

Technical debt can cause performance problems and a decline in quality. Think carefully about how you take on technical debt and manage it.

[Read more](https://www.lucidchart.com/blog/choosing-a-database)

**2. Normalize your data**

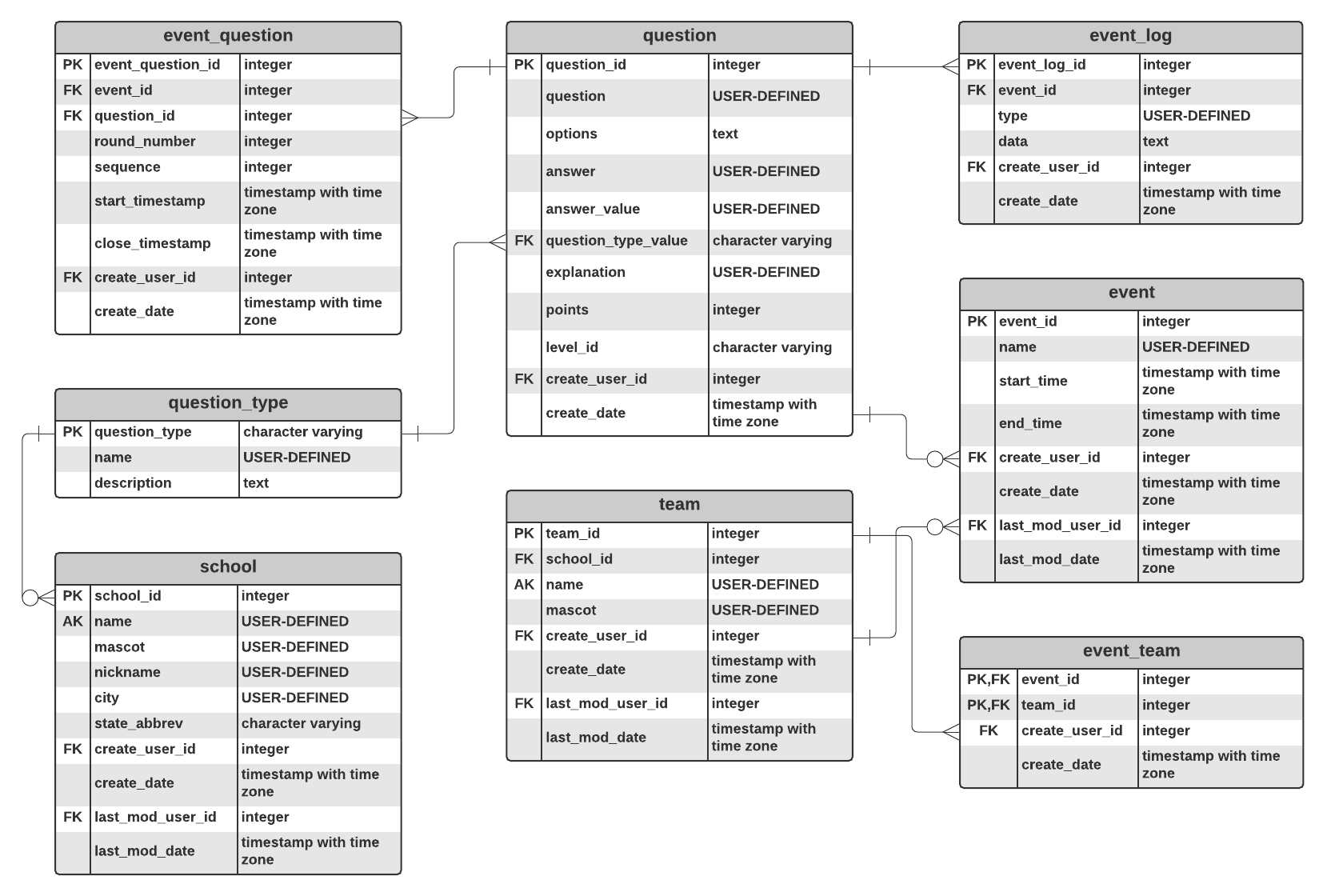
Keep redundancy to a minimum and protect your data’s consistency.

* Verify dependencies: Make sure your dependencies are in order and remain consistent throughout your database.
* Prevent anomalies: Update, insertion, and deletion anomalies can be prevented by double-checking your database dependencies.

**3. Consider the running conditions**

Your database won’t stay in beta forever—at some point, users will be putting your database to use. Real-world conditions may not be ideal, and you need to plan ahead for them so your database is up to the challenge.

* Design for the long term: Look ahead at how your users will need to scale, adapt, or use your database differently from how it’s originally designed.
* Create documentation: Carefully document, even if it’s a pain to do right now, so your users have an easier time later.
* Diagram your data: A visual representation can be a helpful way for your users to understand data relationships and structures.
* Plan for resource limitations: In all likelihood, your database and your app will run alongside others or you’ll be sharing computing resources. Take potential limits into consideration.

[](https://lucid.app/lucidchart/editNewOrRegister/ccb705cb-1ac6-45ce-88bd-019411960373?anonId=0.92be4c3d18931c9a0c6&sessionDate=2023-07-07T19%3A17%3A24.815Z&sessionId=0.45374ab618931c9a0ce)Entity-relationship diagram example (Click on image to modify online)

**4. Collaborate more—bring the devs and DBAs together**

Many organizations still treat the database like it’s distinct enough that the DBA doesn’t need to work closely with DevOps. But this mindset can create discrepancies.

* Communicate expectations: Other teams should know which actions are only for database administrators (for instance, making database changes).
* Loop in the DBA: Find ways for the DBA to easily chat with the developers and vice versa.

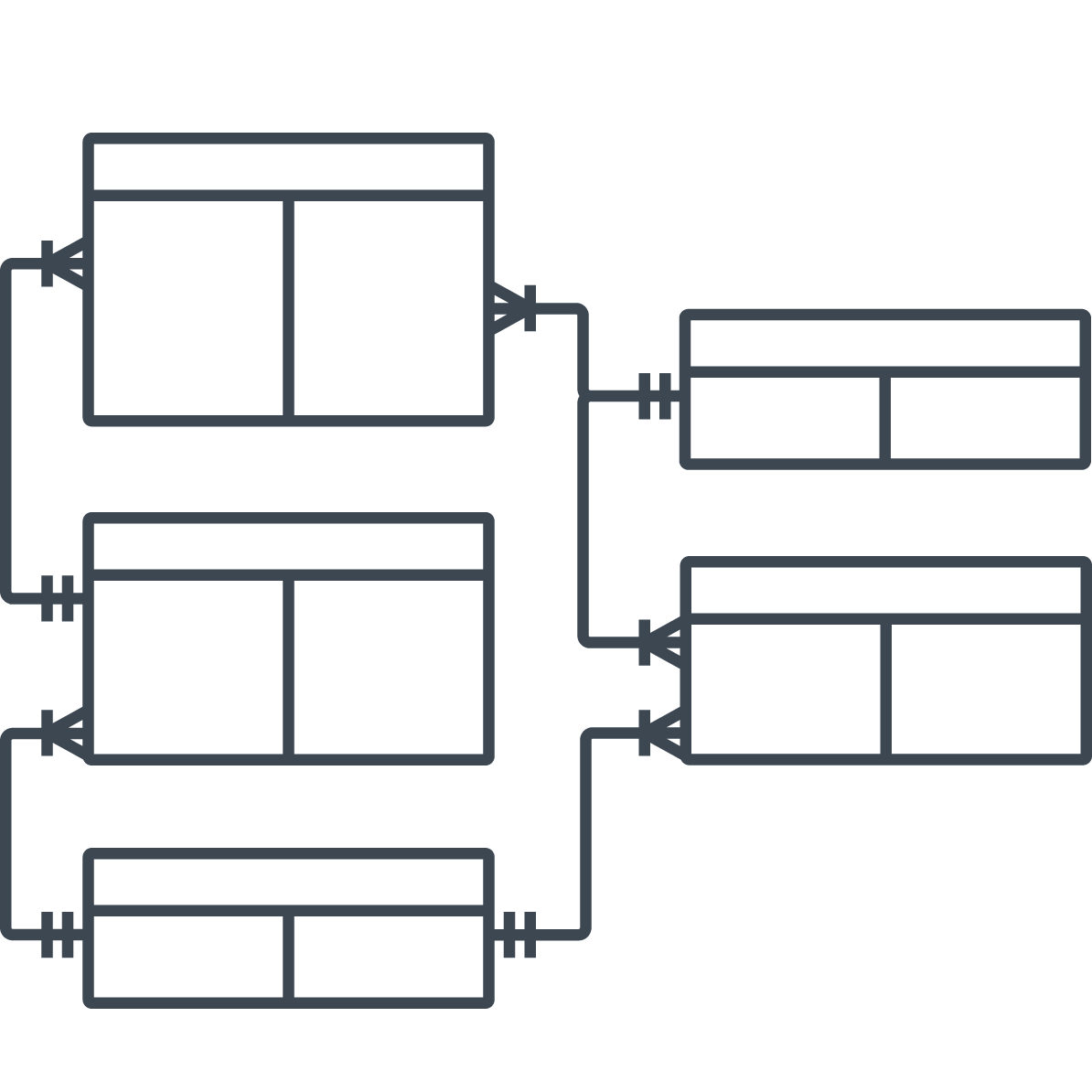
**5. Model your data and look for the right fit for your needs**

Since your database design is so important, take some time with it. Create a model, develop a diagram, and incorporate your team’s input.

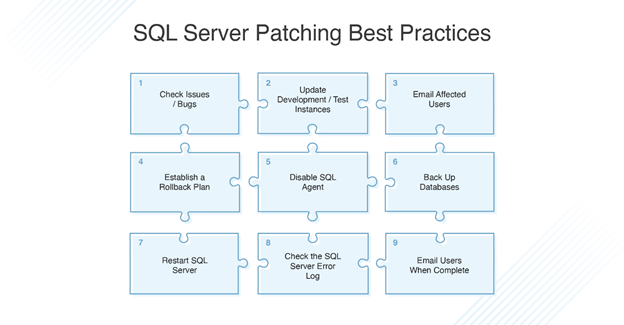
* Minimize unnecessary maintenance: Modeling and thinking about your database can keep your resource investment down and make it easier for you to maintain your database later.
* Use visualization: Create a diagram and test it. Does your diagram fit your data, and does the structure you chose fit your needs?
* Test your database: Spend some time in testing and don’t skip this step. A non-functional database is more costly in the long run than an extended deadline or investing the time.

**Scalable database design for the win**

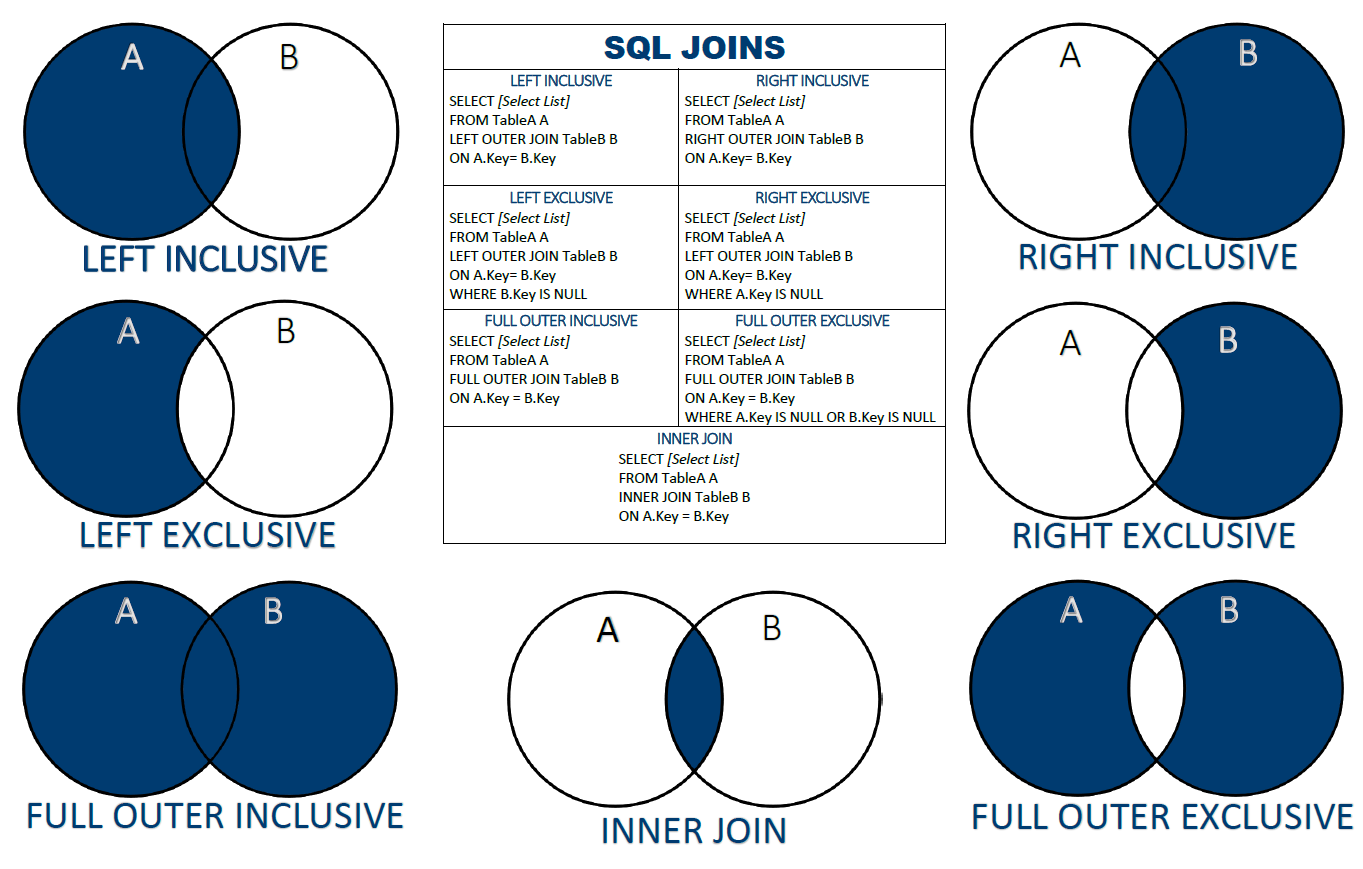
As you scale your database in the future, these database design best practices and careful data modeling will allow you to plan more effectively for how your use of data evolves and grows over time. Making these decisions today maximizes your investment and protects the data you work so hard to collect and optimize.



SQL Server Best Practices :



# SQL Joins



6 Important Types Of Indexes In SQL Server

[[](https://u-next.com/author/ajay-sarangam/)](https://u-next.com/author/ajay-sarangam/)

[Ajay Sarangam](https://u-next.com/author/ajay-sarangam/)

[Author](https://u-next.com/author/ajay-sarangam/)

 16 Jun 2022

**Share**

**Introduction**

The index is named as a design in SQL server stored or maintained wilt in-memory structure or on disk related with a table or views, which is utilized principally to recognize a specific set or a row Table or Views. Indexes in SQL are the individual lookup tables, which are utilized by the data set internet searcher to accelerate the general information recovery.  
The use of the index in SQL is to rapidly discover the data in a data set table without looking through each row of it. In SQL Index, it is basic to keep up more extra storage to make a copy duplicate of the data set. Tables in SQL server are contained inside database item holders that are called Schemas. The schema likewise fills in as a security limit, where you can restrict data set client authorizations to be on a particular schema level as it were. To know what are the different types of Indexes in SQL Server, then read this article to explore them and have a better understanding of them.

**Different Types of Indexes in SQL** Server

There are various types of indexes in SQL server:

1. [**Clustered Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Clustered-Index)
2. [**Non-Clustered Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Non-Clustered-Index)
3. [**Column Store Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Column-store-Index)
4. [**Filtered Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Filtered-Index)
5. [**Hash Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Hash-Index)
6. [**Unique Index**](https://u-next.com/blogs/data-science/types-of-indexes-in-sql-server/#Unique-Index)

1. **Clustered Index**

Clustered Index stores and sort rows of data in a view or table depending on their central values. There may be an instance of having just one clustered index in each table, as it can empower the client to store data in a solitary request. Clustered index store data in an arranged way, and in this way, at whatever point data is contained in the table in an arranged manner implies it is orchestrated with a clustered index.

At the point when a table contains a clustering in SQL server, it is named a clustered table. A clustered index is liked to utilize when adjustment of gigantic information is needed in any data set. If the data put away in a table or data set are not organized in descending or ascending request, at that point, the data table is named as a heap.

2. **Non-Clustered Index**

It represents a structure, which is isolated from data rows. This types of indexes in SQL server covers the non-clustered key values, and each worth pair has a pointer to the data row that comprises vital significance.

In the non-clustered index, the client can undoubtedly add non-key to the leaf level, as it sidesteps the current index key cut-off points and performs completely covered recorded questions. A non-clustered index is made to improve the general exhibition of much of the time posed inquiries, which are not covered by grouped things.

Clustered vs. Non-clustered index in SQL server is that the non-clustered index stores the data at one area and indices at another area, while the clustered index is a kind of index that sorts the data rows in the table on their key values.

3. **Column store Index**

A column store index is one of the types of indexes in SQL Server that has a standard type of index with regards to putting away and questioning enormous data warehousing truth tables. This is an index of SQL, which was intended for development in the presentation of inquiry in the event of jobs with huge measures of data.

The column-store index empowers putting away information inside little impressions, which helps in speeding up. The use of this index takes into account the client to get IO with multiple times higher inquiry execution when contrasted with conventional column arranged capacity. For examination, the Columnstore Index gives a significant degree to have a preferable exhibition over other records in SQL. Column store index esteems from a similar area have comparative qualities, which expands the general pace of information compressions.

4. **Filtered Index**

A filtered index is one of the types of indexes in an SQL server that is made when a column has just a few applicable numbers for questions on the subset of values. If, when a table comprises heterogeneous data rows, a separated list is made in SQL for at least one sort of data.

5. **Hash Index**

Hash Index is one of the types of indexes in SQL server that slots containing a pointer or an array of N buckets and a row on each slot or bucket. It utilizes the Hash function F (K, N), where N is several buckets and K is critical. The capacity delineates the key relating to the bucket of the hash index. Every bucket of the Hash Index comprises eight bytes, which are utilized to stock the memory address of the connected rundown of basic sections.

6. **Unique Index**

The unique index in the SQL server confirms and guarantees that the index key doesn’t contain any copy esteems and along these lines, empowers the clients to examine that each row in the table is exceptional in either way.

The unique index in SQL is extraordinarily utilized when the client needs to have an extraordinary trait of every information. It permits people to guarantee the data respectability of each characterized section of the table in the data set. This index likewise gives extra data about the data table, which is useful to question enhancers.

**Types of Pages in SQL server**

* Data Pages
* Bulk Changed Map
* Text/Image Pages
* Page Free Space
* Index Allocation Map
* Secondary Global Allocation Map
* Differential Changed Map
* Global Allocation Map

**Types of the Database in SQL server**

* tempdb
* msdb
* Master
* Model

**Conclusion**

To create an index in the SQL statement is utilized to make files in tables. Indexes are utilized to recover information from the data set more rapidly than something else. The clients can’t see the lists, they are simply used to accelerate queries/searches.

An Index is a key work from at least one column in the information base that speeds up getting rows from the view or table. This key aids a Database like MySQL, SQL Server, Oracle, and so on to discover the row related to key qualities rapidly.

An index stores the total information in the table, which is coordinated coherently with rows and columns, and truly kept up and puts away in line shrewd data known as row store, and if the records are stored away in segment insightful data, known as Column store.

If you are interested in making a career in the Data Science domain, our 11-month in-person [**Postgraduate Certificate Diploma in Data Science**](https://u-next.com/post-graduate-diploma-in-data-science-pgds-certification-training/) course can help you immensely in becoming a successful Data Science professional.



