Physical vs. logical (hard vs. soft) delete of database record? [closed]

Asked 16 years ago Modified 2 years, 11 months ago

Viewed 108k times

159

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 2 years ago.

Improve this question

What is the advantage of doing a logical/soft delete of a record (i.e. setting a flag stating that the record is deleted) as opposed to actually or physically deleting the record?

Is this common practice?

Is this secure?

sql database database-design soft-delete hard-delete

Share

edited Mar 2, 2021 at 5:24

Improve this question

Follow

Kerem

11.5k ● 5 ● 60 ● 58

asked Dec 18, 2008 at 16:09

user21826

3,634 ● 5 ● 30 ● 28

44 Use delete timestamps, not flags. – Dave Jarvis Dec 19, 2013 at 23:28

4 A flag doesn't provide any information about *when* the row was deleted. Temporal information has many uses, including systems debugging. – Dave Jarvis Mar 13, 2019 at 4:04

3 I soft delete doesn’t add useful audit data. If your intent it to create an audit history create a secondary table focused on that intent. It could even store previous versions and doesn’t make development and reporting a massive pain in the rear. – Matthew Whited Aug 17, 2020 at 10:09

Before implementing a hard delete, consider whether or not you are remove access to data that is still required. For example, on a blogging web site, physically deleting a user from the user table might also delete data that is needed for displaying that user's blog entries. Disabling the user's account is adequate and similar to a soft deletion. – Dave F May 20, 2021 at 3:59

@DaveF You have to be very careful about that line of thinking these days. With GDPR and other legislation in various areas it is NOT enough to simply disable a user's account if they request deletion. You can anonymize rather than delete, but even that has some restrictions. – Luke Aug 30, 2021 at 13:43

26 Answers

Sorted by: Highest score (default)

91

Advantages are that you keep the history (good for auditing) and you don't have to worry about cascading a delete through various other tables in the database that reference the row you are deleting. Disadvantage is that you have to code any reporting/display methods to take the flag into account.

As far as if it is a common practice - I would say yes, but as with anything whether you use it depends on your business needs.

EDIT: Thought of another disadvantange - If you have unique indexes on the table, deleted records will still take up the "one" record, so you have to code around that possibility too (for example, a User table that has a unique index on username; A deleted record would still block the deleted users username for new records. Working around this you could tack on a GUID to the deleted username column, but it's a very hacky workaround that I wouldn't recommend. Probably in that circumstance it would be better to just have a rule that once a username is used, it can never be replaced.)

Share Improve this answer Follow

edited Dec 18, 2008 at 16:22

answered Dec 18, 2008 at 16:13

Chris Shaffer

32.6k ● 5 ● 51 ● 61

Display as Active/Deactivated users =) On another note if it is a unique index (assuming here you mean the database is controlling the unique index) what do you mean by - it would still block the deleted users username for new records?? – Paul C Jan 22, 2013 at 13:48

20 @ChrisShaffer Alternatively, instead of a GUID, one can choose to index only non-deleted rows. E.g.: `CREATE UNIQUE INDEX ... WHERE DELETED_AT is null` (in PostgreSQL) and then all rows with any deletion date are not indexed. (They can be included in a non-unique index instead.) – KajMagnus Jun 19, 2013 at 20:52

11 @Chris Shaffer: Quote "u don't have to worry about cascading a delete through various other tables". Not true, you'll have to forward the soft delete manually, which is a great pain in the ass and causes inconsistencies. This is actually a disadvantage, because there is no more foreign-key relationship enforcement. You will end up with data-garbage very soon. – Stefan Steiger Apr 26, 2014 at 7:57

2 @Chris Shaffer: Quote "you keep the history (good for auditing)". Again no, you don't. If I update (or accidentially upate via SSMS) the values in the table, you have no record of the old value. If I accidentially delete (with real delete) a row, you do not have a record left, either. Soft deletes fail as an audit substitute. – Stefan Steiger Apr 26, 2014 at 8:00

1 abstraction.blog/2015/06/28/… – Abolfazl Aghili Sep 28, 2021 at 13:46

35

Are logical deletes common practice? Yes I have seen this in many places. Are they secure? That really depends are they any less secure then the data was before you deleted it?

When I was a Tech Lead, I demanded that our team keep every piece of data, I knew at the time that we would be using all that data to build various BI applications, although at the time we didn't know what the requirements would be. While this was good from the standpoint of auditing, troubleshooting, and reporting (This was an e-commerce / tools site for B2B transactions, and if someone used a tool, we wanted to record it even if their account was later turned off), it did have several downsides.

The downsides include (not including others already mentioned):

1. Performance Implications of keeping all that data, We to develop various archiving strategies. For example one area of the application was getting close to generating around 1Gb of data a week.
2. Cost of keeping the data does grow over time, while disk space is cheap, the amount of infrastructure to keep and manage terabytes of data both online and off line is a lot. It takes a lot of disk for redundancy, and people's time to ensure backups are moving swiftly etc.

When deciding to use logical, physical deletes, or archiving I would ask myself these questions:

1. Is this data that might need to be re-inserted into the table. For example User Accounts fit this category as you might activate or deactivate a user account. If

this is the case a logical delete makes the most sense.

2. Is there any intrinsic value in storing the data? If so how much data will be generated. Depending on this I would either go with a logical delete, or implement an archiving strategy. Keep in mind you can always archive logically deleted records.

Share Improve this answer

Follow

edited Jan 23, 2022 at 11:59

Pedro Luz

2,772 ● 4 ● 46 ● 56

answered Dec 18, 2008 at 16:36

JoshBerke

67k ● 25 ● 129 ● 170

In your user accounts example, would it be good to keep actived and deactived users in separate tables ? Eg. `Activated` table and `Deactivated` table schema - `Id,Name,etc..` Row in `Activated` - `1001,Smith007,etc...` When he is deactivated, then we can clear all but ID column for smith in `Activated` and add him to `Deactivated` . – Erran Morad Apr 13, 2014 at 2:00

What benefit is there in moving all the data if your going to leave the Id and the row? Maybe if your record is huge but I'd look at that as a micro-optimization. – JoshBerke May 13, 2014 at 13:45

Good luck with cascading foreign key constraints if you are moving data around tables. – CAD bloke Jul 9, 2017 at 13:57

30

It might be a little late but I suggest everyone to check Pinal Dave's blog post about logical/soft delete:

> I just do not like this kind of design [soft delete] at all. I am firm believer of the architecture where only necessary data should be in single table and the useless data should be moved to an archived table. Instead of following the isDeleted column, I suggest the usage of two different tables: one with orders and another with deleted orders. In that case, you will have to maintain both the table, but in reality, it is very easy to maintain. When you write UPDATE statement to the isDeleted column, write INSERT INTO another table and DELETE it from original table. If the situation is of rollback, write another INSERT INTO and DELETE in reverse order. If you are worried about a failed transaction, wrap this code in TRANSACTION.
>
> What are the advantages of the smaller table verses larger table in above described situations?
>
> - A smaller table is easy to maintain
> - Index Rebuild operations are much faster
> - Moving the archive data to another filegroup will reduce the load of primary filegroup (considering that all filegroups are on

different system) – this will also speed up the backup as well.

- Statistics will be frequently updated due to smaller size and this will be less resource intensive.
- Size of the index will be smaller
- Performance of the table will improve with a smaller table size.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Sep 30, 2014 at 16:42

Tohid

6,649 ● 8 ● 58 ● 81

25 how would you take care of foreign keys using such method? There may be 1, 10 or more other tables that reference the record being deleted and move to another table! – sam360 Jun 7, 2015 at 20:21

@sam360 - that's a big challenge. To be honest, I personally failed to implement the above recommendation in my projects, because of handling the PK and relationship between tables. Unfortunately there wasn't a real world example in that article. I'm working on a solution in one of my project, if it turned out to be a good implementation, I'll share the code with you... – Tohid Jun 8, 2015 at 14:09

what is it called ? instead of soft-delete? – eugene Dec 20, 2015 at 10:22

1 @eugene - I don't know any specific term for this solution. It's a **really "delete" rows and keep deleted records in an "archive" table** approach, if it makes sense to ya. – Tohid Dec 20, 2015 at 20:37

1 I believe, "Moving the archive data to another filegroup" can be implemented as partitions in Oracle, so one gets the benefits listed above... – Betlista Oct 28, 2018 at 16:13

18

I'm a NoSQL developer, and on my last job, I worked with data that was always critical for someone, and if it was deleted by accident in the same day that was created, I were not able to find it in the last backup from yesterday! In that situation, soft deletion always saved the day.

I did soft-deletion using timestamps, registering the date the document was deleted:

```
IsDeleted = 20150310  //yyyyMMdd
```

Every Sunday, a process walked on the database and checked the `IsDeleted` field. If the difference between the current date and the timestamp was greater than N days, the document was hard deleted. Considering the document still be available on some backup, it was safe to do it.

EDIT: This NoSQL use case is about big documents created in the database, tens or hundreds of them every day, but not thousands or millions. By general, they were documents with the status, data and attachments of

workflow processes. That was the reason why there was the possibility of a user deletes an important document. This user could be someone with Admin privileges, or maybe the document's owner, just to name a few.

TL;DR My use case was not Big Data. In that case, you will need a different approach.

Share Improve this answer Follow

edited Feb 15, 2017 at 7:05

answered Mar 11, 2015 at 3:35

Mario S

1,984 ● 1 ● 20 ● 32

12

One pattern I have used is to create a mirror table and attach a trigger on the primary table, so all deletes (and updates if desired) are recorded in the mirror table.

This allows you to "reconstruct" deleted/changed records, and you can still hard delete in the primary table and keep it "clean" - it also allows the creation of an "undo" function, and you can also record the date, time, and user who did the action in the mirror table (invaluable in witch hunt situations).

The other advantage is there is no chance of accidentally including deleted records when querying off the primary unless you deliberately go to the trouble of including records from the mirror table (you may want to show live and deleted records).

Another advantage is that the mirror table can be independently purged, as it should not have any actual foreign key references, making this a relatively simple operation in comparison to purging from a primary table that uses soft deletes but still has referential connections to other tables.

What other advantages? - great if you have a bunch of coders working on the project, doing reads on the database with mixed skill and attention to detail levels, you don't have to stay up nights hoping that one of them didn’t forget to not include deleted records (lol, Not Include Deleted Records = True), which results in things like overstating say the clients available cash position which they then go buy some shares with (i.e., as in a trading system), when you work with trading systems, you will find out very quickly the value of robust solutions, even though they may have a little bit more initial "overhead".

Exceptions:
- as a guide, use soft deletes for "reference" data such as user, category, etc, and hard deletes to a mirror table for "fact" type data, i.e., transaction history.

edited May 10, 2019 at 12:27

DJo
2,169 ● 4 ● 30 ● 48

answered Sep 9, 2016 at 0:18

Code Warrior
117 ● 1 ● 6

9

I used to do soft-delete, just to keep old records. I realized that users don't bother to view old records as often as I thought. If users want to view old records, they can just view from archive or audit table, right? So, what's the advantage of soft-delete? It only leads to more complex query statement, etc.

Following are the things i've implemented, before I decided to not-soft-delete anymore:

1. implement audit, to record all activities (add,edit,delete). Ensure that there's no foreign key linked to audit, and ensure this table is secured and nobody can delete except administrators.
2. identify which tables are considered "transactional table", which very likely that it will be kept for long time, and very likely user may want to view the past records or reports. For example; purchase transaction. This table should not just keep the id of master table (such as dept-id), but also keep the additional info such as the name as reference (such as dept-name), or any other necessary fields for reporting.
3. Implement "active/inactive" or "enable/disable" or "hide/show" record of master table. So, instead of deleting record, the user can disable/inactive the master record. It is much safer this way.

Just my two cents opinion.

Share Improve this answer

Follow

answered Jun 6, 2016 at 3:06

David

91 ● 1 ● 2

7

I'm a big fan of the logical delete, especially for a Line of Business application, or in the context of user accounts. My reasons are simple: often times I don't want a user to be able to use the system anymore (so the account get's marked as deleted), but if we deleted the user, we'd lose all their work and such.

Another common scenario is that the users might get re-created a while after having been delete. It's a much nicer experience for the user to have all their data present as it was before they were deleted, rather than have to re-create it.

I usually think of deleting users more as "suspending" them indefinitely. You never know when they'll legitimately need to be back.

Share Improve this answer

Follow

answered Dec 18, 2008 at 16:24

Jon Dewees

2,957 ● 6 ● 31 ● 38

2 Shouldn't we use something like account activation/deactivation instead of logical deletion here ? @jon-dewees – Eagle_Eye May 27, 2020 at 7:26

One common pattern for this is to create a two-level user data structure. So a user will have an login account where you can do SSO with other system and there is another

system account that associates with all business records. You can delete the first level user login account while retaining the ability to relink the underneath system account later if needed. – KuN Feb 14, 2021 at 20:38

6

I almost always soft delete and here's why:

- you can restore deleted data if a customer asks you to do so. More happy customers with soft deletes. Restoring specific data from backups is complex
- checking for `isdeleted` everywhere is not an issue, you have to check for `userid` anyway (if the database contains data from multiple users). You can enforce the check by code, by placing those two checks on a separate function (or use views)
- graceful delete. Users or processes dealing with deleted content will continue to "see" it until they hit the next refresh. This is a very desirable feature if a process is processing some data which is suddenly deleted
- synchronization: if you need to design a synchronization mechanism between a database and mobile apps, you'll find soft deletes much easier to implement

Share Improve this answer Follow

edited May 31, 2019 at 8:14

answered Jun 9, 2018 at 12:44

Gianluca Ghettini
11.6k ● 22 ● 99 ● 167

2 @Jim persist data onto a database it is not illegal. it's illegal if you keep the records even after the customer told you to remove its own data. Soft deletes are perfectly compatible with GDPR: upon request, just overwrite the sensible data with blank data. Moreover, if a user delete a record, it may want to undo the action later in the future or restore the data somehow... it does't mean he/she wants the data to disappear completely from the database – Gianluca Ghettini Jun 20, 2019 at 9:49

5

I commonly use logical deletions - I find they work well when you also intermittently archive off the 'deleted' data to an archived table (which can be searched if needed) thus having no chance of affecting the performance of the application.

It works well because you still have the data if you're ever audited. If you delete it physically, **it's gone**!

Share Improve this answer

Follow

answered Dec 18, 2008 at 16:17

Galwegian
42.2k ● 16 ● 113 ● 158

4

Re: "Is this secure?" - that depends on what you mean.

If you mean that by doing physical delete, you'll **prevent anyone from ever finding the deleted data**, then yes, that's more or less true; you're safer in physically deleting the sensitive data that needs to be erased, because that means it's permanently gone from the database. (However, realize that there may be other copies of the data in question, such as in a backup, or the transaction log, or a recorded version from in transit, e.g. a packet sniffer - just because you delete from your database doesn't guarantee it wasn't saved somewhere else.)

If you mean that by doing logical delete, your data is more secure because **you'll never lose any data**, that's also true. This is good for audit scenarios; I tend to design this way because it admits the basic fact that once data is generated, it'll never *really* go away (especially if it ever had the capability of being, say, cached by an internet search engine). Of course, a real audit scenario requires that not only are deletes logical, but that updates are also logged, along with the time of the change and the actor who made the change.

If you mean that the data won't fall into the hands of anyone who isn't supposed to see it, then that's totally up to your application and its security structure. In that respect, logical delete is no more or less secure than anything else in your database.

Follow

answered Dec 18, 2008 at 16:32

Ian Varley

9,437 ● 5 ● 31 ● 34

3

Logical deletions if are hard on referential integrity.

It is the right think to do when there is a temporal aspect of the table data (are valid FROM_DATE - TO_DATE).

Otherwise move the data to an Auditing Table and delete the record.

On the plus side:

It is the easier way to rollback (if at all possible).

It is easy to see what was the state at a specific point in time.

Share Improve this answer

Follow

answered Dec 18, 2008 at 16:31

pkario

2,220 ● 6 ● 28 ● 30

3

I strongly **disagree** with logical delete because you are exposed to many errors.

First of all queries, each query must take care the IsDeleted field and the possibility of error becomes higher with complex queries.

Second the performance: imagine a table with 100000 recs with only 3 active, now multiply this number for the

tables of your database; another performance problem is a possible conflict with new records with old (deleted records).

The only advantage I see is the history of records, but there are other methods to achieve this result, for example you can create a logging table where you can save info: `TableName,OldValues,NewValues,Date,User,[..]` where `*Values` can be `varchar` and write the details in this form `fieldname : value`; [..] or store the info as `xml`.

All this can be achieved via code or Triggers but you are only **ONE** table with all your history. Another options is to see if the specified database engine are native support for tracking change, for example on SQL Server database there are SQL Track Data Change.

Share Improve this answer

Follow

answered Sep 30, 2014 at 17:18

Max

7,080 ● 3 ● 50 ● 64

1 Nice points, but dealing with performance is doable by adding a partial index on it. – lucastamoios May 4, 2021 at 17:08

3

There are requirements beyond system design which need to be answered. What is the legal or statutory requirement in the record retention? Depending on what the rows are related to, there may be a legal requirement

that the data be kept for a certain period of time after it is 'suspended'.

On the other hand, the requirement may be that once the record is 'deleted', it is truly and irrevocably deleted. Before you make a decision, talk to your stakeholders.

Share Improve this answer

Follow

answered Feb 20, 2015 at 13:03

Dave

31 ● 1

2

It's fairly standard in cases where you'd like to keep a history of something (e.g. user accounts as @Jon Dewees mentions). And it's certainly a great idea if there's a strong chance of users asking for un-deletions.

If you're concerned about the logic of filtering out the deleted records from your queries getting messy and just complicating your queries, you can just build views that do the filtering for you and use queries against that. It'll prevent leakage of these records in reporting solutions and such.

Share Improve this answer

Follow

answered Dec 18, 2008 at 16:34

BQ.

9,413 ● 3 ● 27 ● 35

2

Mobile apps that depend on synchronisation might impose the use of logical rather than physical delete: a server must be able to indicate to the client that a record

has been (marked as) deleted, and this might not be possible if records were physically deleted.

Share Improve this answer Follow

answered Jul 15, 2015 at 9:10

axd

636 ● 5 ● 13

2

I just wanted to expand on the mentioned *unique constraint* problem.

Suppose I have a table with two columns: `id` and `my_column.` To support soft-deletes I need to update my table definition to this:

```
create table mytable (
  id serial primary key,
  my_column varchar unique not null,
  deleted_at datetime
)
```

But if a row is soft-deleted, I want `my_column` constraint to be ignored, because deleted data should not interfere with non-deleted data. My original model will not work.

I would need to update my data definition to this:

```
create table mytable (
  id serial primary key,
  my_column varchar not null,
  my_column_repetitions integer not null default 0,
  deleted_at datetime,
  unique (my_column, my_column_repetitions),
  check (deleted_at is not null and my_column_repetiti
```

```
null and my_column_repetitions = 0)
)
```

And apply this logic: when a row is current, i.e. not deleted, `my_column_repetitions` should hold the default value `0` and when the row is soft-deleted its `my_column_repetitions` needs to be updated to `(max. number of repetitions on soft-deleted rows) + 1.`

The latter logic must be implemented programmatically with a trigger or handled in my application code and there is no check that I could set.

Repeat this is for every unique column!

I think this solution is really hacky and would favor a separate *archive* table to store deleted rows.

Share Improve this answer

Follow

answered Aug 24, 2020 at 10:55

clapas

1,846 ● 3 ● 19 ● 31

1

They don't let the database perform as it should rendering such things as the cascade functionality useless.

For simple things such as inserts, in the case of re-inserting, then the code behind it doubles.

You can't just simply insert, instead you have to check for an existence and insert if it doesn't exist before or update the deletion flag if it does whilst also updating all other columns to the new values. This is seen as an update to

the database transaction log and not a fresh insert causing inaccurate audit logs.

They cause performance issues because tables are getting glogged with redundant data. It plays havock with indexing especially with uniqueness.

I'm not a big fan of logical deletes.

Share Improve this answer

Follow

edited Oct 11, 2012 at 19:13

matthias krull

4,429 ● 3 ● 35 ● 54

answered Jul 4, 2012 at 15:47

Taqveem

11 ● 1

1

To reply to Tohid's comment, we faced same problem where we wanted to persist history of records and also we were not sure whether we wanted `is_deleted` column or not.

I am talking about our python implementation and a similar use-case we hit.

We encountered https://github.com/kvesteri/sqlalchemy-continuum which is an easy way to get versioning table for your corresponding table. Minimum lines of code and captures history for add, delete and update.

This serves more than just `is_deleted` column. You can always backref version table to check what happened

with this entry. Whether entry got deleted, updated or added.

This way we didn't need to have `is_deleted` column at all and our delete function was pretty trivial. This way we also don't need to remember to mark `is_deleted=False` in any of our api's.

Share Improve this answer Follow

edited Sep 8, 2017 at 23:59

answered Sep 8, 2017 at 2:54

Lalit

29 • 7

0

Soft Delete is a programming practice that being followed in most of the application when data is more relevant. Consider a case of financial application where a delete by the mistake of the end user can be fatal. That is the case when soft delete becomes relevant. In soft delete the user is not actually deleting the data from the record instead its being flagged as IsDeleted to true (By normal convention).

In EF 6.x or EF 7 onward Softdelete is Added as an attribute but we have to create a custom attribute for the time being now.

I strongly recommend SoftDelete In a database design and its a good convention for the programming practice.

Share Improve this answer

Follow

answered Nov 14, 2014 at 7:40

Sanu Antony
364 ● 4 ● 15

0

Most of time softdeleting is used because you don't want to expose some data but you have to keep it for historical reasons (A product could become discontinued, so you don't want any new transaction with it but you still need to work with the history of sale transaction). By the way, some are copying the product information value in the sale transaction data instead of making a reference to the product to handle this.

In fact it looks more like a rewording for a visible/hidden or active/inactive feature. Because that's the meaning of "delete" in business world. I'd like to say that Terminators may delete people but boss just fire them.

This practice is pretty common pattern and used by a lot of application for a lot of reasons. As It's not the only way to achieve this, so you will have thousand of people saying that's great or bullshit and both have pretty good arguments.

From a point of view of security, SoftDelete won't replace the job of Audit and it won't replace the job of backup too. If you are afraid of "the insert/delete between two backup case", you should read about Full or Bulk recovery Models. I admit that SoftDelete could make the recovery process more trivial.

Up to you to know your requirement.

Share Improve this answer

Follow

answered Sep 19, 2017 at 14:34

Marco Guignard
643 ● 3 ● 9

0

To give an alternative, we have users using remote devices updating via MobiLink. If we delete records in the server database, those records never get marked deleted in the client databases.

So we do both. We work with our clients to determine how long they wish to be able to recover data. For example, generally customers and products are active until our client say they should be deleted, but history of sales is only retained for 13 months and then deletes automatically. The client may want to keep deleted customers and products for two months but retain history for six months.

So we run a script overnight that marks things logically deleted according to these parameters and then two/six months later, anything marked logically deleted today will be hard deleted.

We're less about data security than about having enormous databases on a client device with limited memory, such as a smartphone. A client who orders 200 products twice a week for four years will have over 81,000 lines of history, of which 75% the client doesn't care if he sees.

answered Feb 9, 2018 at 21:35
TychaBrahe
49 • 6

It all depends on the use case of the system and its data.

For example, if you are talking about a government regulated system (e.g. a system at a pharmaceutical company that is considered a part of the quality system and must follow FDA guidelines for electronic records), then you darned well better not do hard deletes! An auditor from the FDA can come in and ask for all records in the system relating to product number ABC-123, and all data better be available. If your business process owner says the system shouldn't allow anyone to use product number ABC-123 on new records going forward, use the soft-delete method instead to make it "inactive" within the system, while still preserving historical data.

However, maybe your system and its data has a use case such as "tracking the weather at the North Pole". Maybe you take temperature readings once every hour, and at the end of the day aggregate a daily average. Maybe the hourly data will no longer ever be used after aggregation, and you'd hard-delete the hourly readings after creating the aggregate. (This is a made-up, trivial example.)

The point is, it all depends on the use case of the system and its data, and not a decision to be made purely from a technological standpoint.

Share Improve this answer

Follow

edited Nov 16, 2018 at 1:59

Pang

10.1k ● 146 ● 85 ● 124

answered Apr 13, 2018 at 19:19

HardCode

6,746 ● 5 ● 32 ● 56

0

Well! As everyone said, it depends on the situation.

If you have an index on a column like UserName or EmailID - and you never expect the same UserName or EmailID to be used again; you can go with a soft delete.

That said, always check if your SELECT operation uses the primary key. If your SELECT statement uses a primary key, adding a flag with the WHERE clause wouldn't make much difference. Let's take an example (Pseudo):

Table Users (UserID [primary key], EmailID, IsDeleted)

```
SELECT * FROM Users where UserID = 123456 and IsDelete
```

This query won't make any difference in terms of performance since the UserID column has a primary key. Initially, it will scan the table based on PK and then execute the next condition.

Cases where soft deletes cannot work at all:

Sign-up in majorly all websites take EmailID as your unique identification. We know very well, once an EmailID is used on a website like facebook, G+, it cannot be used by anyone else.

There comes a day when the user wants to delete his/her profile from the website. Now, if you make a logical delete, that user won't be able to register ever again. Also, registering again using the same EmailID wouldn't mean to restore the entire history. Everyone knows, deletion means deletion. In such scenarios, we have to make a physical delete. But in order to maintain the entire history of the account, we should always archive such records in either archive tables or deleted tables.

Yes, in situations where we have lots of foreign tables, handling is quite cumbersome.

Also keep in mind that soft/logical deletes will increase your table size, so the index size.

Share Improve this answer

Follow

edited May 10, 2019 at 12:12

DJo

2,169 ● 4 ● 30 ● 48

answered Jun 20, 2015 at 14:12

1JD

325 ● 3 ● 5

I have already answered in another post. However, I think my answer more fit to the question here.

0

My practical solution for soft-delete is archiving by creating a new table with following columns: `original_id`, `table_name`, `payload`, (and an optional primary key `id).

Where `original_id` is the original id of deleted record, `table_name` is the table name of the deleted record (`"user"` in your case), `payload` is JSON-stringified string from all columns of the deleted record.

I also suggest making an index on the column `original_id` for latter data retrievement.

By this way of archiving data. You will have these advantages

- Keep track of all data in history
- Have only one place to archive records from any table, regardless of the deleted record's table structure
- No worry of unique index in the original table
- No worry of checking foreign index in the original table
- No more `WHERE` clause in every query to check for deletion

The is already a discussion here explaining why soft-deletion is not a good idea in practice. Soft-

> delete introduces some potential troubles in future such as counting records, ...

Share Improve this answer

Follow

answered Sep 27, 2019 at 2:41

Sang

4,437 ● 4 ● 40 ● 51

I have written a blog post on all ways of data deletion transang.me/database-design-practice-soft-deletion-to – Sang Sep 27, 2019 at 10:15

0

It depends on the case, consider the below:

Usually, you don't need to "soft-delete" a record. Keep it simple and fast. *e.g. Deleting a product no longer available, so you don't have to check the product isn't soft-deleted all over your app (count, product list, recommended products, etc.).*

Yet, you might consider the "soft-delete" in a data warehouse model. *e.g.* You are viewing an old receipt on a deleted product.*

Share Improve this answer

Follow

answered Jan 17, 2020 at 22:17

Kris Khairallah

1,582 ● 15 ● 16

Advantages are data preservation/perpetuation. A disadvantage would be a decrease in performance when

0

querying or retrieving data from tables with significant number of soft deletes.

In our case we use a combination of both: as others have mentioned in previous answers, we `soft-delete` `users/clients/customers` for example, and `hard-delete` on `items/products/merchandise` tables where there are duplicated records that don't need to be kept.

Share Improve this answer

Follow

edited Aug 24, 2021 at 20:04

marc_s

753k ● 183 ● 1.4k ● 1.5k

answered Nov 15, 2019 at 18:03

Máster

1,001 ● 12 ● 25