How to build a table for a private messaging system that supports replies? [on hold]

I'm writing a private messaging web application in PHP, the application allows replies such that when you view a message, you also get to see to what was that a reply to, and to what was that a reply to and to what was that a reply to and so on and so on.

I'm trying to find a right database structure that would avoid redundancy, so I'm wondering how can I link a single message to all other messages that it is replying to?

I thought about basically having a field called reply_id which would be a serialized array holding the IDs of all messages to be presented as replies

Does anyone have a suggestion on how to do this efficiently? Is my thought a good practice?

php database database-design relational-database

asked Apr 15 at 22:48
Ali
2,699 1 4 21

put on hold as too broad by Louis, Infinite Recursion, jonrsharpe, Rizier123, Qantas 94 Heavy 5 hours ago

There are either too many possible answers, or good answers would be too long for this format. Please add details to narrow the answer set or to isolate an issue that can be answered in a few paragraphs.

If this question can be reworded to fit the rules in the help center, please edit the question.

3 I disagree. Re-open the question. It is not broad, let alone too broad, the scope is clear. The question is understandable to a data modeller or DBA (it may not be to others or to modellers/DBAs who do not understand hierarchies). A "good answer" is a few sort paragraphs. Some responders might add a simple diagram. There are not too too many correct answers, just one AFAIC. – PerformanceDBA Apr 18 at 12:20

1 @@Ali. (1) Please confirm, a Reply is the same as a Message, except for two things: (a) a Reply has a parent Message or Reply, and a Message doesn't have a parent. (b) Messages can start a Thread, and a Reply can't. In that case the solution is straight-forward. (2) Do you want the simple one or the rock-solid, easy-to-enhance one ? – PerformanceDBA Apr 18 at 12:26

@PerformanceDBA You are correct, a reply is the same as a message except that it has a parent message, and the parent message has a parent message, and so on. I prefer the rock-solid. – Ali 2 days ago

@@Ali. (2) The rock-solid one is for people who can play with swords without killing themselves. You will need relevant Relational Database experience. (3) I have opened a discussion in an attempt to have the question re-opened. – PerformanceDBA yesterday

@@Ali. Please review and comment. – PerformanceDBA 11 hours ago

2 Answers

Hierarchy

What you have described is an Hierarchy, a tree structure. The universe is full of hierarchies, they occur naturally. Eg. the famous Unix FileNode system; any organisation chart; etc. Once you have that, you can traverse the tree any way you like.

- The *Relational Model* provides for hierarchies really well, they are fundamental to it.
- It is pretty standard, and common in high-end systems, you need an Hierarchy to support a tree structure, and we have lots of them.

If you are *not* aware that it is an hierarchy, and you do not set it up as one, the result is various complications and complexities, in bothe the database and the code. If you are aware, then the data structure required, and the code to support it, are simple.

Data Model

This is the data model, for the requirement that I have gathered from your question.

- Depending on your level of experience, it may or may not be easy to follow. The fact that it is *so* simple and *so* tight bothers some people ... they think there is something that they are not getting.

- As far as I can ascertain, the top-most Messages (let's call them Threads) and Replies are the same, except that a Thread has no parent Message, and a Reply has a parent Message. If there are no other differences, then this structure is probably final; if there are, you might need a basetype::subtype structure.

Message Data Model

Predicate

Please visit **this Answer**, scroll down to the **Predicate** section, and read it.

At this stage, given this model, and subject to your confirmation:

- Upon table creation, a single row needs to be inserted for the Root Message, before the FK Constraints are declared:

```
INSERT Message VALUES ( 0, 0 "Admin", "Root", <date, "", ... )
```

- A Thread has Root as its parent, a Reply has a parent that is not Root. The method for determining a Thread vs a Reply is therefore dead simple:

```
WHERE MessageNo_Parent  = 0  -- Thread
WHERE MessageNo_Parent != 0  -- Reply
```

- Depending on your INSERT vs Query considerations, I can give you a different PK, but that should be discussed. The simple case is given for now: single `MessageNo` as Primary; `(MessageNo_Parent, MessageNo)` as Alternate.

> *I'm trying to find a right database structure that would avoid redundancy*

This has no redundancy or duplication whatsoever. Genuine Fifth Normal Form, no Update Anomalies.

Actually the efficiency cannot be improved.

- On high-end systems, use the AK as the Clustered Index, and you have a tree straight from Heaven.

> *I'm wondering how can I link a single message to all other messages that it is replying to?*

I understood the rest of your Question completely, but I am a bit uncertain about that sentence.

- Since a Reply is to a single Message, and all the related Messages form a branch of the tree (a pure tree itself), I took that "*link to all Messages*" to mean all the Messages above it in the tree. In that case, it is is easy: just traverse the tree, vertically, seeking parents.

- If that is not the case, please advise, and I will respond.

Recursion

The tree can be traversed easily, either horizontally (across the Leaf Nodes) or vertically (up via parents, or down via children). However, You need recursion.

By way of explanation:

- In high-end platforms, we have had full recursion in the server since 1984. That means, we wrote simple stored procs or functions that were coded to be called recursively, and the fact that the server had recursive capability, produced the result that any tree (unlimited height, unlimited width) could be traversed. No cursors; no loops; no tricks.

- However, we can't perform a traversal via a single SELECT.

- In the last decade or so some platforms have provided CTEs.

  - That provides recursion, and it saves writing a proc
    (Great for those who *cannot write* a recursive proc. Much like the HIERARCHYID Datatype, for those who cannot implement a data model such as I have given).

  - The real benefit of a CTE is, a tree can be traversed via a single SELECT. It is not simple to write, and it is cumbersome as hell to execute, it builds worktables, and uses masses of resources. But that is the cost of being unable to write a recursive proc.

Re recursion or CTEs, I don't know what facilities your Non-SQL platform provides. The worst case is, you might have to write a loop.

edited 2 hours ago

answered 11 hours ago
PerformanceDBA
10.9k 2 22 47

I see the razor gang is already out, sharp and shiny. It is a great compliment to be followed like this. Thanks ! – PerformanceDBA 9 hours ago

1 Thank you, I understand your answer perfectly. I'd also like to thank you for all the hate that's been coming up against you. Although I agree that I didn't write my question in the best way possible, but you've done a great job with your knowledge. It seems like I will have to write a recursive function to find all the replies to a thread and it seems like the best way. Thank you. – Ali 7 hours ago

@Ali. The pleasure is all mine. Dealing with freaks and ignorant people is just something that comes with the territory. I am a bit surprised that the data model fits your exact requirement. God is merciful ! Yes, a recursive function or proc is the highest performance. If you have trouble finding *good* resources, email me, and I will send you some example code. I have tutorials but they are not for the public. – PerformanceDBA 2 hours ago

I'm not sure how efficient this would be, but you could have a table with 2 columns, one column for the ID of the message (storing the message in a different table to prevent duplication/nulls), and one column for the ID of the message being replied to.

So you could have multiple rows in this table for 1 message.

answered Apr 15 at 23:05
Mex
807 2 15