Keeping one table or multiple table for similar type of data which one is best while considering high performance

Asked 4 years, 9 months ago Active 3 months ago Viewed 2k times

2

2

I am designing DATABASE for a Sales and Purchase application like ERP and using MYSQL as RDBMS, I have doubt on creating table for sales and purchase entities to go with single table for each module(Sale/Purchase) or multiple tables for each entities(Sales order, Sale invoice, Sale return, Purchase order, Purchase invoice, Purchase return) in longer run. Below is my use case.

My application will have Sale Order, Sale Delivery, Sale Invoice, Sale Return and Credit Note and same entity for Purchase module also and all these entity may be enter linked in there module. Like Sale Order can be converted into Sale Delivery or Sale Order and Sale Delivery can be converted into Sale Invoice. So there need to maintain reference b/w each entity of a module.

Now, I am little confuse to keep all this in one table for each module say "sale_entity" and "purchase_entity" having entity type Or should I create separate table for each entity type say sale_order, sale_invoice, sale_return, purchase_order, purchase_invoice, purchase_return etc.

Below is what running in my mind for both the cases:

Single Table: I really want to keep this in single table for each module but I am worry about performance in longer run, It will increase the table size quickly and may slow down the performance.

Multiple table: It will be difficult to manage, maintain relationships and fetching data in reports for all entity types of records at once, requires union and all.

My understanding is that large size of table performs slower than the small size of table, Please correct if I am wrong.

Please put some light on it, and suggest me how should I proceed.

Thank You

mysql database database-design rdbms erp

asked Sep 18 '15 at 7:16

Irfan.gwb

502 ● 1 ● 5 ● 26

2 Learn about indexes and how to use them to make your queries efficient. Even tables with many millions of rows can work well and return the needed rows fast in most cases. – jkavalik Sep 18 '15 at 7:39

Honestly, the response that you marked as the answer is 100% not the answer. – stevenbranigan82 Mar 11 at 11:59

2 Answers

Active Oldest Votes

6

Rule of thumb: If two table have identical (or nearly identical) columns, make it one table, not two. You *may* need to add a column to distinguish the two types of data. You *may* need to have a column be `NULL` if it applies to one use but not the other. Too many `NULLable` columns --> don't combine the tables.

Rule of thumb: One-to-many and many-to-many relationships *require* having two or three tables. (The third is for many-to-many.)

"Orders" usually involved "Order Items". That is one row in `Orders` maps to one or more `OrderItems`. Those should be separate tables.

"Purchases" versus "Returns"? *Maybe* they could be in the same table, and distinguish by the sign of the `amount`?

answered Sep 18 '15 at 19:09
Rick James
93.3k ● 7 ● 84 ● 140

There will not be too many NULLable columns. my concern is just with size of the table and it's performance. does the size of table affect it and can we improve it using proper indexing? – Irfan.gwb Sep 19 '15 at 7:14

2 If your dataset is smaller than RAM, then generally table size does not matter much. For a much bigger dataset, I/O considerations dominate, and using smaller datatypes, etc, becomes a factor in speed. Indexing is very important for speed. In general, indexing hides "size". Fetching one particular indexed row in a billion-row table is not much slower than for a hundred-row table. – Rick James Sep 19 '15 at 13:14

5

Use separate tables.

I agree with the notion that you will have similar data in both tables but there is more to a Sales Order header and Sales Orderline than just the design of the table.

The concept of a sales order is very different logically than say the concept of a purchase order.

On it's most basic level one is directly related to a debtor account and one to a creditor account.

Both also have very different procedures when it comes to stock movements. E.g. One is goods inwards, which has a whole set of concepts like costing associated. Whereas the other is a goods outwards which has associated delivery information such as couriers, cost of delivery, etc.

I am assuming if you are trying to replicate an ERP then you will also have to have some sort of manufacturing works orders tables, which when considered together with a sales order and purchase order, interact very differently with stock transactions and stock locations. When you consider the differences between 1. receiving goods to a location during putaway, to 2. consuming raw materials receieved and generating new output items to different stock locations, to 3. sales orders for items that are generated from a works order and not part of the goods inwards receiving process, then you start to see that trying to dump all that logic into a single table would be complete madness.

You also need to consider the affects of having a single table in terms of programmability. What happens when you need to lock the table for a transaction? You have now locked all users from all data related to orders incl. Sales orders, purchase orders, and works orders.

So, in short, you need to separate the data with the logic.

I have never seen an ERP database that has a single table for all order types, or all stock transactions, or all debitor transactions, etc.

Anyone who says different is a bluffer of monumental proportions.

If in doubt remember that RELATIONAL database systems were designed so that you could separate data across multiple tables reducing the issues associated with single table designs, like performance, table locking, separation of logic (stored procedures, triggers, etc). Use sales order header numbers (or equivelant) as the relational key. Its a far tidier design principle.

answered Jun 4 '17 at 5:56

stevenbranigan82
150 ● 1 ● 2 ● 13