Subject: updating/discarding table data on dialogs...
Posted by indiocolifa on Mon, 18 Feb 2008 18:56:13 GMT
View Forum Message <> Reply to Message

This is the first time i'm using U++ for a medium sized data-aware application.

I'm simply asking how you manage data editing through dialogs, for example, how to discriminate accept (commit changes to DB) or cancel (Discarding all).

I think the best method (tell me what you think) is:

1. Load table data on local variables (class).
2. Let user touch data, if Button accept, copy from local to SQL tables.
3. If discarding, do nothing (do not commit changes).

Another approach I was thinking was transactions e.g:

1. Start Transaction.
2. Bind sqlArray to session/table.
3. Let user modify values,
4. If CANCEL, rollback.

How you manage data in your application? Are you using C++ classes which "map" to the DBMS tables? Transactions?

Thx for ya help.

Subject: Re: updating/discarding table data on dialogs...
Posted by mingodad on Mon, 18 Feb 2008 19:54:17 GMT
View Forum Message <> Reply to Message

In my opnion the first approach could allow for one user to overwrite work for another user.
The second approach prevent that but at the cost of block the database, no one will be able to work on that record reliable.

In my apps I use to include a field in every table that has high probability of concurrency and increment it at each update, when I retrieve data for edit I use the value of that field as one clause in that way the database is free and the faster user will be able to work, at expenses of the slow one.

Example:

```
create table test (
id int primary key,
rec_version int default 0,
data text
```

);

insert into test(data) values('First value');
-- id = 1, rec_version = 0, data = 'First value'

-- when one user edit
select * from test;
-- id = 1, rec_version = 0, data = 'First value'
-- let's suppose this user want data = 'Second value'
-- but he went to have a coffe

-- when another user edit too
select * from test;
-- id = 1, rec_version = 0, data = 'First value'
-- let's suppose this user want data = 'Flower value'

update test set
data = 'Flower value',
rec_version = rec_version +1
where id = 1 and rec_version = 0;
-- rec_version should be equal to when we got data to edit
-- id = 1, rec_version = 1, data = 'Flower value'

-- First user return from coffe
-- try update

update test set
data = 'Second value',
rec_version = rec_version +1
where id = 1 and rec_version = 0;

-- when checkin updated records we found 0
-- tell user that another user has changed the record
-- reload new data to replay

No locks to the database biggest concurrency possible, when updating more than one table, use transactions and the same
behavior.

When the update involve more than one record at randon, I use to have one record as semaphore.

I hope that this can help you and others !

Subject: Re: updating/discarding table data on dialogs...
Posted by unodgs on Mon, 18 Feb 2008 22:04:41 GMT

This is what I do. My primary condition is user can accept/reject values connected with data from one table. This condition simplify code a lot in more complicated dialogs.
In the simplest case when widgets on dialog represent data from one table I use SqlCtrl class (I made my own version which I think release to public) which allow to bind ctrl with table field. Now if user presses OK insert to database is geneated, if user presses Cancel nothing is done. If dialog is opened in update mode (I mean one is trying to alter existing data) update of modified fields is generated. So in this case you don't have to even open any transaction as insert/update on one table is atomic.
When dialog reflects data of connected tables (tables with foregin keys exactely) I organize my dialogs in sheets (tabctrl). First tab contain widgets connected with "top" - the most important table. As long as you stay in this first tab you can press cancel and discard inserting the record to database. If you change tab I do insert (after fields validation of course) and if insert ends with success active sheet is changed and cancel button goes to disable state. After succesful insert I update internal id (taken from database) and pass it to depentent inserts. The point is every operation in dialog within widgets binded to one table is immediate. I mean I don't use complicated data structures to store all values and I don't trace if record was inserted deleted etc and wait to final OK press to do antother complicated routine that persits that all in database. This way I can organize my code in much more cleaner way. And you can react much earlier if something goes wrong during inserting/updating.

This is your first scenario, second option is wrong IMO. Rollback should not be used to handle cancel operation. What about sequences or autincremented fields. You will waste to many numbers if user presses cancel too many times. Of course it's hard to reach end of counter (if it's 64bit) but it's not impossible.