Lost Update Problem

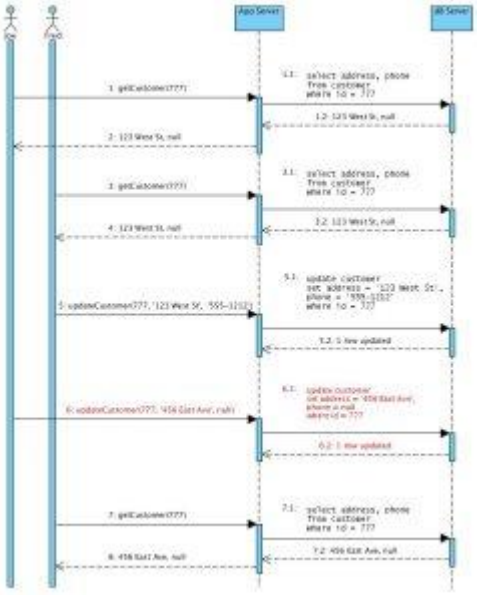
To illustrate the lost update problem, consider a web application which allows salespeople to maintain a list of clients. This web application allows any salesperson to view a list of clients and then click on any one client to update that client's information. The user must click the "Save" button to persist their changes to the database.

Using this web application, Joe (Salesperson) wants update Bob's (Client) record by adding Bob's new cell phone number. Joe performs a search, the application displays a matching list of client records, and Joe clicks on Bob's record. At this time, the data currently stored in the database for Bob is retrieved and displayed on a page in Joe's web browser. Joe decides to get a cup of coffee from the break room before making his change to Bob's record.

While Joe gets his coffee, Fred (Salesperson) wants to update the address on Bob's record. Like Joe, Fred accesses Bob's record via a search and is shown the data currently stored in the database for Bob. At this point, both Fred and Joe have the same data displayed in their web browser for Bob. Fred changes Bob's address in the form, clicks the "Save" button, and goes to Lunch.

Joe comes back with his coffee. At this point, Joe's view of Bob's data is not the current data for Bob in the database. Fred changed Bob's address and that change is not shown in Joe's web browser. Most web applications operate in this manner – they only send data to the user (Joe) when the user requests it. Now Joe adds Bob's cell phone number and clicks the "Save" button. This causes Bob's new phone number as well as the outdated address that was shown in his form to be saved in the database. Fred's update is now lost.
Fred gets back from lunch, searches for and views Bob's record again, and sees Bob's old address along with a new phone number. Fred is angered and fires off an email to his boss telling him the cool new sales web application is broken and he wants to use the old application.

As you can imagine, this is not good for the application development team or the salespeople. The diagram below illustrates the sequence of events leading up to the lost update.



As is often the case, each time the user clicks “Save” on the page in their web browser, all the data in their web form gets sent to the application server and written to the database. The flaw in this case is the application assumes the data submitted by the user is the current state of the data in the database. As you can see with the above example with Joe and Fred, this is not necessarily true.

To handle the lost update problem, you can use either optimistic concurrency control or pessimistic concurrency control.