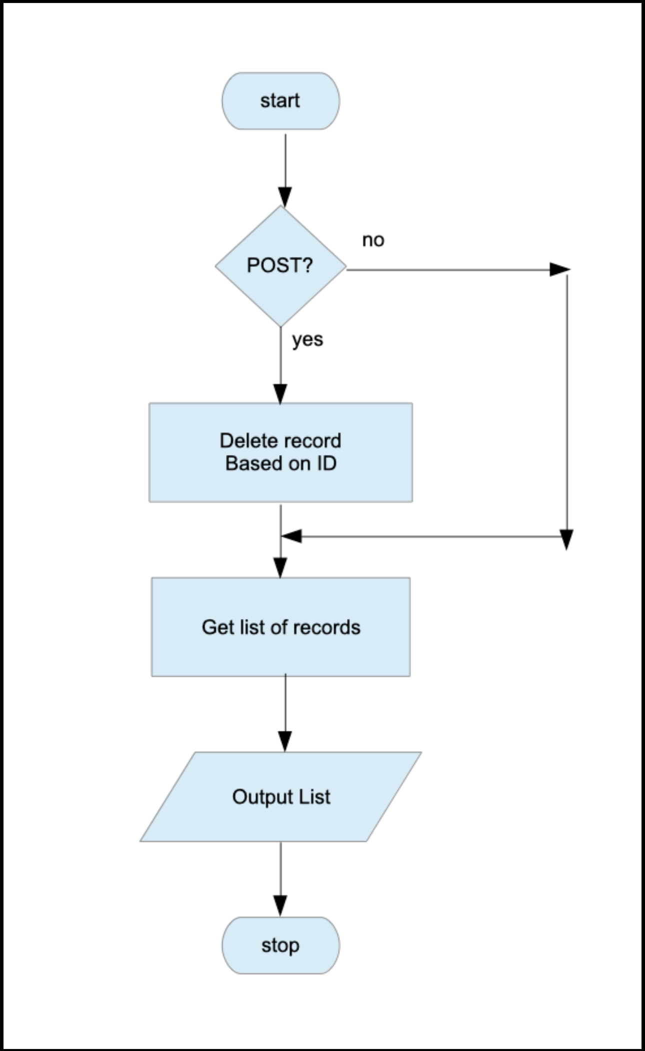
MySQL & PHP Deletes

Intro PHP

Download sample code from Nexus

Deleting records

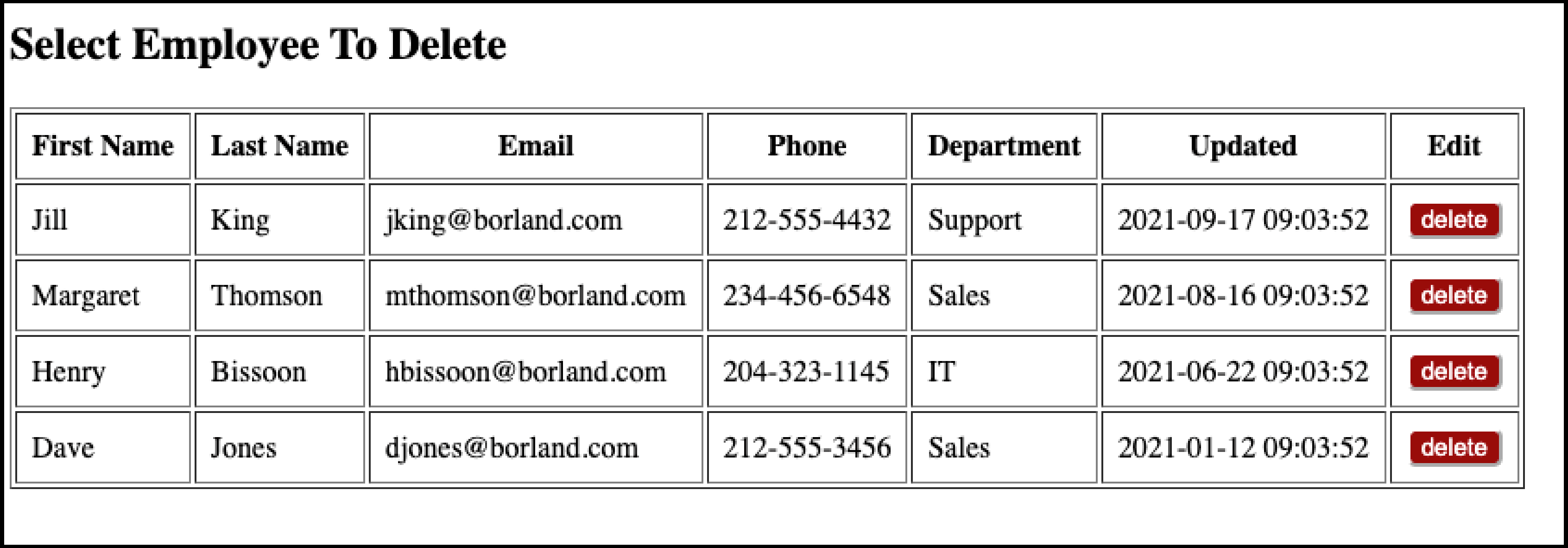
This is a simplified flowchart illustrating the procedure for deleting a record



02_edit.php

Start with a list view

Because we are working with existing records that need to be deleted, we generally start with a list view so we can find them.



Select Employee To Delete

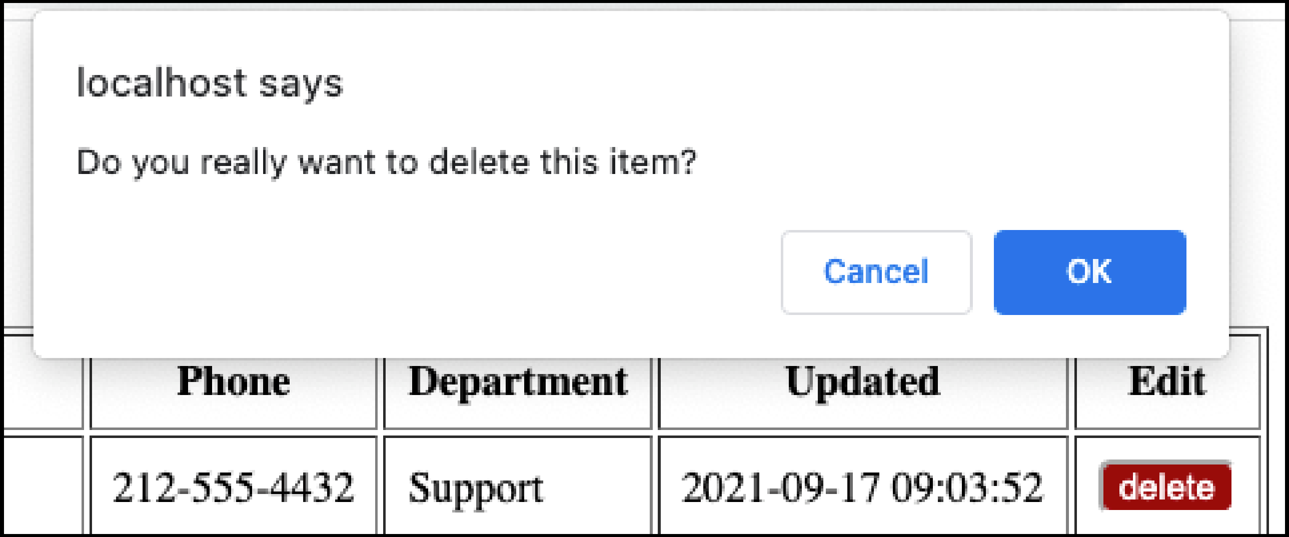
First Name	Last Name	Email	Phone	Department	Updated	Edit
Jill	King	jking@borland.com	212-555-4432	Support	2021-09-17 09:03:52	delete
Margaret	Thomson	mthomson@borland.com	234-456-6548	Sales	2021-08-16 09:03:52	delete
Henry	Bissoon	hbissoon@borland.com	204-323-1145	IT	2021-06-22 09:03:52	delete
Dave	Jones	djones@borland.com	212-555-3456	Sales	2021-01-12 09:03:52	delete

Note that we have a button, not a link, to execute the delte. This is because a POST request is required, and the button is a form.

01_hard_delete.php

Confirming the deletion

Unlike inserting or updating, there is not form or second page that must load. We will simply execute the delete query on the list view… but it's a good idea to confirm the user's intention!



localhost says

Do you really want to delete this item?

	Phone	Department	Updated	Edit
	212-555-4432	Support	2021-09-17 09:03:52	delete

01_hard_delete.php

Hard Delete – POST request

The simplest delete handler might look something like this:

```
if(isset($_POST['delete'])) {

  $query = "DELETE FROM employees WHERE id = :id";
  $stmt = $dbh->prepare($query);
  $stmt->bindValue(':id', $_POST['id'], PDO::PARAM_INT);
  $stmt->execute();

}
```

After the delteion is completed, we simply fall through to perform another query to display the list view... this time minus the record that was just deleted

01_hard_delete.php

Hard Delete – archive old record

In most cases, however, we will want to take the extra precaution of archiving any record we are deleting

```
$dbh->beginTransaction();

// Do the backup
$query = "REPLACE INTO employees_arc SELECT * FROM employees
WHERE id = :id";
$stmt = $dbh->prepare($query);
$stmt->bindValue(':id', $_POST['id'], PDO::PARAM_INT);
$stmt->execute();
// End backup

// Do the delete
$query = "DELETE FROM employees WHERE id = :id";
$stmt = $dbh->prepare($query);
$stmt->bindValue(':id', $_POST['id'], PDO::PARAM_INT);
$stmt->execute();
// End Delete

$dbh->commit();
} catch (Exception $e) {
```

Note, because we are executing more than one query, we must wrap our queries in a transaction to maintain ACID consistency, committing the transaction permanently ONLY if both queries execute successfully.

01_hard_delete.php

Hard Delete - transaction

If the transaction fails at any point, we must rollback any of the completed queries.

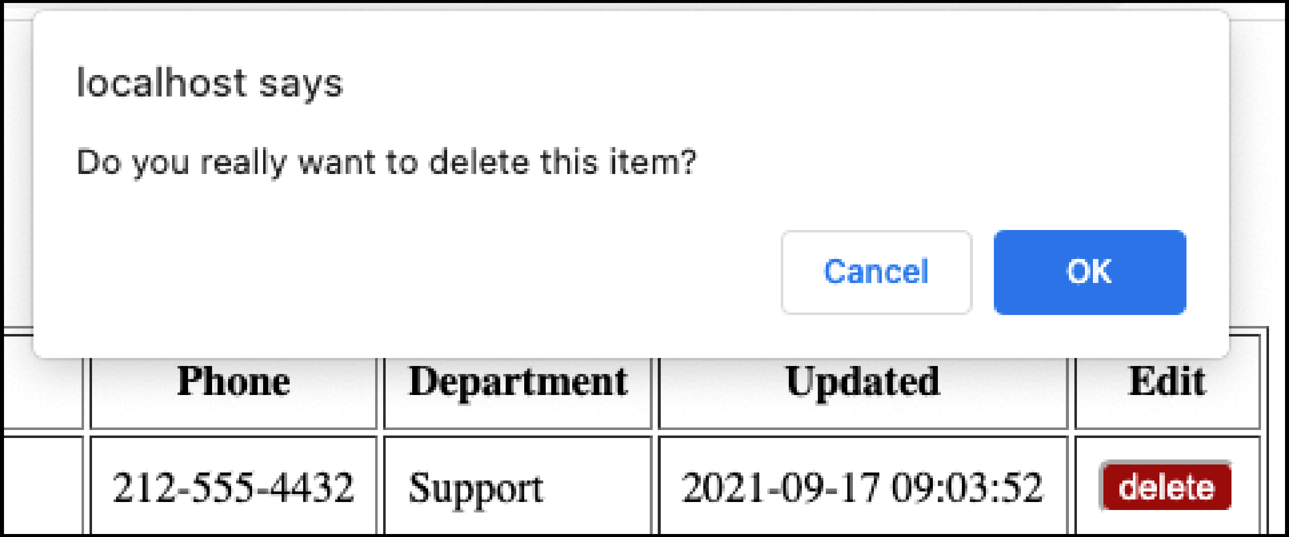
To perform the rollback, we need to wrap our entire transaction in a try/catch block, which will allow us to catch any exceptions and respond as required.

```
try{

  $dbh->beginTransaction();

  // query one here

  // query two here

  $dbh->commit();

} catch (Exception $e) {

  $dbh->rollback();
  echo $e->getMessage();

}
```

01_hard_delete.php

Soft Delete – begins the same way

Unlike inserting or updating, there is not form or second page that must load. We will simply execute the delete query on the list view... but it's a good idea to confirm the user's intention!


localhost says

Do you really want to delete this item?

	Phone	Department	Updated	Edit
	212-555-4432	Support	2021-09-17 09:03:52	delete

01_soft_delete.php

Soft Delete – update, don't delete

When performing a soft delete, we never actually delete the record. Instead, we simply update it, and set the value of a flag field to 1 or true, to show the record is deleted.

```
$query = "UPDATE employees SET deleted = 1 WHERE id = :id";
$stmt = $dbh->prepare($query);
$stmt->bindValue(':id', $_POST['id'], PDO::PARAM_INT);
$stmt->execute();
```

Because we don't actually delete the record, there is no need to create an archive copy. And because there is no need for a second query, there is no requirement for a transaction.

01_soft_delete.php

Soft Delete – requirements

Although no transaction or archive table is required for soft deletes, they do come with some added requirements.

```
CREATE TABLE employees (
  id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  phone VARCHAR(255),
  department VARCHAR(255),
  deleted BOOL NOT NULL DEFAULT 0,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP
);
```

First, we need to add an extra field to our table to facilitate the soft delete. By default, it is set to 0, or false, indicating the record is not deleted.

01_soft_delete.php

Soft Delete – requirements

Although no transaction or archive table is required for soft deletes, they do come with some added requirements.

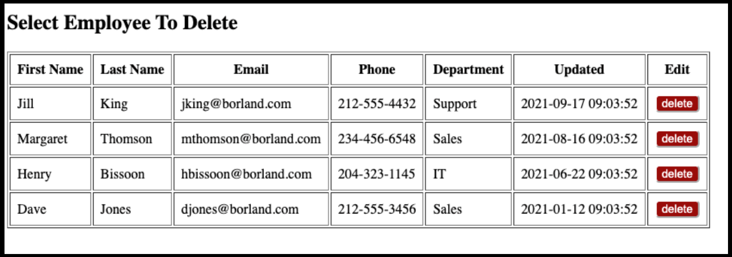
```
$query = "SELECT *
        FROM employees
        WHERE deleted != 1
        ORDER BY
        updated_at DESC";
```

Second, every single query we make in every other part of the site MUST take into account deleted records and take care never to display them

01_soft_delete.php

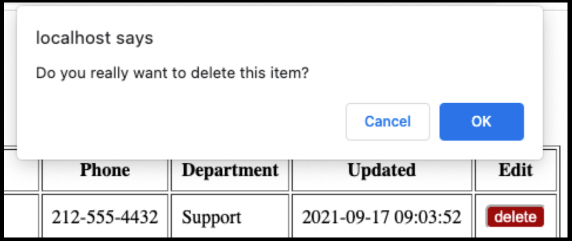
The Delete workflow

Deleting records is such a common administrative task it has an established workflow.



Select Employee To Delete

First Name	Last Name	Email	Phone	Department	Updated	Edit
Jill	King	jking@borland.com	212-555-4432	Support	2021-09-17 09:03:52	delete
Margaret	Thomson	mthomson@borland.com	234-456-6548	Sales	2021-08-16 09:03:52	delete
Henry	Bissoon	hbissoon@borland.com	204-323-1145	IT	2021-06-22 09:03:52	delete
Dave	Jones	djones@borland.com	212-555-3456	Sales	2021-01-12 09:03:52	delete



localhost says

Do you really want to delete this item?

Phone	Department	Updated	Edit
212-555-4432	Support	2021-09-17 09:03:52	delete

1. Start at the list view
2. Click the "delete" button
3. Confirm the deletion
4. Redirect back to list view
5. Repeat as necessary

02_edit.php

End of

Intro PHP