APPENDIX D

## Exercises

Even though we've made a pretty solid database, there are a few enhancements we could make if we chose to do so. These potential enhancements are presented now for you, the reader, to undertake. Good luck!

1. **Add a unique constraint to the PhoneNumberTypes table**

   At the moment, this table allows the same phone number type to be added multiple times. I could have four "Home" phone number types, each with a different ID. Try to add a unique constraint to ensure the phone number type must be unique.

2. **Bulk import using staging tables**

   This is a harder exercise. Do you remember our `BULK INSERT` chapter? We created a set of text files to insert into our tables. The problem with this approach was we needed to know the `ContactId` values to successfully import contact phone numbers and other contact-related data. This isn't an ideal approach.

   Try to resolve this issue by modifying the child record import files to use the contact's first and last name instead of the contact's ID. Then import these files to staging tables (tables to which you effectively copy the file contents like for like). Once the data is in the staging tables, write code that creates the contacts. With the contacts created, populate each child table using SELECT statements with INNER JOINs to retrieve the newly generated `ContactId` from `Contacts`. You'll need to use `FirstName` and `LastName` from `Contact` to join to the appropriate child table (e.g., `Staging.ContactAddresses`).

3. **ContactPhoneNumbers clustered primary key**

   The `ContactPhoneNumbers` table uses `ContactPhoneNumberId` as its clustered index. This is a poor choice—ContactId would be better, as we use it regularly in joins. Try to remodel the table so `ContactPhoneNumberId` becomes a nonclustered primary key, and `ContactId` becomes the clustered index.

4. **ContactName function code reduction**

   Your final challenge. Our `ContactName` function has a `SELECT` statement that checks if `FirstName` and/or `LastName` are `NULL`, and if they are, sets them to empty strings via the `COALESCE` statement. These lines could be removed and incorporated into the `CASE` statement. See if you can do this.

PREV Appendix C: Common SQL Serv…

NEXT Index