- Use as simple a key as possible, with values that do not need to be updated.

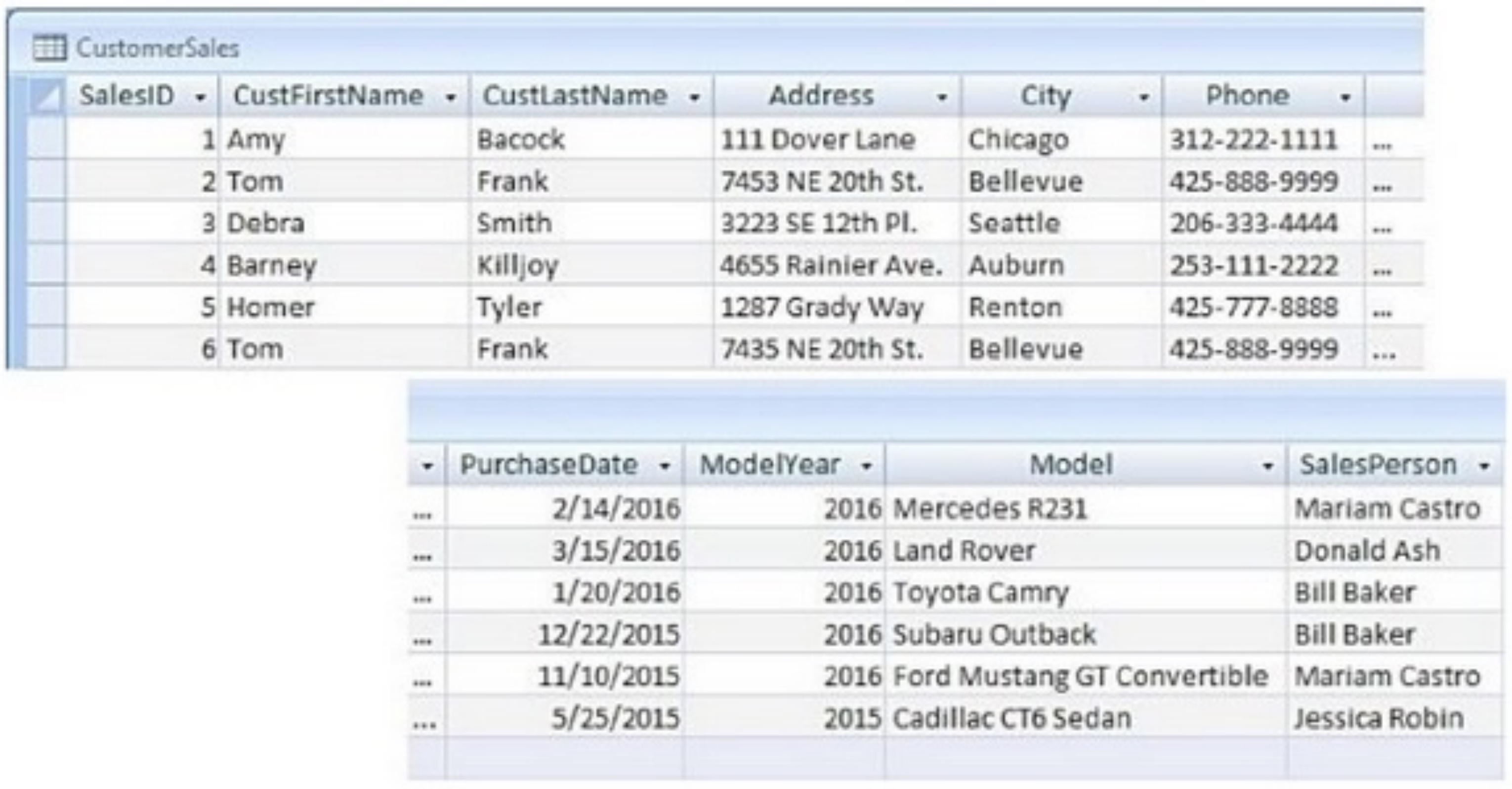
Item 2: Eliminate Redundant Storage of Data Items

Redundant storage of data causes many problems, including inconsistent data; insert, update, and delete anomalies; and wasted disk space. Normalization is a process that involves dividing information by subject to help eliminate problems associated with storing duplicate data. Note that by "redundant" we do not mean the apparent duplication of a primary key value from one table as a foreign key in another table. We are more concerned with cases where users enter the same piece of data in more than one place. Such redundancy is necessary to maintain the relational link between tables.

Although we cannot go into too much depth on the topic of database normalization because of space constraints, it is very important that people working with databases have a thorough understanding of this subject. There are many excellent resources available in books and on the Web that go into greater detail.

One goal of normalization is to minimize the need to repeat data, either in the same table or in different tables throughout a database. A few examples of the redundant storage of data are shown in the Customer Sales database shown Figure 1.3 on the next page.

Sách có bản quyền



CustomerSales

SalesID	CustFirstName	CustLastName	Address	City	Phone	PurchaseDate	ModelYear	Model	SalesPerson
1	Amy	Bacock	111 Dover Lane	Chicago	312-222-1111	2/14/2016	2016	Mercedes R231	Mariam Castro
2	Tom	Frank	7453 NE 20th St.	Bellevue	425-888-9999	3/15/2016	2016	Land Rover	Donald Ash
3	Debra	Smith	3223 SE 12th Pl.	Seattle	206-333-4444	1/20/2016	2016	Toyota Camry	Bill Baker
4	Barney	Killjoy	4655 Rainier Ave.	Auburn	253-111-2222	12/22/2015	2016	Subaru Outback	Bill Baker
5	Homer	Tyler	1287 Grady Way	Renton	425-777-8888	11/10/2015	2016	Ford Mustang GT Convertible	Mariam Castro
6	Tom	Frank	7435 NE 20th St.	Bellevue	425-888-9999	5/25/2015	2015	Cadillac CT6 Sedan	Jessica Robin

Figure 1.3 Example of redundant storage of data in a single table

An example of inconsistent data is the address for customer Tom Frank. In the second record, the numeric portion of his address is 7453, whereas in the sixth record, the numeric portion is 7435. Similar inconsistencies in data could be present in any of the columns.

An insertion anomaly is present because you cannot enter information for a given model of automobile until you have a sale that is entered with a customer record. Also, the design requires repeating most data when a customer purchases additional cars. This represents unnecessary data entry that is wasteful of disk space, memory, network resources, and even the time spent by a data entry clerk. In addition, repeating data entry greatly increases the risk of data entry errors, such as transposing numbers in an address as shown in the example in Figure 1.3.

An update anomaly exists because if, for example, a salesperson gets married and changes his or her name, you would need to run an update query to update all occurrences of the person's name. This can present real challenges if you are dealing with a large number of records in a database that many people use concurrently. In addition, such an update will be successful only if all occurrences of the person's name are spelled exactly the same (meaning no

Sách có bản quyền

inconsistent data) and if more than one person does not share the name.

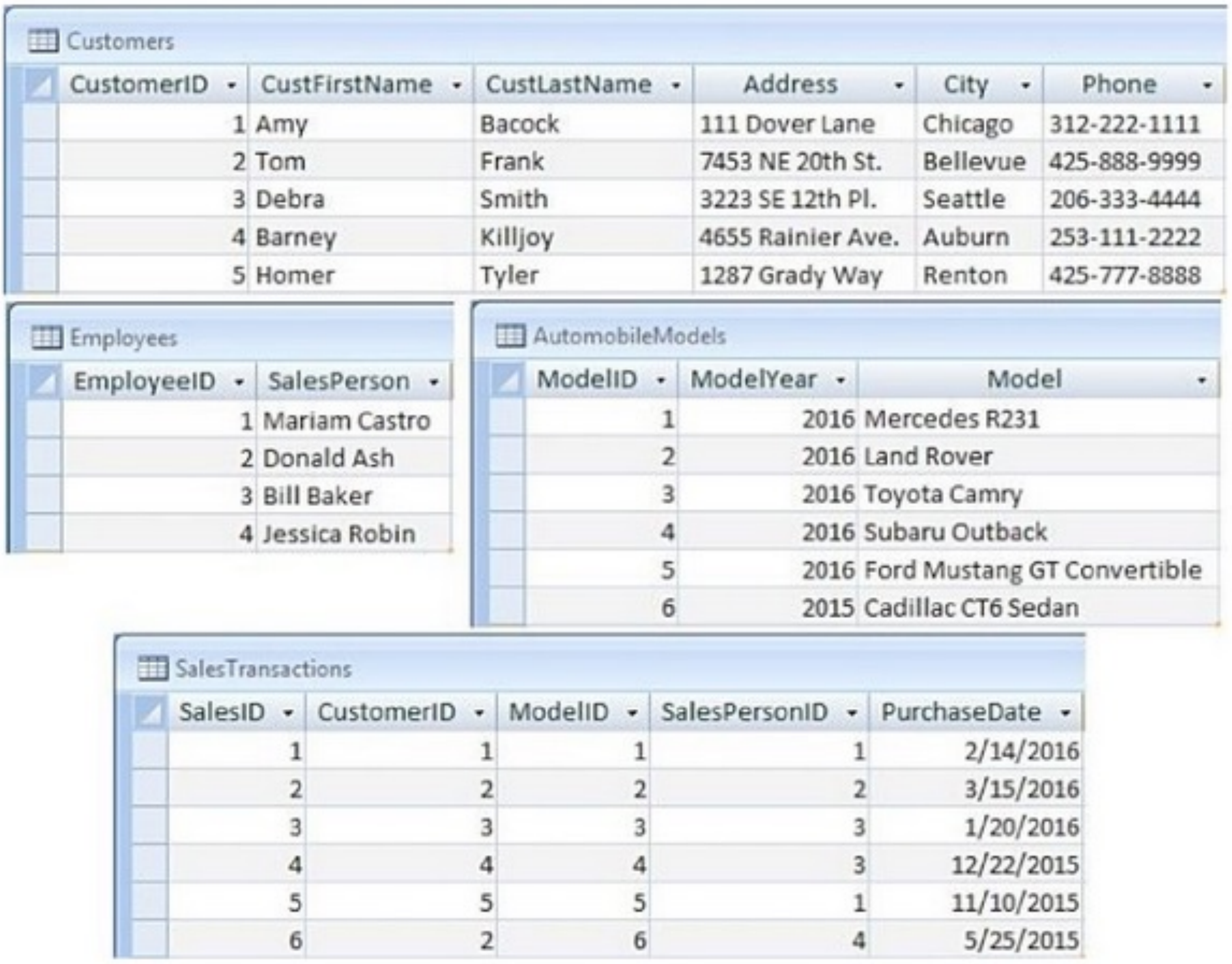
A deletion anomaly exists because if a row is deleted, you may lose data you did not intend to remove from your database.

The customer sales data shown in Figure 1.3 can logically be divided into four tables:

1. `Customers` table (name, address, etc.)
2. `Employees` table (salesperson name, hire date, etc.)
3. `AutomobileModels` table (model year, model, etc.)
4. `SalesTransactions` table

This design allows you to enter customer, employee, and automobile model information once into the respective tables. All tables include a unique identifier that can be set as a primary key. The `SalesTransactions` table uses foreign keys to store the details of each sales transaction. See Figure 1.4.

Sách có bản quyền



Customers

CustomerID	CustFirstName	CustLastName	Address	City	Phone
1	Amy	Bacock	111 Dover Lane	Chicago	312-222-1111
2	Tom	Frank	7453 NE 20th St.	Bellevue	425-888-9999
3	Debra	Smith	3223 SE 12th Pl.	Seattle	206-333-4444
4	Barney	Killjoy	4655 Rainier Ave.	Auburn	253-111-2222
5	Homer	Tyler	1287 Grady Way	Renton	425-777-8888

Employees

EmployeeID	SalesPerson
1	Mariam Castro
2	Donald Ash
3	Bill Baker
4	Jessica Robin

AutomobileModels

ModelID	ModelYear	Model
1	2016	Mercedes R231
2	2016	Land Rover
3	2016	Toyota Camry
4	2016	Subaru Outback
5	2016	Ford Mustang GT Convertible
6	2015	Cadillac CT6 Sedan

SalesTransactions

SalesID	CustomerID	ModelID	SalesPersonID	PurchaseDate
1	1	1	1	2/14/2016
2	2	2	2	3/15/2016
3	3	3	3	1/20/2016
4	4	4	3	12/22/2015
5	5	5	1	11/10/2015
6	2	6	4	5/25/2015

Figure 1.4 Example of splitting data into tables by subject

The astute reader may have noticed that one duplicate customer record was eliminated in this process as a result of determining the correct address for customer Tom Frank.

We can create relationships (sometimes referred to as foreign key constraints) by joining the primary key from the three parent tables (`Customers`, `AutomobileModels`, and `Employees`) to the foreign key columns in the `SalesTransactions` child table, as shown in Figure 1.5 on the next page. We created the example shown in the figure using the relationships editor in Microsoft Access. Each relational database has a different way of representing relationships between tables.

Sách có bản quyền

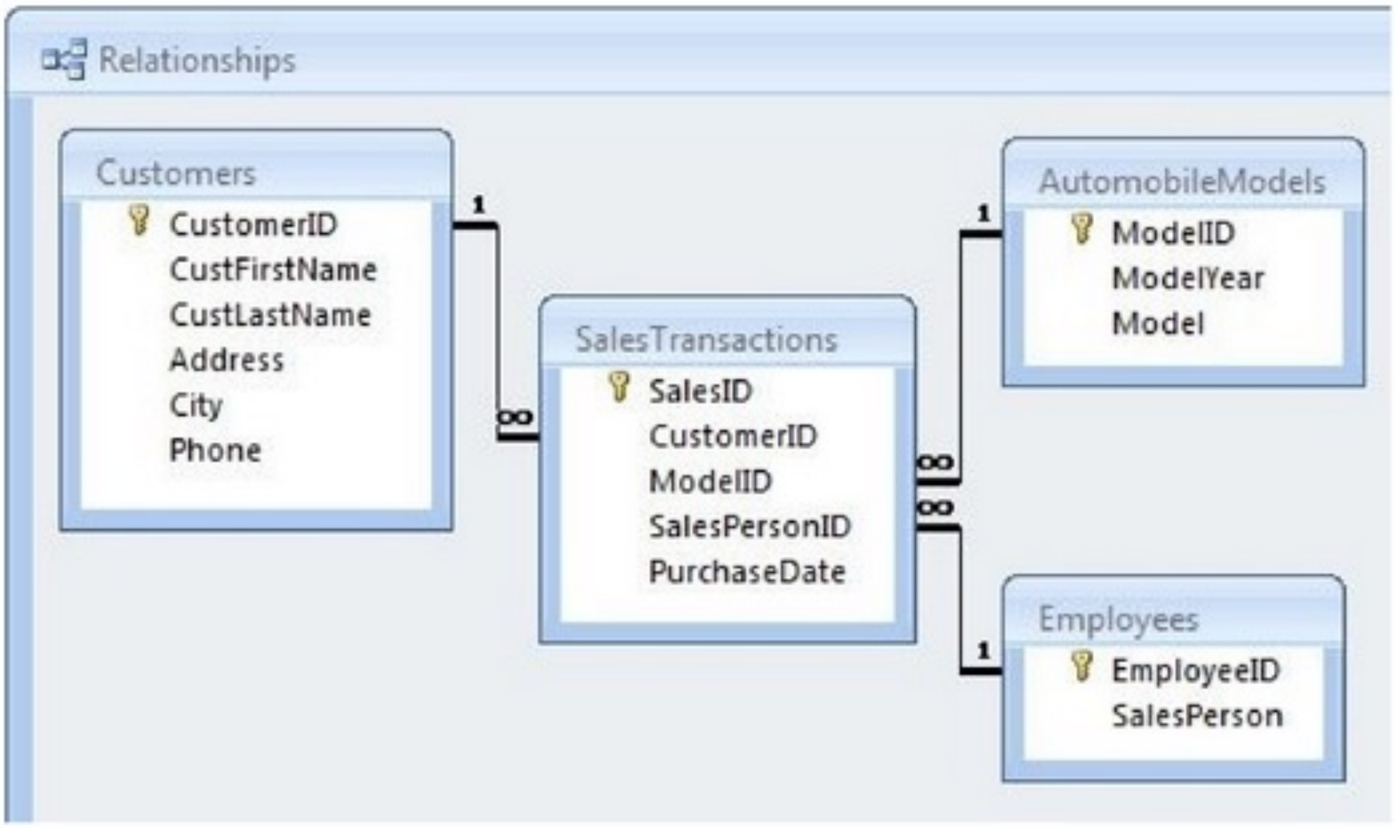


Figure 1.5 Four tables that have been related using primary keys joined to foreign key columns

You can easily re-create the original data, shown earlier in Figure 1.3, by constructing a virtual table (query) as shown in Listing 1.1 on the next page without the penalties imposed by storing redundant data. (Construction of the virtual table is a perfect use for a CTE, or common table expression, as discussed in Item 42, "If possible, use common table expressions instead of subqueries.")

Listing 1.1 An SQL statement that returns the original data

Click here to view code image

```
SELECT st.SalesID, c.CustFirstName, c.CustLastName,
c.Address,
  c.City, c.Phone, st.PurchaseDate, m.ModelYear, m.Model,
  e.SalesPerson
FROM SalesTransactions st
  INNER JOIN Customers c
    ON c.CustomerID = st.CustomerID
  INNER JOIN Employees e
    ON e.EmployeeID = st.SalesPersonID
  INNER JOIN AutomobileModels m
```

Sách có bản quyền

```
    ON m.ModelID = st.ModelID;
```

Things to Remember

- A goal of database normalization is the elimination of redundant data and minimizing resource use when processing data.
- By eliminating redundant data, you eliminate insert, update, and delete anomalies.
- By eliminating redundant data, you minimize the occurrence of inconsistent data.

References

If you want to explore the correct ways to design a relational database, here are a couple of books we recommend; the first one is accessible to beginners and a good place for the novice to start:

- Hernandez, Michael J. *Database Design for Mere Mortals* (Addison-Wesley, 2013). ISBN-10: 0-321-88449-3.
- Fleming, Candace C., and Barbara von Halle. *Handbook of Relational Database Design* (Addison-Wesley, 1989). ISBN-10: 0-201-11434-8.

Item 3: Get Rid of Repeating Groups

It is common to see spreadsheets that include repeating groups of similar data. Often information workers simply import this data into a new database without any consideration of database normalization. An example of repeating groups of data is shown in Figure 1.6, with a `DrawingNumber` being associated with up to five `Predecessor`s. The table has a one-to-many relationship between drawing numbers and predecessor values.

Sách có bản quyền