12. MS Access Tables, Relationships, and Queries

12.1 Creating Tables and Relationships

Suppose we want to build a database to hold the information for computers (also refer to parts in the text) and suppliers who supply the computers. The schemas for these two entity types are as follows:

(1) tblSupplier(S_No, Sname, Status, City)
(2) tblComputer(P_No, Pname, Color, Weight, City)

Figure 1 is a table design window, in which you can follow the information to design the supplier and computer tables.

Apparently, the relationship between Supplier and Computer is many-to-many. Therefore, we need to create another table to represent this relationship (Figure 2).

(3) tblSupplierComputer(S_No, P_No, Qty)

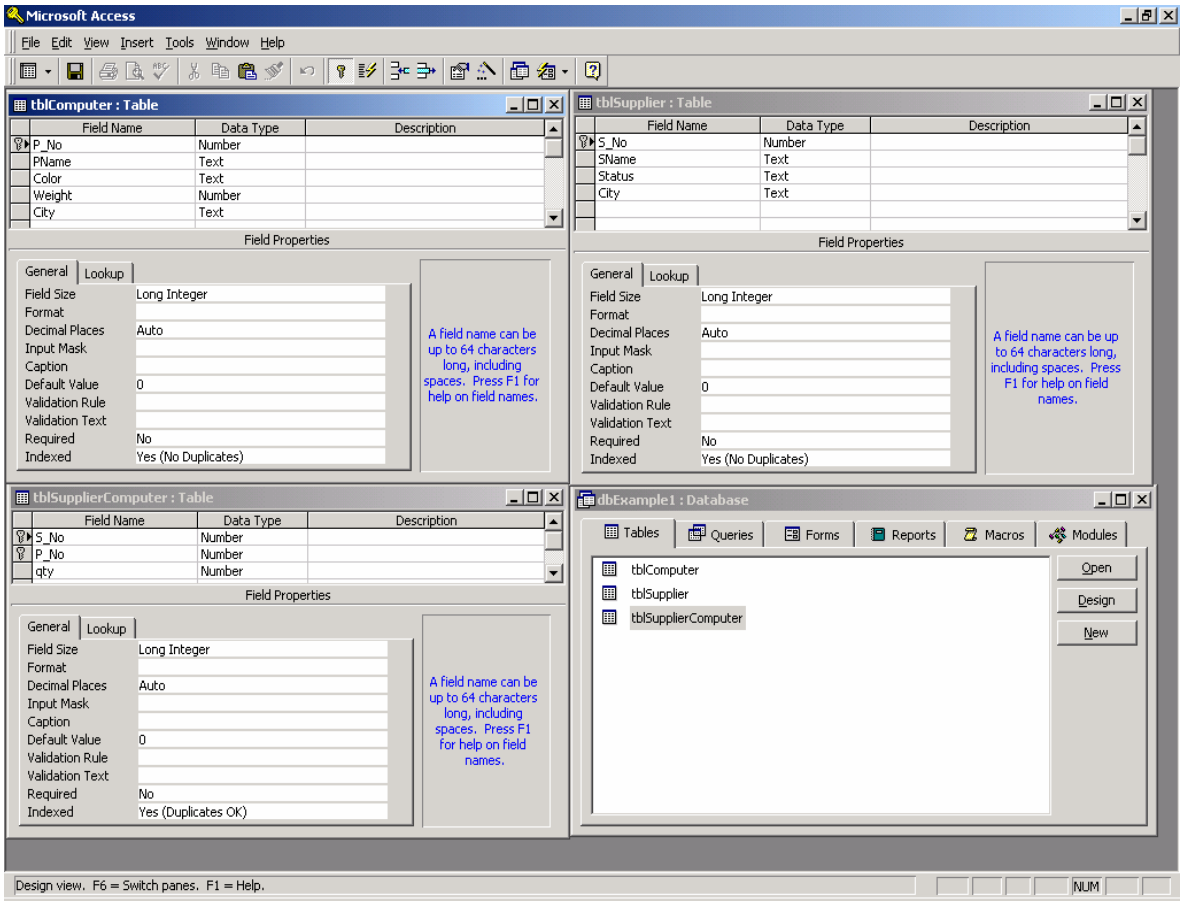

Figure 1. Schemas of tables.

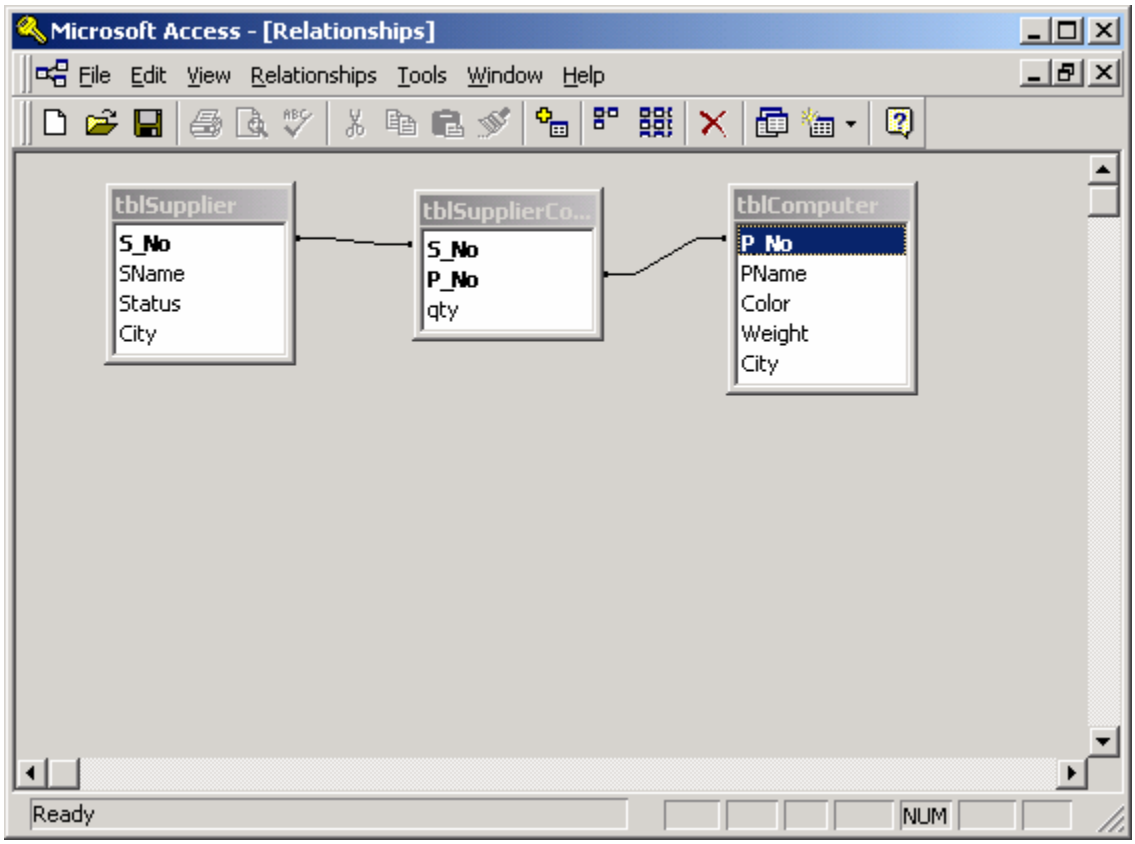

Figure 2. Building n-n relationship.

Once the relationship is built, enter some data into the tables as shown in Figure 3.

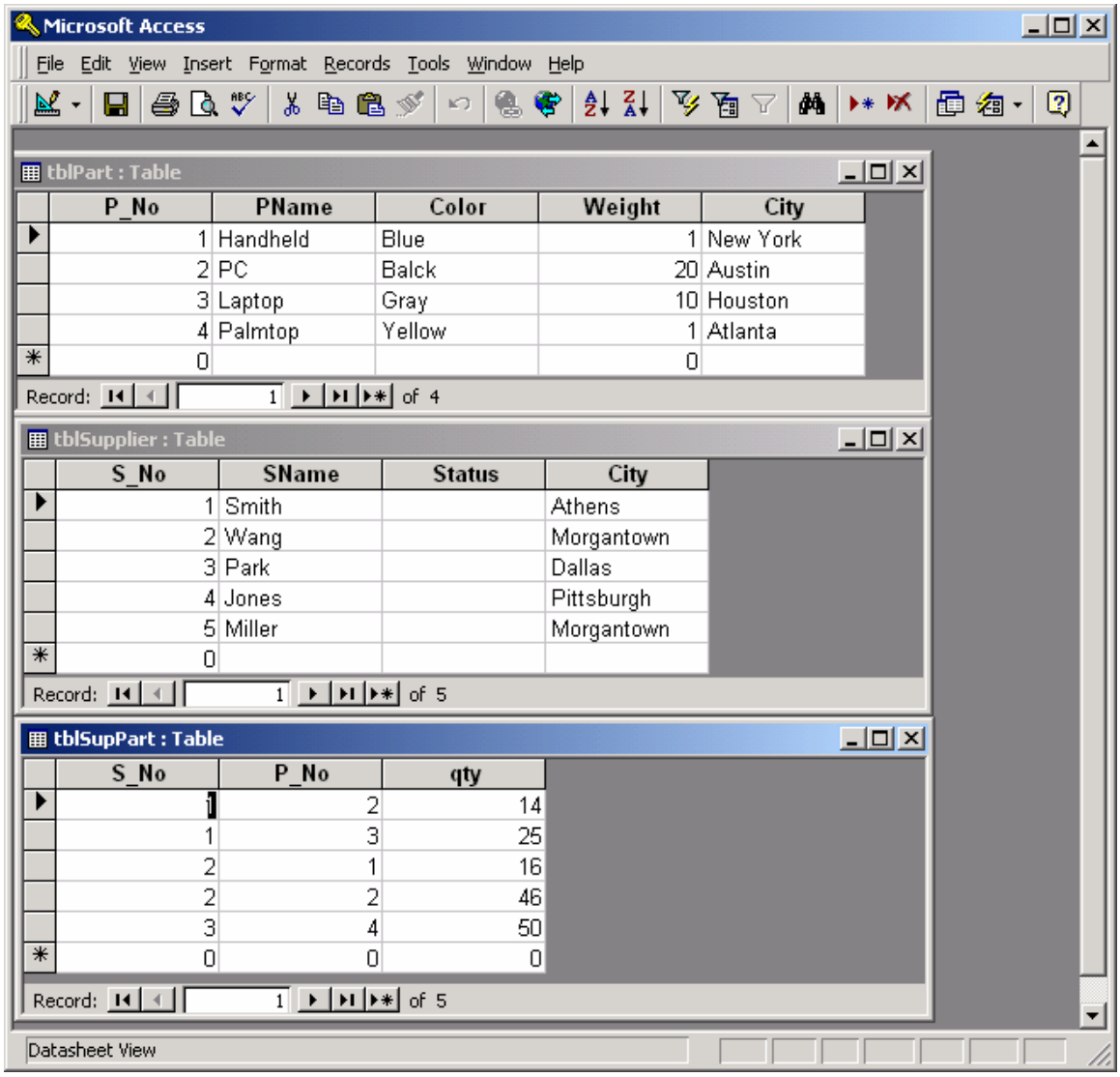

Figure 3. Data entries in tables.

12.2 Querying Tables

The basic syntax of the query SQL commands is:

Select attributes | expressions [[as] alias], …
From table(s)
[Where Boolean conditions];

(1) List supplier number, part number, quantity supplied and quantity supplied if increased by 15%.

```
Select s_no, p_no, qty, "increased by 15%", qty*1.15 as NewQuantity
From tblSupplierComputer;
```

(2) List all records in table Computer.

```
Select * from tblComputer;
```

(3) List supplier numbers of suppliers who supply at least one computer in quantity between 10 and 30 and remove duplicate suppliers.

```
Select distinct s_no
From tblSupplierComputer
Where qty between 10 and 30;
```

Cartesian Product

The Cartesian product of two or more tables is the logical table created by juxtaposing all records from participating tables. We usually use the Cartesian product to resolve query involving multiple tables.

(4) List supplier name, part number, and supplied quantity increased by 10%; sort the result in ascending order of supplier name and the increased quantity in descending order.

```
Select Sname, P_No, qty*1.1 as NewQty
From tblSupplier, tblSupplierComputer
Where tblSupplier.S_No = tblSupplierComputer.S_No
Order by Sname asc, NewQty desc;
```

Subqueries or Nested Queries

A SQL query returns a set of records; thus, we should be able to apply set operations to test, for example, membership in a set. Our test is to decide if an attribute value is or is not related to a set of values.

The syntax is:

….

Where {attribute values} comparators =, <>, >,< {any or all (set of values)}

(5) Find supplier names of suppliers who supply computer or part #3.

```
Select sname
From tblSupplier
Where S_No = any
                (select s_no
                    from tblSupplierComputer
                    where P_No=3);
```

Notes: a. The nested select must be enclosed in parentheses ();
b. The comparison '= any' may be replaced by "in";

(6) List computer or part names of computers that are not currently supplied.

```
Select Pname
From tblComputer
Where P_No <> all (select P_No from tblSupplierComputer);
```

Notes: (1) The comparator '<> all' may be replaced w/ 'not in';

(7) List supplier names of suppliers who are located in the same city as supplier #2.

```
Select Sname
From tblSupplier
Where city in
        (select city
                from tblSupplier
                where S_No=2);
```

(8) A literal set is denoted by a list of values and separated by commas. List supplier numbers of suppliers who supply at least one type of computer 2, 3, 4.

```
Select distinct S_No
From tblSupplierComputer
Where P_No in (2,3,4);
```

Table Aliases

We may define an alias or variable dynamically for a table by simply following the table name in the From clause by alias that is separated by a white space.

(9) List supplier numbers of suppliers who supply at least 2 computers or parts.

```
Select distinct x.S_No
From tblSupplierComputer x, tblSupplierComputer y
Where x.S_No=y.S_No and x.P_No <> y.P_No;
```

(10) List supplier names of suppliers who currently supply no computers or parts.

```
Select Sname
From tblSupplier
Where not exists
(select * from tblSupplierComputer where S_No = tblSupplier.S_No);
```

(11) Find supplier number of suppliers who supply the maximum quantity of computers.

```
Select S_No
From tblSupplierComputer
Where qty >= all
            (select qty from tblSupplierComputer);
```

Built-in Statistical Functions

MS Access and other DBMS such as Oracle supports Count, Max, Min, Sum, Avg, StDev, and Var statistical functions.

(11a) For query 11, we rewrite as:

```
SELECT S_No
FROM tblSupplierComputer
WHERE qty >= (select max(qty) from tblSupplierComputer);
```

(12) Display the total number of suppliers.

```
Select count(*) as NofSupplier from tblSupplier;
```

(13) List supplier number of suppliers who supply at least one part.

```
Select distinct S_No from tblSupplierComputer;
```

(14) Get the total quantity of computer #2 (part #2) that is supplied.

```
Select sum(qty)
From tblSupplierComputer
Where P_No=2;
```

Similarly, we can get the average quantity of part #2 by simply replacing Sum With Avg.

(15) List supplier names of suppliers who supply at least two different computers.

```
Select Sname
From tblSupplier
Where 2<= (select count(S_No) from tblSupplierComputer where
S_No=tblSupplier.S_No);
```

Group By Option

We wish logically to group records together in order to apply an aggregate statistical function to each group. The enhanced syntax is:

```
Select attributes (that are unique to the group)
From …
[where Boolean conditions]
group by (attributes)
[order by …];
```

(16) List each supplier number with average quantity supplied.

```
Select S_No, Avg(qty)
From tblSupplierComputer
Where qty is not NULL
Group by S_No
Order by S_No;
```

Having Clause

The having clause complements the "Group By" by providing a Boolean condition to decide if a group should or should not be displayed. The syntax is:

```
Select attributes (that are unique to the group)
From …
[where Boolean conditions]
group by (attributes)
[having …]
[order by …];
```

(17) List supplier names of supplier who supply exactly two different computers.

```
Select Sname
From tblSupplier
Where S_No in
(select S_No from tblSupplierComputer
group by S_No
having count(*)=2);
```