COP 4710: Database Systems
Spring 2004

-Day 17 – March 3, 2004 –
Introduction to SQL

Instructor : Mark Llewellyn
markl@cs.ucf.edu
CC1 211, 823-2790
http://www.cs.ucf.edu/courses/cop4710/spr2004

School of Electrical Engineering and Computer Science
University of Central Florida

History of SQL

- SQL, pronounced "S-Q-L" by some and "sequel" by others (mostly old-timers), has become the de facto standard language for creating and querying relational databases.
- It has been accepted by ANSI (American National Standards Institute) and ISO (International Standards Organization) as well as being a FIPS (Federal Information Processing Standard).
- Between 1974 and 1979, workers at the IBM Research Laboratory in San Jose, California undertook the development of System R. This was shortly after Codd's classic paper defining the relational database was published. The goal of the System R project was to demonstrate the feasibility of implementing the relational model in a DBMS. They used a language named SEQUEL (Structured English QUEry Language), which was a descendent of SQUARE (Specifying QUeries As Relational Expressions), both of which were developed at IBM, San Jose.
- SEQUEL was renamed to SQL during this project.

History of SQL (cont.)

- System R itself was never produced commercially, but directly led to the development of SQL/DS (1981 running under DOS/VE OS, a VM version followed in 1982) which was IBM's first commercial relational DBMS.
- IBM however, did not produce the first commercial implementation of a relational DBMS. That honor went to Oracle (Relational Software) in 1979.
- Today, the relational DBMS system of virtually all vendors is based on SQL.
- Each vendor provides all the standard features of SQL. Most vendors also provide additional features of their own, called extensions to standard SQL. These extensions lead to portability issues when moving SQL-based applications across various RDBMS. Vendors attempt to distinguish their SQL versions through these extensions.

History of SQL (cont.)

- The current version of ANSI standard for SQL is SQL-99 (also referred to as SQL3). This standard has also been accepted by ISO.
- Although many different extensions of SQL exist, we'll look at the core SQL that will be found on any RDBMS that you will encounter. Whether you use Oracle, Microsoft SQL Server, IBM's DB2, Microsoft Access, MySQL, or any other well-established RDBMS, you'll be able to get up to speed on that system with the information in this set of notes.

SQL

- SQL is a complete relational database language in the sense that it contains both a data definition language (DDL) and a data manipulation language (DML).
- We'll examine components of both parts of SQL.
- If you use Microsoft Access, for example, you'll need to know less about the DDL side of SQL than you will if you use Oracle 9i or MySQL.
- The table on the following pages summarize the commands in the DDL portion of SQL. The entries in the table do not correspond to the order in which you will use the commands, but simply give a quick summary of those available. The table does not contain a complete listing of the commands in the DDL portion of SQL.

Summary of SQL DDL Commands

Command or Option	Description
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's DB schema
NOT NULL	Constraint that ensures a column will not have null values
UNIQUE	Constraint that ensures a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Constraint used to validate data in a column
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows/columns from 1 or more tables
ALTER TABLE	Modifies a table's definition: adds/deletes/updates attributes or constraints
DROP TABLE	Permanently deletes a table (and thus its data) from the DB schema
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

The DDL Component Of SQL

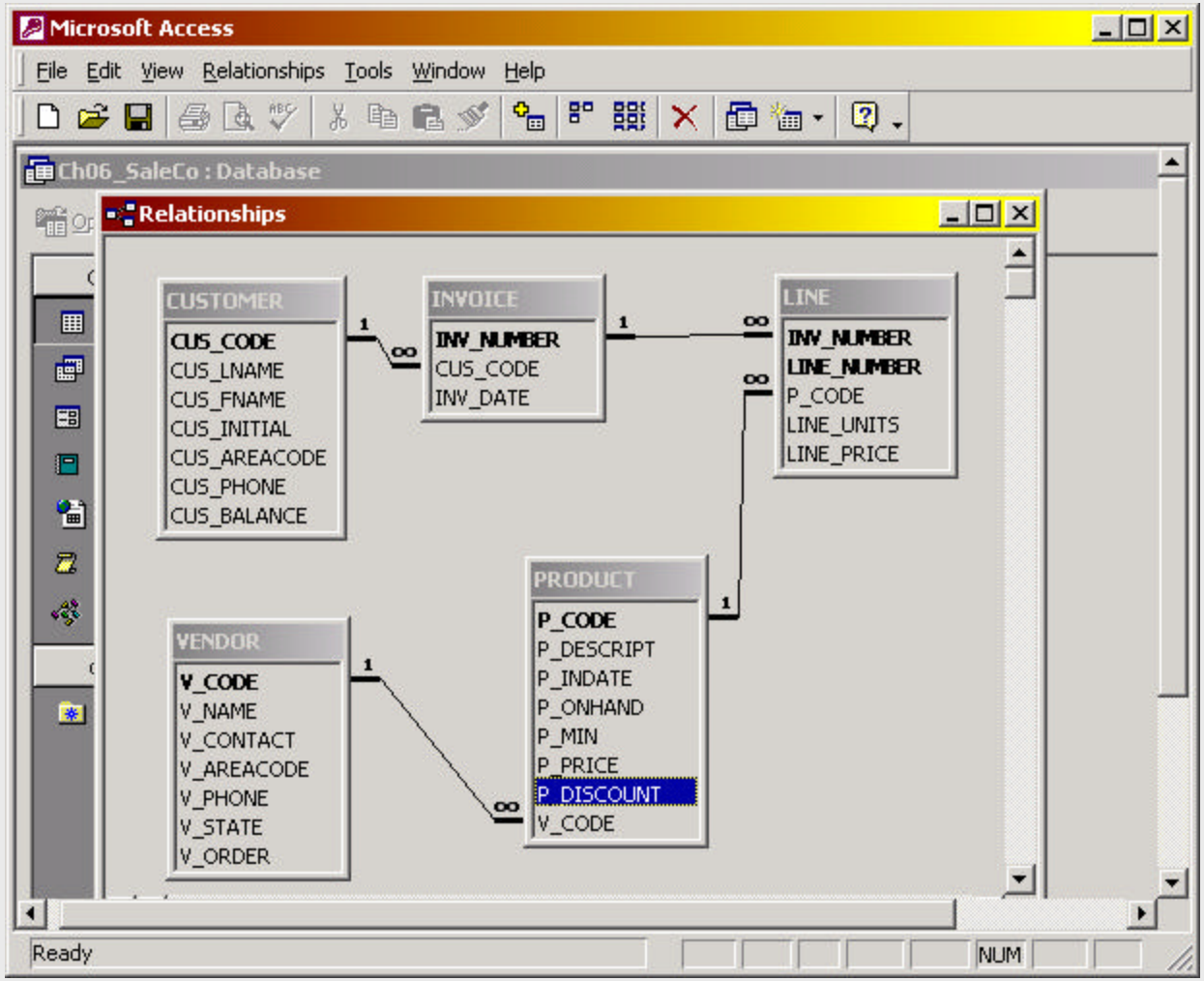
- Before you can use a RDMS two tasks must be completed: (1) create the database structure, and (2) create the tables that will hold the end-user data.
- Completion of the first task involves the construction of the physical files that hold the database. The RDBMS will automatically create the data dictionary tables and create a default database administrator (DBA).
  - Creating the physical files requires interaction between the host OS and the RDBMS. Therefore, creating the database structure is the one feature that tends to differ substantially from one RDBMS to another.
- With the exception of the creation of the database, most RDBMS vendors use SQL that deviates very little from ANSI standard SQL. Nevertheless, you might occasionally encounter minor syntactic differences. For example, most RDBMSs require that any SQL command be ended with a semicolon. However, some SQL implementations do not use a semicolon. I'll try to point out most of the common syntactic differences, or at least the ones of which I am aware.

Use Of DDL Commands In SQL

- We'll use the database shown on the next page for illustrating the DDL commands of SQL. This database is a bit more involved than our supplier-parts-jobs-shipments database, but its along the same lines. The business rules that apply to this database are:

1. A customer may generate many invoices. Each invoice is generated by one customer.
2. An invoice contains one or more invoice lines. Each invoice line is associated with one invoice.
3. Each invoice line references one product. A product may be found in many invoice lines. You can sell more than one hammer to more than one customer.
4. A vendor may supply many products. Some vendors may not supply any products,
5. If a product is vendor-supplied, that product is supplied by only one vendor.
6. Some products are not supplied by a vendor, they may be made "in-house" or obtained through other means.

An Example Database



SQL Syntax Notation

Notation	Description
CAPITALS	Required SQL command keyword
italics	An end-user provided parameter – normally required
{a | b | ... }	A mandatory parameter, use one from option list
[...]	An optional parameter – everything in brackets is optional
tablename	The name of a table
column	The name of an attribute in a table
data type	A valid data type definition
constraint	A valid constraint definition
condition	A valid conditional expression – evaluates to true or false
columnlist	One or more column names or expressions separated by commas
tablelist	One or more table names separated by commas
conditionlist	One or more conditional expressions separated by logical operators
expression	A simple value (e.g., 76 or 'married') or a formula (e.g., price-10)

Creating Table Structures Using SQL

- The CREATE TABLE syntax is:

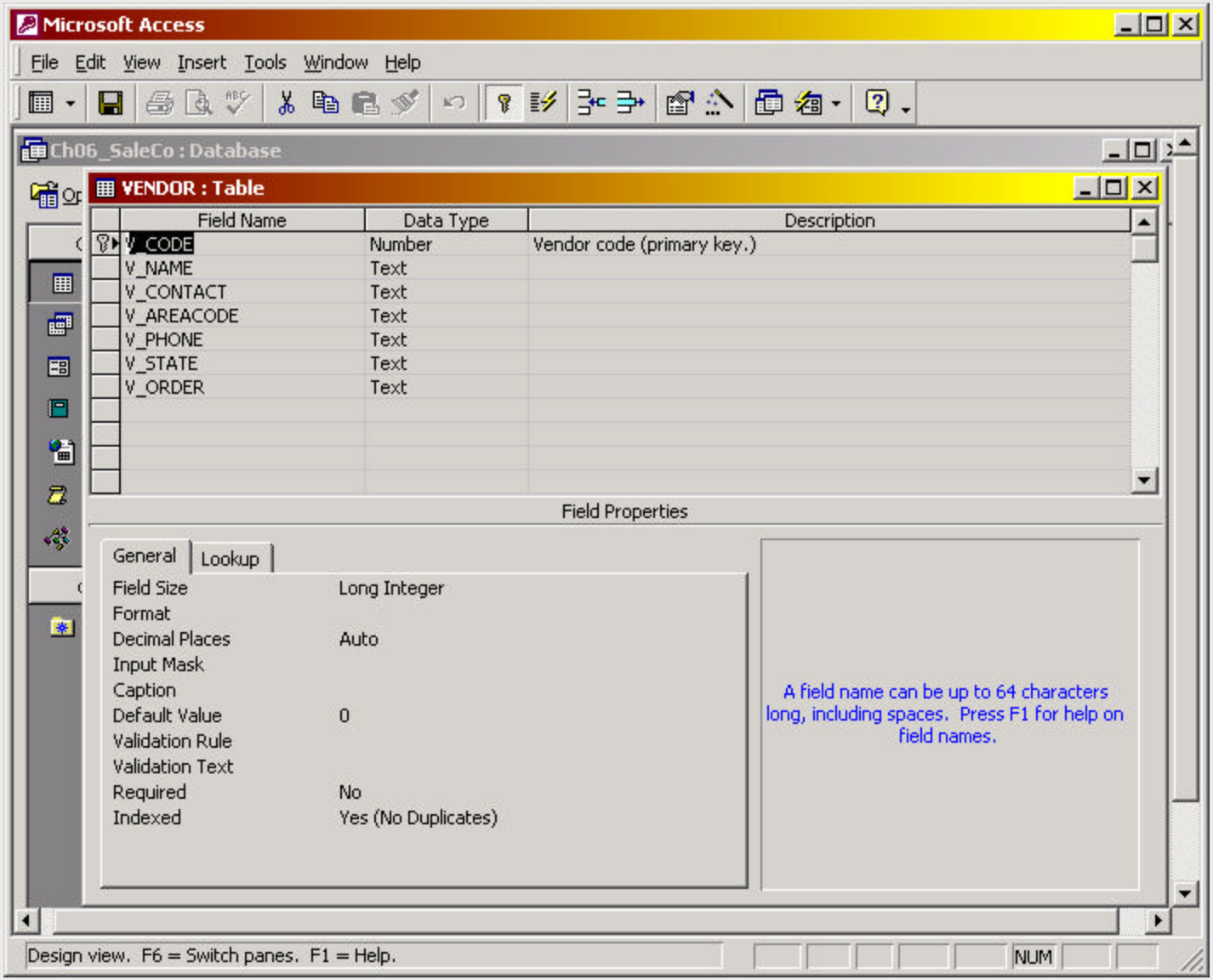
```
CREATE TABLE tablename (
    column1     data type     [constraint] [,
    column2     data type     [constraint] ]  [,
    PRIMARY KEY (column1 [,column2] )]  [,
    FOREIGN KEY (column1 [,column2] ) REFERENCES tablename ] [,
    CONSTRAINT constraint ] ) ;
```


Example – Table Creation

- As an example, let's create the VENDOR table as described on page 11.

```
CREATE TABLE VENDOR (
   V_CODE          INTEGER        NOT NULL      UNIQUE,
   V_NAME          VARCHAR(35)    NOT NULL,
   V_CONTACT       VARCHAR(15)    NOT NULL,
   V_AREACODE      CHAR(3)        NOT NULL,
   V_PHONE         CHAR(8)        NOT NULL,
   V_STATE         CHAR(2)        NOT NULL,
   V_ORDER         CHAR(1)        NOT NULL,
   PRIMARY KEY ( V_CODE));
```


The VENDOR Table in Access

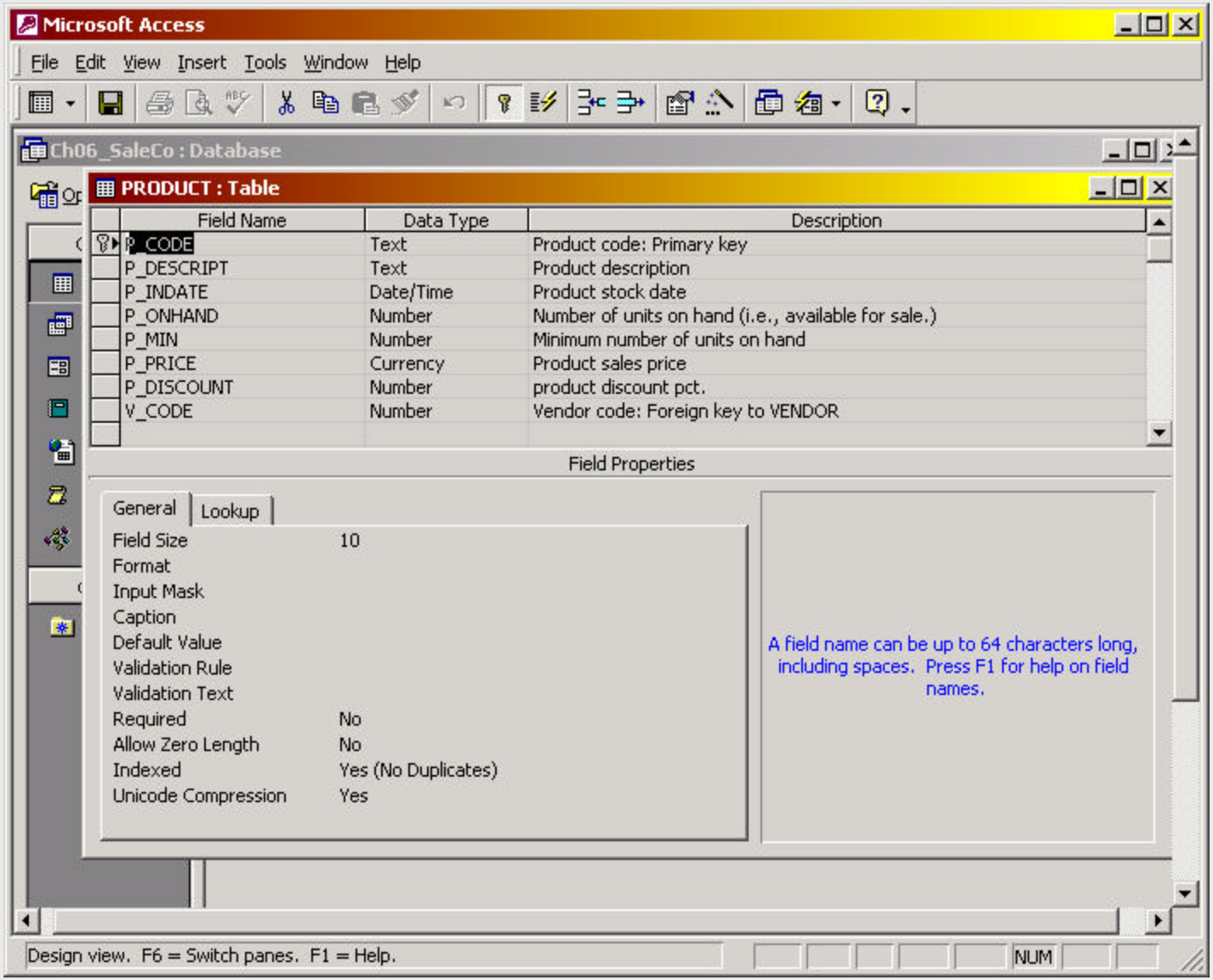


Example – Table Creation

- Now let's create the PRODUCT table as described on page 11.

```
CREATE TABLE PRODUCT (
    P_CODE          VARCHAR(10)     NOT NULL        UNIQUE,
    P_DESCRIPT      VARCHAR(35)     NOT NULL,
    P_INDATE        DATE            NOT NULL,
    P_ONHAND        SMALLINT        NOT NULL,
    P_MIN           SMALLINT        NOT NULL,
    P_PRICE         NUMBER(8,2)     NOT NULL,
    P_DISCOUNT      NUMBER(4,2)     NOT NULL,
    V_CODE          INTEGER,
    PRIMARY KEY ( P_CODE),
    FOREIGN KEY (V_CODE) REFERENCES VENDOR  ON UPDATE CASCADE);
```


The PRODUCT Table in Access



Example – Table Creation

- Now let's create the CUSTOMER table as described on page 11.

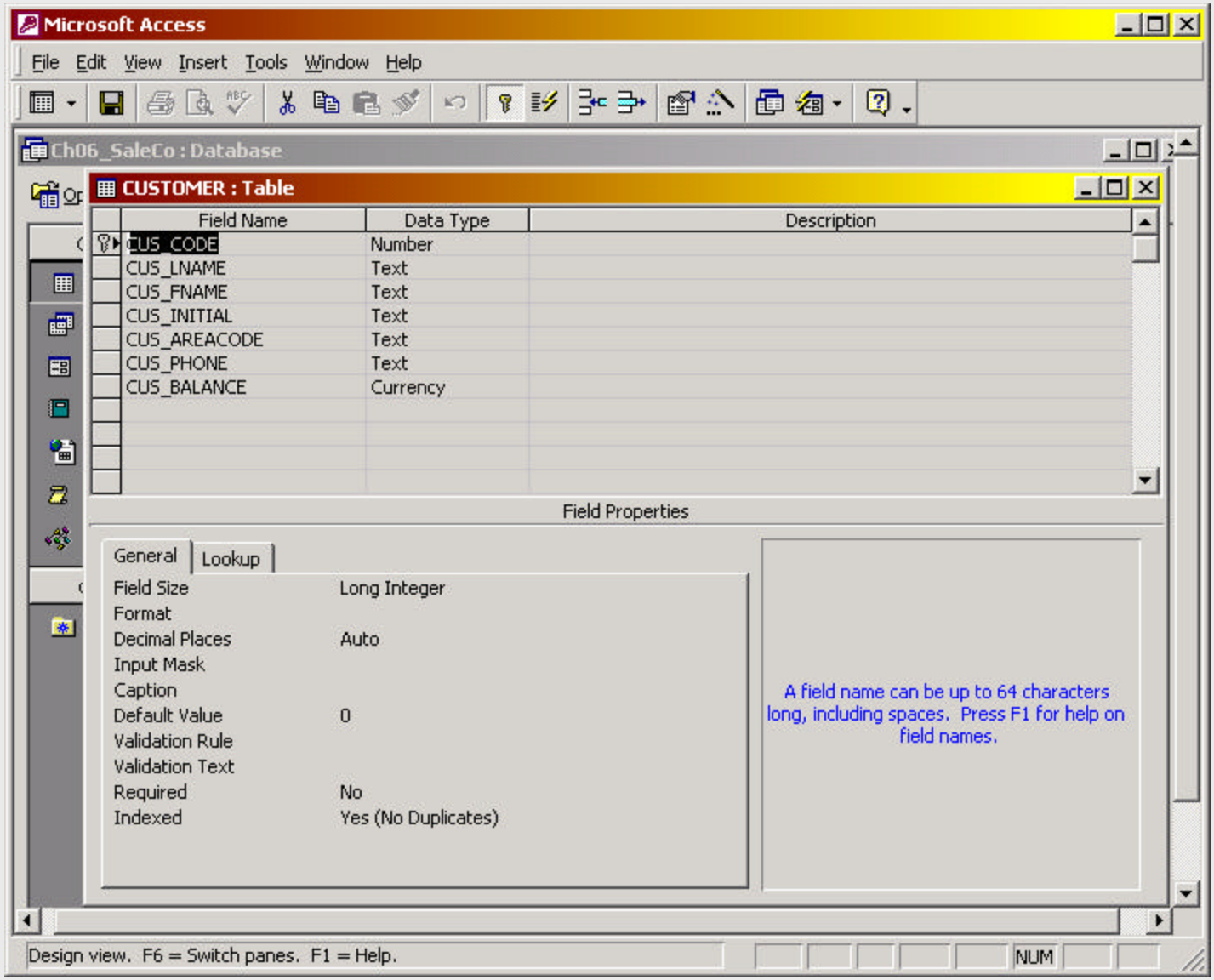
```
CREATE TABLE CUSTOMER (
   CUS_CODE          NUMBER          PRIMARY KEY,
   CUS_LNAME         VARCHAR(15)     NOT NULL,
   CUS_FNAME         VARCHAR(15)     NOT NULL,
   CUS_INITIAL       CHAR(1),
   CUS_AREACODE   CHAR(3)            DEFAULT '615' NOT NULL
                                     CHECK (CUS_AREACODE IN ('615', '713', '931')),
   CUS_PHONE         CHAR(8)         NOT NULL,
   CUS_BALANCE       NUMBER(9,2)     DEFAULT 0.00,
   CONSTRAINT  CUS_UI1  UNIQUE (CUS_LNAME, CUS_FNAME));
```

Column constraint

Table constraint

Creates a unique index constraint named CUS_UI1 on the customer's last name and first name.

The CUSTOMER Table in Access



Example – Table Creation

- Now let's create the INVOICE table as described on page 11.

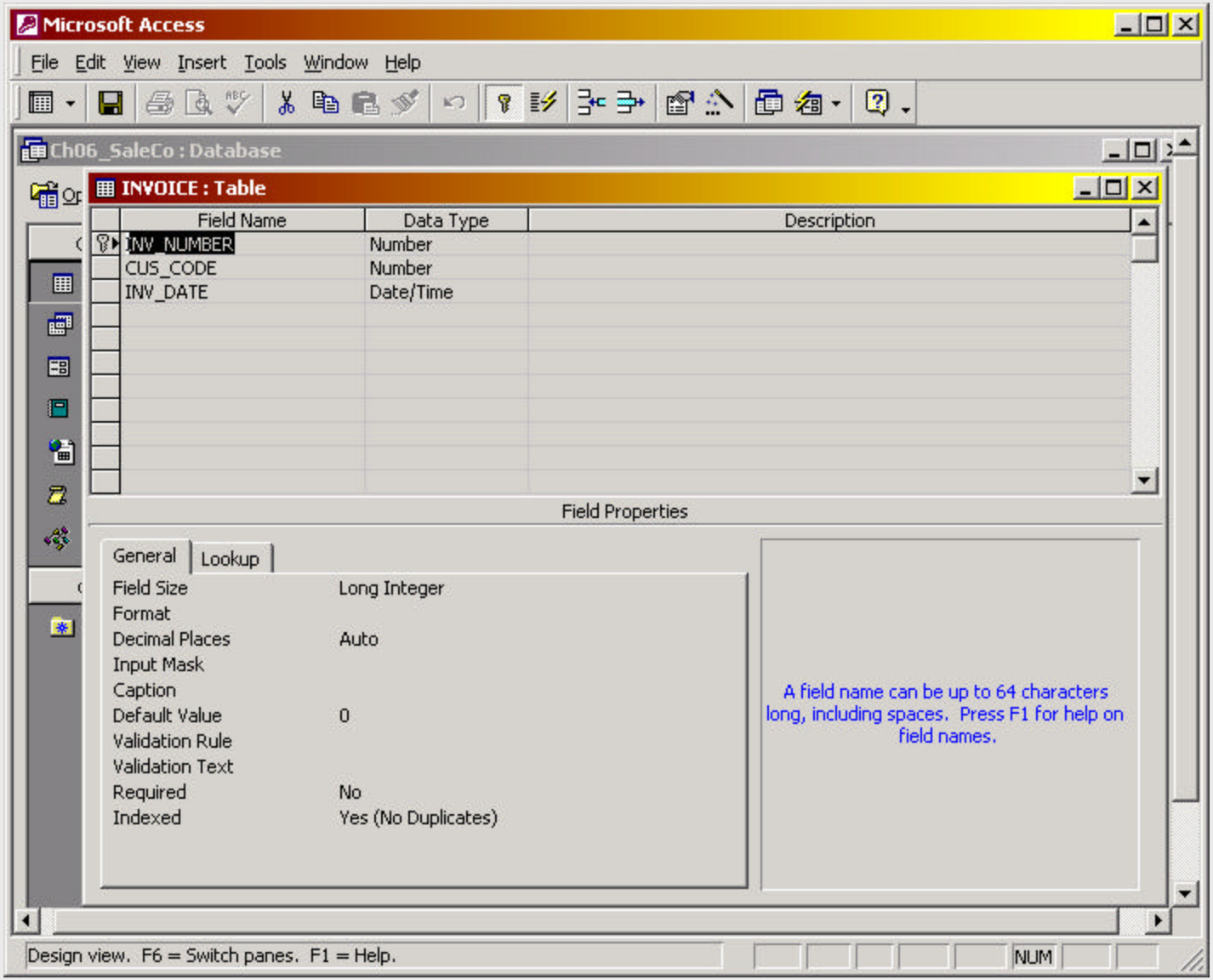
```
CREATE TABLE INVOICE (
   INV_NUMBER      NUMBER          PRIMARY KEY,
   CUS_CODE        NUMBER          NOT NULL, REFERENCES CUSTOMER(CUS_CODE)
   INV_DATE        DATE            DEFAULT SYSDATE NOT NULL,
CONSTRAINT  INV_CK1  CHECK (INV_DATE > TO_DATE('01-JAN-2002', 'DD-MON-YYYY')));
```

Alternative way to define a foreign key

Special function that returns today's date

Check constraint is used to validate that the invoice date is greater than January 1, 2002. The TO_DATE function requires two parameters, the literal date and the date format used.

The INVOICE Table in Access



Example – Table Creation

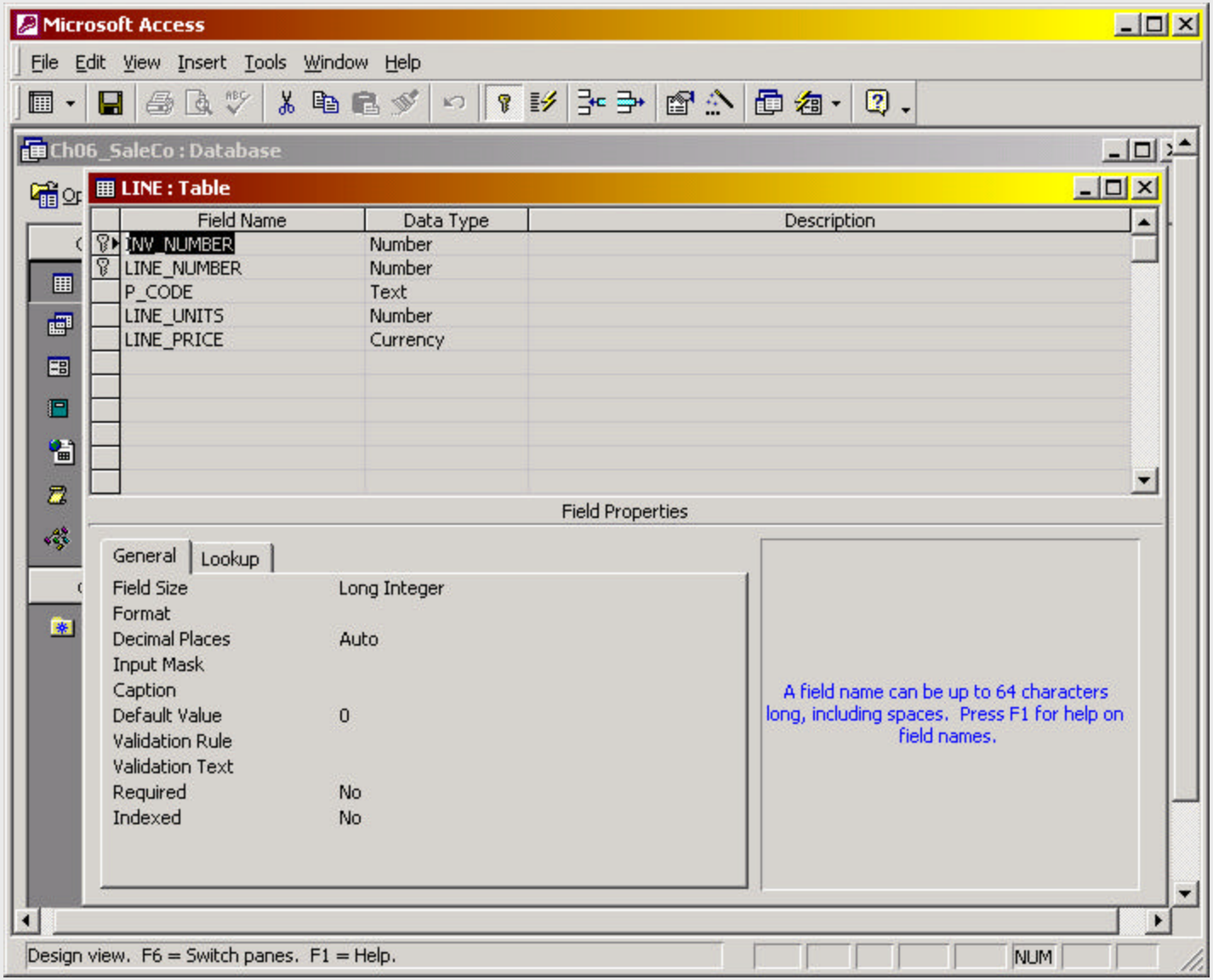
- As a final example of table creation, let's create the LINE table as described on page 11.

```
CREATE TABLE LINE (
   INV_NUMBER      NUMBER          NOT NULL,
   LINE_NUMBER     NUMBER(2,0)     NOT NULL,
   P_CODE          VARCHAR(10)     NOT NULL,
   LINE_UNITS      NUMBER(9,2)     DEFAULT 0.00  NOT NULL,
   LINE_PRICE      NUMBER(9,2)     DEFAULT 0.00  NOT NULL,
   PRIMARY KEY (INV_NUMBER, LINE_NUMBER),
   FOREIGN KEY (INV_NUMBER)  REFERENCES INVOICE     ON DELETE CASCADE
   FOREIGN KEY (P_CODE)  REFERENCES PRODUCT(P_CODE),
  CONSTRAINT  LINE_UI1  UNIQUE(INV_NUMBER, P_CODE));
```

The use of ON DELETE CASCADE is recommended for weak entities to ensure that the deletion of a row in the strong entity automatically triggers the deletion of the corresponding rows in the dependent weak entity.

Table constraint prevents the duplication of an invoice line.

The LINE Table in Access



Some Notes On Table Creation

- Given our sample database, the PRODUCT table contains a foreign key that references the VENDOR table. Thus, the VENDOR table must be created first. In general, the table on the "1" side of a 1:M relationship must be created before the table on the "M" side can be created.
- In Oracle 9i, if you use the PRIMARY KEY designation you do not specify the NOT NULL and UNIQUE specifications. In fact, you will get an error message if you do so.
- ON UPDATE CASCADE is part of the ANSI standard but several RDBMSs do not support it. Oracle is one which does not support this specification.
- If the primary key is a composite key, all of the attributes of the key are contained within a set of parentheses and are separated by commas. For example, the table LINE on page 11 would have its primary key defined as:

PRIMARY KEY (inv_number, line_number).

Some Notes On Table Creation (cont.)

- Support for referential constraints varies widely from RDBMS to RDBMS.
  - MS Access, SQL Server, and Oracle support ON DELETE CASCADE.
  - MS Access and SQL Server, support ON UPDATE CASCADE.
  - Oracle does not support ON UPDATE CASCADE.
  - Oracle supports SET NULL.
  - MS Access and SQL Server do not support SET NULL.
- MS Access does not support ON DELETE CASCADE or ON UPDATE CASCADE at the SQL line level, however, it does support it through the relationship window interface (see Day 16 notes).

The DML Portion of SQL

- The DML portion of SQL can be viewed as two separate components which overlap in certain areas. The two components are the non-query DML commands and the query DML commands.
- Non-query DML commands allow you to populate tables (INSERT), modify data in tables (UPDATE), delete data from tables (DELETE) as well as make changes permanent (COMMIT) and undo changes (to some extent with ROLLBACK).
- The query DML commands essentially consist of a single statement (SELECT) with many different optional clauses.
- We'll look at the non-query part of the DML first.

Summary of SQL DML Commands

Command or Option	Description
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows
UPDATE	Modifies attribute values in one or more of a table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values
Comparison Operators	
=, <, >, <=, >=, <>	Used in conditional expressions
Logical Operators	
AND, OR, NOT	Used in conditional expressions

Summary of SQL DML Commands (cont.)

Command or Option	Description
Special Operators	*used in conditional expressions*
BETWEEN	Checks whether an attributes values is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks if a subquery returns any rows or not
DISTINCT	Limits values to unique values, i.e., eliminates duplicates
Aggregate Functions	*used with SELECT to return mathematical summaries on columns*
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Adding Rows To Tables

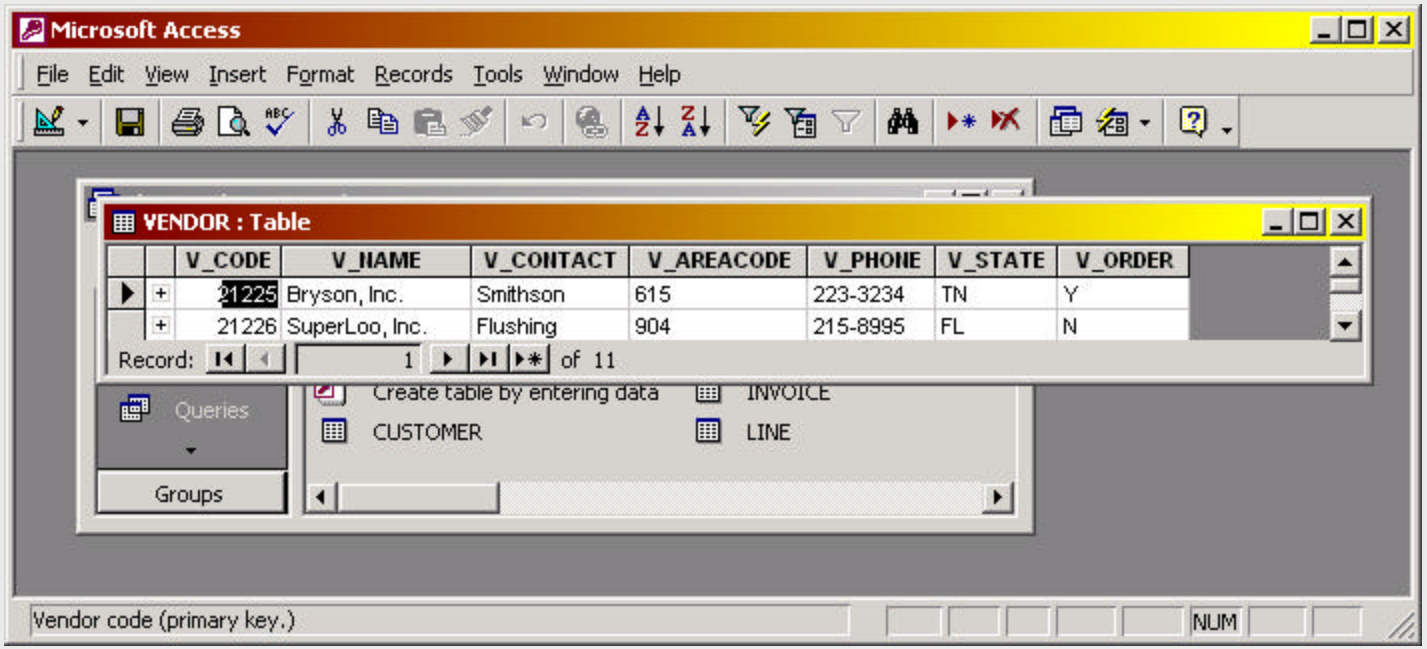
- SQL requires the use of the INSERT command to enter data into a table.
- The syntax of the INSERT command is:

```
INSERT INTO tablename
        VALUES (value1, value 2, ...value n);
```


Example - Adding Rows To Tables

- In order to add the two rows to the VENDOR table shown below, we would need to execute the following two SQL commands:

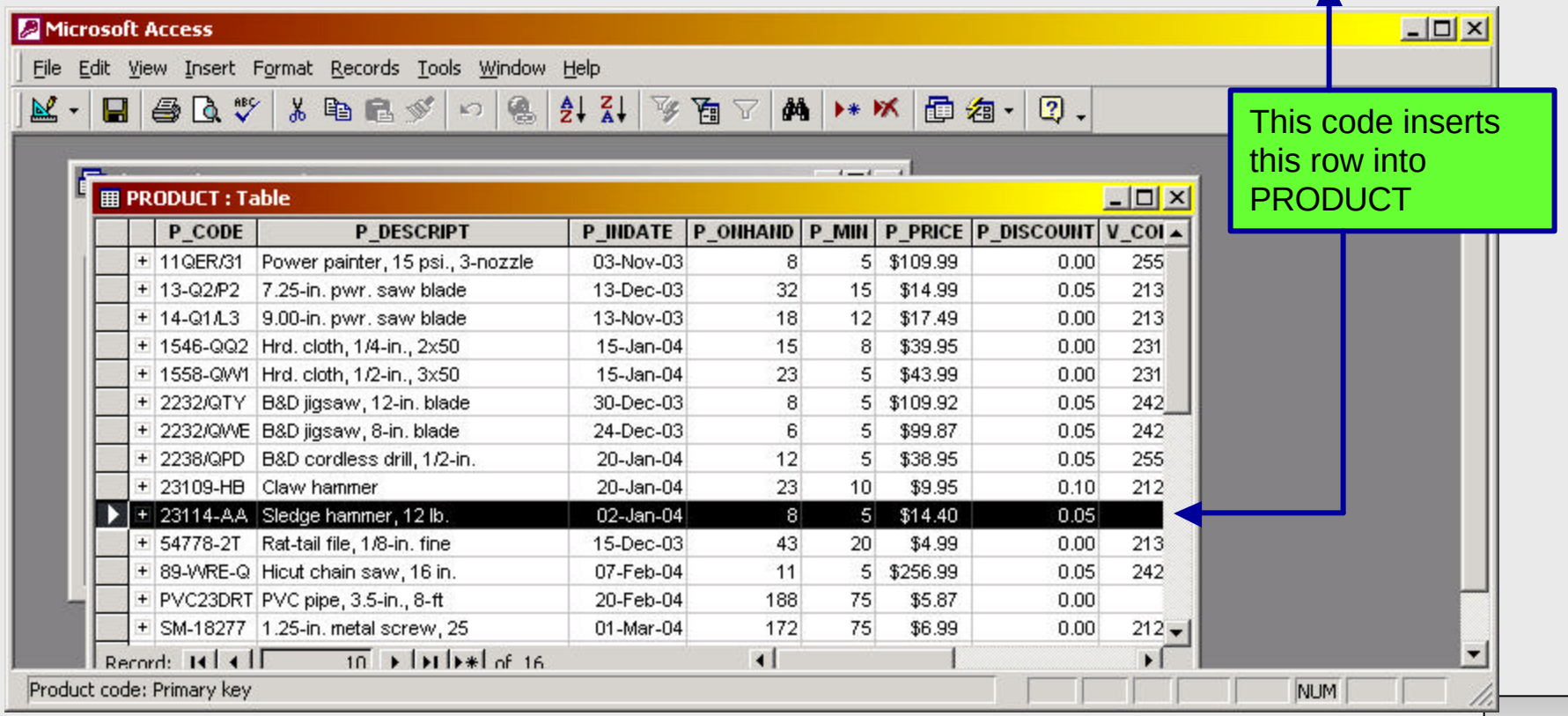
```
INSERT INTO VENDOR
    VALUES (21225, 'Bryson, Inc.', 'Smithson', '615', '223-3234', 'TN', 'Y');
INSERT INTO VENDOR
    VALUES (21226, 'SuperLoo, Inc.', 'Flushing', '904', '215-8995', 'FL', 'N');
```



Example - Adding Rows With Nulls To Tables

- If an attribute in a row has no value (i.e., is null) you would use the following syntax to enter the row into the table:

```
INSERT INTO PRODUCT
    VALUES ('23114-AA',  'Sledge hammer, 12 lb.', '02-Jan-02', 8, 5, 14.40, 0.05, NULL);
```



Example - Adding Rows With Optional Values To Tables

- There may be occasions on which more than one attribute is optional (i.e., can be null). Rather than declaring each attribute as NULL in the INSERT command, you can just indicate the attributes that have required values.
- This is done by listing the attribute names for which values are being inserted inside parentheses after the table name.
- For the purposes of example, suppose that only the P_CODE and P_DESCRIPT are required attributes in the PRODUCT table. If this is the case, then either of the following syntactic forms could be used:

```
INSERT INTO PRODUCT
    VALUES ('23114-AA',  'Sledge hammer, 12 lb.', NULL, NULL, NULL, NULL, NULL, NULL);
-or-
INSERT INTO PRODUCT(P_CODE, P_DESCRIPT)
    VALUES('23114-AA', 'Sledge hammer, 12 lb.');
```


Deleting Rows From A Table

- It is easy to use SQL to delete a row from a table. This is handled via the DELETE command.
- The syntax of the DELETE command is:

```
DELETE FROM tablename
    [WHERE conditionlist ];
```

- To delete a row of a table based on a primary key value you would use a command such as:

```
DELETE FROM PRODUCT
    WHERE P_CODE = '23114-AA';
```

Deleting Rows From A Table (cont.)

- Deletion also works to remove potentially multiple rows from a table.
  - For example, suppose that we want to delete every product from the PRODUCT table where the value of the P_MIN attribute is equal to 5. To accomplish this you would issue the following command:

```
DELETE FROM PRODUCT
WHERE P_MIN = 5;
```

- DELETE is a set-oriented command. This means that since the WHERE condition is optional, if it is not specified, all rows from the specified table will be deleted!

Updating the Rows of a Table

- To modify the data within a table the UPDATE command is used.
- The syntax of the UPDATE command is:

```
UPDATE  tablename
   SET columnname = expression [, columnname = expression ]
   [ WHERE conditionlist ];
```

- Notice that the WHERE condition is optional in the UPDATE command. If the WHERE condition is omitted, then the update is applied to all rows of the specified table.

Updating the Rows of a Table (cont.)

- As an example, suppose that we want to modify the P_INDATE from December 13, 2003 to January 18, 2004 in the second row of the PRODUCT table. We need to use the primary key value 13-Q2/P2 to locate the correct row of the table, which gives the following command syntax:

```
UPDATE PRODUCT
  SET P_INDATE = '18-Jan-2004'
  WHERE P_CODE = '13-Q2/P2';
```

- If more than one attribute is to be updated in a row, the updates are separated by commas:

```
UPDATE PRODUCT
    SET P_INDATE = '18-JAN-2004', P_PRICE = 16.99, P_MIN = 10
    WHERE P_CODE = '13-Q2/P2';
```


Saving Changes to a Table

- Any changes made to the table contents are not physically saved into the underlying physical table (the file system) until a COMMIT command has been executed.
- Depending on the sophistication of the system on which you are working, if the power should fail during the updating of a table (or database in general), before the COMMIT command was executed, your modifications are simply lost. More sophisticated systems will be able to recover from such disasters, but for small PC-based systems you'd better have a UPS installed!
- The syntax for the COMMIT command is:

```
COMMIT [ tablename ];

-or-

COMMIT;    //saves all changes made in any modified tables
```


Restoring Table Contents

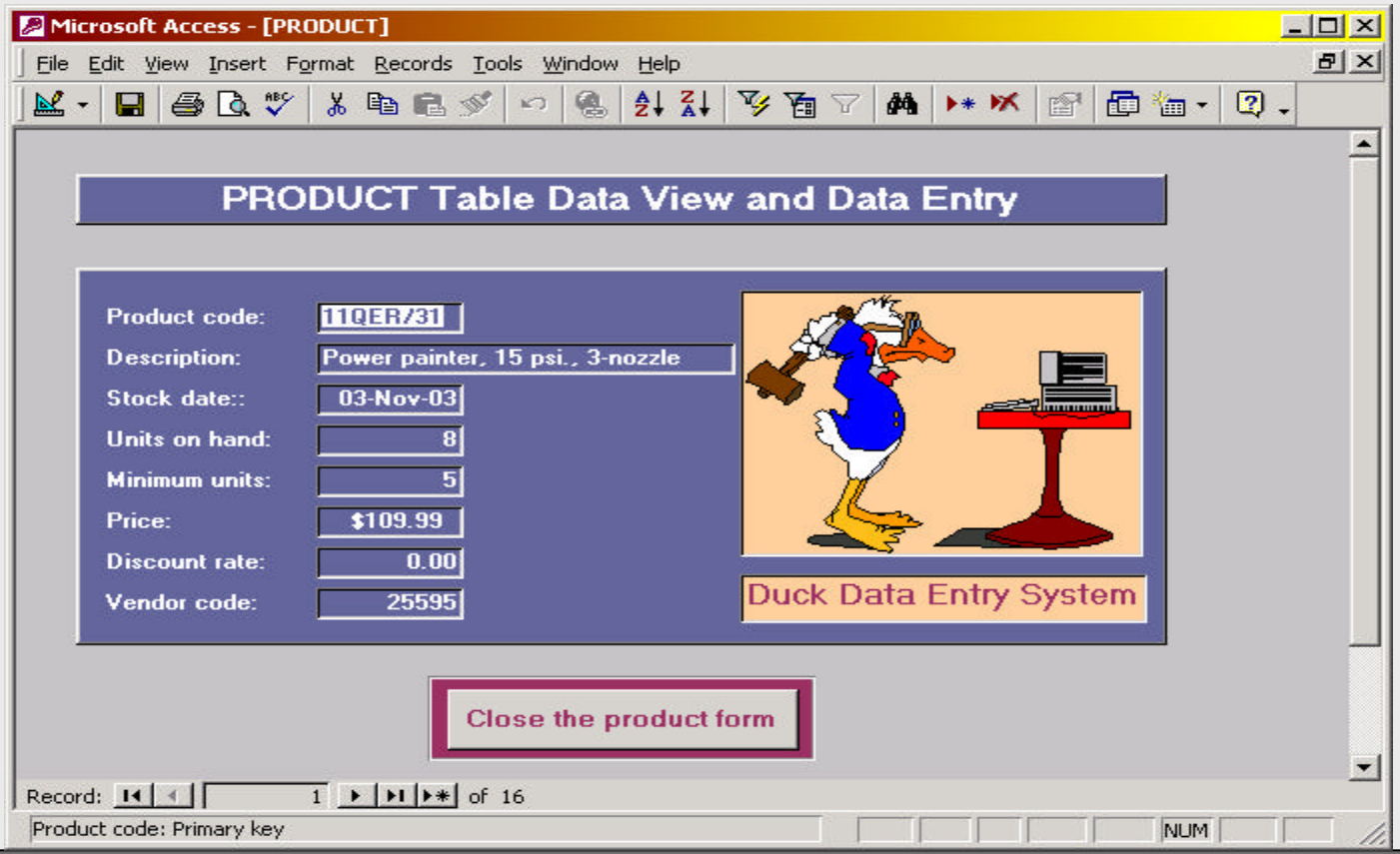
- If you have not yet used the COMMIT command to permanently store the changes in the database, you can restore the database to its previous state (i.e., the one that was the result of the last COMMIT) with the ROLLBACK command.
- ROLLBACK undoes any changes made and brings the data back to the values that existed before the changes were made.
- The syntax for the ROLLBACK command is:

```
ROLLBACK;
```

- MS Access does not support ROLLBACK! Some RDBMSs like Oracle automatically COMMIT data changes when issuing DDL commands, so ROLLBACK won't do anything on these systems.
- ROLLBACK rolls back everything since the last COMMIT, which means that even changes that you might not want undone will be if no commit has been issued.

Summary of SQL Non-Query DML Commands

- As you can see, data entry is rather cumbersome in SQL.
- End-user applications are best created with utilities that generate attractive and easy to use input screens. As we saw in Day 16 notes, MS Access handles data entry far better than does straight SQL.



Query Portion of the DML of SQL

- The query portion of the DML of SQL consists of a single command called the SELECT command.
- The syntax of the SELECT command is:

```
SELECT [ ALL | DISTINCT] columnlist
FROM tablelist
[ WHERE condition ]
[GROUP BY  columnlist ]
[HAVING condition ]
[ORDER BY columnlist ];
```

- We'll examine most of the features of the SELECT command, starting with simple queries and working our way toward more complex queries. I'll continue to use the same database that we've developed in this set of notes.

Simple Selection Queries in SQL

- Perhaps the simplest query to form is that which retrieves every row from some specified table.
- For example, suppose that we wanted to list every attribute value in every row of the PRODUCT table. In other words, to view this table. The following command will accomplish this task:

```
SELECT  P_CODE, P_DESCRIPT, P_INDATE, P_ONHAND, P_MIN,
        P_PRICE, P_DISCOUNT, V_CODE
FROM PRODUCT;
```

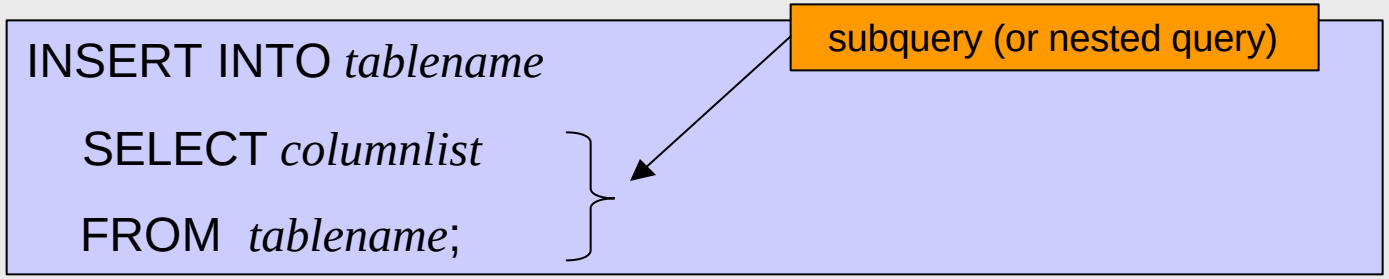
-or-

```
SELECT *
FROM PRODUCT;
```

* is a wildcard character representing all attributes in a table

Inserting Table Rows with a Select Subquery

- Although this is technically a non-query DML operation, it also includes a query command, so I've included an example here before we move on to more complex query expressions.
- SQL allows you to enter rows into a table using the data from another table as the populating basis. The syntax for this type of insert command is:



- The inner query is always executed first by the RDBMS and the values extracted by the inner query will be used as input to the outer query (in this case the INSERT command). The values returned by the inner query must match the attributes and data types of the table in the INSERT statement.

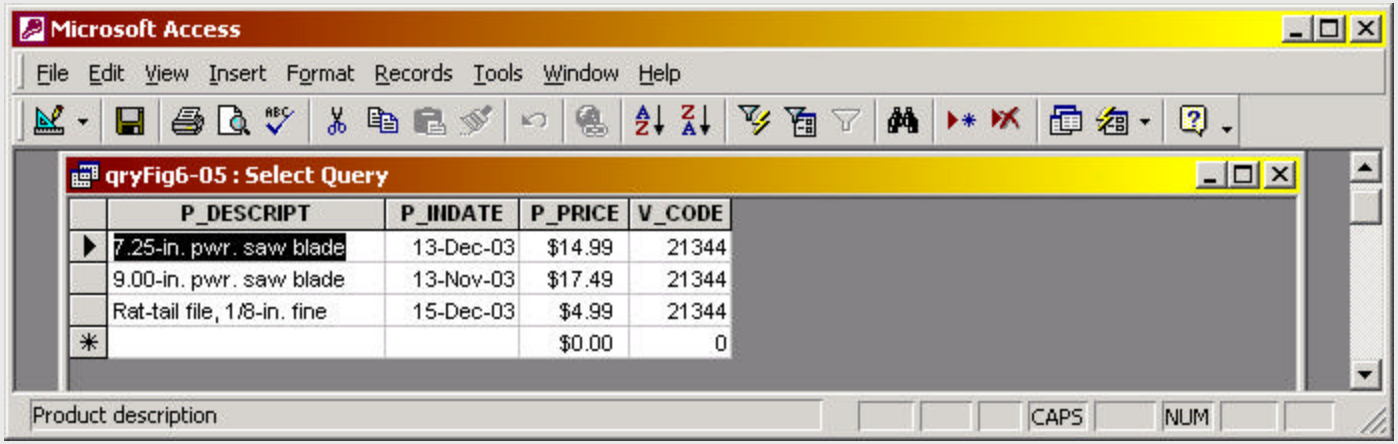
Selection Queries With Conditional Restrictions

- You can select partial table contents by placing restrictions on the rows to be included in the result. This is accomplished using the WHERE clause:

```
SELECT columnlist
    FROM tablelist
    WHERE conditionlist ;
```

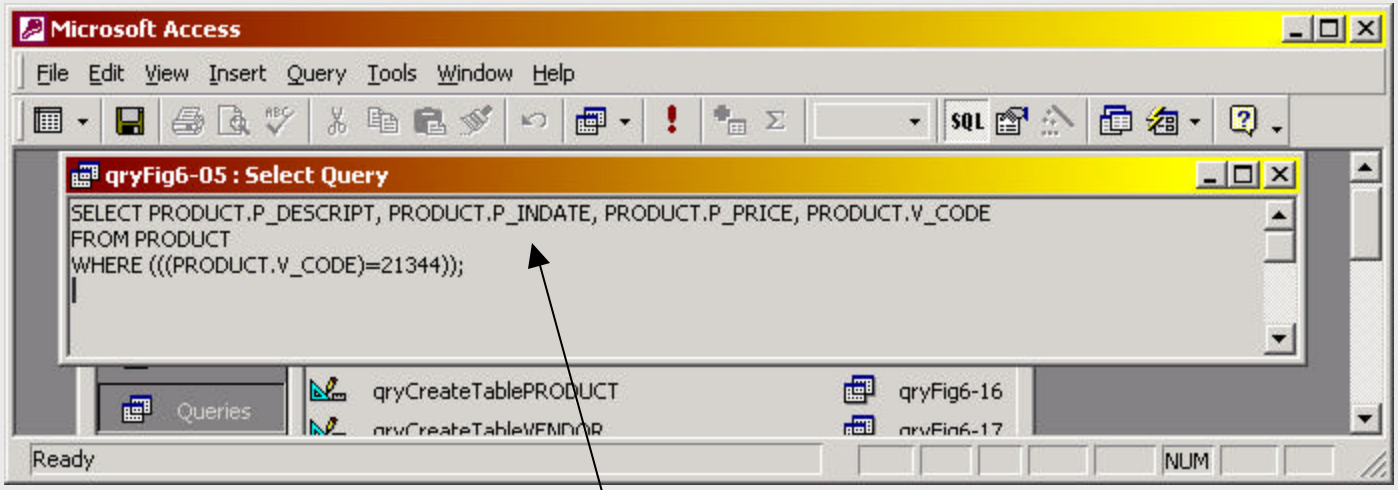
- The SELECT statement will retrieve all rows that match the specified condition(s) specified in the WHERE clause.
  - For example:

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
    FROM PRODUCT
    WHERE V_CODE = 21344;
```



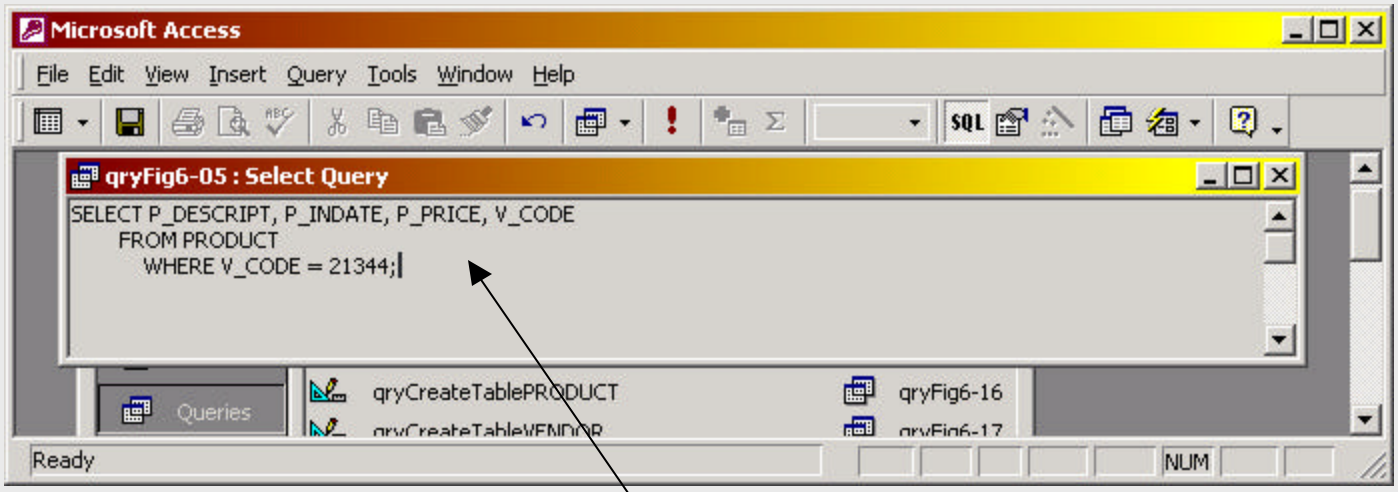
Note on Access QBE Interface for SQL

- Microsoft Access provides the Access QBE query generator. Although Access QBE generates its own "native" version of SQL, you can also elect to type standard SQL in the Access SQL window as shown on the next page.

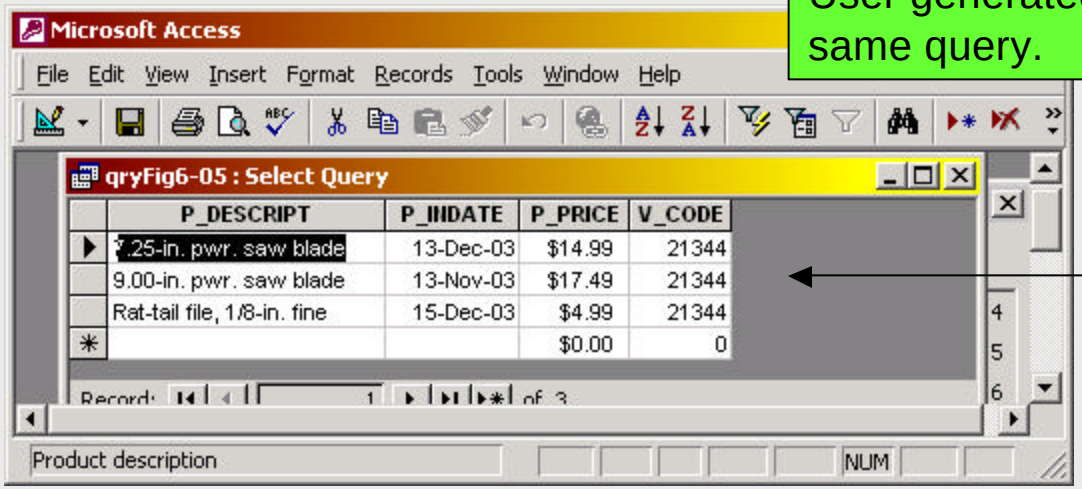


Access QBE "native" SQL code for the query on the previous page.

Note on Access QBE Interface for SQL



User generated SQL code for the same query.



Results of the user generated SQL code showing the same set of tuples as before in the result.

Conditional Restrictions in SQL Queries

- The SQL command structure provides almost limitless query flexibility. Numerous conditional restrictions may be placed on the selected table contents.
- Unless specifically testing for attribute values which are null, SQL does not include rows for which a selected attribute value is null in the result.
- Consider the following query:

```
SELECT  P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
    FROM PRODUCT
    WHERE V_CODE <> 21344;
```

- The PRODUCT table is shown on the next page and the output from this query is shown on the following page. Notice that rows 10 and 13 in the PRODUCT table do not appear in the results of this query.

Conditional Restrictions in SQL Queries (cont.)

Microsoft Access

File Edit View Insert Format Records Tools Window Help

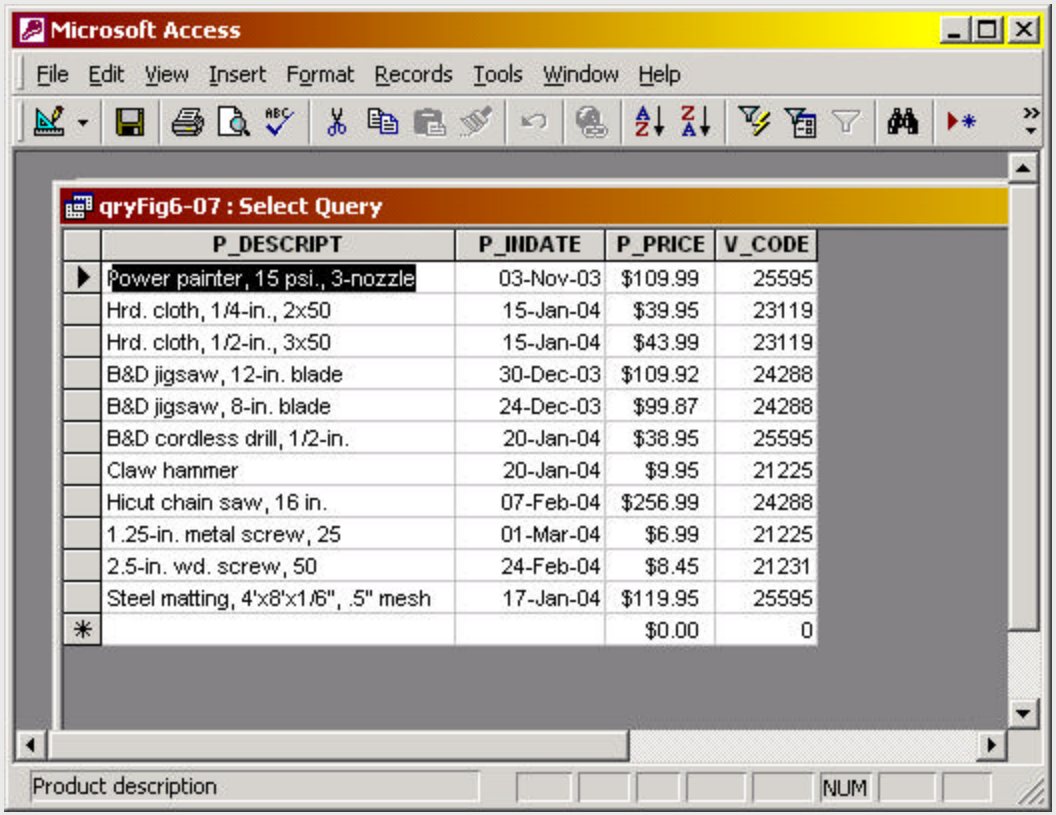
PRODUCT : Table

P_CODE	P_DESCRIPT	P_INDATE	P_ONHAND	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
1QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	$109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	$14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	$17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	$39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	$43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	$109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	$99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	$38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-04	23	10	$9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	$14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	$4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	$256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	$5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	$6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-04	237	100	$8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	18	5	$119.95	0.10	25595
			0	0	$0.00	0.00	0

Product code: Primary key NUM

These two rows do not appear in the result on the following page.

Conditional Restrictions in SQL Queries (cont.)



P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	03-Nov-03	$109.99	25595
Hrd. cloth, 1/4-in., 2x50	15-Jan-04	$39.95	23119
Hrd. cloth, 1/2-in., 3x50	15-Jan-04	$43.99	23119
B&D jigsaw, 12-in. blade	30-Dec-03	$109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-03	$99.87	24288
B&D cordless drill, 1/2-in.	20-Jan-04	$38.95	25595
Claw hammer	20-Jan-04	$9.95	21225
Hicut chain saw, 16 in.	07-Feb-04	$256.99	24288
1.25-in. metal screw, 25	01-Mar-04	$6.99	21225
2.5-in. wd. screw, 50	24-Feb-04	$8.45	21231
Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	$119.95	25595
		$0.00	0

Results of the query:

```
SELECT P_SDESCRIPT,
P_INDATE,    P_PRICE,
                V_CODE
     FROM PRODUCT
     WHERE
          V_CODE <> 21344;
```

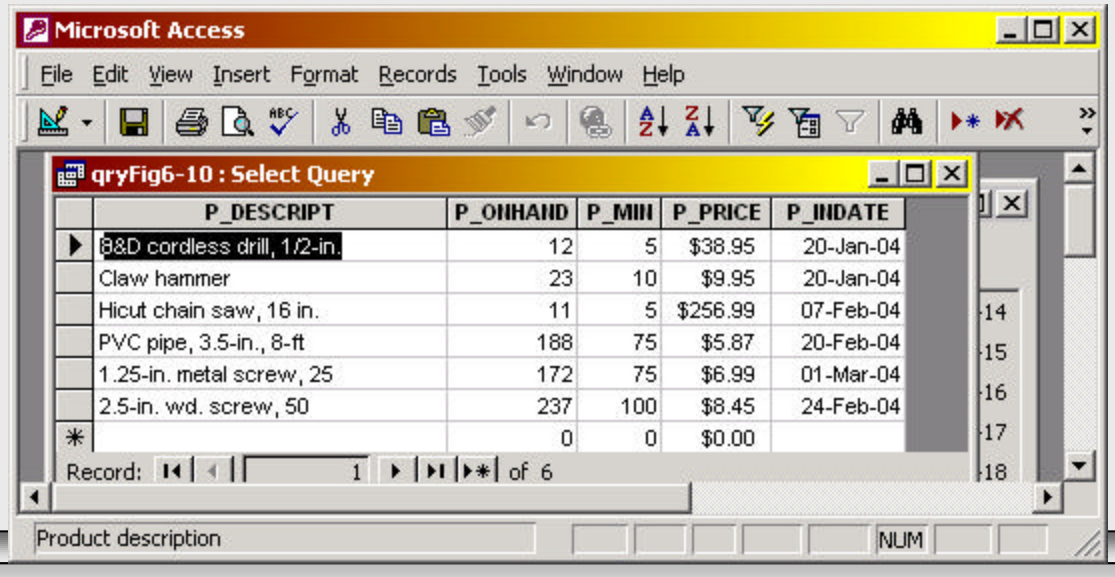

Comparisons Involving Dates in SQL Queries

- Date procedures are often more software-specific than most other SQL procedures. For example, the query to list all the rows in which the inventory stock dates occur on or after January 20, 2004, will look like this:

```
SELECT  P_DESCRIPT, P_ONHAND, P_MIN, P_PRICE, P_INDATE
    FROM PRODUCT
    WHERE P_INDATE >= '20-Jan-2004';
```

- Note that in Access the delimiters for dates is #, so in Access this query would look like:

```
SELECT  P_DESCRIPT, P_ONHAND, P_MIN, P_PRICE, P_INDATE
    FROM PRODUCT
    WHERE P_INDATE >= #20-Jan-2004#;
```



Using Computed Columns and Column Aliases

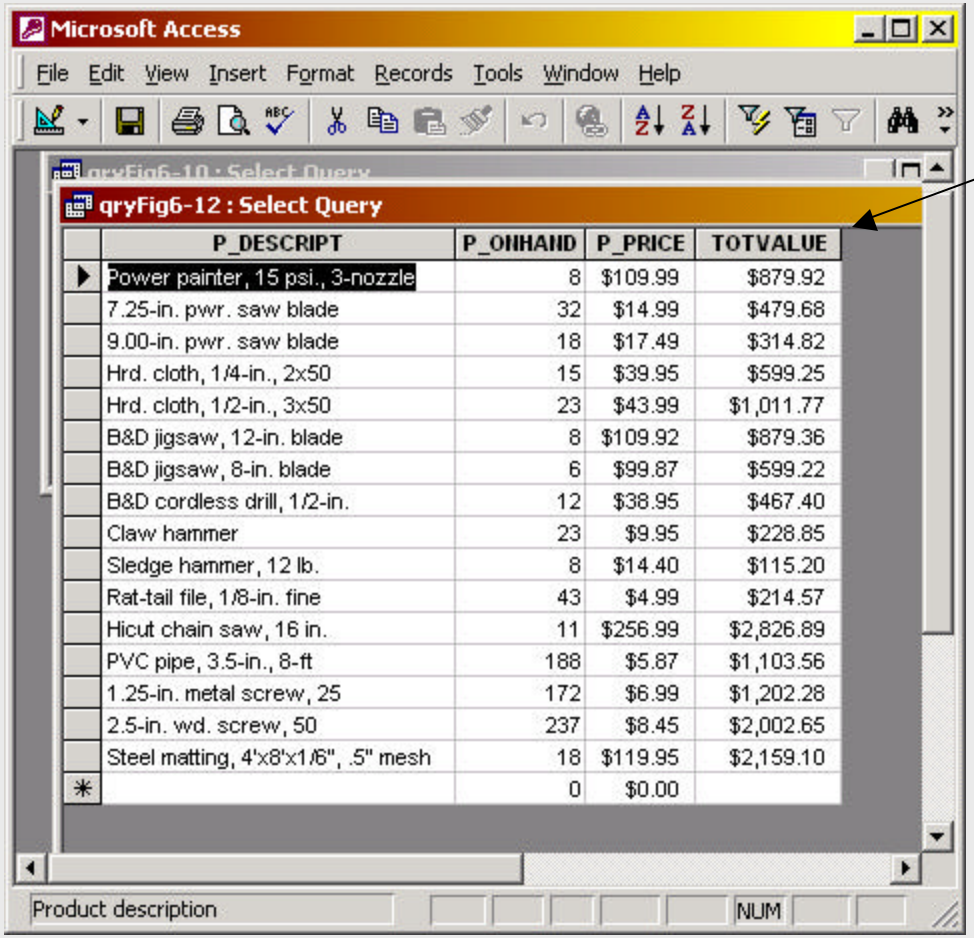
- Suppose that your query needs to determine a value which is not physically stored in the database but is calculated from data that is in the database.
- For example, let's suppose that we want to determine the total value of each of the products currently held in inventory. Logically, this determination requires the multiplication of each product's quantity on hand by its current price. The SQL query for this is shown below and the resulting output is on the next page.

```
SELECT  P_DESCRIPT, P_ONHAND,  P_PRICE, P_ONHAND * P_PRICE   AS TOTVALUE
        FROM PRODUCT
```

SQL will accept any valid expression in the computed columns that apply to the attributes in any of the tables specified in the FROM clause. Note that Access will automatically add an Expr label to all computed columns. Oracle uses the actual expression to label the computed column.

Standard SQL permits the use of aliases for any column in a SELECT statement. The alias for any column is preceded by the keyword AS.

Using Computed Columns and Column Aliases (cont.)



P_DESCRIPT	P_ONHAND	P_PRICE	TOTVALUE
Power painter, 15 psi., 3-nozzle	8	$109.99	$879.92
7.25-in. pwr. saw blade	32	$14.99	$479.68
9.00-in. pwr. saw blade	18	$17.49	$314.82
Hrd. cloth, 1/4-in., 2x50	15	$39.95	$599.25
Hrd. cloth, 1/2-in., 3x50	23	$43.99	$1,011.77
B&D jigsaw, 12-in. blade	8	$109.92	$879.36
B&D jigsaw, 8-in. blade	6	$99.87	$599.22
B&D cordless drill, 1/2-in.	12	$38.95	$467.40
Claw hammer	23	$9.95	$228.85
Sledge hammer, 12 lb.	8	$14.40	$115.20
Rat-tail file, 1/8-in. fine	43	$4.99	$214.57
Hicut chain saw, 16 in.	11	$256.99	$2,826.89
PVC pipe, 3.5-in., 8-ft	188	$5.87	$1,103.56
1.25-in. metal screw, 25	172	$6.99	$1,202.28
2.5-in. wd. screw, 50	237	$8.45	$2,002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	$119.95	$2,159.10
	0	$0.00	

The computed column with its alias.

Using A Computed Column an Alias and Date Arithmetic in a Single Query

- Suppose that we want to get a list of "out-of-warranty" products. In this case, let's assume that we've arbitrarily defined out-of-warranty products as those that have been stored more than 90 days. Therefore, the P_INDATE is at least 90 days less than the current date. The Access version of this query is shown below followed by the Oracle version, with the resulting output shown on the next page.

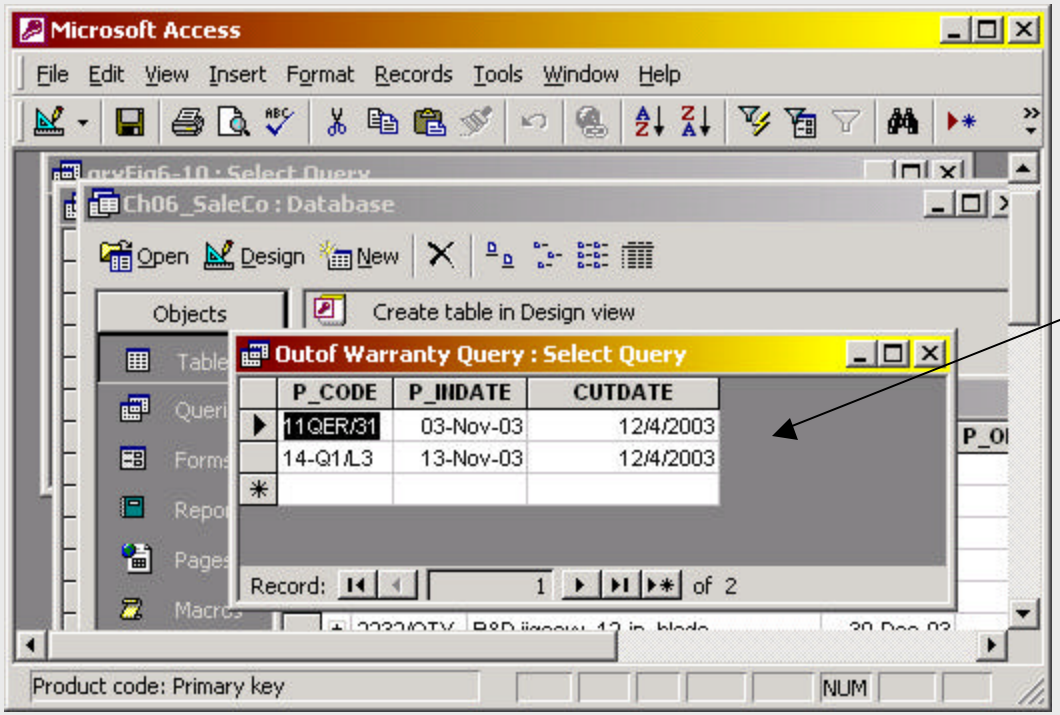
Access Version

```
SELECT  P_CODE, P_INDATE, DATE() – 90 AS CUTDATE
        FROM PRODUCT
        WHERE P_INDATE <= DATE() – 90;
```

Oracle Version

```
SELECT  P_CODE, P_INDATE, SYSDATE – 90 AS CUTDATE
        FROM PRODUCT
        WHERE P_INDATE <= SYSDATE – 90;
```


Using A Computed Column an Alias and Date Arithmetic in a Single Query



Verify that these are the only two products that are out of range for the warranty by checking the dates of products in the PRODUCTS table on page 45.

Using The Logical Operators AND, OR, and NOT

- In the real world, a search of data normally involves multiple conditions. SQL allows you to express multiple conditions in a single query through the use of logical operators.
- The logical operators supported by SQL are: AND, OR, and NOT.
- Suppose you want a list of the table of PRODUCTS for either V_CODE = 21344 or V_CODE = 24288. The SQL query to accomplish this is:

```
SELECT    P_DESCRIPT,
          P_INDATE,
          P_PRICE,
          V_CODE
FROM PRODUCT
WHERE
      V_CODE = 21344
      OR
      V_CODE = 24288;
```