

SQL Programming

DDL

In this 'final' module we examine the SQL statements that are used to create the database structures.

These statements are referred to as the data definition language (DDL) statements, because they are the statements that are used to DEFINE the DATA (Structures) in the database.

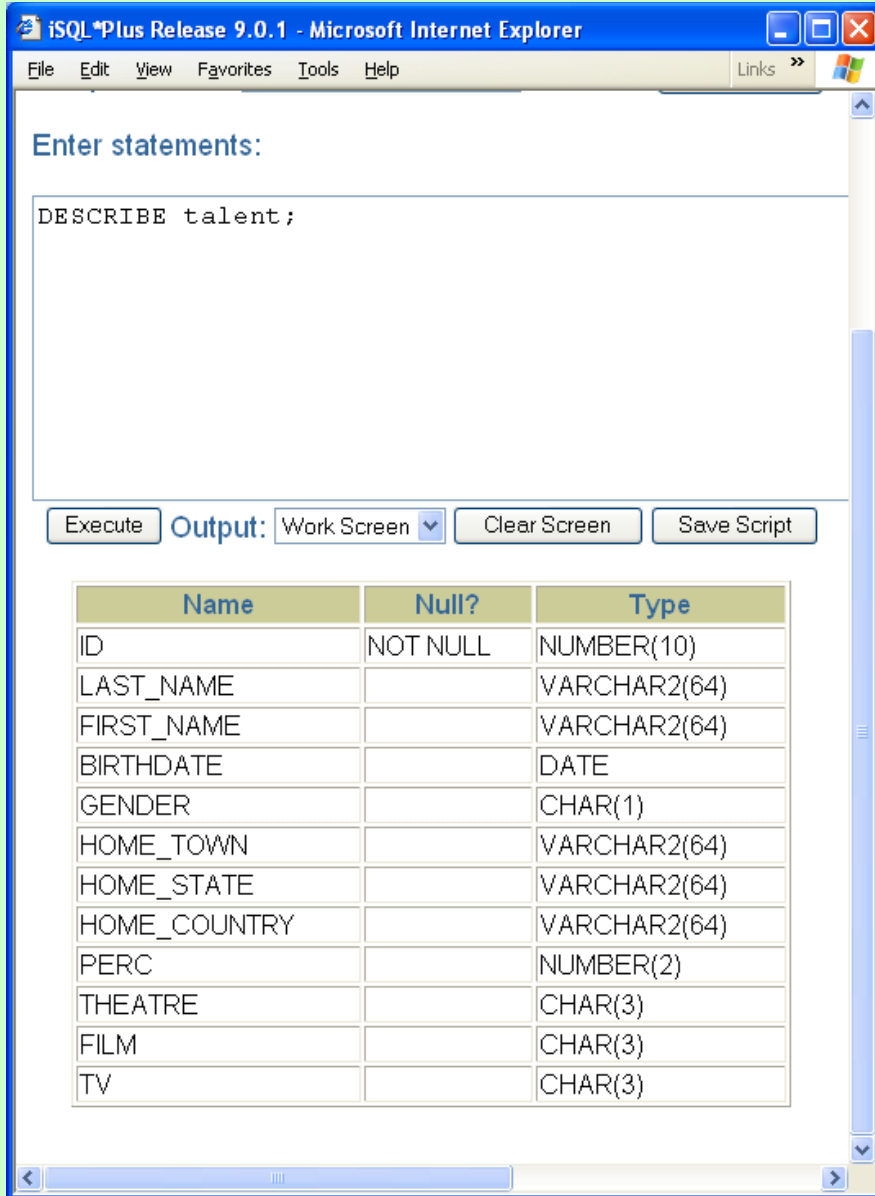
Module 17: DDL

Page B-1 DESCRIBE

We used the DESCRIBE command before to examine the structure of a table.

Let's take another look at the talent table.

What are the structural elements of the talent table?



The screenshot shows the iSQL*Plus Release 9.0.1 interface within a Microsoft Internet Explorer browser. The "Enter statements:" text area contains the command `DESCRIBE talent;`. Below the text area are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The output is displayed as a table with three columns: "Name", "Null?", and "Type".

Name	Null?	Type
ID	NOT NULL	NUMBER(10)
LAST_NAME		VARCHAR2(64)
FIRST_NAME		VARCHAR2(64)
BIRTHDATE		DATE
GENDER		CHAR(1)
HOME_TOWN		VARCHAR2(64)
HOME_STATE		VARCHAR2(64)
HOME_COUNTRY		VARCHAR2(64)
PERC		NUMBER(2)
THEATRE		CHAR(3)
FILM		CHAR(3)
TV		CHAR(3)

Module 17: DDL

Page B-2 Table Structure

Every table has a name.

Every table contains at least one column.

Every column has a name.

Every column is of a particular datatype and in this regard we often say that columns define a domain of values from which their members may be drawn.

The domain specifies the following:

Datatype: number, character, date

Size/format: eg. 30 character name field,
date in DD-Mon-YYYY format

The screenshot shows the iSQL*Plus interface with the following components:

- Header:** iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer
- Menu:** File, Edit, View, Favorites, Tools, Help
- Input Area:** Enter statements: DESCRIBE talent;
- Buttons:** Execute, Output: Work Screen, Clear Screen, Save Script
- Table Structure:**

Name	Null?	Type
ID	NOT NULL	NUMBER(10)
LAST_NAME		VARCHAR2(64)
FIRST_NAME		VARCHAR2(64)
BIRTHDATE		DATE
GENDER		CHAR(1)
HOME_TOWN		VARCHAR2(64)
HOME_STATE		VARCHAR2(64)
HOME_COUNTRY		VARCHAR2(64)
PERC		NUMBER(2)
THEATRE		CHAR(3)
FILM		CHAR(3)
TV		CHAR(3)

The CREATE command is used to create structures in the database.

And the syntax for the CREATE statement is something like this:

```
CREATE structure-type structure-name  
      [details about that structure]
```

Let's build a table named music.

What kind of structure are we building?

What kind of structure are we building?

What kind of structure are we building?

CREATE *structure-type structure-name*
[*details about that structure*]

A table

CREATE TABLE *structure-name*
[*details about that structure*]

So step one is to specify TABLE as our structure type.

What's the name for this structure?

Module 17: DDL

Page B-5 CREATE TABLE name

CREATE *structure-type structure-name*
[*details about that structure*]

What kind of structure are we building?

A table

CREATE TABLE *structure-name*
[*details about that structure*]

So step one is to specify TABLE as our structure type.

What's the name for this structure?

music

CREATE TABLE music
[*details about that structure*]

So the next step is to plug in the name of our table

Now all we have to do is 'flesh out' the details.

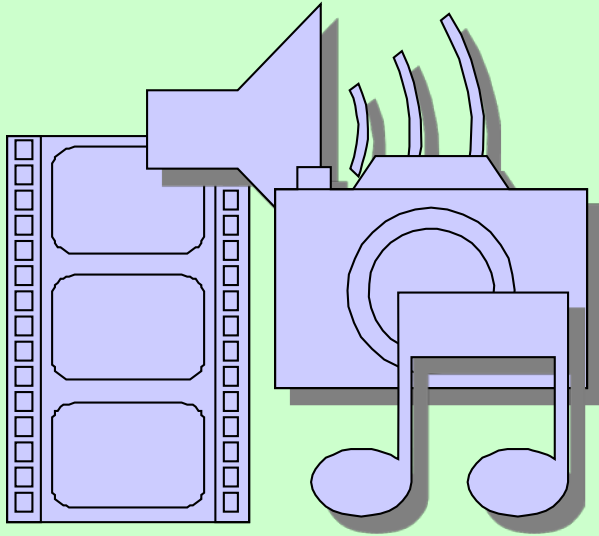
I'd like the music table to keep track of the music that I have at home in my CD collection.

This will be a simple table with only a few fields of information.

I want to know the name of the artist (or band), and the title of the album. As well as the media/format, eg. CD, cassette, MP3.

Oh yeah, I read somewhere that records in a database should have a key? Or an id number? Or something like that. So let's throw one of those in too.

Now this process of investigating my needs and coming up with a suitable table falls under the work heading of systems analysis (and design) or database analysis and design. You're only a programmer! Or more correctly, this is only a *programming* class. So we won't delve into the design aspects.



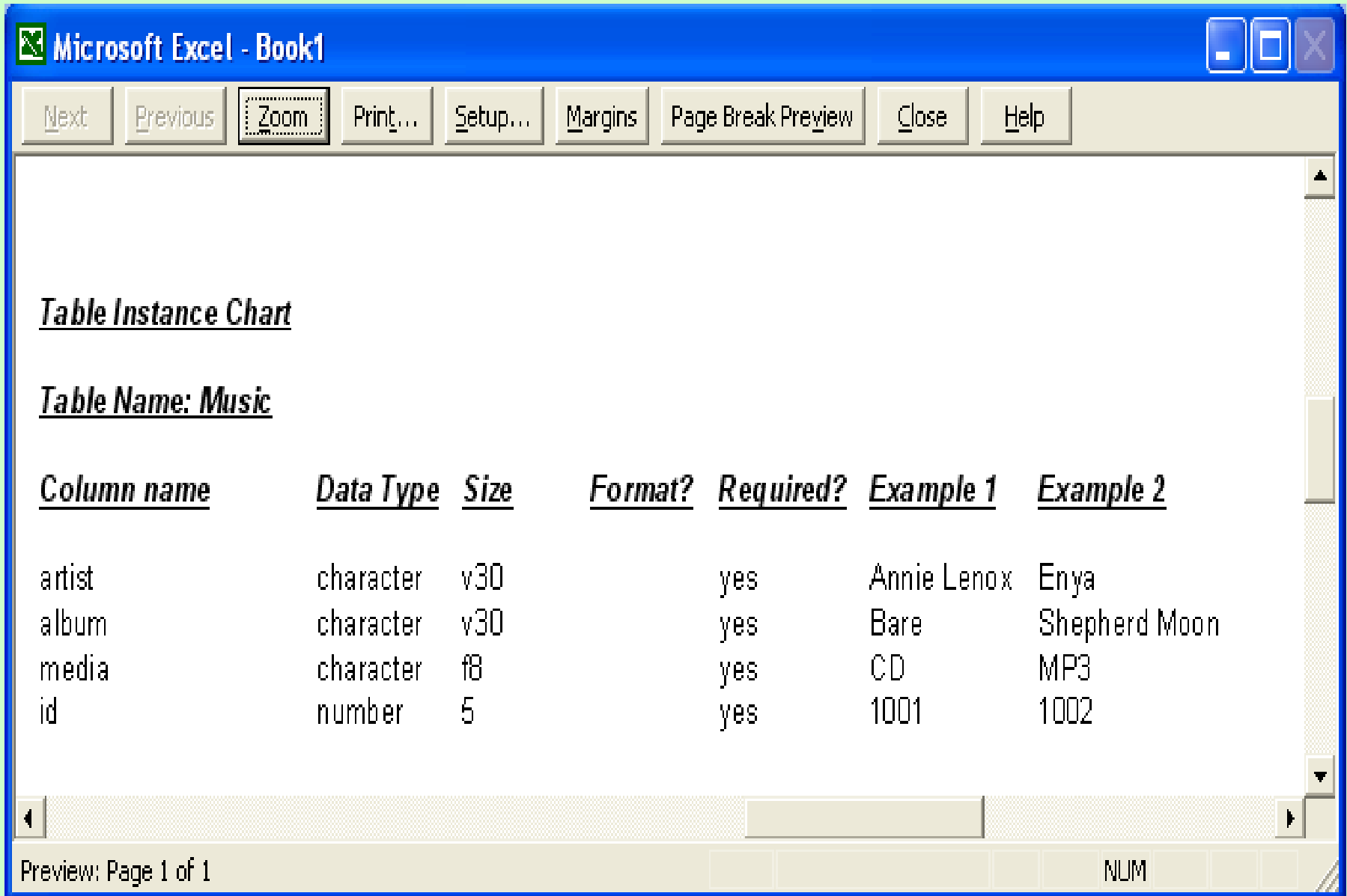
Although you won't develop the design, as a programmer you'll be expected to read the design and then implement all of the features specified therein.

Designs generally come in one of two formats:

- Table Instance Charts

- Entity Relationship Diagrams

In this module we're only going to focus on Table Instance Charts



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Book1". The window has a menu bar with "Next", "Previous", "Zoom", "Print...", "Setup...", "Margins", "Page Break Preview", "Close", and "Help". The main content area displays a table titled "Table Instance Chart" with the subtitle "Table Name: Music". The table has seven columns: "Column name", "Data Type", "Size", "Format?", "Required?", "Example 1", and "Example 2". The table contains four rows of data: "artist" (character, v30, yes, Annie Lenox, Enya), "album" (character, v30, yes, Bare, Shepherd Moon), "media" (character, f8, yes, CD, MP3), and "id" (number, 5, yes, 1001, 1002). The status bar at the bottom shows "Preview: Page 1 of 1" and "NUM".

<u>Column name</u>	<u>Data Type</u>	<u>Size</u>	<u>Format?</u>	<u>Required?</u>	<u>Example 1</u>	<u>Example 2</u>
artist	character	v30		yes	Annie Lenox	Enya
album	character	v30		yes	Bare	Shepherd Moon
media	character	f8		yes	CD	MP3
id	number	5		yes	1001	1002

Microsoft Excel - SQL lecture notes table instance charts

Next Previous Zoom Print... Setup... Margins Page Break Preview Close Help

<u>Table Instance Chart</u>						
<u>Table Name: Music</u>						
<u>Column name</u>	<u>Data Type</u>	<u>Size</u>	<u>Format?</u>	<u>Required?</u>	<u>Example 1</u>	<u>Example 2</u>

Preview: Page 1 of 1 NUM

The general format for a table instance chart allows the designer to specify these features about the table:

Table name

Column name

Data type

Size

Format

Required?

Samples

So now all you need to know as the programmer is how to map a table instance chart into the SQL CREATE statement.

Datatypes come in three main flavors: character, numeric, and date.

As to character datatype, they are defined in the database in one of two ways: either as a fixed length character field, or as a variable length character field.

Which of these two subtypes you choose is based primarily on performance (and space) considerations.

A fixed length 10 character field, will always use 10 characters, regardless of the size of the entry. A variable length 10 character field will allow no more than 10 characters in the entry, and it will readjust the amount of space it takes up, based on the size of the entry.

Fixed length character fields are defined with the CHAR datatype, followed by the size of the column in parentheses, eg.

```
name CHAR(30),  
city CHAR(20)
```

Variable length character fields (in Oracle) are defined with the VARCHAR2 datatype, in a similar fashion, eg.

```
major VARCHAR2(16),  
street VARCHAR2(32)
```

Number fields are defined with the NUMBER datatype.

Integer values, or whole numbers, are defined simply by specifying the precision (number of digits) that you need for the value.

2-digit value	NUMBER(2)
5-digit value	NUMBER(5)
9-digit value	NUMBER(9)

Decimal values, require you to specify both the number of digits that are to be displayed, as well as how many positions to the right of the decimal point are to be permitted.

Here are a few examples of sample data, and the datatype specification needed to correctly store that value.

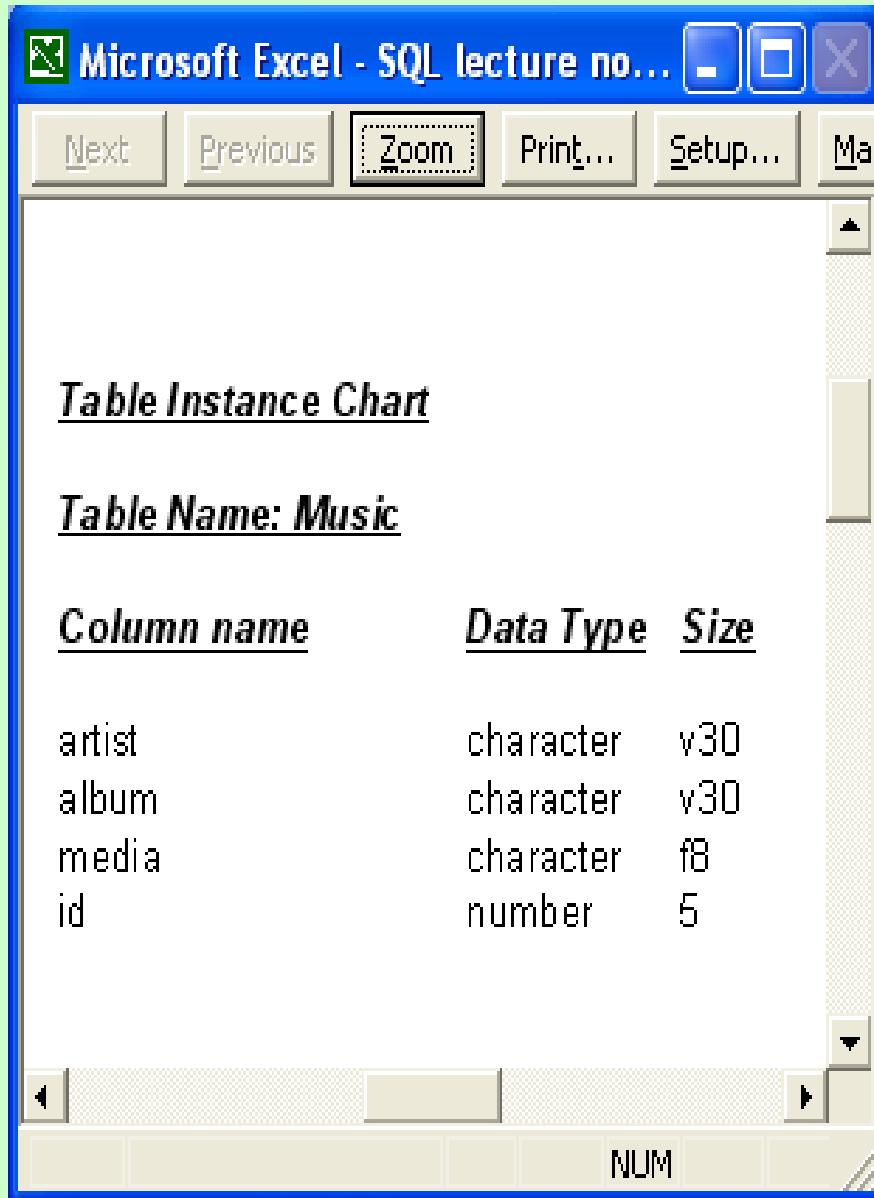
<u>Value</u>	<u>Definition</u>
30	Number (2)
30.1	NUMBER(3,1)
30.22	NUMBER(4,2)
30000.123	NUMBER(8,3)

Date (and time) values are stored in the database using the DATE datatype.

For example,

birthdate	DATE,
application_date	DATE

Let's return to our table instance chart and finish the program we re writing to create the music table structure.



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - SQL lecture no...". The window has a menu bar with "Next", "Previous", "Zoom...", "Print...", "Setup...", and "Mac". The main content area displays a table instance chart for a table named "Music". The table has three columns: "Column name", "Data Type", and "Size". The rows are:

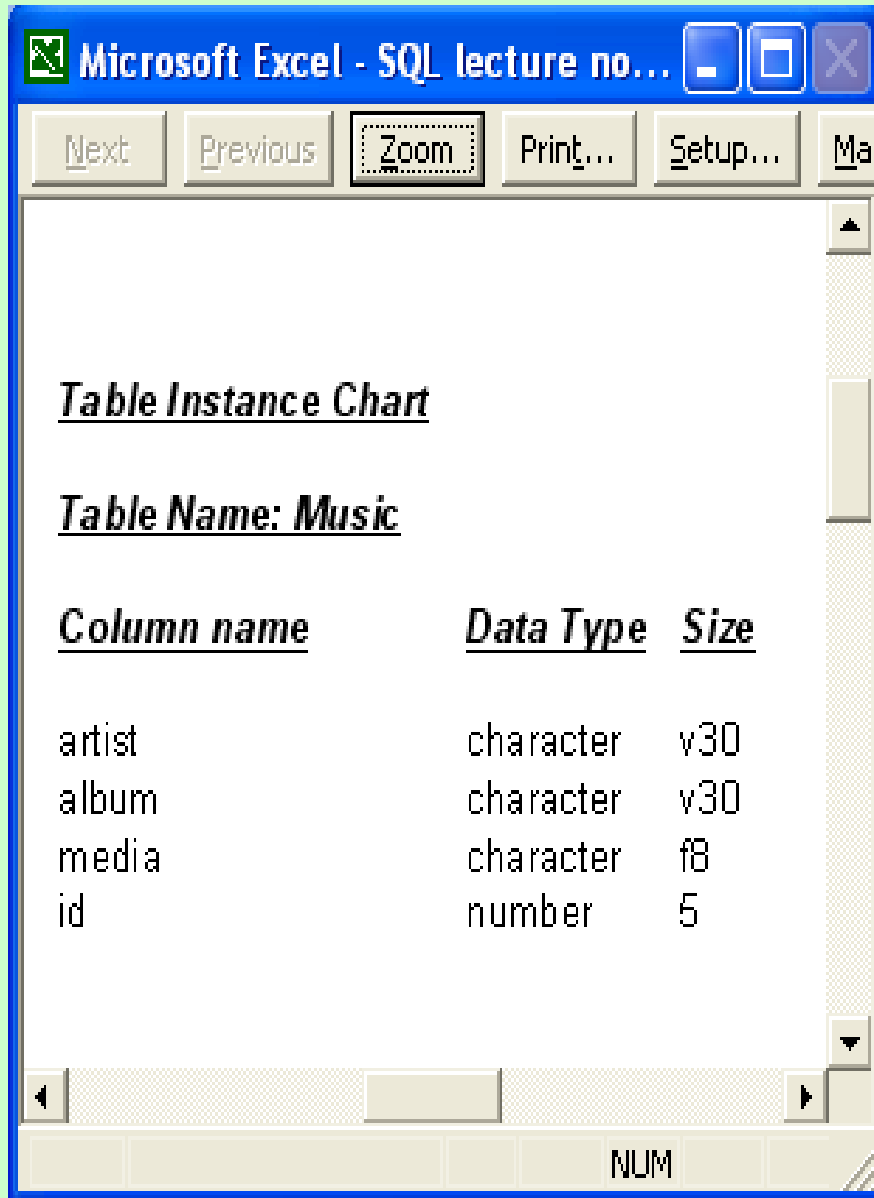
<u>Column name</u>	<u>Data Type</u>	<u>Size</u>
artist	character	v30
album	character	v30
media	character	f8
id	number	5

The status bar at the bottom shows "NUM".

We had gotten this far:

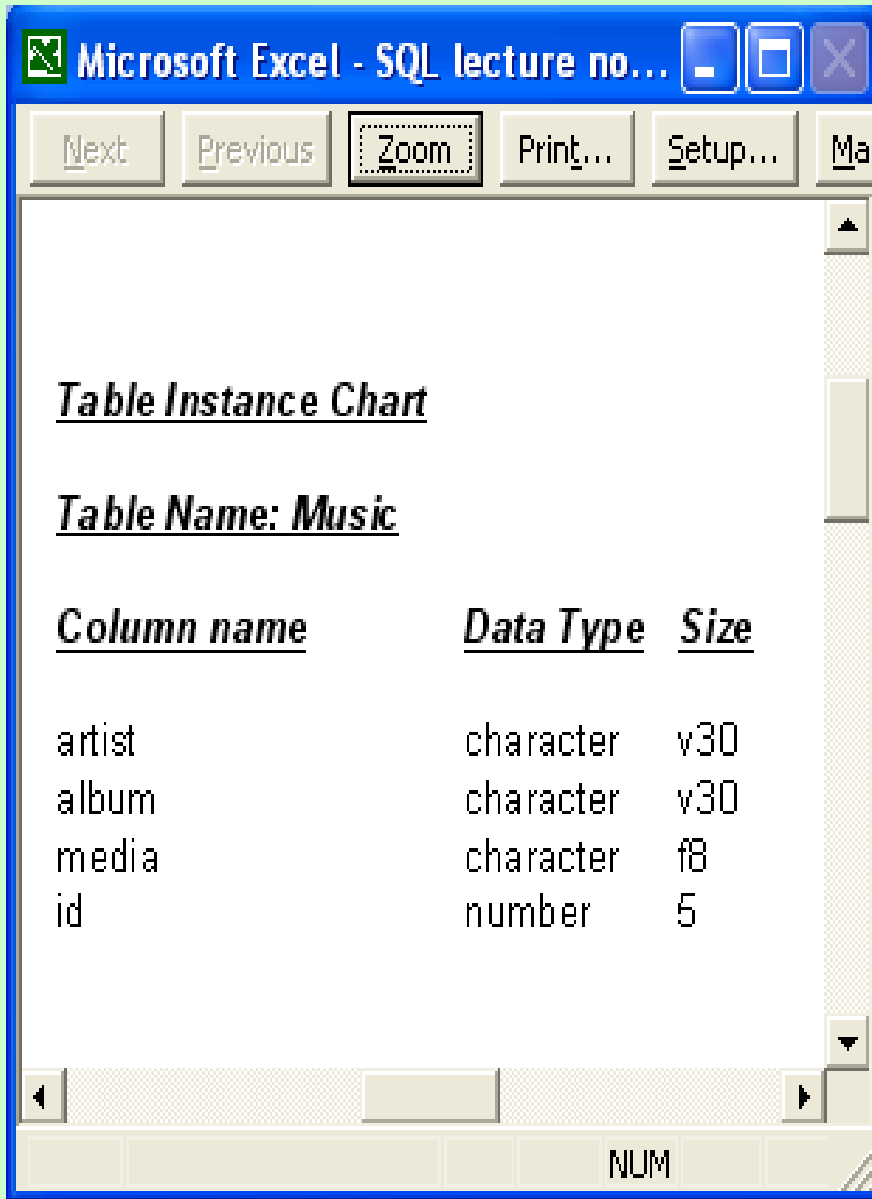
```
CREATE TABLE music
    [details about that structure]
```

Now we need to provide the details for the table structure. Minimally we need to specify the name and datatype for each of the columns in the table.



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - SQL lecture no...". The window has a menu bar with "Next", "Previous", "Zoom...", "Print...", "Setup...", and "Mac". Below the menu bar, the text "Table Instance Chart" is displayed. Underneath, the text "Table Name: Music" is shown. A table with three columns is displayed: "Column name", "Data Type", and "Size". The table contains four rows of data: "artist" (character, v30), "album" (character, v30), "media" (character, f8), and "id" (number, 5). The bottom status bar shows "NUM".

<u>Column name</u>	<u>Data Type</u>	<u>Size</u>
artist	character	v30
album	character	v30
media	character	f8
id	number	5



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - SQL lecture no...". The window has a menu bar with "Next", "Previous", "Zoom...", "Print...", "Setup...", and "Mac...". Below the menu bar, the text "Table Instance Chart" is displayed. Underneath, the text "Table Name: Music" is shown. A table with three columns is displayed: "Column name", "Data Type", and "Size". The table contains four rows of data: "artist" (character, v30), "album" (character, v30), "media" (character, f8), and "id" (number, 5). The table is displayed in a grid format with a blue border. The status bar at the bottom shows "NUM".

<u>Column name</u>	<u>Data Type</u>	<u>Size</u>
artist	character	v30
album	character	v30
media	character	f8
id	number	5

All of the column information immediately follows the table-name, and needs to be enclosed within parentheses.

So the syntax now looks something like this:

```
CREATE TABLE table-name
(
    column-name1 datatype,
    column-name2 datatype
    ...
);
```

And we should come up with something like this for our music table:

```
CREATE TABLE music
(
    artist VARCHAR2(30),
    album VARCHAR2(30),
    media CHAR(08),
    id NUMBER(05)
);
```

Module 17: DDL



Page D-4 Stylistic Convention

Notice the way I've used white space to highlight my code, and hopefully, make it more understandable.

```
CREATE TABLE  music
(
    artist  VARCHAR2(30),
    album   VARCHAR2(30),
    media   CHAR(08),
    id      NUMBER(05)
);
```

CREATE TABLE stands on a line of its own. All details are indented underneath that 'heading'.

Details are coded in a columnar fashion, ie. Column names appears to be in a column of their own, datatypes appear to be in a column of their own.

One columns worth of information at a time.

Module 17: DDL

Page D-5 Stylistic Convention

Remember, these conventions are for the reader, not SQL.

SQL would be happy with this code.

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

ORACLE iSQL*Plus

[Password](#) [Log Out](#) [Help](#)

Script Location:

Enter statements:

```
CREATE TABLE music
(artist
VARCHAR2(30), album
VARCHAR2(30), media
CHAR(08), id NUMBER(05)
);
```

Output:

Table created.

Module 17: DDL

Page D-6 Verify

Use the describe command to check that the table was created as you had intended.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The title bar reads "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The main content area features the Oracle logo and "iSQL*Plus" text, along with icons for Password, Log Out, and Help. Below these are links for "Password", "Log Out", and "Help". A "Script Location:" field is followed by "Browse..." and "Load Script" buttons. The "Enter statements:" section contains a text area with the command "DESCRIBE music;". Below the text area are "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script" buttons. The output is displayed in a table with three columns: "Name", "Null?", and "Type".

Name	Null?	Type
ARTIST		VARCHAR2(30)
ALBUM		VARCHAR2(30)
MEDIA		CHAR(8)
ID		NUMBER(5)

Now that you can build tables that can store data, let's look at building virtual tables.

What's a virtual table?

A virtual table is a 'fake' table. It doesn't store any data, it just behaves like a table. In this regard, it has the behavior of a table, but not the data.

Pretty cool huh?

You can actually build something in the database that won't store data, but acts like a table???

Who dreamed this up? And why the heck would you ever use it?

Give me a moment and we'll get there, but first,

A virtual table is stored in the database as a VIEW structure.

I realize I just said that a virtual table (View) doesn't store data, so what is it that's getting stored in the database?

A SQL program, specifically a SELECT statement.

Think of a view as a SELECT subquery. Subqueries don't store data do they? But they do behave like tables, don't they?

So back to the question we phrased a moment ago.

Why? What's the point of having a view?

A VIEW can be set up to give different user groups their own *views* of the database.

Consider this scenario. The Personnel department stores all of the information about every employee in a single table. But not everyone in the company should have access to all of that information.

The phone directory office should be able to access name department and phone number, and nothing else. Line managers should be able to access information about their team, but nothing about other team members.

The two primary reasons for using views are to provide a level of security, and to simplify access.

I have a few friends and we share our movies between us.

I've got some old BETA tapes and LaserDiscs that none of these folks are interested in. They only borrow my DVDs.

They'll hook up to my database and check my collection to see if there's anything they want to watch. But every time they run a query they need to include this predicate:

```
WHERE format <> 'DVD'
```

I want to create a view that'll simplify their using my database

Recall the general syntax description for the CREATE statement. It went something like this:

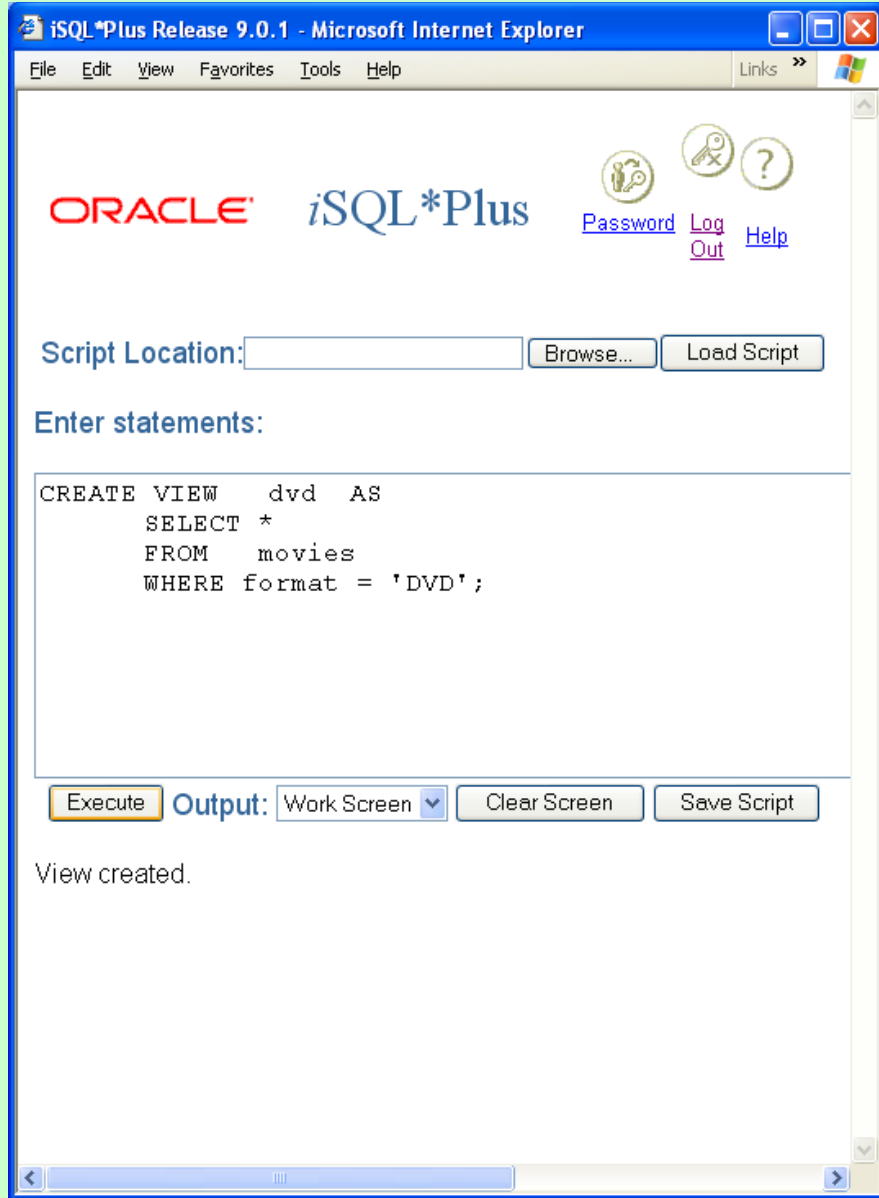
```
CREATE structure-type structure-name  
    [details about that structure]
```

When creating a view structure you specify the term VIEW, ie.

```
CREATE VIEW structure-name  
    [details about that structure]
```

Each view needs to be named, and the details for view structure is simply a SELECT statement.

Module 17: DDL



Page E-6 Problem 18-1 Design

To design a view I need a name, and a SELECT program.

```
CREATE VIEW structure-name
[SELECT program]
```

Step 1. Name the view

```
CREATE VIEW dvd AS
```

Step 2. Write the SELECT program

```
SELECT *
FROM movies
WHERE format = 'DVD';
```

Put it all together

```
CREATE VIEW dvd AS
SELECT *
FROM movies
WHERE format = 'DVD';
```

Module 17: DDL

Page E-7 Using a view

This view should behave like any other table in the database.

Seems to have the same number of titles.

The screenshot shows the iSQL*Plus web interface in a Microsoft Internet Explorer browser window. The page title is "iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The iSQL*Plus logo is displayed, along with links for Password, Log Out, and Help. There is a "Script Location:" field with a "Browse..." button and a "Load Script" button. Below this is a section titled "Enter statements:" containing two SQL queries. The first query is:
SELECT format, count (*)
FROM movies
GROUP BY format;
The second query is:
SELECT format, count (*)
FROM dvd
GROUP BY format;
Below the queries are buttons for "Execute", "Output:" (with a dropdown menu set to "Work Screen"), "Clear Screen", and "Save Script". The results of the queries are displayed in two tables. The first table, corresponding to the first query, has two columns: "FORM" and "COUNT(*)". It contains three rows: DVD with a count of 18, LD with a count of 21, and VHS with a count of 62. The second table, corresponding to the second query, also has two columns: "FORM" and "COUNT(*)". It contains one row: DVD with a count of 18.

ORACLE iSQL*Plus Password Log Out Help

Script Location: Browse... Load Script

Enter statements:

```
SELECT format, count (*)
FROM movies
GROUP BY format;

SELECT format, count (*)
FROM dvd
GROUP BY format;
```

Execute Output: Work Screen Clear Screen Save Script

FORM	COUNT(*)
DVD	18
LD	21
VHS	62

FORM	COUNT(*)
DVD	18

Module 17: DDL

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

Enter statements:

```
SELECT title, leng, format
FROM movies
WHERE leng < 85;

SELECT title, leng, format
FROM dvd
WHERE leng < 85;
```

Execute Output: Work Screen Clear Screen Save Script

TITLE	LENG	FORM
METROPOLIS	45	DVD
GODZILLA: KING OF THE MONSTERS	80	VHS
THE CRAWLING EYE	84	VHS
THE TINGLER	82	VHS
DARK STAR	83	VHS
HER ALIBI	84	VHS
RUN LOLA RUN	81	DVD

7 rows selected.

TITLE	LENG	FORM
METROPOLIS	45	DVD
RUN LOLA RUN	81	DVD

Page E-8 Using a view

iSQL*Plus Release 9.0.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help Links »

ORACLE iSQL*Plus [Password](#) [Log Out](#) [Help](#)

Script Location: Browse... Load Script

Enter statements:

```
SELECT title, genre, format
FROM movies
WHERE genre = 'SH';

SELECT title, genre, format
FROM dvd
WHERE genre = 'SH';
```

Execute Output: Work Screen Clear Screen Save Script

TITLE	GEN	FORM
HAMLET	SH	LD
HAMLET	SH	VHS
MUCH ADO ABOUT NOTHING	SH	VHS
THE TAMING OF THE SHREW	SH	VHS

no rows selected

Structure are removed from the database with the DROP command.

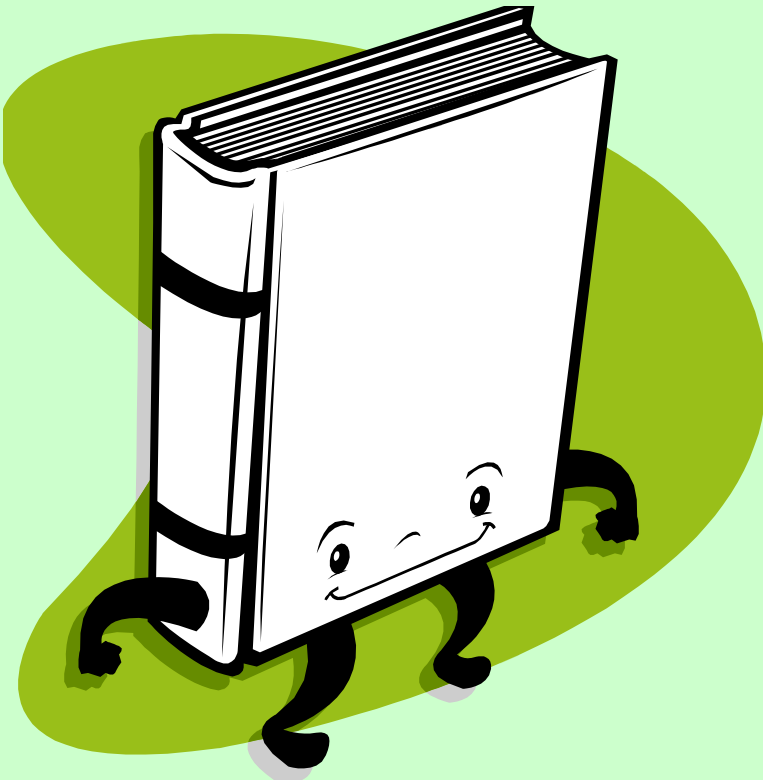
The syntax for the DROP statement is something like this:

DROP *structure-type structure-name*

CREATE
CREATE TABLE
CREATE VIEW

Datatype, domains
Numeric – NUMBER
Character – CHAR, VARCHAR2
Date – DATE

DROP
DROP TABLE
DROP VIEW



Please drop me an email if you noticed any errors in this module. I'd also appreciate reading your comments, criticisms, and or suggestions as to how this module could be improved.

Thanks,

bil



That's All