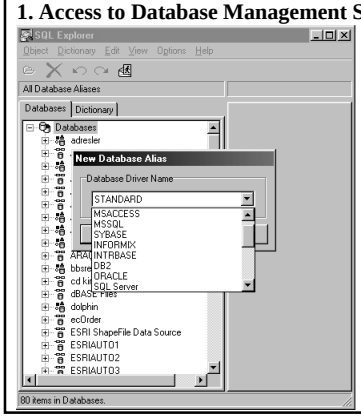
1. Access to Database Management System



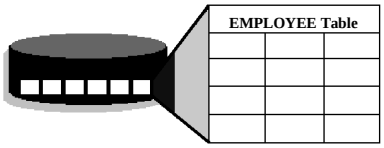
Example; Using Generic Tools

Borland SQL Explorer can be connected to any supported database management system

2. DATA TYPES

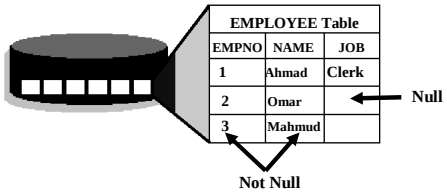
Data are held in tables in Database Management System (DBMS)
The most important entity in a Relational Database is table.

2. DATA TYPES



Data are held in tables in Database Management System (DBMS)
Tables consist of ROWS and COLUMNS.

2. DATA TYPES



Data are held in tables in Database Management System (DBMS)
Tables consist of ROWS and COLUMNS.
Each COLUMN has a DATA TYPE and may have a constraint not to have null values.

2. DATA TYPES

Null Values

***Null* values are used to represent data for which we don't always have an applicable value. They represent optional attributes in data models. Think of the column values below which can take NULL values.**

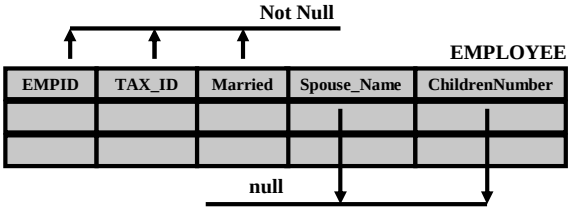
EMPLOYEE

EMPID	TAX_ID	Married	Spouse_Name	ChildrenNumber

For mandatory columns we have to have a constraint called NOT NULL.

2. DATA TYPES

Null Values



In EMPLOYEE Table, *Spouse_Name and ChildrenNumber* COLUMNS are not applicable for all EMPLOYEES. Therefore, they are nullable but other columns should take *NOT NULL* constraint.

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
MS SQL SERVER	BINARY BINARY() IMAGE VARBINARY VARBINARY()	DATETIME SMALLDATETIME TIMESTAMP	INT MONEY NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT SMALLMONEY TINYINT	CHAR CHAR() TEXT VARCHAR VARCHAR()
IBM DB2	GRAPHIC GRAPHIC() LONG VARGRAPHIC VARGRAPHIC()	DATE TIME TIMESTAMP	DECIMAL DECIMAL(,) FLOAT FLOAT() INTEGER REAL SMALLINT	CHAR CHAR() character character() LONG VARCHAR VARCHAR()

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
ORACLE	LONG LONG RAW MLSLABEL RAW MLSLABEL RAW()	DATE	DECIMAL() DECIMAL(,) FLOAT INTEGER NUMBER NUMBER(*) NUMBER(,) SMALLINT	CHAR() character() LONG VARCHAR VARCHAR() VARCHAR2()
INTERBASE	BLOB	DATE	DECIMAL DECIMAL() DECIMAL(,) DOUBLE PRECISION FLOAT INTEGER NUMERIC NUMERIC() NUMERIC(,) SMALLINT	CHAR VARCHAR()

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
INGRES	BYTE VARYING LONG BYTE	DATE	BYTE DECIMAL FLOAT FLOAT() FLOAT4 FLOAT8 INTEGER INTEGER1 INTEGER2 INTEGER4 MONEY SMALLINT	C CHAR() LONG VARCHAR TEXT() VARCHAR()
MS ACCESS	OLE OBJECT	DATE/TIME	AUTONUMBER BYTE CURRENCY DOUBLE INTEGER LONG INTEGER REPLICATION ID SINGLE YES/NO	MEMO TEXT()

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
PARADOX	BINARY BINARY() GRAPHIC GRAPHIC() OLE OLE()	DATE TIME TIMESTAMP	AUTOINCREMENT BCD BCD() BYTES() LOGICAL LONG INTEGER MONEY NUMBER SHORT	ALPHA() FORMATTED MEMO FORMATTED MEMO() MEMO()
SYBASE	BINARY() IMAGE VARBINARY()	DATE/TIME SMALLDATETIME TIMESTAMP	BIT DECIMAL DECIMAL() DECIMAL(,) FLOAT INT MONEY NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT SMALLMONEY TINYINT	CHAR() TEXT VARCHAR()

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
INFORMIX	BYTE	DATE	DEC DEC(,) DECIMAL DECIMAL(,) DOUBLE PRECISION DOUBLE PRECISION() FLOAT FLOAT() INT INTEGER MONEY MONEY(,) NUMERIC NUMERIC(,) REAL SERIAL SERIAL() SERIAL(1) SMALLFLOAT SMALLINT	CHAR CHAR() character character() NCHAR NCHAR() NVARCHAR() TEXT VARCHAR()

2. DATA TYPES

Data Base Name	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMBER DATA TYPE	STRING DATA TYPE
PROGRESS		DATE	DECIMAL DECIMAL() DECIMAL(,) FLOAT FLOAT() INTEGER LOGICAL NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT	CHAR() character()
FOXPRO ve DBASE IV	MEMO	DATE	FLOAT(,) LOGICAL NUMERIC(,)	character()

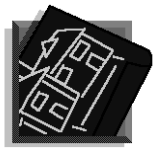
3. Data Definition Language Statements: CREATE TABLE

Table structure is defined according to specific data types available under a certain DBMS

3. Data Definition Language Statements: CREATE TABLE

There are 2 steps to construct a table.

1. DESIGN **2. DEFINITION**



3. Data Definition Language Statements: CREATE TABLE

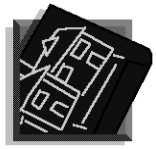
Steps in DESIGN Phase;

1. **Define Data Elements (Entities) to trace (Construct Data Model)**
2. **Define the Columns under Tables according to attributes of entities**
3. **Name the COLUMNS in Tables. Decide DATA TYPE in detail**
4. **Decide which COLUMN is *null* and which COLUMN has *not null* constraint. Mandatory attributes in data model become not null columns.**
5. **Define *Primary key Column(s)* and *Foreign key Column(s)***

3. Data Definition Language Statements: CREATE TABLE

Following steps are for definition Stage

1. **Create Table**
2. **Alter the Table for defining primary and/or foreign keys**



3. Data Definition Language Statements: CREATE TABLE

A SQL Statement to create a table must contain

- **Table Name**
- **COLUMN Name**
- **COLUMN DATA TYPE**

Now let's look at the rules to name tables;

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

1. **Depending on the DBMS, do not use more than some specific number of characters when naming the Table (For example; Table name should have a value of 1-30 character for ORACLE and 1-31 for INTERBASE respectively)**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

1. **Depending on the DBMS, do not use more than some specific number of characters when naming the Table (For example; Table name should have a value of 1-30 character for ORACLE and 1-31 for INTERBASE respectively)**
2. **The first character in Table name should be alphabetic one.**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

1. **Depending on the DBMS, do not use more than some specific number of characters when naming the Table (For example; Table name should have a value of 1-30 character for ORACLE and 1-31 for INTERBASE respectively)**
2. **The first character in Table name should be alphabetic one.**
3. **Table name should consist of following characters. (Note: Although DBMSs like MS Access, Paradox allow using special characters it is better to avoid them for transferability)**
   - **A-Z, a-z (in English Alphabet), 0-9,_(underscore)**
   - **$ and # is valid for Oracle but not suggested**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

4. **Do not use a reserved word in DBMS. (For example; you cannot name a Table column as foreign)**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

4. **Do not use a reserved word in DBMS. (For example; you cannot name a Table name as Table)**
5. **You cannot use a word that you defined before (For example; if you defined a Table named Employees before, you cannot create a table with the same name)**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table (specific to Oracle but many applied to other DBMS)

4. **Do not use a reserved word in DBMS. (For example; you cannot name a Table column as foreign)**
5. **You cannot use a word that you defined before (For example; if you defined a Table named Employees before, you cannot create a table with the same name)**

Most DBMSs are not case-sensitive, i.e. EMPLOYEE and EmpLoYeE are same thing.

3. Data Definition Language Statements: CREATE TABLE

Which of the followings is not a valid table name?

a . CUSTOMER ORDERS

b . CUSTOMER_ORDERS

c . ORDERS

d . CUST_ORDERS

3. Data Definition Language Statements: CREATE TABLE

Now Rules for naming COLUMNS:

Table Name

COLUMN Name

COLUMN DATA TYPE

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table Column
(specific to Oracle but many applied to other DBMS)

1. **Depending the DBMS, the number of characters are limited for naming the COLUMNS. (For Example; this number is 1-30 for ORACLE and 1-31 for INTERBASE respectively.**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table Column
(specific to Oracle but many applied to other DBMS)

1. **Depending the DBMS, the number of characters are limited for naming the COLUMNS. (For Example; this number is 1-30 for ORACLE and 1-31 for INTERBASE respectively.**
2. **The first character in COLUMN Name should be alphabetic one**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table Column
(specific to Oracle but many applied to other DBMS)

1. **Depending the DBMS, the number of characters are limited for naming the COLUMNS. (For Example; this number is 1-30 for ORACLE and 1-31 for INTERBASE respectively.**
2. **The first character in COLUMN Name should be alphabetic one**
3. **It should only contain following characters. (Some DBMSs such as MS Access, Paradox accept special characters but avoid using special characters for transferability.**

- **A-Z, a-z (in English Alphabet), 0-9,_(underscore)**
- **$ and # signs are valid for Oracle but avoid for transferability**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table Column
(specific to Oracle but many applied to other DBMS)

4. **You cannot use certain reserved words in DBMS for naming the columns (For example; you cannot have a column named *foreign*)**

3. Data Definition Language Statements: CREATE TABLE

Rules for Naming a Table Column
(specific to Oracle but many applied to other DBMS)

4. **You cannot use certain reserved words in DBMS for naming the columns (For example; you cannot have a column named *foreign*)**
5. **Under the same table you cannot have two different columns with the same name. (You can have same column names under different tables)**

3. Data Definition Language Statements: CREATE TABLE

Tablo Name

COLUMN Name

COLUMN DATA TYPE

The last thing to do is to decide COLUMN DATA TYPES

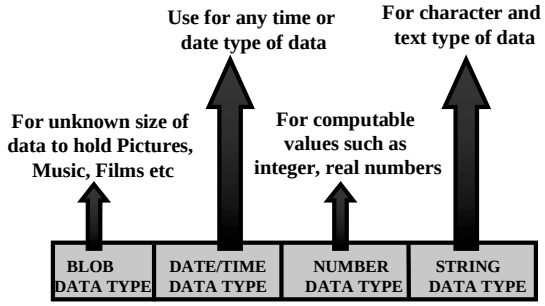
3. Data Definition Language Statements: CREATE TABLE

Be careful in defining COLUMN DATA TYPES.

For Example; Under ORACLE if you choose Number as data type, you can make calculations based on that column but not with data type called RAW.

3. Data Definition Language Statements: CREATE TABLE

USE OF DATA TYPES



3. Data Definition Language Statements: CREATE TABLE

USE OF DATA TYPES (EXAMPLE; ORACLE)

STRING DATA TYPE: `CHAR()` and `VARCHAR()`

For example; If you define a column as `CHAR(10)` ;

'Jeddah'
'United_Sta'
'UNITED_STA' are examples of data.

Queries are case sensitive for string data types, i.e. 'London' and 'LONDON' are two different things.

3. Data Definition Language Statements: CREATE TABLE

USE OF DATA TYPES (EXAMPLE; ORACLE)

STRING DATA TYPE: `CHAR()` and `VARCHAR()`

Difference between `VARCHAR(10)` and `CHAR(10)`

If you use CHAR(10) defining a column, it reserves a place of 10 characters in DBMS even if you insert less characters. If you enter value of 'Jeddah', DBMS stores it as 'Jeddah '.

VARCHAR(10) is more economical to use.

3. Data Definition Language Statements: CREATE TABLE

USE OF DATA TYPES (EXAMPLE; ORACLE)

BLOB DATA TYPE: `LONG`

This data type holds data up to 2 *gigabytes*. You can store text with more than 255 characters or pictures.
For each table, you can only have one LONG Data Type.

3. Data Definition Language Statements: CREATE TABLE

USE OF DATA TYPES (EXAMPLE; ORACLE)

NUMBER DATA TYPE: `NUMBER(,)`

`NUMBER(,)` DATA TYPE is used to hold numbers. For example; By defining a column as `NUMBER(7,2)` you can hold `12345.67`. If you enter a value of 12345.678, it will be rounded to 12345.68.

3. Data Definition Language Statements: CREATE TABLE

Now you can use CREATE TABLE Statement. This statement consists of 5 things.

1. **Define Table Name**
2. **Define number and names of columns**
3. **Define data type for each column**
4. **Define for each column whether we allow null values or not.**
5. **Define other constraints for each column (primary key or foreign key constraints)**

3. Data Definition Language Statements: CREATE TABLE

CREATE TABLE *syntax is as follows:*

```
CREATE TABLE [User_Name.]table_name
(Column_Name Data_Type [column_constraint],
 Column_Name Data_Type [column_constraint],
              .....                  )
```


3. Data Definition Language Statements: CREATE TABLE

Following statement defines a table named EMPLOYEE .

```
CREATE TABLE EMPLOYEE
(EMP_ID              INTEGER     NOT NULL,
 NAME                VARCHAR(19) NOT NULL,
 SURNAME             VARCHAR(19) NOT NULL,
 IS_MARRIED          VARCHAR(1)  NOT NULL,
 NUMBER_OF_CHILDREN  SMALLINT    );
```

COLUMN NAMES **DATA TYPES** **NULL/NOT NULL**

3. Data Definition Language Statements: CREATE TABLE

Maximum number of columns is 255 for ORACLE DBMS. Usually we don't need that much if we have a proper relational design)

```
CREATE TABLE EMPLOYEE
(EMP_ID              INTEGER     NOT NULL,
 NAME                VARCHAR(19) NOT NULL,
 SURNAME             VARCHAR(19) NOT NULL,
 IS_MARRIED          VARCHAR(1)  NOT NULL,
 NUMBER_OF_CHILDREN  SMALLINT    );
```

COLUMN NAMES **DATA TYPES** **NULL/NOT NULL**

3. Data Definition Language Statements: CREATE TABLE

Second column is for data type of each column.

```
CREATE TABLE EMPLOYEE
(EMP_ID              INTEGER     NOT NULL,
 NAME                VARCHAR(19) NOT NULL,
 SURNAME             VARCHAR(19) NOT NULL,
 IS_MARRIED          VARCHAR(1)  NOT NULL,
 NUMBER_OF_CHILDREN  SMALLINT    );
```

COLUMN NAMES **DATA TYPES** **NULL/NOT NULL**

3. Data Definition Language Statements: CREATE TABLE

Third column shows whether the column is allowed to accept Null values. If the column is required then place NOT NULL.

```
CREATE TABLE EMPLOYEE
(EMP_ID              INTEGER     NOT NULL,
 NAME                VARCHAR(19) NOT NULL,
 SURNAME             VARCHAR(19) NOT NULL,
 IS_MARRIED          VARCHAR(1)  NOT NULL,
 NUMBER_OF_CHILDREN  SMALLINT    );
```

COLUMN NAMES **DATA TYPES** **NULL/NOT NULL**

3. Data Definition Language Statements: CREATE TABLE

What is the DATA TYPE for DEPT_ID?

```
CREATE TABLE DEPARTMENT
(DEPT_ID        INTEGER     NOT NULL,
 NAME           VARCHAR(32) NOT NULL,
 LOCATION       NUMBER(2),
 HEAD_DEPT_ID   INTEGER );
```

a. VARCHAR(32)
b. NOT NULL
c. NUMBER(2)
d. INTEGER

3. Data Definition Language Statements: CREATE TABLE

COLUMN CONSTRAINTS

You can place one or more constraint on the same column. In the following example, the column *DEPT_ID* takes two constraints (*not null* constraint and *primary key* constraint).

```
CREATE TABLE DEPARTMENT
(DEPT_ID    INTEGER NOT NULL CONSTRAINT pk_DEPT_ID PRIMARY KEY,
 NAME       VARCHAR(32) NOT NULL,
 LOCATION   NUMERIC(2));
```


3. Data Definition Language Statements: CREATE TABLE

COLUMN CONSTRAINTS

You can also write constraints at the end.

```
CREATE TABLE DEPARTMENT
(DEPT_ID    INTEGER NOT NULL CONSTRAINT pk_DEPT_ID PRIMARY KEY,
 NAME       VARCHAR(32) NOT NULL,
 LOCATION   NUMERIC(2));
```

```
CREATE TABLE DEPARTMENT
(DEPT_ID    INTEGER NOT NULL,
 NAME       VARCHAR(32) NOT NULL,
 LOCATION   NUMERIC(2),
 CONSTRAINT pk_DEPT_ID PRIMARY KEY);
```

Therefore, the statements above give the same result.

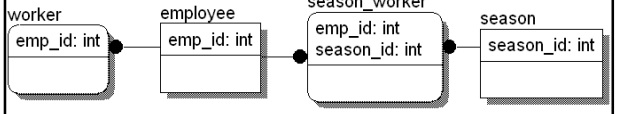
3. Data Definition Language Statements: EXAMPLE

```
create table employee (emp_no int not null constraint pk_employee primary key);

create table worker    (emp_no int not null constraint pk_worker primary key
                                constraint fk_worker references employee);

create table season(season_id int not null constraint pk_season primary key);

create table season_worker (emp_no int not null,
   season_id int not null,
   constraint pk_season_worker primary key (emp_no,season_id),
   constraint fk_season_worker1 foreign key(season_id) references season(season_id),
   constraint fk_season_worker2 foreign key(emp_no) references employee(emp_no)
               );
```



3. Data Definition Language Statements: CREATE TABLE

COPYING A TABLE UNDER ORACLE

If you have an existing table, you can create another table based on that one by using CREATE TABLE AS under ORACLE. This is not available in all DBMS.

Note: It does not copy primary key or foreign key constraints.

The syntax for CREATE TABLE AS is given below;

```
CREATE TABLE table_name[(COLUMN_Name,COLUMN_Name...)]
AS
  [Query]
```

3. Data Definition Language Statements: CREATE TABLE AS

Following CREATE TABLE AS statement creates a table called HDATES. New table contains three columns based on the query. They are EMPNO,ENAME and HIREDATE. New table includes three columns in all rows.

```
SQL> CREATE TABLE HDATES
2    AS
3         SELECT EMPNO,ENAME,HIREDATE
4         FROM EMP;
```


3. Data Definition Language Statements: CREATE TABLE AS

HDATES contains all data in EMP table.

```
SQL> CREATE TABLE HDATES
2    AS
3         SELECT *
4         FROM EMP;
```


3. Data Definition Language Statements: CREATE TABLE AS

What happens in the following example?

```
SQL> CREATE TABLE BLANK_EMP
2    AS
3         SELECT * FROM EMP
4         WHERE 2=3;
```

BLANK_EMP table is created with all columns as in EMP. But it contains no data from EMP because 2=3 condition is not met.

3. Data Definition Language Statements: ALTER TABLE

After using CREATE TABLE statements, you can change the structure of your database by using ALTER TABLE ... Statements.

3. Data Definition Language Statements: ALTER TABLE

Using ALTER TABLE ... statements you can make changes in table structures as follows:

- **Adding new a column or constraint. For example, adding a *NOT NULL* or *foreign key* constraint.**
- **Changing an existing column definition. You can change data type or constraint for an existing column.**

3. Data Definition Language Statements: ALTER TABLE

Using ALTER TABLE ... statements you can make changes in table structures as follows:

- **Adding new a column or constraint. For example, adding a *NOT NULL* or *foreign key* constraint.**
- **Changing an existing column definition. You can change data type or constraint for an existing column.**

Who can change a Table structure?

Owner of the table or the persons allowed by the owner can change the structure of a table using ALTER TABLE statements.

Also DBA has got the privilege of access and change to any table.

3. Data Definition Language Statements: ALTER TABLE

True or False?

ANYONE CAN CHANGE A TABLE.

TRUE | FALSE

3. Data Definition Language Statements: ALTER TABLE

Adding a New COLUMN to a Table

The first change to a table is to add a column and constraint. For this purpose, you can use ALTER TABLE statement as follows:

```
ALTER TABLE Table_Name
  ADD ( {COLUMN_Name data_type | COLUMN_constraint},
        [{COLUMN_Name data_type | COLUMN_constraint}]..)
```

ORACLE

```
ALTER TABLE Table_Name
  ADD {COLUMN_Name data_type | COLUMN_constraint},
 [ADD {COLUMN_Name data_type | COLUMN_constraint}] ..
```

InterBase

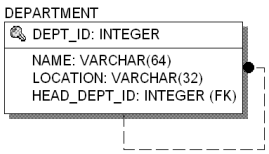
3. Data Definition Language Statements: ALTER TABLE

By using ALTER TABLE ... Statement, you can change an existing column.

```
CREATE TABLE DEPARTMENT (
       DEPT_ID       INTEGER      NOT NULL PRIMARY KEY,
       NAME          VARCHAR(64)  NOT NULL,
       LOCATION      VARCHAR(32),
       HEAD_DEPT_ID  INTEGER      NOT NULL );
```

First construct a CREATE TABLE statement to convert the data model into design. The name of table is *DEPARTMENT*.

***HEAD_DEPT_ID* is not referenced yet as *foreign key*.**

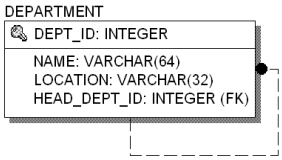


3. Data Definition Language Statements: ALTER TABLE

ALTER TABLE Table_Name ADD FOREIGN KEY...

```
ALTER TABLE DEPARTMENT
       ADD FOREIGN KEY (HEAD_DEPT_ID)
                       REFERENCES DEPARTMENT;
```

For recursive relationship, *HEAD_DEPT_ID* is referenced from the same table.



3. Data Definition Language Statements: ALTER TABLE

ALTER TABLE Table_Name
ADD COLUMN_Name data_type constraint;

```
ALTER TABLE DEPARTMENT
       ADD NUMBER_OF_POSITIONS INTEGER;
```

We add a column named *NUMBER_OF_POSITIONS*

3. Data Definition Language Statements: ALTER TABLE

You can add many columns at the same time.
For example;

```
ALTER TABLE DEPARTMENT
       ADD NUMBER_OF_POSITIONS INTEGER,
       ADD MANAGER INTEGER NOT NULL;
```

3. Data Definition Language Statements: ALTER TABLE

CHANGING COLUMN DEFINITIONS

Using ALTER TABLE statement, you can change column structure and its constraints

Examples;

- **You can increase the width of a string column**
- **You can reduce the width of a string column**

3. Data Definition Language Statements: ALTER TABLE

Changing COLUMN Definitions

Syntax for changing COLUMN definitions;

```
ALTER TABLE Table_Name
 MODIFY ({COLUMN_Name data_type | COLUMN_constraint}
        [{COLUMN_Name data_type | COLUMN_constraint}] ..)
```

ORACLE

MODIFY is available under ORACLE. For most of the DBMSs, this choice is not available. For other DBMSs, you can first delete column using drop choice. And later use add choice to add column. Do not forget to get a backup of column before deleting.

3. Data Definition Language Statements: ALTER TABLE

ALTER TABLE Table_Name DROP COLUMN_Name

```
ALTER TABLE DEPARTMENT DROP NUMBER_OF_POSITIONS;
```

InterBase

Drop choice is used to delete a column in a table (ALTER TABLE ... DROP statement)

3. Data Definition Language Statements: DROP TABLE

DROP TABLE Statement

DROP TABLE statement deletes a table from the database permanently.

3. Data Definition Language Statements: DROP TABLE

DROP TABLE Statement

Syntax is very simple but the result is

```
DROP TABLE Table_Name
```

Following statement deletes the table EMPLOYEE from database permanently.

```
DROP TABLE EMPLOYEE
```


3. Data Definition Language Statements: DROP TABLE

Do not Forget!

DROP TABLE Table_Name statement will delete not only the contents of a table but also the structure of table

SQL Data Definition Language

3. Data Definition Language Statements

OTHER DATA DEFINITION LANGUAGE STATEMENTS

DEFINITON STATEMENTS	RESTRUCTURING STATEMENTS	DROP STATEMENTS
CREATE DATABASE CREATE DOMAIN CREATE EXCEPTION CREATE GENERATOR CREATE INDEX CREATE PROCEDURE CREATE ROLE CREATE SHADOW CREATE TABLE CREATE TRIGGER CREATE VIEW	ALTER DATABASE ALTER DOMAIN ALTER EXCEPTION ALTER INDEX ALTER PROCEDURE ALTER TABLE ALTER TRIGGER	DROP DATABASE DROP DOMAIN DROP EXCEPTION DROP EXTERNAL DROP FILTER DROP INDEX DROP PROCEDURE DROP ROLE DROP SHADOW DROP TABLE DROP TRIGGER DROP VIEW

NOTE: Some statements above are for INTERBASE DBMS

cetinerg@itu.edu.tr

Assoc.Prof.Dr.B.Gültekin Çetiner ©2000