Introduction to DBMS

Lecture Objectives

- Some common Definitions of database systems.
- Characteristics of file-based systems.
- Problems with file-based approach.
- Database Management System (DBMS).
- Relational Database Management System (RDBMS)

Definitions

- ***Data***: stored representations of **meaningful objects** and **events** or
- Referred to facts concerning objects and events that could be recorded and **stored** on computer media
  - ***Structured*:** numbers, text, dates
  - ***Unstructured***: images, video, documents
- ***Information***: data **processed** to increase **knowledge** in the person using the data
- ***Metadata:*** data that describes the **properties** and **context** of user data

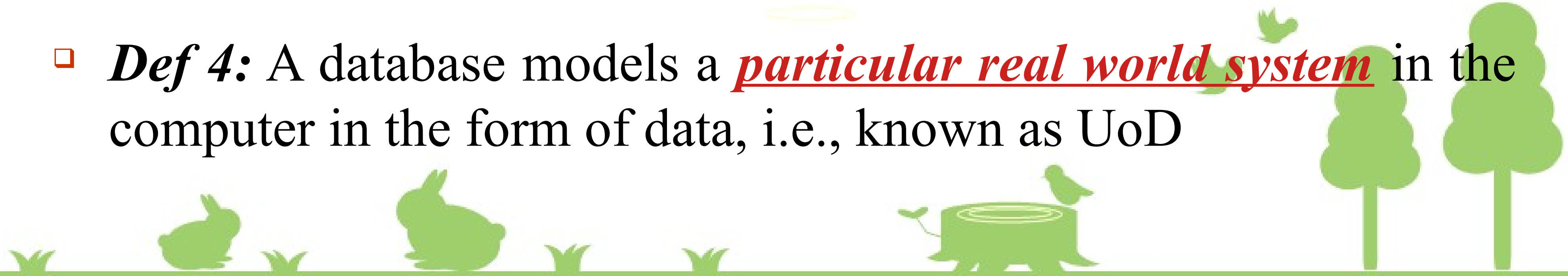
Definitions of *Database*

- ***Def 1:*** Database is an ***organized collection*** of ***logically related data***

- ***Def 2:*** A database is a ***shared*** collection of ***logically related data*** that is stored to meet the requirements of ***different users*** of an organization

- ***Def 3:*** A database is a ***self-describing*** collection of integrated records

- ***Def 4:*** A database models a ***particular real world system*** in the computer in the form of data, i.e., known as UoD



Examples of Database Applications

- Library catalogues
- Medical records
- Bank accounts
- Stock control
- Product catalogues
- Telephone directories
- Train timetables
- Airline bookings
- Credit card details
- Student records
- Customer histories
- Stock market prices
- and **so on…**

A bit of History

- Computer initially used for computational/ engineering purposes
- Commercial applications introduced **File Processing System**

Flat-File Database

- A flat file database is a database that stores data in a **plain text file** or a **binary file**
- Each line of the text file holds one record. fields separated by delimiters, such as commas or tabs.

Characteristics of a Flat file database:

-- Simple structure

-- Cannot contain **multiple tables** like a relational database does

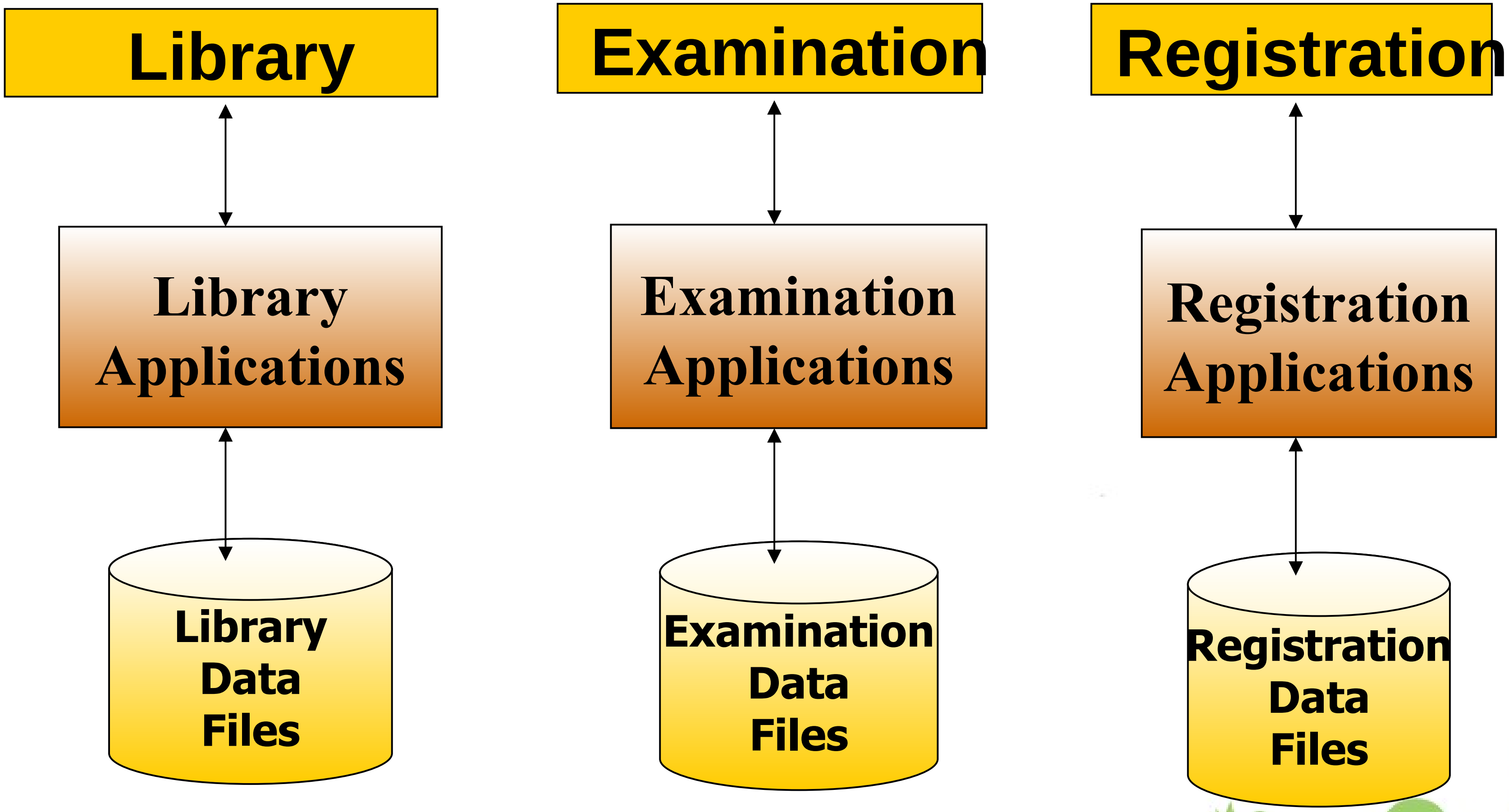
-- Records follow a **uniform format**, with fields **separated** by delimiters.

-- **No** structure for **indexing** or recognizing relationships between records.

-- System becomes increasingly inefficient as more data is added.

-- Some common **flat file formats**: XML, **CSV** or comma-delimited files often representing **spreadsheets**.

File Processing Systems



Library

Library Applications

Library Data Files

Examination

Examination Applications

Examination Data Files

Registration

Registration Applications

Registration Data Files

Fig: Program and Data Interdependence

File Processing Systems

Library
Reg_Number
Name
Father’s Name
Books Issued
Fine

Examination
Reg_Number
Name
Address
Class
Semester
Grade

Registration
Reg_Number
Name
Father’s Name
Phone
Address
Class

Disadvantages of File Processing

- **Program-Data Dependence**
  - File structure is defined in the program code.
  - All programs maintain metadata for each file they use
- **Duplication of Data (Data Redundancy)**
  - Different systems/programs have separate copies of the same data
- **Limited Data Sharing**
  - No centralized control of data
  - Application programs are written in different languages, and so cannot easily access other's files.

Data Inconsistency

When data changes in one file but not reflected in other files.

Compromises Data Integrity ***(Data Reliability)***

SOLUTION:
Database System Approach

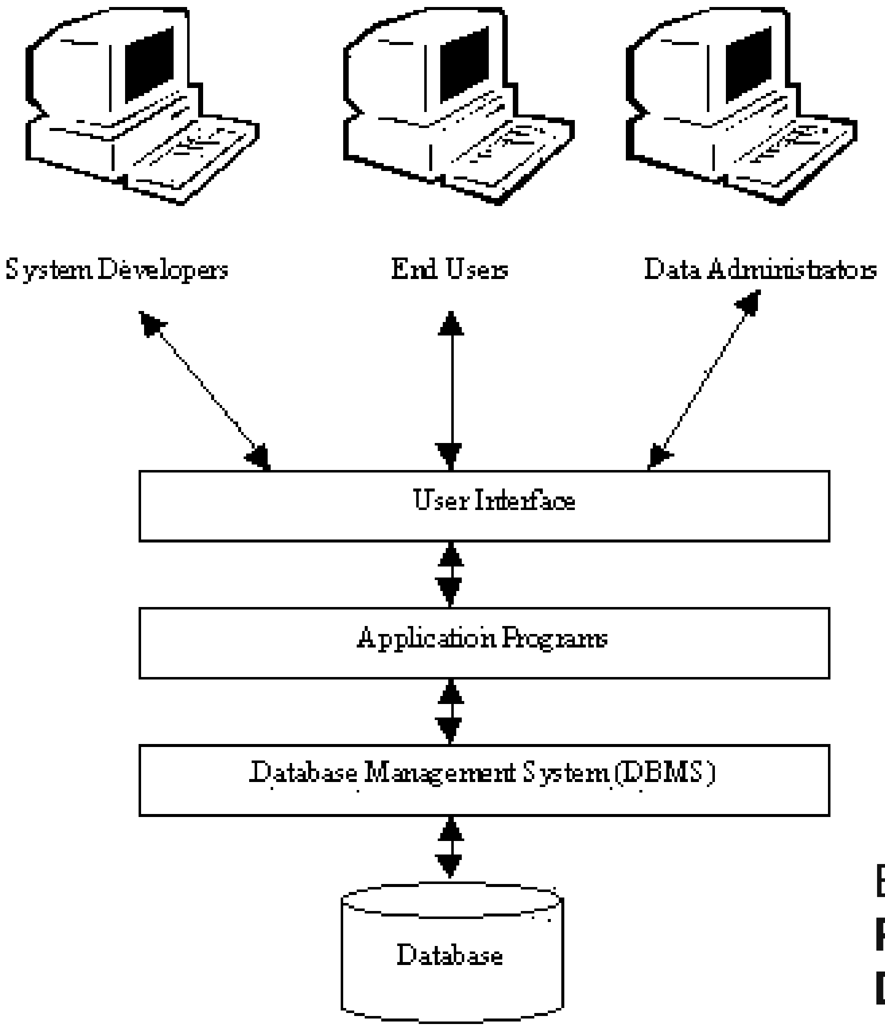
Integrated: Distinct data files have been logically organised to eliminate or reduce **redundancy** and to facilitate **data access**.

Shared: All qualified **users** in the organisation have **access** to the same data for use in a variety of activities.

Interrelated: Structured in a manner that is **logically meaningful** to the organisation.

This ***requires*** a
Database and ***D***ata***b***ase ***M***anagement ***S***ystem (DBMS)

Database System components



A database system components:

--***Data (the database)***

Schema, Data catalogue

--***Software***

DBMS, Application Programs

--***Hardware***

Client-server architecture

--***Users***

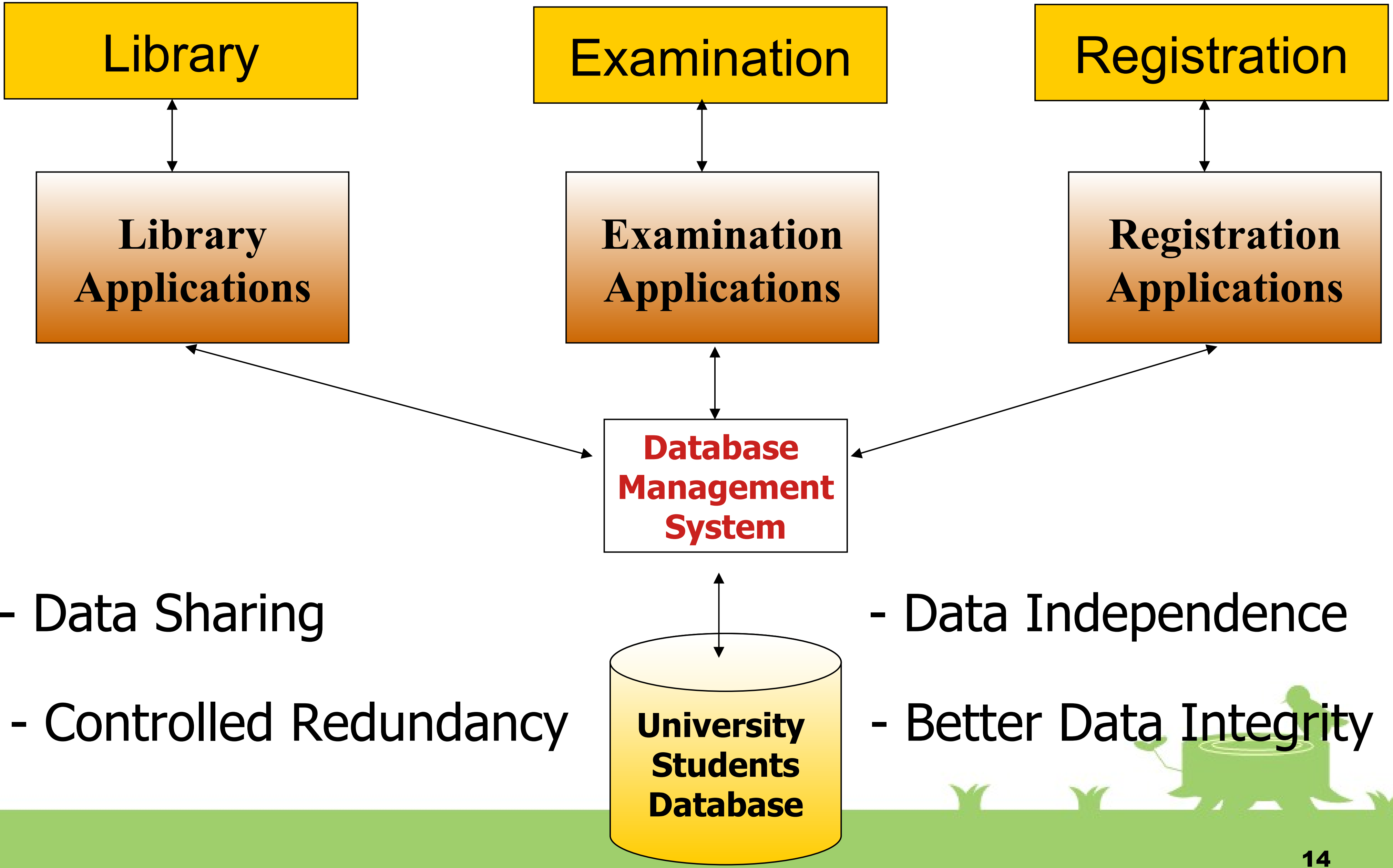
Database administrator, Database designer, Application programmer

End-user: naive & sophisticated

E.g., **MySQL**, **Ms Access**, **Ms SQL Server**, **FileMaker Pro**, **Oracle Database**, **dBASE, DB2 etc.,.**

Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a ***Database system***

Advantages of Database Approach


Library
Examination
Registration
Library Applications
Examination Applications
Registration Applications
Database Management System
University Students Database

- Data Sharing

- Controlled Redundancy

- Data Independence

- Better Data Integrity

History of Database Systems

First generation *(in 1960s)*

--Hierarchical model
--Network model

Limitation:
– Complex program for simple query
– Minimum data independence

Second generation *(in 1980s)*

--Relational model (E. F. Codd)
DB2, Oracle, MySQL etc.,

Limitation:
--Limited data modeling

Third generation *(in 1990s)*

--Object-relational DBMS
--Object-oriented DBMS

Next generation *(in late 2000s)*

--NoSQL Database

MongoDB

--Cloud DBMS

DBaaS (Database as a ***Service***)

Relational DBMS (*RDBMS*)

Relational databases became dominant in the 1980s.

Pioneer of Relational Database Model: **E.F. Codd**

Items in a relational database are organized as a set of **tables (two-dimensional)** with columns and rows.

Relational database technology provides the **most efficient** and **flexible way** to access **structured information**.

Follows **Codd's Golden Rules** (**13 rules**)

Features of RDBMS Systems

-- All data stored are in the form of **two-dimensional Table**

-- Facilitates **primary key** for **unique** identification of the rows

-- Facilitates a **common column** to be shared amid two or **more tables**

-- **Integrity constraints** maintain data consistency across multiple tables.

-- **Multi-user accessibility** facilitated to be controlled by individual users

-- **Index creation** for **retrieving data** at a **higher speed**

-- **Column** (**attributes**) values are **atomic**.

-- All of the **values** in a **column** have the same **data type**.

-- Each **column** has a **unique name**.

-- Each **row** (**tuple**) is **unique**.

-- The **sequence** of **rows and columns** is **insignificant**.

Thank You

--End of *1st Session*--

SQL (Structured Query Language)

--*Commands*--

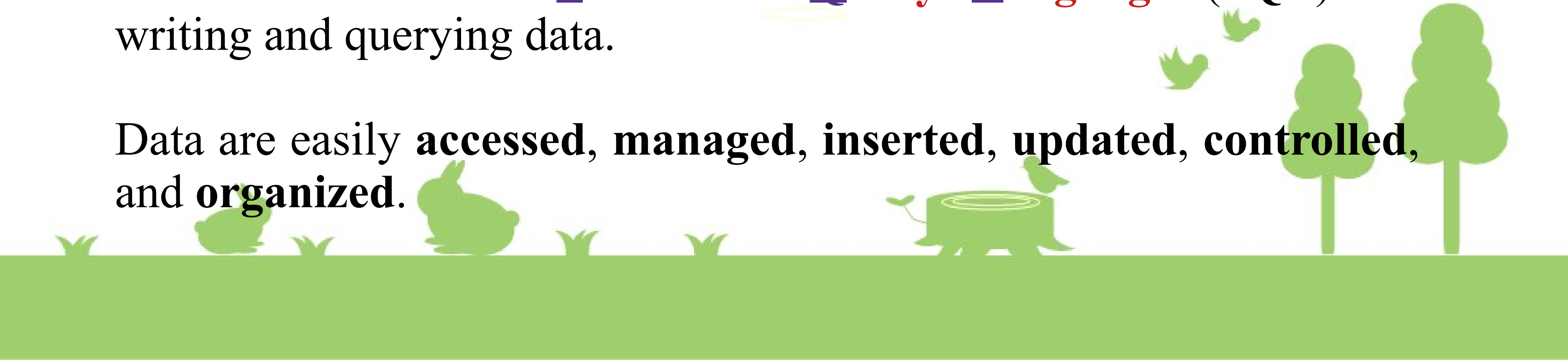
SQL: Basics

Both **ANSI (American National Standards Institute)** and the **ISO/IEC** have accepted SQL as the standard language for **Relational Databases**.

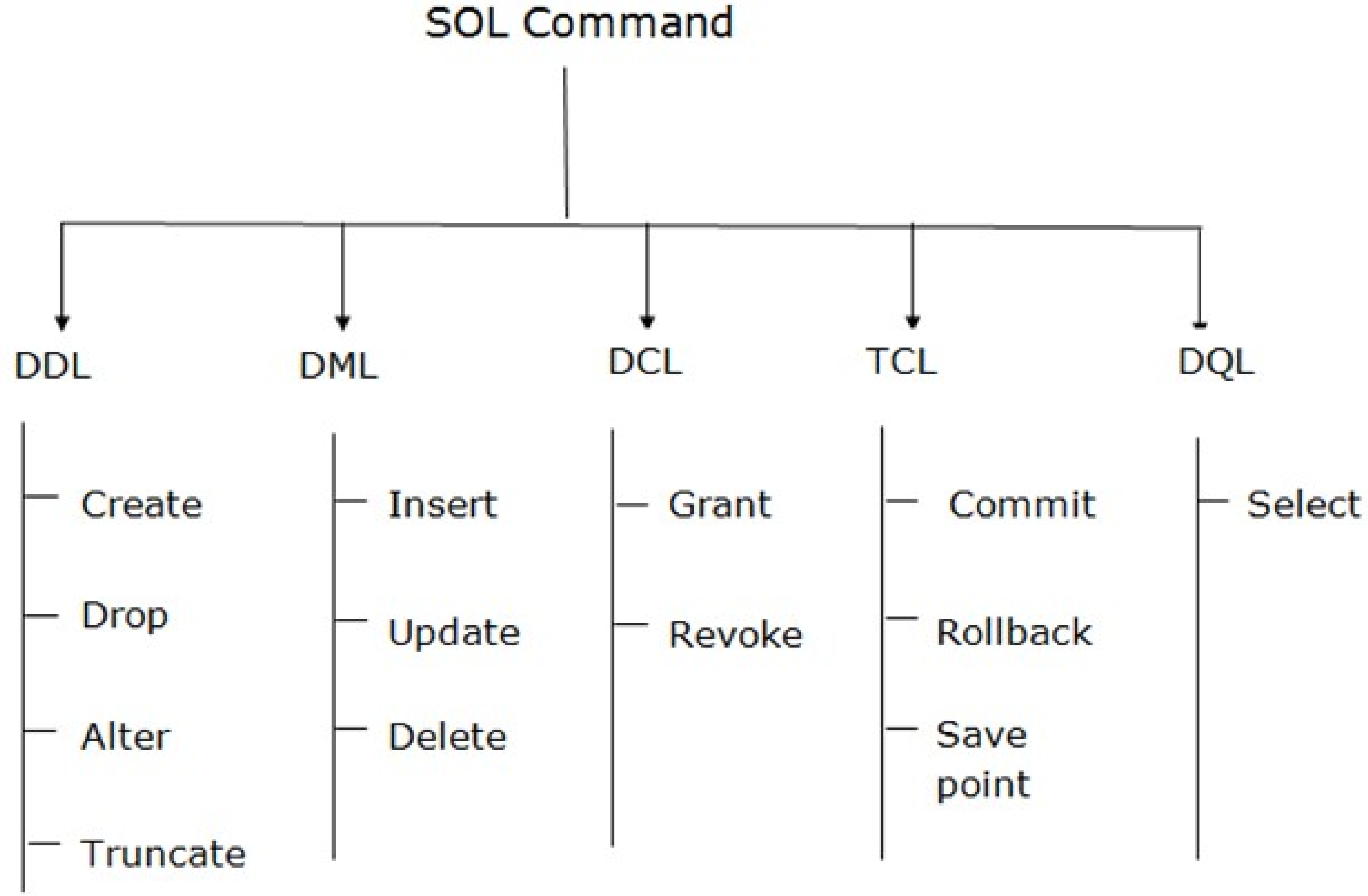
Data in the most common types of databases in operation today is typically **modelled** in **rows** and **columns** in the tables to make processing and **data querying efficient.**

Most databases use **Structured Query Language** **(SQL)** for writing and querying data.

Data are easily **accessed**, **managed**, **inserted**, **updated**, **controlled**, and **organized**.



SQL: Languages



SOL Command

- DDL
  - Create
  - Drop
  - Alter
  - Truncate
- DML
  - Insert
  - Update
  - Delete
- DCL
  - Grant
  - Revoke
- TCL
  - Commit
  - Rollback
  - Save point
- DQL
  - Select

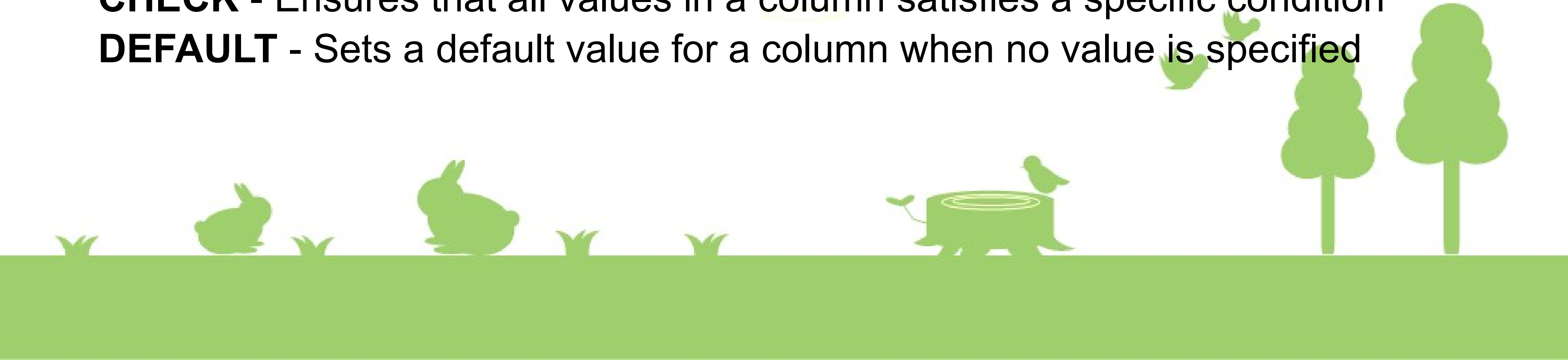
SQL: CONSTRAINTS

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified



------------DDL-----------

SQL *CREATE* TABLE

SQL: DDL

To create a table, you need to define three things:

- **Its name**
- **Its columns**
- **Data types of these columns**

Syntax:

```
CREATE table <table_name> (
  <column1_name> <data_type>,
  <column2_name> <data_type>,
  <column3_name> <data_type>,
  ... );
```

Data Types

	Data Type	Description	MAX Size
Character (alphanumeric values, or strings)	**CHAR(size)**	Fixed length character	2000 bytes. Min: 1 byte
	VARCHAR2(size)	Variable length character string	4000 bytes. Min: 1 byte
Numeric	**NUMBER(p,s)**	Numeric data, with a precision of p and scale of s.	Can store up to **38 digits**
DATE	**DATE**	A date value	Dec 31 9999 AD

Number (3,1), allows ***44.5*** to be stores exactly, but not 444.5 or 0.32.

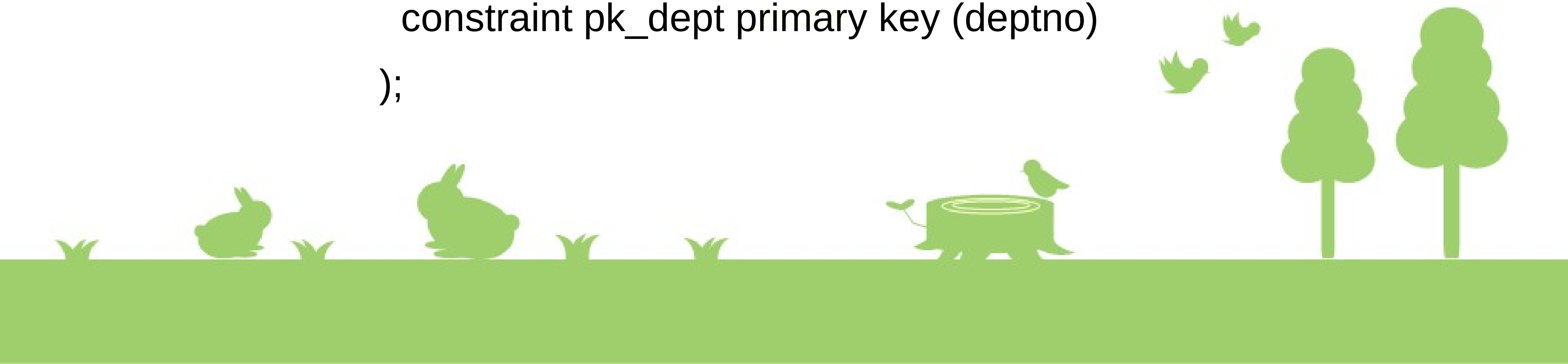
Default **DATE** format **`DD-MON-YY’**. E.g., 12th October 2016: ***’12-OCT-16'***

SQL: DDL

Create DEPT table which will be the parent table of the EMP table.

SCHEMA: DEPT(deptno, dname, loc)

```
CREATE TABLE DEPT(
  deptno     number(2),
  dname      varchar2(14),
  loc        varchar2(13),
  constraint pk_dept primary key (deptno)
);
```



SQL: DDL

Create the EMP table which has a foreign key reference to the DEPT table.

SCHEMA: EMP**(empno, ename, job, mgr, hiredate, sal, comm, deptno)**

```
CREATE TABLE EMP(
  empno    number(4),
  ename    varchar2(10),
  job      varchar2(9),
  mgr      number(4) constraint fk_mgr references EMP(empno),
  hiredate date,
  sal      number(7,2),
  comm     number(7,2),
  deptno   number(2),
  constraint pk_emp primary key (empno),
  constraint fk_deptno foreign key (deptno) references DEPT (deptno)
);
```

------------DDL-----------

SQL *ALTER* TABLE

SQL: DDL

- **SQL ALTER TABLE Statement**
- The ALTER TABLE statement is used to **add**, **delete**, **modify or rename columns** in an existing table.
- The ALTER TABLE statement is also used to **add** and **drop** various **constraints** on an existing table.

SQL: DDL (Contd.)

ALTER TABLE - ADD Column

%To add a column in a table

Syntax:

ALTER TABLE table_name **ADD** column_name datatype;

Example:

ALTER **TABLE** EMP **ADD** Phone number(10);

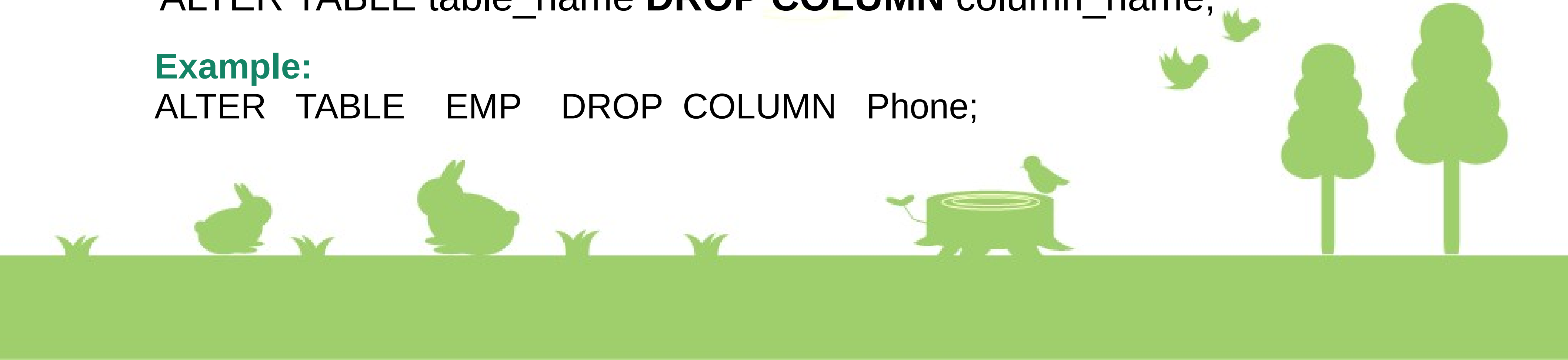
ALTER TABLE - DROP Column

Syntax:

ALTER TABLE table_name **DROP COLUMN** column_name;

Example:

ALTER TABLE EMP DROP COLUMN Phone;



SQL: DDL (Contd.)

ALTER TABLE - MODIFY Column

%To modify a column in a table

Syntax:

ALTER TABLE table_name **MODIFY** column_name datatype;

Example:

ALTER **TABLE** EMP ***MODIFY*** comm number(5,2); **(Oracle 10G and later)**

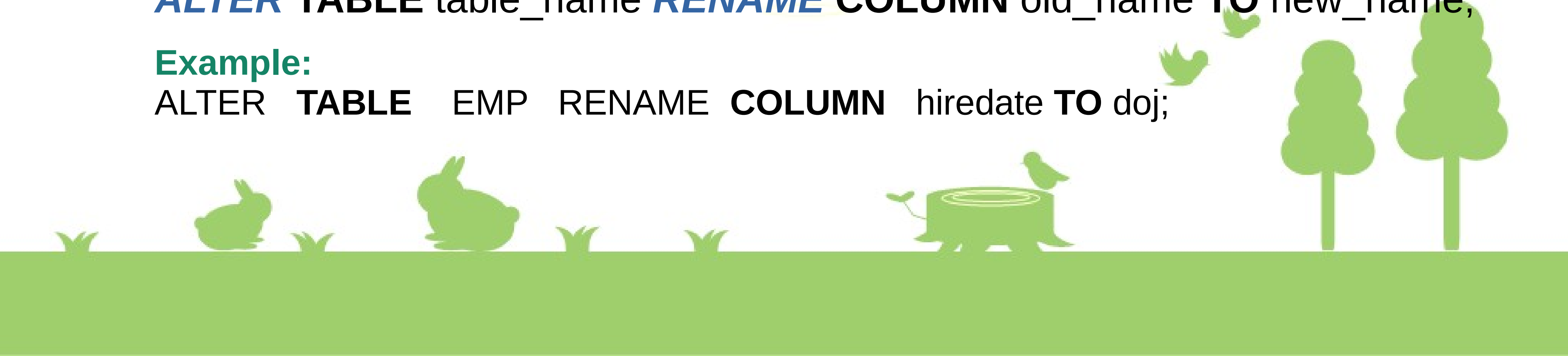
ALTER TABLE - RENAME Column

Syntax:

ALTER **TABLE** table_name ***RENAME*** **COLUMN** old_name **TO** new_name;

Example:

ALTER **TABLE** EMP RENAME **COLUMN** hiredate **TO** doj;



SQL: DDL (Contd.)

ALTER TABLE - ADD Constraint

%To add a constraint

Syntax:

ALTER TABLE table_name **ADD CONSTRAINT** cont_name Primary key (col_name);

Example:

ALTER **TABLE** EMP ADD **CONSTRAINT** pk_empno Primary key (empno);

ALTER TABLE - DROP Constraint

%To delete a constraint

Syntax:

ALTER TABLE table_name DROP CONSTRAINT cont_name;

Example:

ALTER **TABLE** EMP DROP **CONSTRAINT** fk_mgr;

------------DDL-----------

SQL *RENAME* TABLE

SQL: DDL (Contd.)

RENAME TABLE – *Type 1*

%To rename a table

Syntax:

RENAME **TABLE** old_table_name ***TO*** new_table_name;

Example:

RENAME **TABLE** EMP ***TO*** EMPLOYEE;

RENAME TABLE – *Type 2 (ALTER)*

ALTER **TABLE** table_name ***RENAME*** **TO** new_table_name;

------------DDL-----------

SQL *DROP* TABLE

SQL: DDL (Contd.)

DROP TABLE - %To delete a table along with its schema

Syntax:

```
DROP TABLE table_name;
```

Example:

```
DROP   TABLE    EMP;
```

------------DML-----------

SQL *INSERT* DATA

SQL: DML

INSERT DATA - COLUMN-WISE--*Type 1*

%To add a record/row

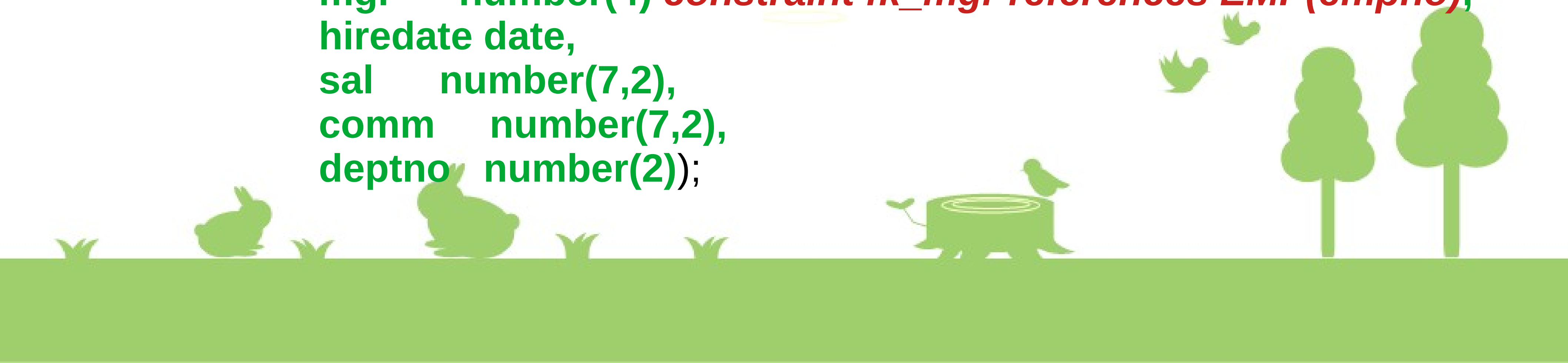
Syntax:

```
INSERT INTO table_name (column_list) VALUES( value_list);
```

Example:

```
INSERT   INTO    EMP (empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES (7839, ‘KING’, ‘President’, NULL, ‘17-NOV-1981’, 5000.0, Null, 10);
```

```
CREATE table EMP(
  empno    number(4),
  ename    varchar2(10),
  job      varchar2(9),
  mgr      number(4) constraint fk_mgr references EMP(empno),
  hiredate date,
  sal      number(7,2),
  comm     number(7,2),
  deptno   number(2));
```



SQL: DML (Contd..)

INSERT DATA – ROW-WISE—*Type 2*

%To add a record/row

Syntax:

INSERT INTO table_name VALUES (value_list);

Example:

INSERT   **INTO**   EMP **VALUES** (7839, ‘KING’, ‘President’, NULL, ‘17-NOV-1981’, 5000.0, Null, 10);

If the **value list** has the **same order** as the **table columns**, you can skip the **column list** although this is **not considered as a good practice.**

SQL: DML (Contd..)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	Allen	Salesman	7698	20/2/81	1600	300	30
7521	Ward	Salesman	7698	22/2/81	1250	500	30
7566	Jones	Manager	7839	2/4/81	2975		20
7654	Martin	Salesman	7698	28/9/81	1250	1400	30
7698	Blake	Manager	7839	1/5/81	2850		30
7782	Clark	Manager	7839	9/6/81	2450		10
7788	Scott	Analyst	7566	9/12/82	3000		20
7839	King	President		17/11/81	5000		10
7844	Turner	Salesman	7698	8/9/81	1500	0	30

------------DML-----------

SQL *DELETE* DATA

SQL: DML (Contd..)

DELETE DATA – *One or Multiple*

%To delete a record/row

Syntax:

```
DELETE  FROM  table_name  WHERE   condition_list;
```

Example:

```
DELETE   FROM   EMP  WHERE mgr=7698 AND sal=1250;
```

DELETE DATA – *All Rows*

%To delete all records/rows

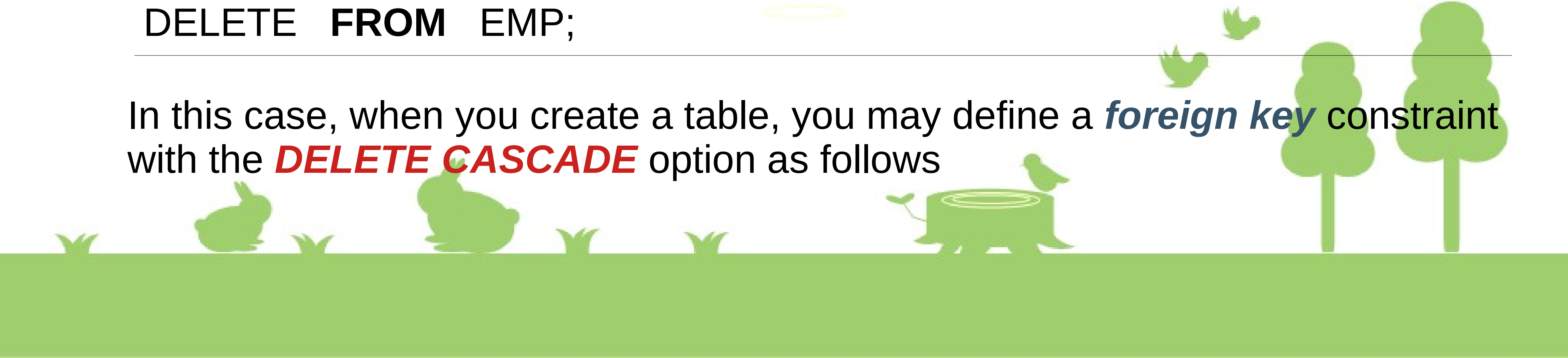
Syntax:

```
DELETE  FROM  table_name;
```

Example:

```
DELETE   FROM   EMP;
```

In this case, when you create a table, you may define a ***foreign key*** constraint with the ***DELETE CASCADE*** option as follows



------------DML-----------

SQL *UPDATE* DATA

SQL: DML (Contd..)

UPDATE DATA – *One or Multiple*

%To delete a record/row

Syntax:

UPDATE table_name **SET** col_name1 = value1, col_name2= value2
WHERE condition_list;

Example:
UPDATE EMP **SET** JOB=’NULL’, SAL=0, COMM=0 **WHERE** deptno = 10 AND JOB <> 'Manager';

UPDATE DATA – *All Rows*

%To delete all records/rows

Syntax:

UPDATE table_name **SET** col_name1 = value1;

Example:
UPDATE EMP **SET** SAL=1.5* SAL;

Thank You

--End of *2nd Session*--