Database Design & Applications

The Database Language - Creating and Managing Tables

Objectives

- Describe the main database objects
- Create tables
- Describe the data types that can be used when specifying column definition
- Alter table definitions
- Drop and truncate tables
- Create and Maintain Constraints

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries

Naming Rules

Table names and column names:

- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, #,and $
- Must not duplicate the name of another object owned by the same user
- Must not be an MS SQL server reserved word

CREATE DATABASE

A SQL Server database can be created, altered and dropped using:

1. Graphically
2. Using Query

Syntax:

CREATE DATABASE databasename;

Example:

CREATE DATABASE Sample;

- When a database is created the following two file get generated:
  - **.MDF File :** Data File (Contains actual data)
  - **.LDF File:** Transaction Log file (Used to recover the database)

ALTER DATABASE

Syntax:

ALTER DATABASE databasename MODIFY Name= newdatabasename;

Example:

ALTER DATABASE Sample MODIFY NAME =Sample1;

- Alternatively, you can also use system stored procedure:

Execute sp_renameDB ‘olddatabasename’ ,’newdatabasename’;

Example :

Execute sp_renameDB ‘Sample’ , ‘Sample1’;

DROP DATABASE

Syntax:

```
DROP DATABASE databasename;
```

Example:

```
DROP DATABASE Sample;
```

- Dropping a database, deletes the LDF and MDF file.
- You cannot drop a database which is in use.

Data Types

- A database table contains multiple columns with specific data types such as numeric or string.
- Each data type in MSSQLSERVER can be determined by the following characteristics:
  - The kind of values it represents.
  - The space that takes up and whether the values are a fixed-length or variable length.
  - The values of the data type can be indexed or not.
  - How TSQL compares the values of a specific data type.

Numeric Data Types

Data Type	Description	Storage
TINYINT	Signed nonnegative 0 to 255 integers.	1 byte
SMALLINT	Signed integers from -32768 to 32767 Unsigned integers from 0 to 65535	2 bytes
INTEGER or INT	Signed integers from -2147483638 to 214747483637 Unsigned integers 0 to 4294967925	4 bytes
BIGINT	Signed integers from -9223372036854775808 to 9223372036854775807 and Unsigned integers 0 to 18446744073709551615 unsigned integers.	8 bytes

Numeric Data Types

Data Type	Description
Decimal(size,[d]) **Or** **Dec**	Allows small numbers with floating decimal point. **size** : specifies maximum number of digits **d**: specifies the maximum number of digits to the right of the decimal
Numeric(size,[d])	Synonym of Decimal
Real	Used for floating point value.
Money	Used for representing monetary values. Money value corresponds to 8 byte Decimal values and are rounded to 4 digits after the decimal point.
Small Money	Corresponds to Money data type but stored in 4 bytes

String Data Types

Data Type	Description
CHAR[(size)]	Holds up to 8000 characters and allows a fixed length string. If size is omitted the length of the string assumed to be 1.
VARCHAR(size)	Holds up to 8000 characters and allows a variable length string.
NCHAR	Fixed Length string of Unicode character with a maximum length of 4000 characters. In CHAR datatype each character is stored in 1byte, but in NCHAR datatype each character is stored in 2 bytes.
NVARCHAR(size)	Stores variable length Unicode character data with a maximum length of 4000 characters.

Date and Time Data Types

Data Type	Description
DATETIME	A combination of date and time values in the format: YYYY-MM-DD HH:MI:SS, where the supported range is from ‘1753-01-01 00:00:00’ to ‘9999-12-31 23:59:59’. Time component is stored in 4 bytes
SMALLDATETIME	A combination of date and time values in the format: YYYY-MM-DD HH:MI:SS, where the supported range is from ‘1900-01-01 00:00:00’ to ‘2079-12-31 23:59:59’. Time component is stored in 2 bytes
DATE	Supported range is from ‘0001-01-01 00:00:00’ to ‘9999-12-31’. Stored in 3 bytes.
TIME	Stored in 3-5 bytes and has an accuracy of 100 nanoseconds.

Data Definition Language (DDL)

- **Data Definition Language** (**DDL**) is a syntax for creating and modifying database objects such as tables, indexes, and users.
- Following commands comes under the DDL category:
- CREATE
- ALTER
- DROP
- TRUNCATE

The Create Table Statement

```
CREATE TABLE tbl_name
          (column_definition,. . .);
```

Arguments:

Name	Description
tbl_name	name of the table

```
column_definition:
     column data_type [NOT NULL | NULL] [DEFAULT default_value]
```

Arguments:

Name	Description
NOT NULL | NULL	If neither NULL nor NOT NULL is specified, the column is treated that, NULL had been specified.
DEFAULT	▪ Literal values, expressions, or SQL functions are legal values. ▪ Another column’s name or a pseudocolumn are illegal values. ▪ The default data type must match the column data type. ▪ DEFAULT does not apply to the BLOB or TEXT types.

Creating Tables

```
CREATE TABLE dept
(deptno INT,
 dname NVARCHAR(14),
 loc   NVARCHAR(13)NOT NULL
 Budget MONEY DEFAULT 1000);
```

Copy Table Structure

- SqlServer allows you to create a table identical to another by using INTO clause in SELECT.
- Following statement will create a table ‘employees_copy' whose structure is identical to the table ‘employees’

```
SELECT *
INTO employees_copy
FROM employees;
```

ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns.

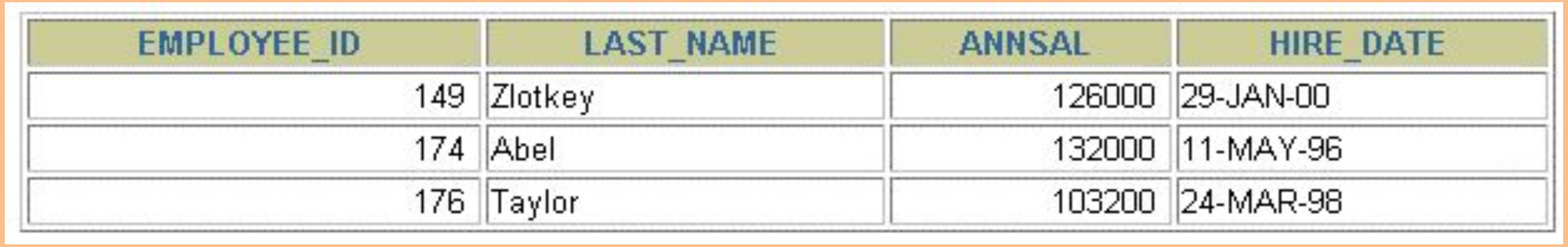
```
ALTER TABLE table
ADD            column datatype [DEFAULT expr],
               [column datatype]...;
```

```
ALTER TABLE table
ALTER COLUMN   column datatype [DEFAULT expr] [NOT NULL];
```

```
ALTER TABLE table
DROP   column1, [column2 ]...;
```

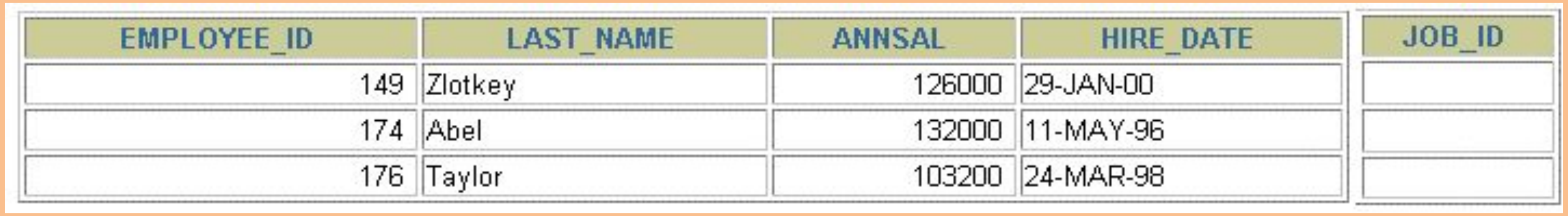
Adding a Column

DEPT80



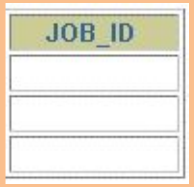
EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

DEPT80



EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

New column



JOB_ID

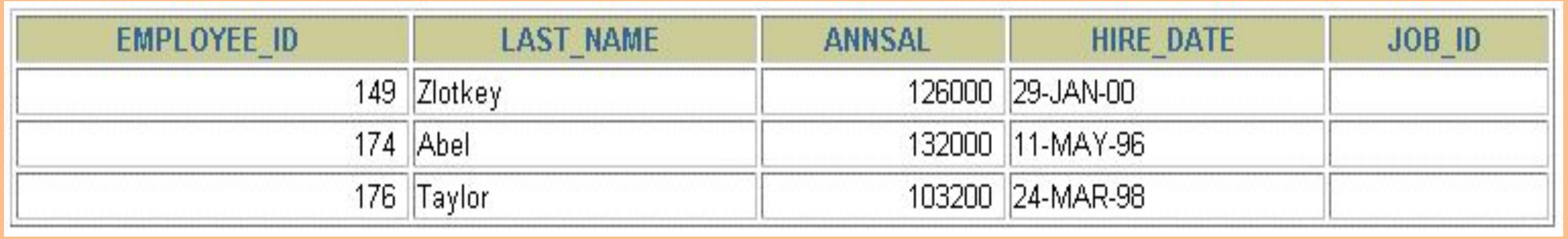
“Add a new column to the DEPT80 table.”

Adding a Column

- You use the ADD clause to add columns.

```
ALTER TABLE dept80
ADD   job_id NVARCHAR(9);
```

- The new column becomes the last column.



EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Modifying a Column

- You can change a column’s data type, size, and default value.

```
ALTER TABLE dept80
ALTER COLUMN job_id NVARCHAR(15)NOT NULL;
```

- A change to the default value affects only subsequent insertions to the table.

Dropping a Column

- Use the DROP COLUMN clause to drop columns you no longer need from the table.

```
ALTER TABLE dept80
DROP  column job_id ;
```

Truncating a Table

- The TRUNCATE TABLE statement:
  - Removes all rows from a table
  - Releases the storage space used by that table

```
TRUNCATE TABLE detail_dept;
```

- Alternatively, you can remove rows by using the
  - DELETE statement.

Renaming a Table

- With the help of inbuilt procedure sp_rename we can rename a table.
- Syntax:
  - Execute sp_rename ‘oldtablename’, ‘new_tablename’

```
Execute sp_rename ‘Employee’ , ‘Emp’;
```

Dropping a Table

- All data and structure in the table is deleted.
- All indexes are dropped.

```
DROP TABLE dept80;
```

Adding Constraints?

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- **The following constraint types are valid:**
  - DEFAULT
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Constraint Guidelines

- SQLServer automatically generates a constraint name
- Create a constraint either:
  - At the same time as the table is created, or
  - After the table has been created
- Define a constraint at the column or table level.

Defining Constraints

```
CREATE TABLE [schema.]table
             (column datatype  [column_constraint],
              ...
              [table_constraint][,...]);
```

- **column datatype**: specifies the name of the column. Each column has a specific data type and optional size e.g.,VARCHAR(255)

Defining Constraints

- Column level constraint

```
column_name constraint_type, ...
```

- Table level constraint

```
column,...
  constraint_type (column, ...),
```

```
CREATE TABLE employees(
               employee_id  INT,
               first_name   VARCHAR(20) NOT NULL,
               ...
               job_id       VARCHAR(10),
                            PRIMARY KEY
                            (EMPLOYEE_ID));
```

NOT NULL Constraint

- NOT NULL constraint ensures that a column cannot have a null value.
- NOT NULL constraint can be defined at column level only.

```
CREATE TABLE employees
(       employee_id    INT,
        Salary     MONEY NOT NULL,
        Last_name VARCHAR(25),
        commission_pct DECIMAL(8,2),
        hire_date      DATE


   ...
```

DEFAULT Constraint

Default constraint is used to set a default for a column.

```
CREATE TABLE employees
(       employee_id    INT,
        Salary     MONEY DEFAULT 10000,
        Last_name VARCHAR(25),
        commission_pct DECIMAL(8,2),
        hire_date      DATE


   ...
```

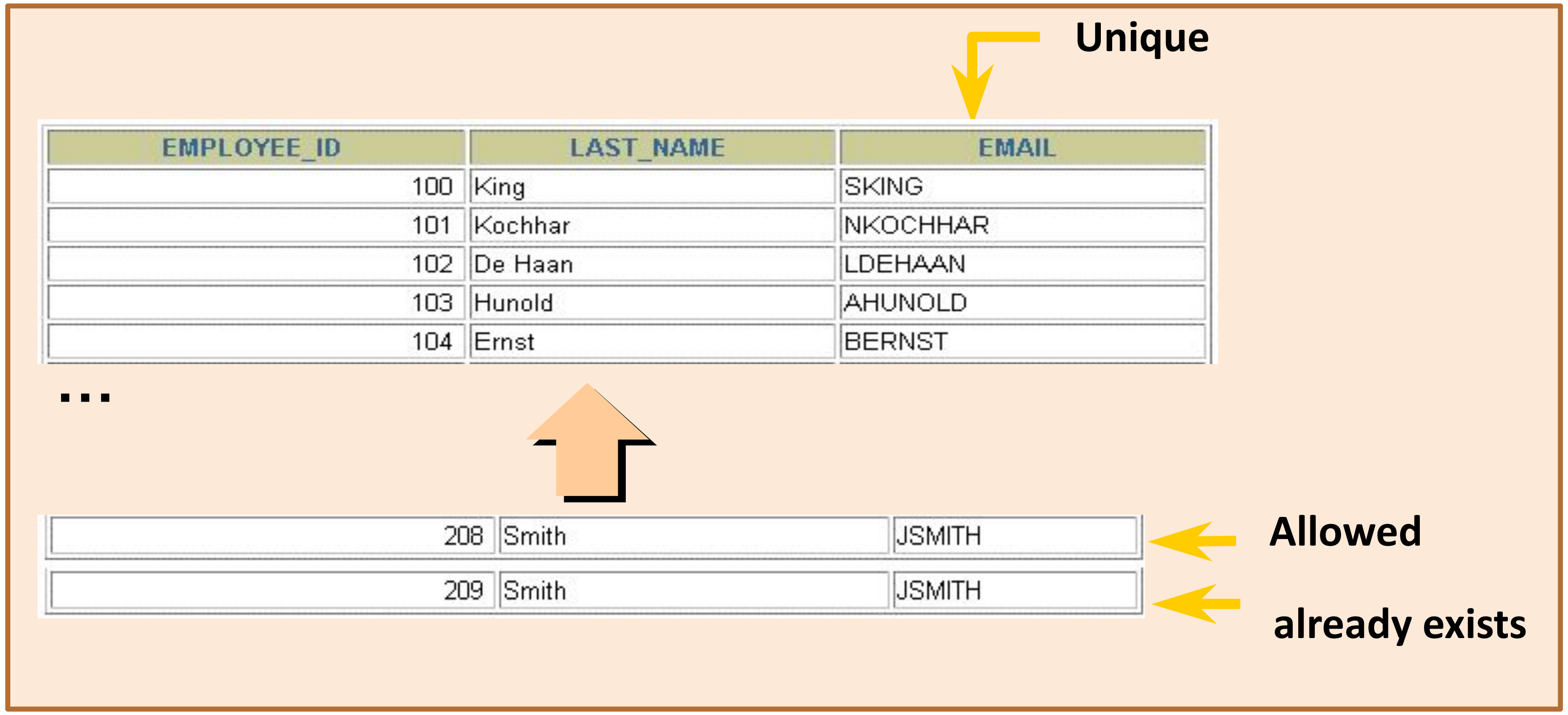
DEFAULT Constraint

```
CREATE TABLE employees
(       employee_id    INT,
        Salary    MONEY CONSTRAINT employees_salary_df DEFAULT 10000,
        Last_name VARCHAR(25),
         commission_pct FLOAT(8,2),
          hire_date       DATE


              ...
```

UNIQUE Constraint

SQL Server UNIQUE constraints allow you to ensure that the data stored in a column, or a group of columns, is unique among the rows in a table.



EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

...

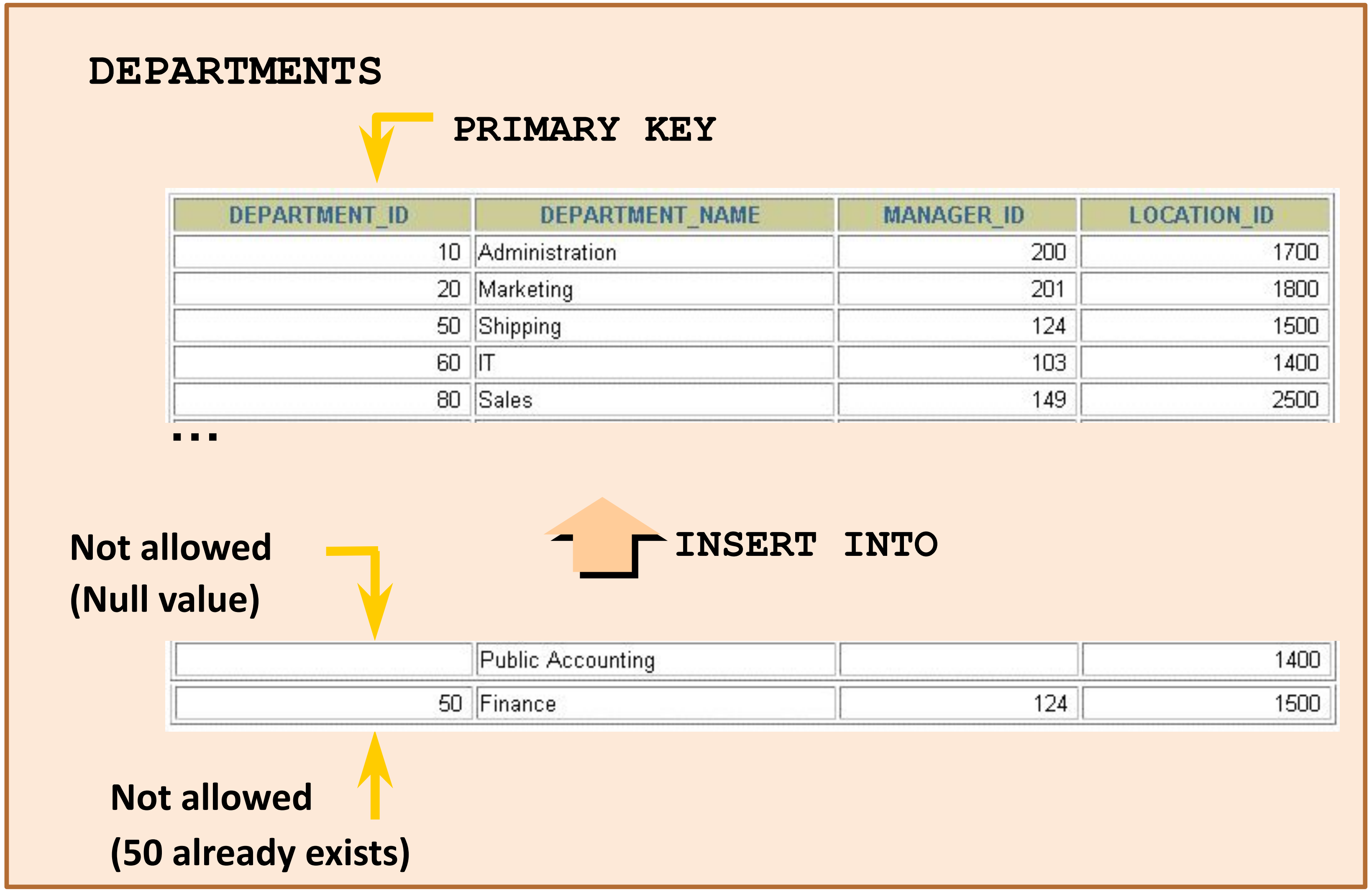
208	Smith	JSMITH
209	Smith	JSMITH

UNIQUE Constraint

Defined at either the table level or the column level:

```
  CREATE TABLE employees(
    employee_id        INT,
    adhaar             VARCHAR(25),
    last_name          VARCHAR(25) NOT NULL,
    email              VARCHAR(25) UNIQUE,
    salary             DEC(8,2),
    commission_pc      DEC(4,2),
... t  hire_date       DATE NOT
    UNIQUE(adhaar));   NULL,
```

PRIMARY KEY Constraint



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...

	Public Accounting		1400
50	Finance	124	1500

PRIMARY KEY Constraint

- A primary key is a column or a group of columns that uniquely identifies each row in a table
- Defined at either the table level or the column level:

```
CREATE TABLE  departments
(   department_id          INT,
    department_name        VARCHAR(30) NOT NULL,
    manager_id            INT,
      PRIMARY KEY(department_id));
```

PRIMARY KEY Constraint

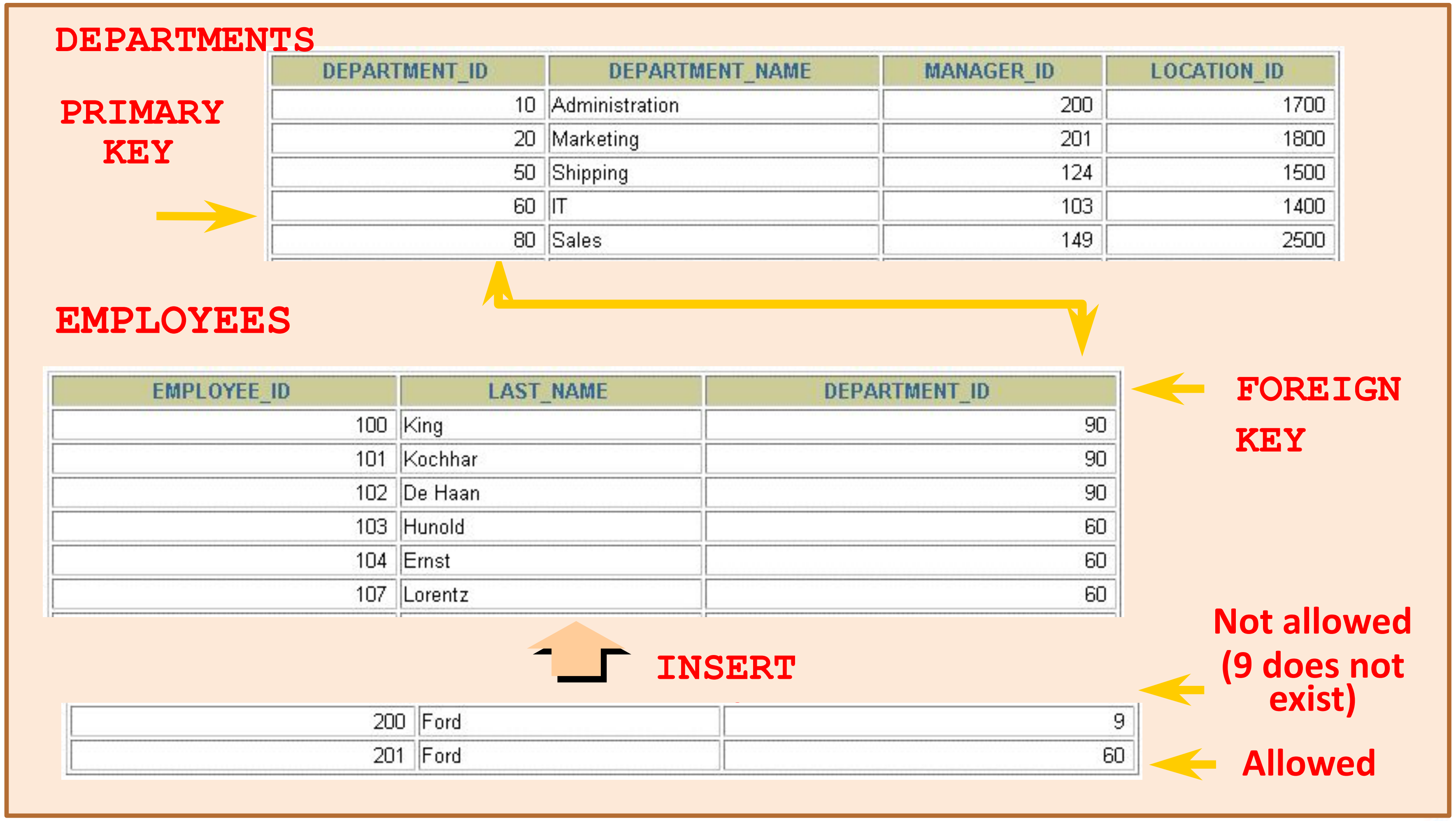
- The CHECK constraint allows you to specify the values in a column that must satisfy a Boolean expression.

- **Syntax:**

```
CHECK (expr)
```

```
CREATE TABLE departments
(   department_id          INT PRIMARY KEY,
    department_name        VARCHAR(30) NOT NULL,
    manager_id           INT,
       CHECK(department_id BETWEEN 10 AND 999));
```

FOREIGN KEY Constraint



DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

INSERT

200	Ford	9
201	Ford	60

FOREIGN KEY Constraint

- A foreign key is a column or a group of columns in one table that uniquely identifies a row of another table or the same.

```
FOREIGN KEY (col_name,….)
REFERENCES tbl_name (col_name, …)
[ON DELETE reference_option]
[ON UPDATE reference_option]
reference option :
NO ACTION| CASCADE| SET NULL | SET DEFAULT
```

FOREIGN KEY Constraint

Example:

```
 CREATE TABLE Department
 (department_id        INT PRIMARY KEY,
d_name                 VARCHAR(25)  NOT NULL,
location_id            INT,
FOREIGN KEY (location_id)
  REFERENCES location(location_id));
```

FOREIGN KEY Constraint Keywords

- **FOREIGN KEY:** Defines the column in the child table at the table constraint level
- **REFERENCES:** Identifies the table and column in the parent table
- When an UPDATE or DELETE operation affects a key value in the parent table that has matching rows in the child table, the result depends on the referential action specified by ON UPDATE and ON DELETE subclauses
- **CASCADE:** Delete or update the row from the parent table and automatically delete or update the matching rows in the child table
- **SET NULL:** Delete or update the row from the parent table and set the foreign key column or columns in the child table to NULL
- **SET DEFAULT:**. Delete or update the row from the parent table and set the foreign key column or columns in the child table to DEFAULT value if specified.
- **NO ACTION:** Default action, Will not allow to delete a row in parent table if there are one or many rows present in child table .

FOREIGN KEY Constraint

Example:

```
CREATE TABLE Department
        (department_id    INT PRIMARY KEY,
       d_name              VARCHAR(25)  NOT NULL,
       location_id        INT,
      FOREIGN KEY (location_id)
        REFERENCES location(location_id) ON UPDATE CASCADE);
```

Example:

```
CREATE TABLE (Department

       department_id      INT PRIMARY KEY,
       d_name              VARCHAR(25)  NOT NULL,
       location_id         INT,
         FOREIGN KEY (location_id)
         REFERENCES location(location_id)
         ON DELETE CASCADE ON UPDATE CASCADE);
```

Adding a Constraint

- Add a FOREIGN KEY constraint to the EMPLOYEES table indicating that a manager must already exist as a valid employee in the EMPLOYEES table.

```
ALTER TABLE  employees
ADD FOREIGN KEY(manager_id)
REFERENCES employees(employee_id);
```

- Add a PRIMARY KEY constraint on EMPLOYEE_ID colum, in EMPLOYEES

```
ALTER TABLE Employee
ADD CONSTRAINT employee_employee_id_pk PRIMARY KEY
 (employee_id)
```

Adding Constraints

- Altering an existing column to add a default constraint.

```
ALTER TABLE  tablename
ADD CONSTRAINT constraintname DEFAULT (default_value)FOR column_name
```

- Adding a new column, with default value, to an existing table.

```
   ALTER TABLE tablename
   ADD column_name datatype [NOT NULL]
   CONSTRAINT constraintname DEFAULT (default_value)
```

Adding a DEFAULT Constraint

- Altering an existing column to add a default constraint.

```
ALTER TABLE  Employee
ADD CONSTRAINT employee_salary_df
DEFAULT (1000)FOR (salary)
```

- Adding a new column, with default value, to an existing table.

```
ALTER TABLE Employee
ADD salary money NOT NULL
CONSTRAINT employee_salary_df DEFAULT (1000)
```

Dropping a Constraint

- **Syntax:** Dropping a constraint.

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name
```

- Remove the manager constraint from the EMPLOYEES table.

```
ALTER TABLE      employees
DROP             emp_manager_fk
CONSTRAINT       ;
```

THANK YOU!