2

# Database Interfaces

# Managing Schema Objects

Oracle for Base

# Objectives

After completing this lesson, you should be able to do the following:

- Describe the logical structure of tables
- Managing Schema Objects
- Modeling and Accessing Relational Data

# Objectives

After completing this lesson, you should be able to do the following:

- **Describe the logical structure of tables**
- Managing Schema Objects
- Modeling and Accessing Relational Data

# What Is SQL ?

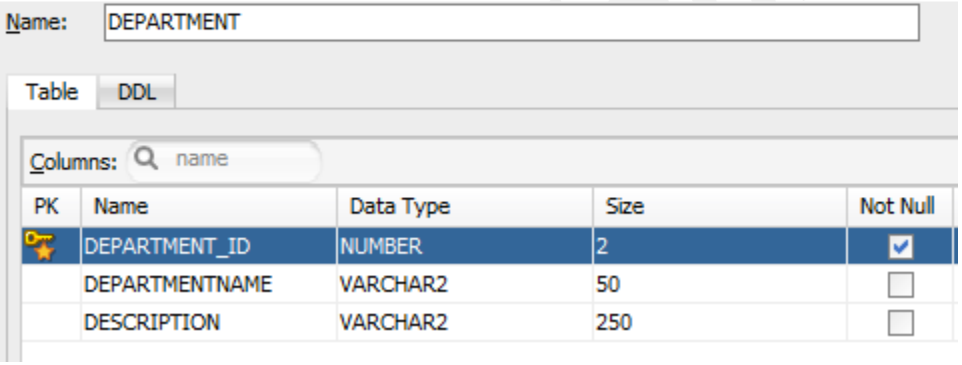
Structured query language (**SQL**) is the set of statements with which all programs and users access data in an Oracle database. SQL statements are divided into the following categories:

- Data definition  language (DDL) statements
- Data manipulation language (DML) statements
- Transaction control statements
- Session control statements
- System control statements
- Embedded SQL statements

# Defining Data

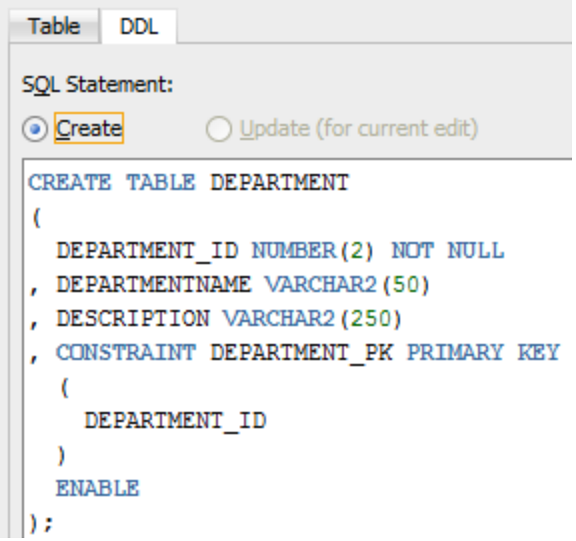
The statements that make up the data definition language (DDL) are used to work with database objects.

- The **CREATE** statement defines new objects in the database. You can create many different objects in the database, including tables, indexes, sequences, views, and others.



Name: DEPARTMENT

| PK | Name | Data Type | Size | Not Null |
|---|---|---|---|---|
| 🔑 | DEPARTMENT_ID | NUMBER | 2 | ☑ |
| | DEPARTMENTNAME | VARCHAR2 | 50 | □ |
| | DESCRIPTION | VARCHAR2 | 250 | □ |



SQL Statement: ● Create ○ Update (for current edit)

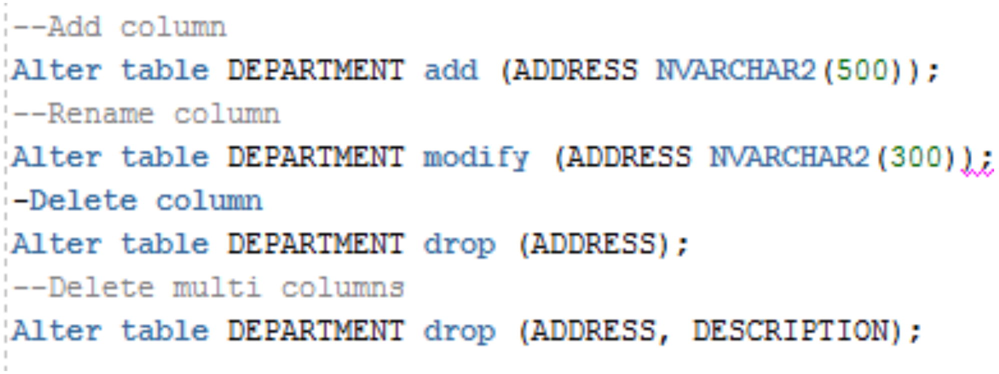
```
CREATE TABLE DEPARTMENT
(
  DEPARTMENT_ID NUMBER(2) NOT NULL
, DEPARTMENTNAME VARCHAR2(50)
, DESCRIPTION VARCHAR2(250)
, CONSTRAINT DEPARTMENT_PK PRIMARY KEY
  (
    DEPARTMENT_ID
  )
  ENABLE
);
```

# Defining Data

The statements that make up the data definition language(DDL) are used to work with database objects.

- The **ALTER** statement modifies existing objects in the database. After an object is defined, you may need to modify that object. Depending on the object, there are different things that can be modified. For example, you can add a column to a table, but you cannot add a column to an index.

**Example**



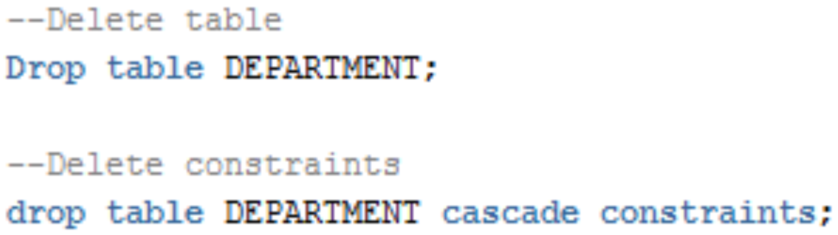
```
--Add column
Alter table DEPARTMENT add (ADDRESS NVARCHAR2(500));
--Rename column
Alter table DEPARTMENT modify (ADDRESS NVARCHAR2(300));
-Delete column
Alter table DEPARTMENT drop (ADDRESS);
--Delete multi columns
Alter table DEPARTMENT drop (ADDRESS, DESCRIPTION);
```

# Defining Data

The statements  that make up the data definition language(DDL) are used  to work with database objects.

- The **DROP** command removes objects in the database. After an object is dropped  it can  no longer be referenced. It is possible toreinstate a dropped a table.

**Example**



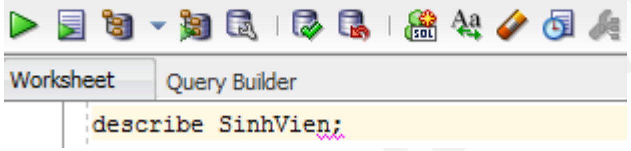
```
--Delete table
Drop table DEPARTMENT;

--Delete constraints
drop table DEPARTMENT cascade constraints;
```

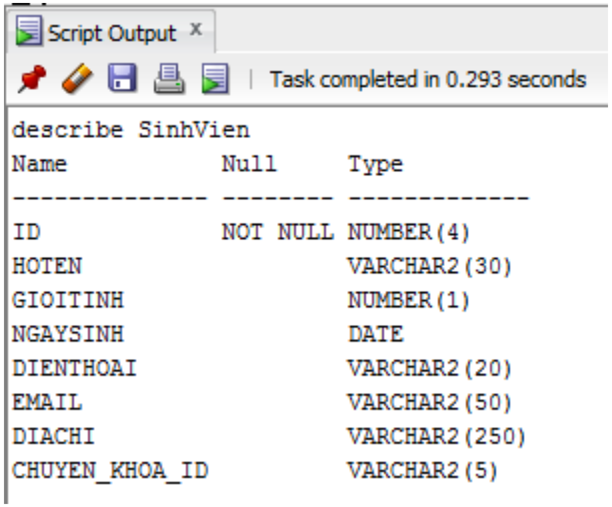
# Describing Data

The **DESCRIBE** command is used to see the description of a database object. This command is not part of theSQL standard, but is one of severalSQL*Plus command unique to Oracle tools

**Example**



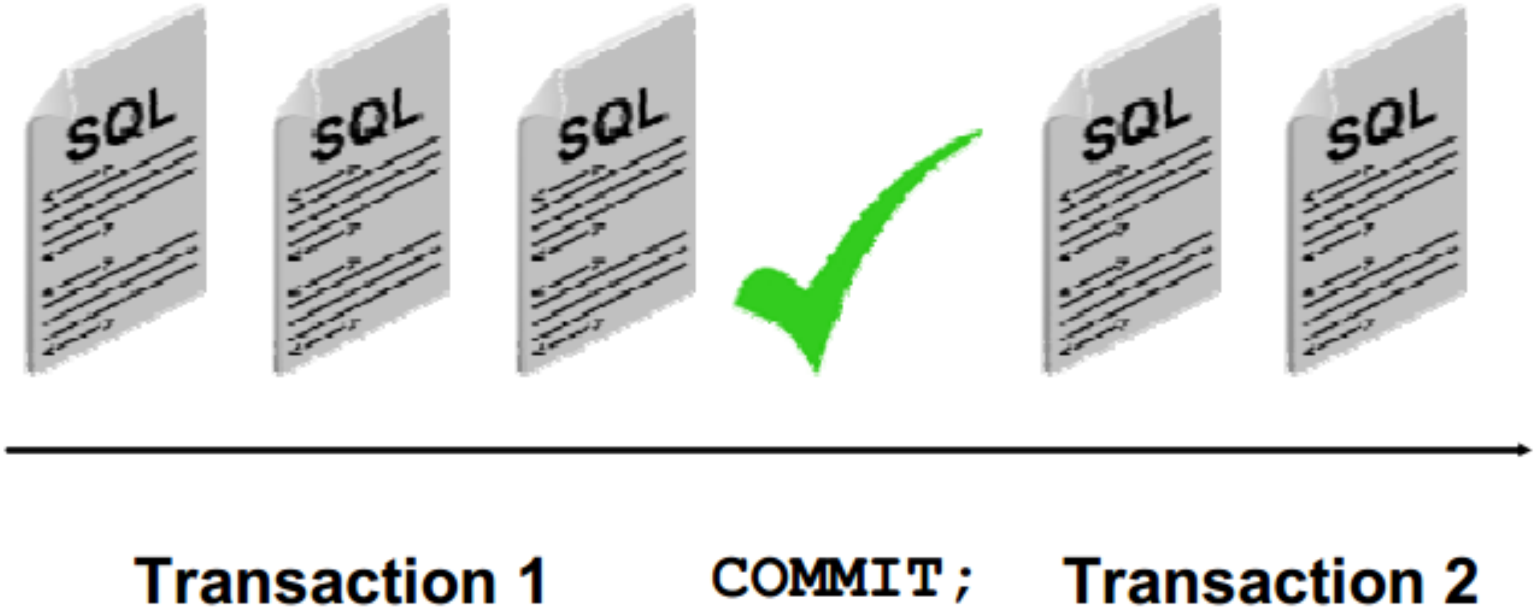
```
describe SinhVien;
```



```
describe SinhVien
Name           Null     Type
-------------- -------- -------------
ID             NOT NULL NUMBER(4)
HOTEN                   VARCHAR2(30)
GIOITINH                NUMBER(1)
NGAYSINH                DATE
DIENTHOAI               VARCHAR2(20)
EMAIL                   VARCHAR2(50)
DIACHI                  VARCHAR2(250)
CHUYEN_KHOA_ID          VARCHAR2(5)
```

# Overview of Transactions

When an executable SQL statement  is executed a transaction begins.



# Overview of Transactions

A transaction ends when any of the following occurs:

- A user issues a **COMMIT** or **ROLLBACK** statement without a SAVEPOINT clause.
- A user runs a DDL statement such as CREATE, DROP, RENAME, or GRANT. If the current transaction contains any DML statements, Oracle first commits the transaction, and then runs and commits the DDL statement as a new, single statement transaction.
- A user disconnects from Oracle. The current transaction is committed.
- A user process terminates abnormally. The current transaction is rolled back.

After one transaction ends, the next executable SQL statement automatically starts the next transaction.

# Objectives

After completing this lesson, you should be able to do the following:

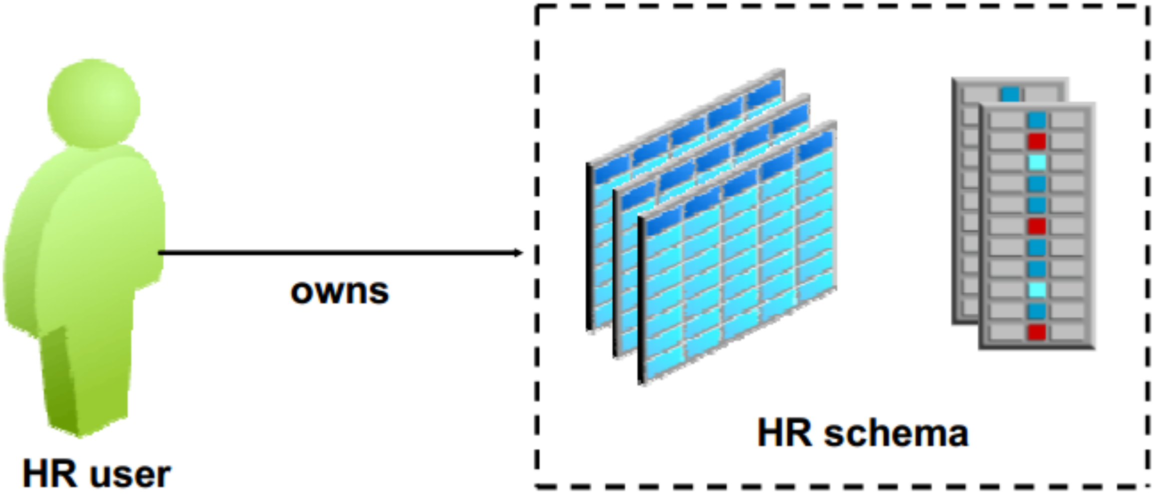
- Describe the logical structure of tables
- **Managing Schema Objects**
- Modeling and Accessing Relational Data

# Managing Schema Objects

- Create and modify tables
- Define constraints
- View the attributes of a table
- View the contents of a table

# What Is a Schema ?

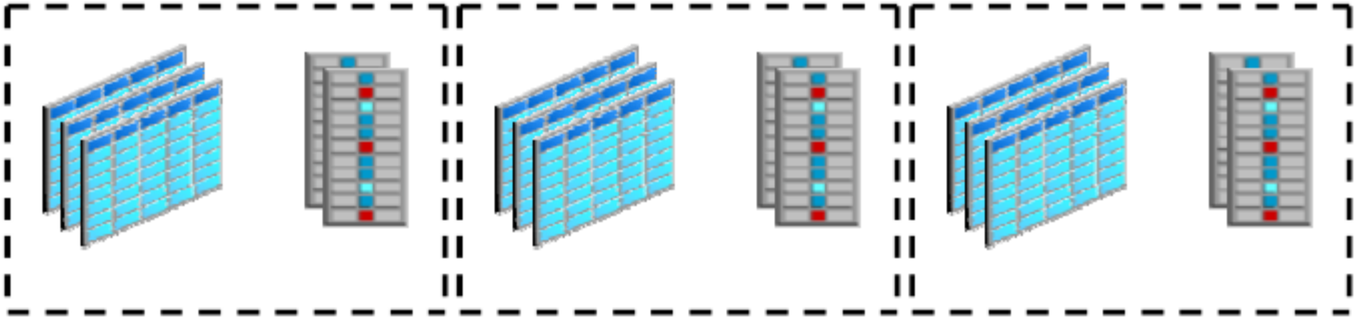
- A schema is a collection of database objects owned by a particular user. A schema has the same name as the user that owns the schema.
- Schema objects are the logical structures that directly refer to the database's data.
- Schema objects include structures such as tables, views, and indexes.



# Schema

Schemas created as part of the database creation process:

- SYS
- SYSTEM
- Sample schemas



# Schema

When you create the database, a number of schemas are created for you. Two of particular importance are:

- **SYS schema**: All of the base tables and views that constitute the database's data dictionary are created in the SYS schema. The data dictionary is a collection of tables that describe the Oracle database. The data dictionary is created in the SYSTEM tablespace when the database is created and is updated by the Oracle database server when a data definition language (DDL) statement is executed. The data dictionary contains information about users, schema objects, and storage structures.

# Schema

When you create the database, a number of schemas are created for you. Two of particular importance are:

- **SYSTEM schema**: Contains additional tables and views that store administrative information, and internal tables and views used by various Oracle options and tools. You should not create any additional objects in the SYSTEM schema.

# Schema

During a complete installation of an Oracle database, the sample schemas are installed automatically with the seed database.

- **HR**: The Human Resources schema is a simple schema for introducing basic topics. An extension to this schema supports Oracle Internet Directory demonstrations.
- **OE**: The Order Entry schema is for dealing with matters of intermediate complexity. A multitude of data types is available in the OE schema. The OC (Online Catalog) subschema is a collection of object-relational database objects built inside the OE schema.
- **PM**: The Product Media schema is dedicated to multimedia data types.

# Schema

During a complete installation of an Oracle database, the sample schemas are installed automatically with the seed database.

- **QS**: The Queued Shipping schema contains a set of schemas that are used to demonstrate Oracle Advanced Queuing capabilities.
- **SH**: The Sales History schema is designed to allow demonstrations with larger amounts of data. An extension to this schema provides support for advanced analytic processing.

2

# Using DDL Statements to Create and Manage Tables

Oracle for Base

# Objectives

After completing this lesson, you should be able to do the following:

- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation
- Describe how schema objects work

# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user’s tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# Naming Database Objects

- Names must be from 1 to 30 bytes long with these exceptions:
  - Names of databases are limited to 8 bytes
  - Names of database links can be as long as 128 bytes
- Non quoted names cannot be Oracle reserved words.
- Non quoted names must begin with an alphabetic character from your database character set.

# Naming Database Objects

- Non quoted names can contain only
  - Alphanumeric characters from your database character set
  - The underscore(_)
  - Dollar sign($)
  - Pound sign(#)
- No two objects can have the same name within the same namespace.

# Database Objects

| Object | Description |
|---|---|
| **Table** | Basic unit of storage; composed of rows |
| **View** | Logically represents subsets of data from one or more tables |
| **Sequence** | Generates numeric values |
| **Index** | Improves the performance of some queries |
| **Synonym** | Gives alternative name to an object |

# Lesson Stanford

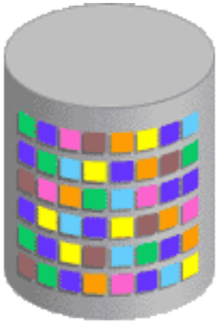
- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user's tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# CREATE TABLE Statement

- You must have:
  - CREATE TABLE privilege
  - A storage area

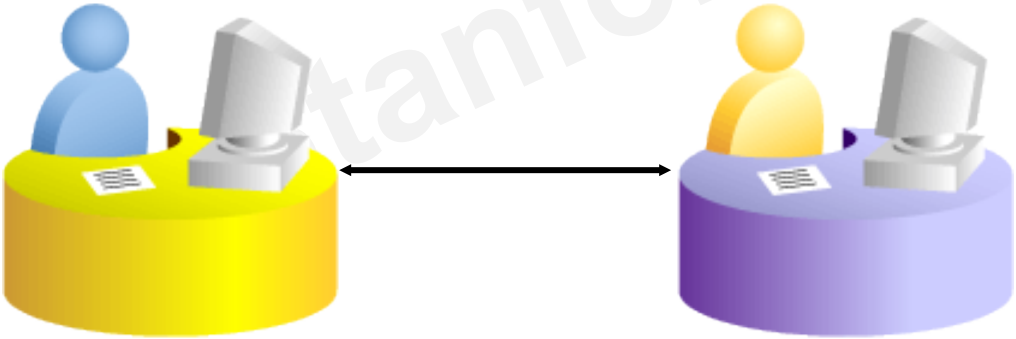
```
CREATE TABLE [schema.]table
          (column datatype [DEFAULT expr][, ...]);
```

- You specify:
  - Table name
  - Column name, column data type, and column size



# Referencing Another User’s Tables

- Tables belonging to other users are not in the user’s schema.
- You should use the owner’s name as a prefix to those tables.



```
SELECT *
FROM userB.employees;
```

```
SELECT *
FROM userA.employees;
```

# DEFAULT Option

- Specify a default value for a column during an insert.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- 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.

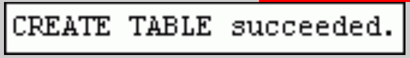
```
CREATE TABLE hire_dates
        (id              NUMBER(8),
         hire_date DATE DEFAULT SYSDATE);
```

```
CREATE TABLE succeeded.
```

# Creating Tables

- Create the table:

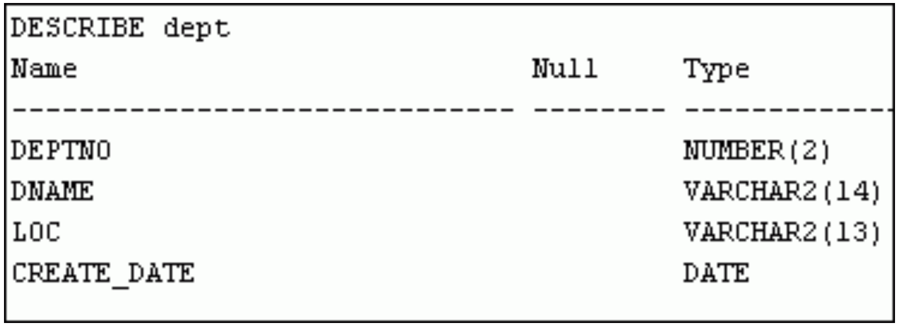
```
CREATE TABLE dept
        (deptno      NUMBER(2),
         dname       VARCHAR2(14),
         loc         VARCHAR2(13),
         create_date DATE DEFAULT SYSDATE);
```



```
CREATE TABLE succeeded.
```

- Confirm table creation:

```
DESCRIBE dept
```



```
DESCRIBE dept
Name                           Null     Type
------------------------------ -------- ------------
DEPTNO                                  NUMBER(2)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
CREATE_DATE                             DATE
```

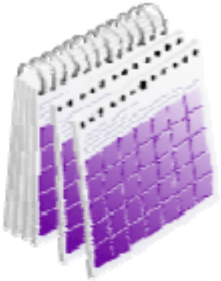
# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user's tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# Specifying Data Types in Tables

Common data types:

- **CHAR(size):** Fixed-length character data of length size bytes. Maximum size is 2000 bytes.
- **VARCHAR2(size):** Variable-length character string having maximum length size bytes or characters.
- **DATE:** Valid date range from January 1, 4712 BC to December 31, 9999 AD
- **NUMBER(p,s):** Number having precision p and scale s



# Specifying Data Types in Tables

When you create a table, you must specify a data type for each of its columns. When you create a procedure or function, you must specify a data type for each of its arguments. These data types define the domain of values that each column can contain or each argument can have.

- **CHAR(size):** Fixed-length character data of length size bytes. Maximum size is 2000 bytes. Default and minimum size is 1 byte.
- **VARCHAR2(size):** Variable-length character string having maximum length size bytes or characters. Maximum size is 4000 bytes. You must specify size for VARCHAR2

# Specifying Data Types in Tables

- **DATE:** Valid daterange from January 1, 4712 BC to December 31, 9999 AD; also stores the time down to the second.
- **NUMBER:** Number having precision p and scale s. The precision canrange from 1 to 38. The scale can range from -84 to 127.

Refer to the Oracle Database SQL Reference for a complete list of built-in data types and userdefined types.

# Specifying Data Types in Tables

- **FLOAT(p):** This  is an ANSI data type. The FLOAT data type is a floating-point number with a binary precision p. The default precision for this data type is 126 binary, or 38 decimal.
- **INTEGER:** This is equivalent to NUMBER(p,0).
- **NCHAR(length):** The NCHAR data type is a Unicode-only data type. When you create a table with an NCHAR column, you define the column length in characters. The maximum column size allowed is 2000 bytes. If you insert a value that is shorter  than  the column length, then Oracle blank-pads the value to column length. You cannot insert a CHAR value into an NCHAR column, nor  can you insert an NCHAR value into a CHAR column.

# Specifying Data Types in Tables

- **NVARCHAR2(length):** The NVARCHAR2 data type is a Unicode-only data type. It is like NCHAR expect that its maximum length is 4000 bytes and it is not blank-padded to the length specified.
- **LONG:** Character data of variable length up to 2 gigabytes, or 2 ^31 -1 bytes.
- **LONG RAW:** Raw binary data of variable length up to 2 gigabytes.
- **RAW(size):** Raw binary data of length size bytes. Maximum size is 2000 bytes. You must specify size for a RAW value.

# Specifying Data Types in Tables

- **ROWID:** Base 64 string representing the unique address of arow in its table. This data type is primarily for values returned by the ROWID pseudo column.
- **UROWID:** Base 64 stringrepresenting the logical address of a row of an index-organized table. The optional size  is the size of a column of type UROWID. The maximum size and default is 4000 bytes.

# Specifying Data Types in Tables

- **BLOB:** A binary large object.
- **CLOB:** A character large object containing single-byte or multibyte characters. Both fixedwidth and variable-width character sets are supported, and both use the CHAR database character set.
- **NCLOB:** A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, and both use the NCHAR database character set. Stores national character set data.

# Specifying Data Types in Tables

- **BFILE:** Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.
- **TIMESTAMP (fractional_seconds_precision):** Year, month, and day values of date, as well as hour, minute, and second values of time, where fractional_seconds_precision is the number of digits in the fractional part of the SECOND date time field(as in fractions of a second). Accepted values of fractional_seconds_precision are 0 to 9. The default is 6.

# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user’s tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# Including 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:
  - `NOT NULL`
  - `UNIQUE`
  - `PRIMARY KEY`
  - `FOREIGN KEY`
  - `CHECK`

# Constraint Guidelines

- You can name a constraint, or the Oracle server generates a name by using the `SYS_Cn` format.
- Create a constraint at either of the following times:
  - At the same time as the creation of the table
  - After the creation of the table
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

# Defining Constraints

- Syntax:

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

- Column-level constraint syntax:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Table-level constraint syntax:

```
column,...
   [CONSTRAINT constraint_name] constraint_type
   (column, ...),
```

# Defining Constraints

- Example of a column-level constraint:

```
CREATE TABLE employees(
  employee_id  NUMBER(6)
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,
  first_name   VARCHAR2(20),
  ...);
```

1

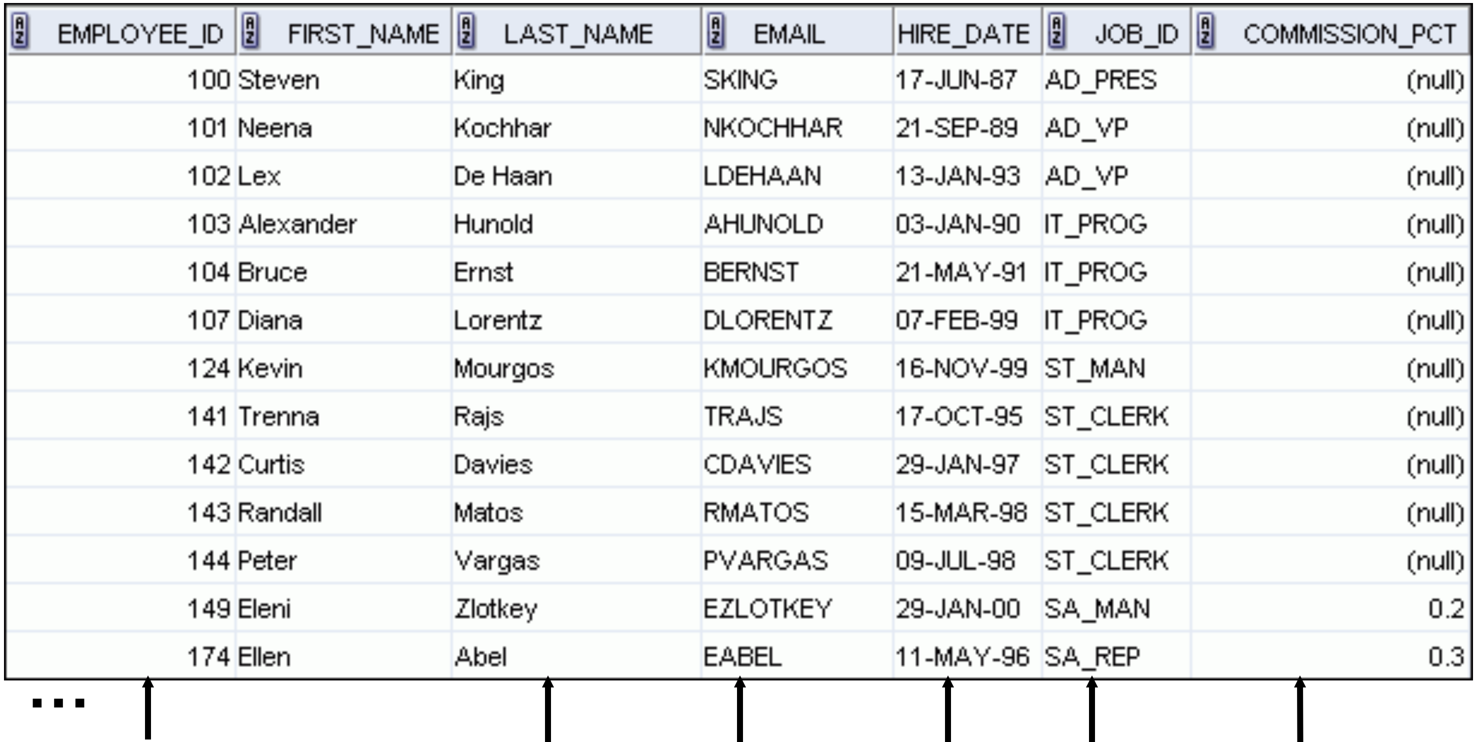
- Example of a table-level constraint:

```
CREATE TABLE employees(
  employee_id  NUMBER(6),
  first_name   VARCHAR2(20),
  ...
  job_id       VARCHAR2(10) NOT NULL,
  CONSTRAINT emp_emp_id_pk
    PRIMARY KEY (EMPLOYEE_ID));
```

2

# NOT NULL Constraint

Ensures that null values are not permitted for the column:



| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | HIRE_DATE | JOB_ID | COMMISSION_PCT |
|---|---|---|---|---|---|---|
| 100 | Steven | King | SKING | 17-JUN-87 | AD_PRES | (null) |
| 101 | Neena | Kochhar | NKOCHHAR | 21-SEP-89 | AD_VP | (null) |
| 102 | Lex | De Haan | LDEHAAN | 13-JAN-93 | AD_VP | (null) |
| 103 | Alexander | Hunold | AHUNOLD | 03-JAN-90 | IT_PROG | (null) |
| 104 | Bruce | Ernst | BERNST | 21-MAY-91 | IT_PROG | (null) |
| 107 | Diana | Lorentz | DLORENTZ | 07-FEB-99 | IT_PROG | (null) |
| 124 | Kevin | Mourgos | KMOURGOS | 16-NOV-99 | ST_MAN | (null) |
| 141 | Trenna | Rajs | TRAJS | 17-OCT-95 | ST_CLERK | (null) |
| 142 | Curtis | Davies | CDAVIES | 29-JAN-97 | ST_CLERK | (null) |
| 143 | Randall | Matos | RMATOS | 15-MAR-98 | ST_CLERK | (null) |
| 144 | Peter | Vargas | PVARGAS | 09-JUL-98 | ST_CLERK | (null) |
| 149 | Eleni | Zlotkey | EZLOTKEY | 29-JAN-00 | SA_MAN | 0.2 |
| 174 | Ellen | Abel | EABEL | 11-MAY-96 | SA_REP | 0.3 |

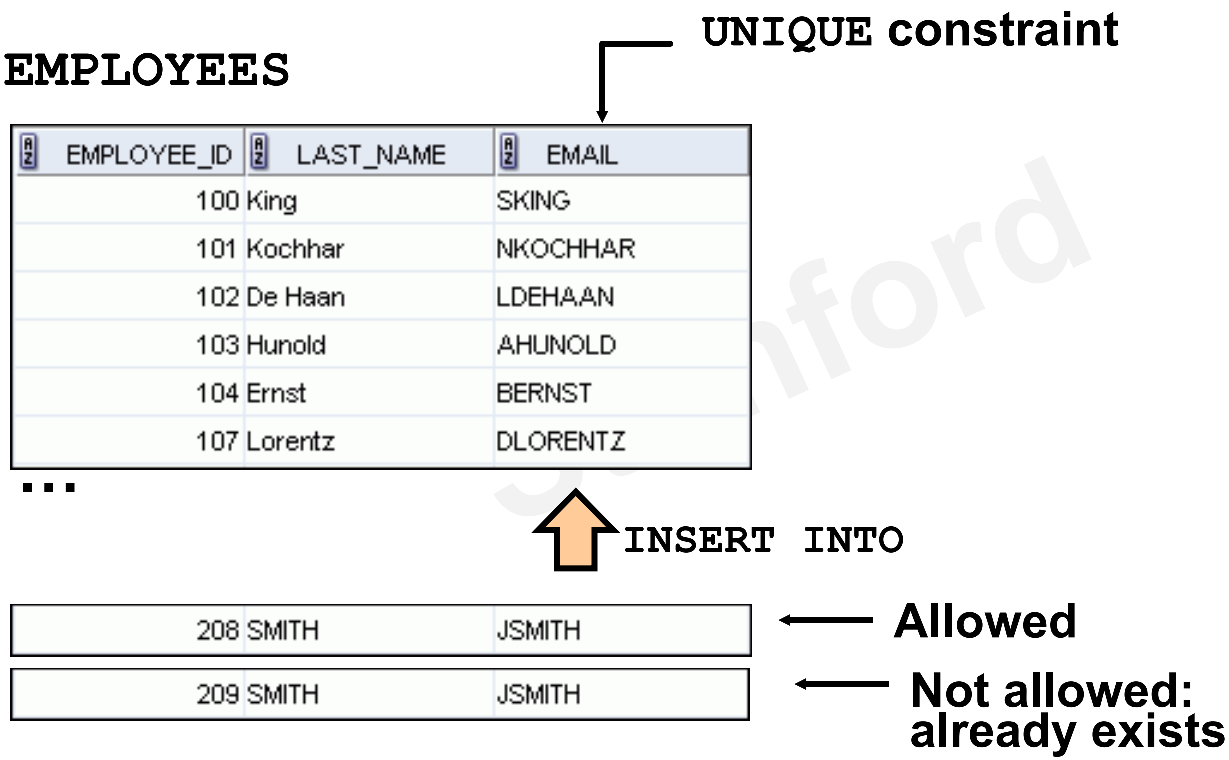
…

**NOT NULL constraint (Primary Key enforces NOT NULL constraint.)**

**NOT NULL constraint**

**Absence of NOT NULL constraint (Any row can contain a null value for this column.)**

# UNIQUE Constraint



EMPLOYEES

UNIQUE constraint

| EMPLOYEE_ID | LAST_NAME | EMAIL |
|---|---|---|
| 100 | King | SKING |
| 101 | Kochhar | NKOCHHAR |
| 102 | De Haan | LDEHAAN |
| 103 | Hunold | AHUNOLD |
| 104 | Ernst | BERNST |
| 107 | Lorentz | DLORENTZ |

...

INSERT INTO

| | | |
|---|---|---|
| 208 | SMITH | JSMITH |

Allowed

| | | |
|---|---|---|
| 209 | SMITH | JSMITH |

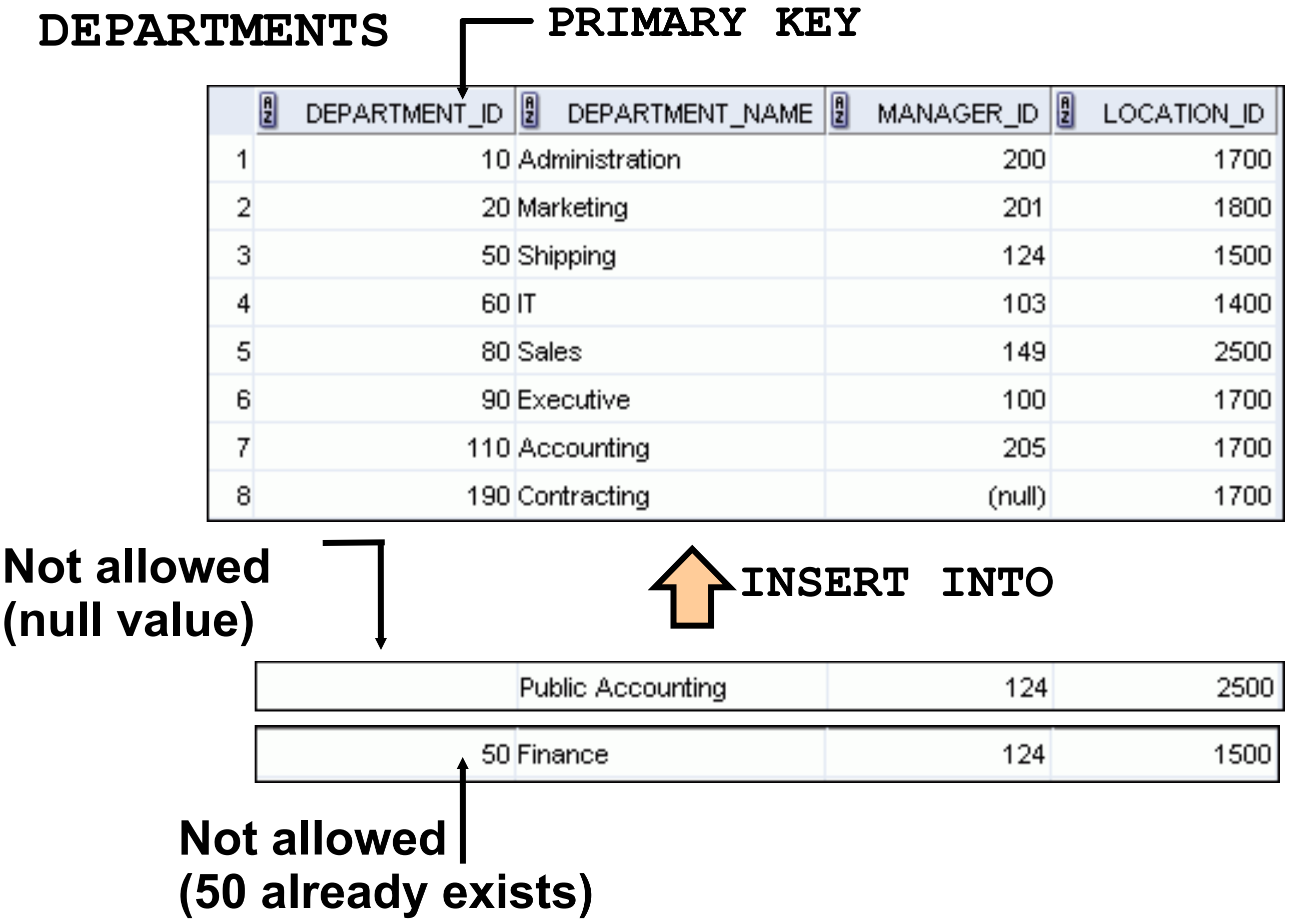
Not allowed:
already exists

# UNIQUE Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

# PRIMARY KEY Constraint



DEPARTMENTS

PRIMARY KEY

| | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
| 1 | 10 | Administration | 200 | 1700 |
| 2 | 20 | Marketing | 201 | 1800 |
| 3 | 50 | Shipping | 124 | 1500 |
| 4 | 60 | IT | 103 | 1400 |
| 5 | 80 | Sales | 149 | 2500 |
| 6 | 90 | Executive | 100 | 1700 |
| 7 | 110 | Accounting | 205 | 1700 |
| 8 | 190 | Contracting | (null) | 1700 |

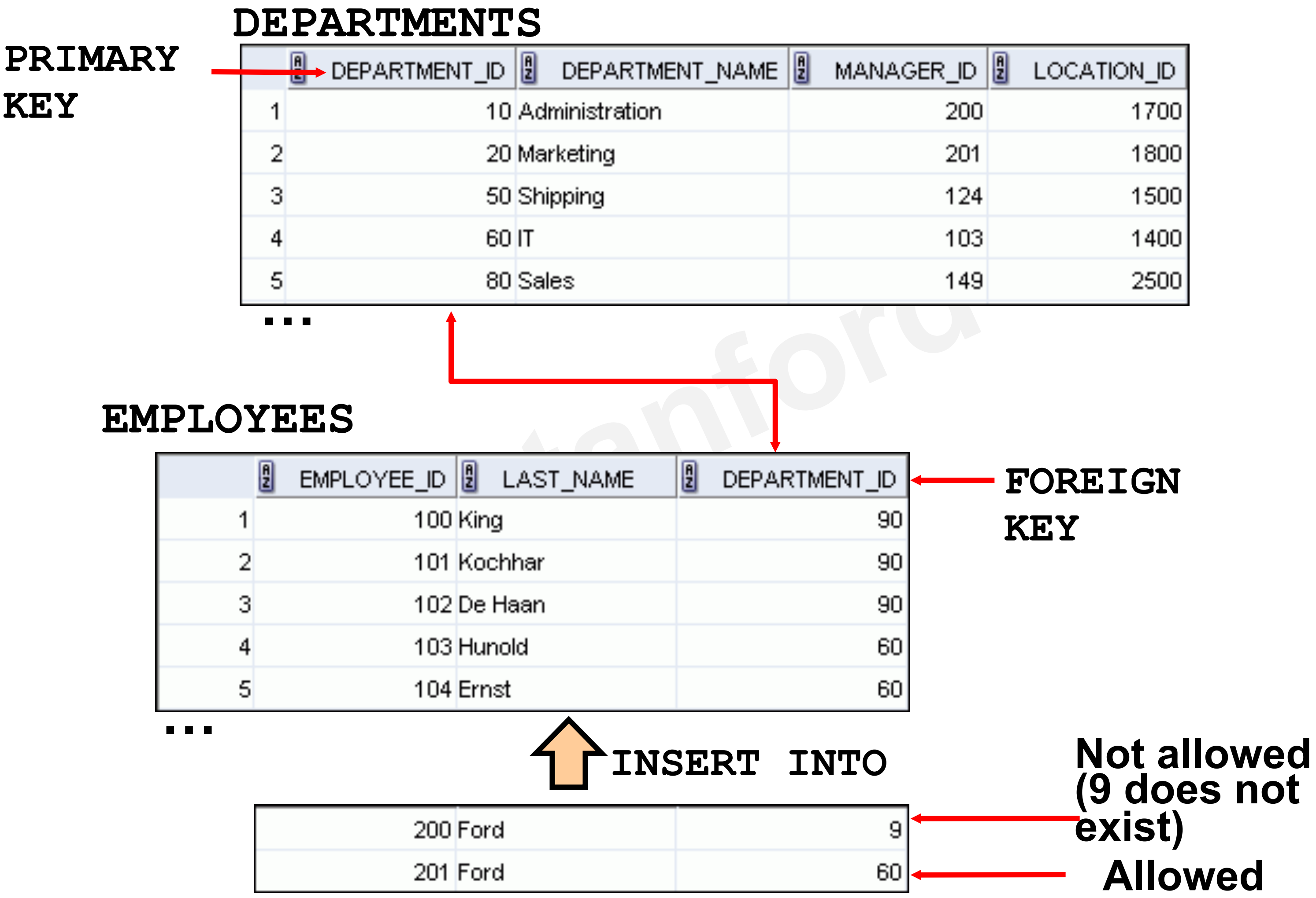
Not allowed
(null value)

INSERT INTO

| | | | |
|---|---|---|---|
| | Public Accounting | 124 | 2500 |
| 50 | Finance | 124 | 1500 |

Not allowed
(50 already exists)

# FOREIGN KEY Constraint



DEPARTMENTS

PRIMARY KEY

| | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
| 1 | 10 | Administration | 200 | 1700 |
| 2 | 20 | Marketing | 201 | 1800 |
| 3 | 50 | Shipping | 124 | 1500 |
| 4 | 60 | IT | 103 | 1400 |
| 5 | 80 | Sales | 149 | 2500 |

...

EMPLOYEES

| | EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|---|
| 1 | 100 | King | 90 |
| 2 | 101 | Kochhar | 90 |
| 3 | 102 | De Haan | 90 |
| 4 | 103 | Hunold | 60 |
| 5 | 104 | Ernst | 60 |

...

FOREIGN KEY

INSERT INTO

| | | |
|---|---|---|
| 200 | Ford | 9 |
| 201 | Ford | 60 |

Not allowed (9 does not exist)

Allowed

# FOREIGN KEY Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(
    employee_id       NUMBER(6),
    last_name         VARCHAR2(25) NOT NULL,
    email             VARCHAR2(25),
    salary            NUMBER(8,2),
    commission_pct    NUMBER(2,2),
    hire_date         DATE NOT NULL,
...
    department_id     NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

# 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
- ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted
- ON DELETE SET NULL: Converts dependent foreign key values to null

# CHECK Constraint

- Defines a condition that each row must satisfy
- The following expressions are not allowed:
  - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
  - Calls to SYSDATE, UID, USER, and USERENV functions
  - Queries that refer to other values in other rows

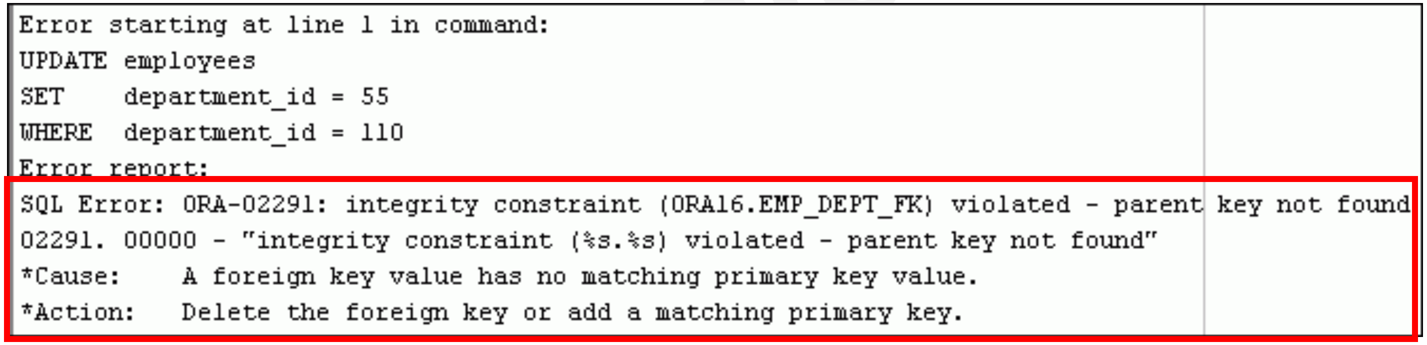
```
..., salary   NUMBER(2)
    CONSTRAINT emp_salary_min
          CHECK (salary > 0),...
```

# CREATE TABLE: Example

```
CREATE TABLE employees
    ( employee_id    NUMBER(6)
      CONSTRAINT       emp_employee_id   PRIMARY KEY
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
      CONSTRAINT       emp_last_name_nn  NOT NULL
    , email          VARCHAR2(25)
      CONSTRAINT       emp_email_nn      NOT NULL
      CONSTRAINT       emp_email_uk      UNIQUE
    , phone_number   VARCHAR2(20)
    , hire_date      DATE
      CONSTRAINT       emp_hire_date_nn  NOT NULL
    , job_id         VARCHAR2(10)
      CONSTRAINT       emp_job_nn        NOT NULL
    , salary         NUMBER(8,2)
      CONSTRAINT       emp_salary_ck     CHECK (salary>0)
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
        CONSTRAINT emp_manager_fk REFERENCES
         employees (employee_id)
    , department_id  NUMBER(4)
      CONSTRAINT       emp_dept_fk       REFERENCES
         departments (department_id));
```

# Violating Constraints

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```



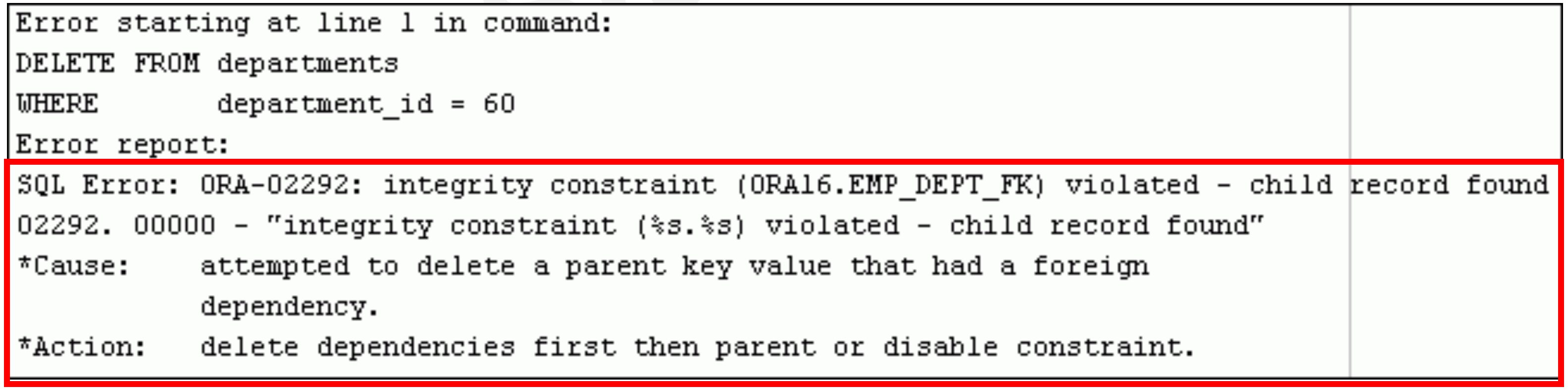
```
Error starting at line 1 in command:
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110
Error report:
SQL Error: ORA-02291: integrity constraint (ORA16.EMP_DEPT_FK) violated - parent key not found
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"
*Cause:    A foreign key value has no matching primary key value.
*Action:   Delete the foreign key or add a matching primary key.
```

Department 55 does not exist.

# Violating Constraints

You cannot delete a row that contains a primary key that is used as a foreign key in another table.

```
DELETE FROM departments
WHERE department_id = 60;
```



```
Error starting at line 1 in command:
DELETE FROM departments
WHERE       department_id = 60
Error report:
SQL Error: ORA-02292: integrity constraint (ORA16.EMP_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user's tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# Creating a Table Using a Subquery

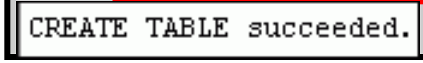
- Create a table and insert rows by combining the `CREATE TABLE` statement and the `AS` *`subquery`* option.

```
CREATE TABLE table
           [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

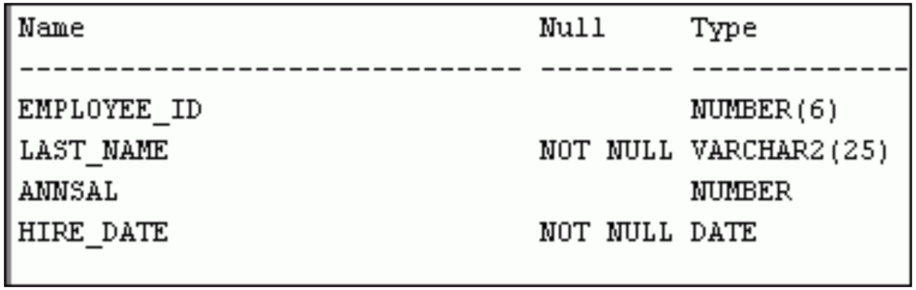
# Creating a Table Using a Subquery

```
CREATE TABLE  dept80
  AS
    SELECT   employee_id, last_name,
             salary*12 ANNSAL,
             hire_date
    FROM     employees
    WHERE    department_id = 80;
```



```
CREATE TABLE succeeded.
```

```
DESCRIBE dept80
```



```
Name                           Null     Type
------------------------------ -------- ------------
EMPLOYEE_ID                             NUMBER(6)
LAST_NAME                      NOT NULL VARCHAR2(25)
ANNSAL                                  NUMBER
HIRE_DATE                      NOT NULL DATE
```

# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user's tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change table to read-only status

# Read-Only Tables

Use the `ALTER TABLE` syntax to put a table into the read-only mode:

- Prevents DDL or DML changes during table maintenance
- Change it back into read/write mode

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

# Lesson Stanford

- Database objects
  - Naming rules
- CREATE TABLE statement:
  - Access another user’s tables
  - DEFAULT option
- Data types
- Overview of constraints: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE
  - Read-only tables
- DROP TABLE statement

# Dropping a Table

- Moves a table to the recycle bin
- Removes the table and all its data entirely if the `PURGE` clause is specified
- Invalidates dependent objects and removes object privileges on the table

```
DROP TABLE dept80;
```

```
DROP TABLE dept80 succeeded.
```

# Summary

In this lesson, you should have learned how to use the `CREATE TABLE` statement to create a table and include constraints:

- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation
- Describe how schema objects work

# Practice: Overview

This practice covers the following topics:

- Creating new tables
- Creating a new table by using the `CREATE TABLE AS` syntax
- Verifying that tables exist
- Setting a table to read-only status
- Dropping tables

# Objectives

After completing this lesson, you should be able to do the following:

- Describe the logical structure of tables
- Managing Schema Objects
- **Modeling and Accessing Relational Data**

# Modeling and Accessing Relational Data

Data is organized in a relational database as *tables*—two-dimensional matrices made up of columns and rows.

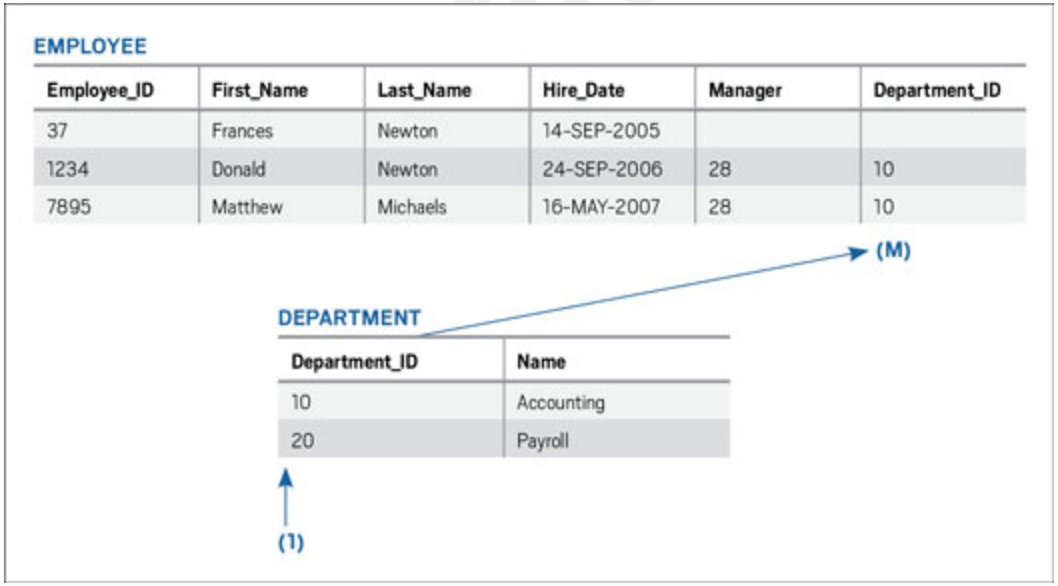
A table's *primary key*, enforced by a primary key constraint (which will be defined in a future article in this series), is a column or combination of columns that ensures that every row in a table is uniquely identified.

Two tables that have a common column are said to have a *relationship* between them; the common column is the *foreign key* in one table and the primary key in the other.

# Modeling and Accessing Relational Data

## One-to-many (1:M)

The most common type of relationship cardinality is a 1:M relationship. The common column is DEPARTMENT_ID (which is the primary key in the **DEPARTMENT** table and the foreign key in the **EMPLOYEE** table).



EMPLOYEE

| Employee_ID | First_Name | Last_Name | Hire_Date | Manager | Department_ID |
|---|---|---|---|---|---|
| 37 | Frances | Newton | 14-SEP-2005 | | |
| 1234 | Donald | Newton | 24-SEP-2006 | 28 | 10 |
| 7895 | Matthew | Michaels | 16-MAY-2007 | 28 | 10 |

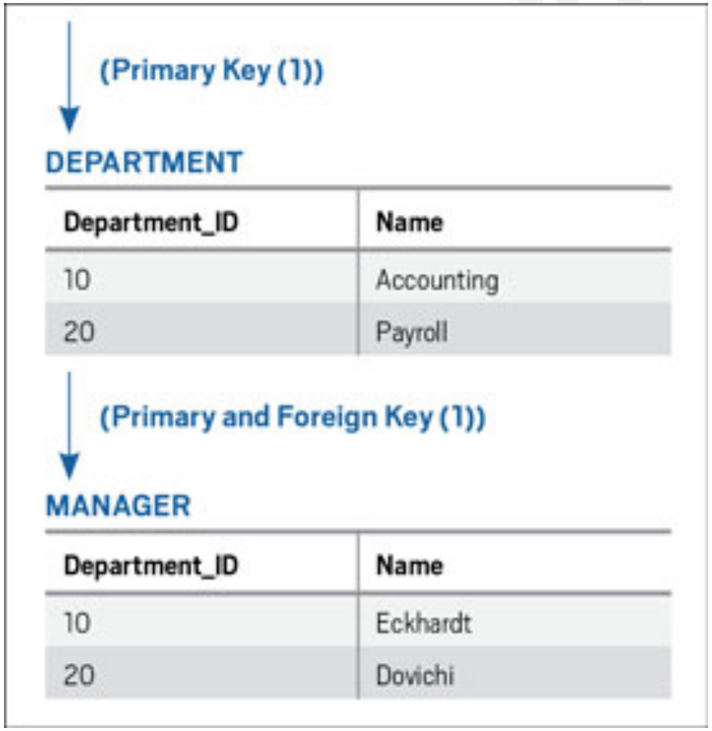
DEPARTMENT

| Department_ID | Name |
|---|---|
| 10 | Accounting |
| 20 | Payroll |

# Modeling and Accessing Relational Data

## One-to-one (1:1)

1:1 relationship between the **DEPARTMENT** table and the **MANAGER** table. For every row in the **DEPARTMENT** table, only one matching row exists in the **MANAGER** table.



(Primary Key (1))

DEPARTMENT

| Department_ID | Name |
|---|---|
| 10 | Accounting |
| 20 | Payroll |

(Primary and Foreign Key (1))

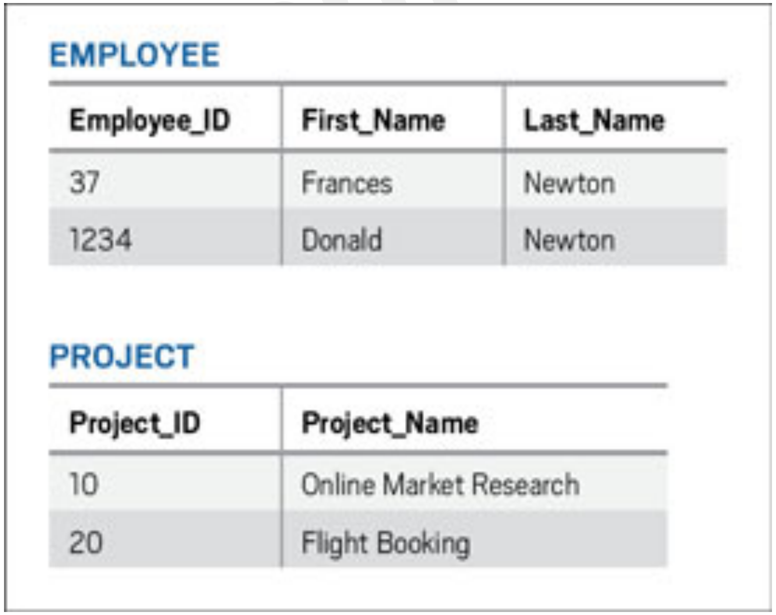
MANAGER

| Department_ID | Name |
|---|---|
| 10 | Eckhardt |
| 20 | Dovichi |

# Modeling and Accessing Relational Data

## Many-to-many (M:M)

Consider the **EMPLOYEE** and **PROJECT** tables. The business rule is as follows: *One employee can be assigned to multiple projects, and one project can be supported by multiple employees.* Therefore, it is necessary to create an M:M relationship to link these two tables.



EMPLOYEE

| Employee_ID | First_Name | Last_Name |
|---|---|---|
| 37 | Frances | Newton |
| 1234 | Donald | Newton |

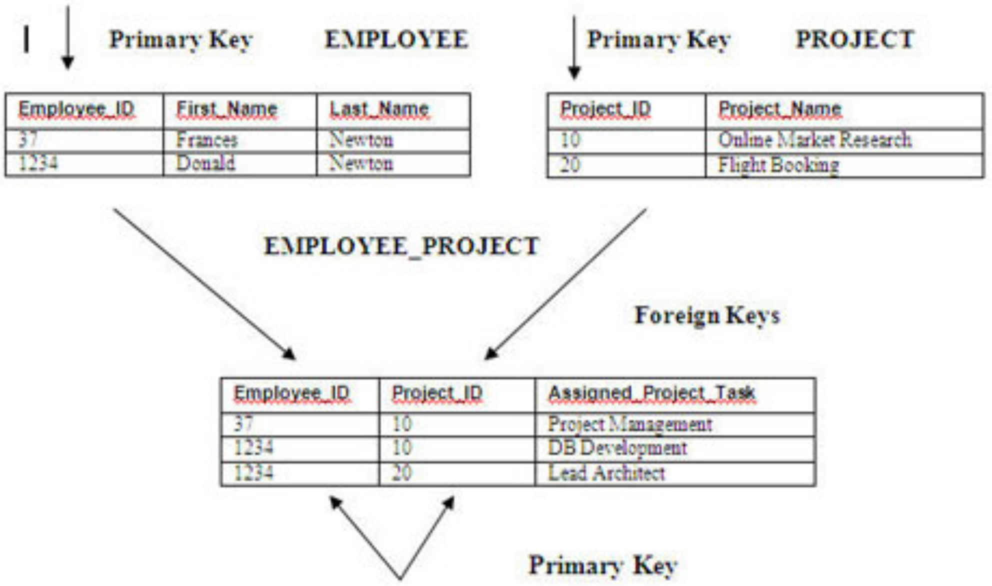
PROJECT

| Project_ID | Project_Name |
|---|---|
| 10 | Online Market Research |
| 20 | Flight Booking |

# Modeling and Accessing Relational Data

## Many-to-many (M:M)

To support the relational database model, an M:M relationship must be resolved into 1:M relationships.



Primary Key EMPLOYEE

| Employee_ID | First_Name | Last_Name |
|---|---|---|
| 37 | Frances | Newton |
| 1234 | Donald | Newton |

Primary Key PROJECT

| Project_ID | Project_Name |
|---|---|
| 10 | Online Market Research |
| 20 | Flight Booking |

EMPLOYEE_PROJECT

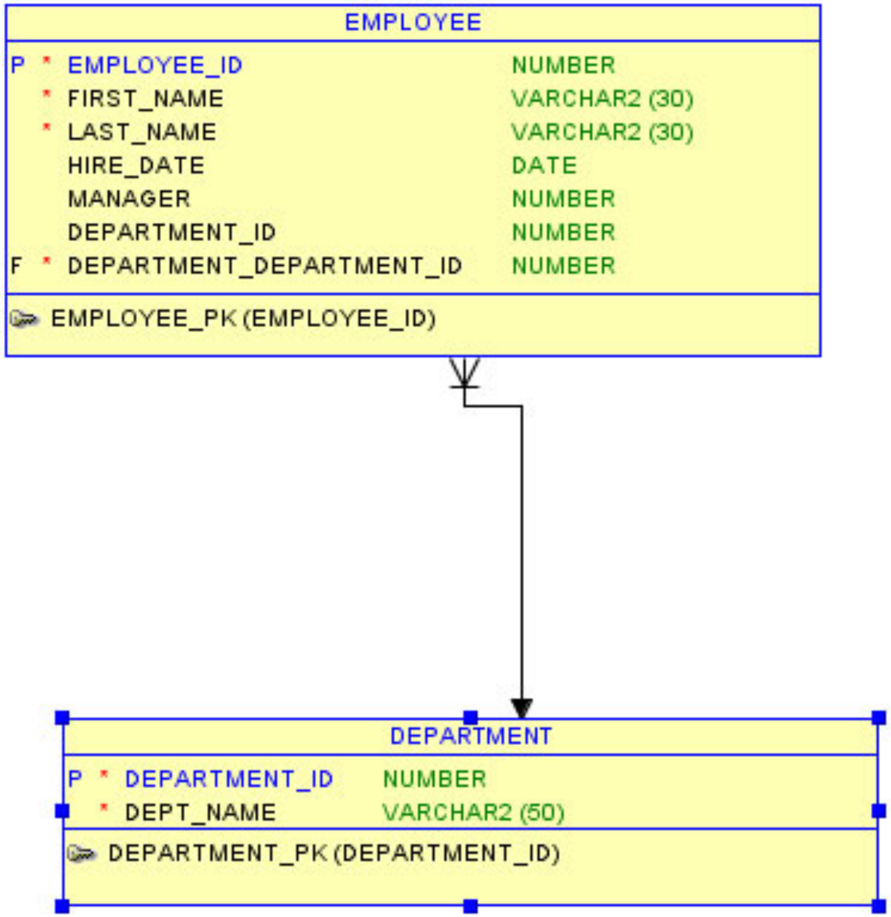
Foreign Keys

| Employee_ID | Project_ID | Assigned_Project_Task |
|---|---|---|
| 37 | 10 | Project Management |
| 1234 | 10 | DB Development |
| 1234 | 20 | Lead Architect |

Primary Key

Figure 4: Associative EMPLOYEE_PROJECT table that resolves the M:M relationship

# Modeling and Accessing Relational Data



EMPLOYEE

| | | | |
|---|---|---|---|
| P | * | EMPLOYEE_ID | NUMBER |
| | * | FIRST_NAME | VARCHAR2 (30) |
| | * | LAST_NAME | VARCHAR2 (30) |
| | | HIRE_DATE | DATE |
| | | MANAGER | NUMBER |
| | | DEPARTMENT_ID | NUMBER |
| F | * | DEPARTMENT_DEPARTMENT_ID | NUMBER |

EMPLOYEE_PK (EMPLOYEE_ID)

DEPARTMENT

| | | | |
|---|---|---|---|
| P | * | DEPARTMENT_ID | NUMBER |
| | * | DEPT_NAME | VARCHAR2 (50) |

DEPARTMENT_PK (DEPARTMENT_ID)

# Practice

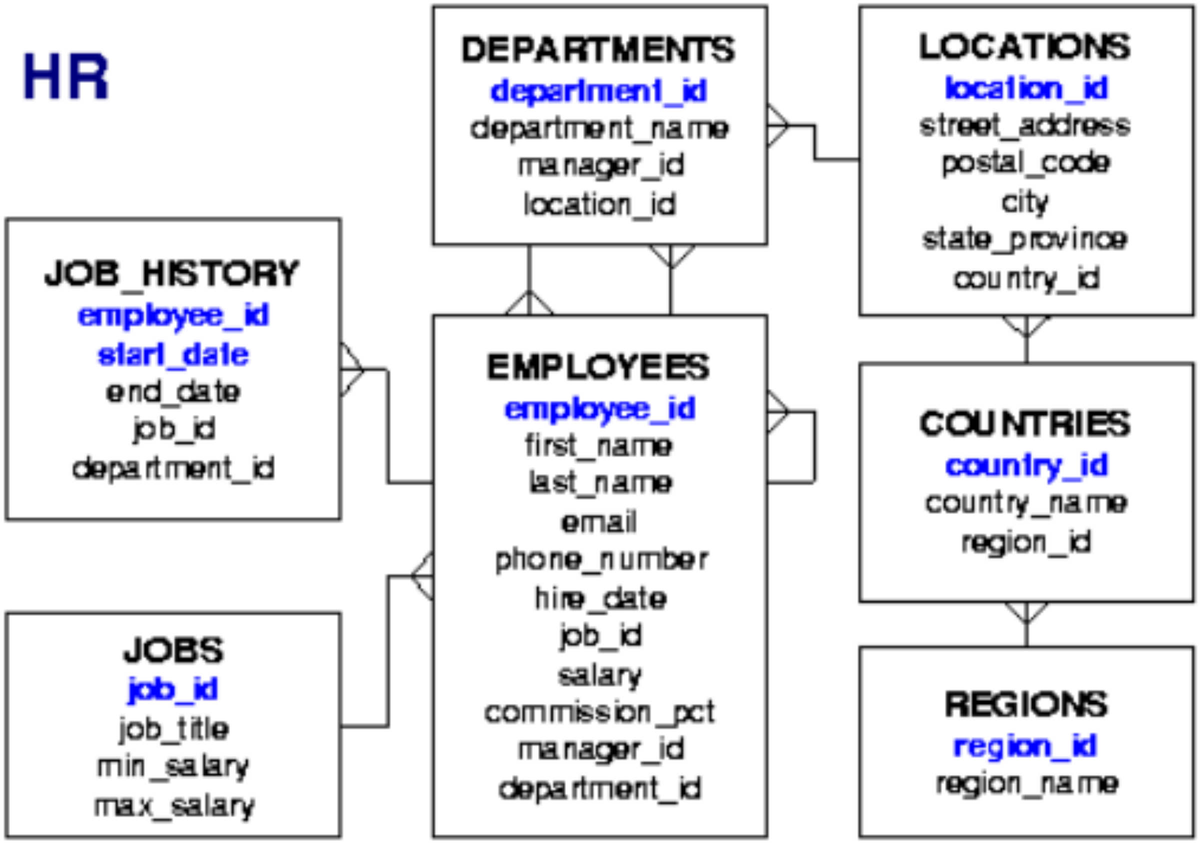
## Working with Tables

This practice covers the following:

- Creating tables andindexes
- Modifying tables
- Dropping a table
- Creating a view

# Practice

## Creating an Oracle Database



HR

DEPARTMENTS
department_id
department_name
manager_id
location_id

LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id

JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id

EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id

COUNTRIES
country_id
country_name
region_id

JOBS
job_id
job_title
min_salary
max_salary

REGIONS
region_id
region_name

# THANK YOU !

Stanford – Dạy kinh nghiệm lập trình