

Sistemas de Base de Datos I

MODELO RELACIONAL

Dr. Gustavo Zepeda Valles

Zacatecas, Zac. 25 de Abril de 2022

1

Modelo Relacional

- SQL (STRUCTURED QUERY LANGUAGE)
LENGUAJE ESTRUCTURADO DE CONSULTAS
- SQL soporta DDL y DML
- Sistema Manejador de Base de Datos: ORACLE

2

Modelo Relacional

DDL

3

Modelo Relacional

Objetos de Oracle

- **TABLAS:** Unidades básicas de almacenamiento.
- **VISTAS:** Representación lógica de un subconjunto de datos.
- **SECUENCIAS:** Genera valores numéricos.
- **ÍNDICES:** Mejoran el rendimiento de las consultas.
- **SINÓNIMOS:** Nombres alternos a los objetos.

4

Modelo Relacional

Reglas para nombrar a las tablas y columnas:

- Deben iniciar con una letra
- Longitud del nombre de 1 a 30 caracteres
- a-z, A-Z, 0-9, \$, _, #
- No se permiten nombres duplicados
- No se permiten palabras reservadas

5

Modelo Relacional

Tipos de Datos

- **CHAR**(Longitud)
- **VARCHAR2**(Longitud)
- **NUMBER**(Precisión, Escala)
- **DATE**

6

Modelo Relacional

CONSTRAINT (Restricciones)

- NOT NULL (NN)
- UNIQUE (UK)
- PRIMARY KEY (PK)
- FOREIGN KEY (FK)
- CHECK (CK)

7

Modelo Relacional

Las restricciones se pueden crear en dos momentos diferentes:

- Al momento de crear la tabla (CREATE TABLE)
- Después de creada la tabla (ALTER TABLE)

Existen dos niveles para definir las restricciones al momento de crear la tabla:

- Nivel columna
- Nivel tabla

8

Modelo Relacional

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, ...),
```

9

Modelo Relacional

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),  
  ...);
```

- 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));
```

10

Modelo Relacional

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));
```

11

Modelo Relacional

CREATE TABLE: Example

```
CREATE TABLE employees  
(  
  employee_id NUMBER(6)  
  CONSTRAINT emp_employee_id PRIMARY KEY  
  , first_name VARCHAR2(20)  
  , last_name VARCHAR2(25)  
  , 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));
```

12

Modelo Relacional

Nomenclatura para asignar un nombre a las restricciones

- **Primary Key:** Nombre de la tabla, un guion bajo y PK

Ejemplo:

Tabla Alumno: alumno_pk

- **Not Null:** Identificador de la tabla, un guion bajo, identificador del campo, guion bajo y NN

Ejemplo:

Tabla alumno y Campo apaterno: al_ap_nn

13

Modelo Relacional

- **Unique:** Identificador de la tabla, un guion bajo, identificador del campo, guion bajo y UK

Ejemplo:

Tabla alumno y Campo email: al_email_uk

- **Check:** Identificador de la tabla, un guion bajo, identificador del campo, guion bajo y CK

Ejemplo:

Tabla alumno y Campo edad: al_ed_ck

14

Modelo Relacional

- **Foreign Key:** Identificador de la tabla origen, guion bajo, identificador de la tabla destino, guion bajo y FK

Ejemplo:

Relación entre la tabla alumno y asignatura:
alumno_asignatura_fk

15

Modelo Relacional

DML

16

- Comando: **INSERT**
- Permite insertar registros en una tabla
- `INSERT INTO NOMBRE_TABLA [columnas a insertar separadas por coma] VALUES (lista de valores separados por coma y en el orden de las columnas a insertar o si se omiten, el orden de la tabla)`
- Ejemplos:
- `Insert into Alumno (Matricula, Apaterno, Amaterno, Nombre, Edad) Values ('20202020','PEREZ','FLORES','LUIS',21);`
- `Insert into Alumno Values ('20202030','DIAZ','HERNANDEZ','JORGE',22);`
- Nota: Los valores de los tipos de dato **CHAR**, **VARCHAR2** y **DATE** van entre comilla sencilla.

17

- Comando: **UPDATE**
- Permite actualizar datos en una tabla
- `UPDATE nombre_tabla SET columna a modificar = nuevo valor [, siguiente columna a modificar] WHERE condición_de_búsqueda`
- Ejemplo:
- `Update Alumno Set Edad=22 Where Matricula ='20202020'`

18

- Comando: **DELETE**
- Permite ELIMINAR datos (Registros) de una tabla
- **DELETE [FROM] NOMBRE_TABLA WHERE CONDICIÓN**
- Ejemplos;
- **DELETE FROM Alumno;**
- **DELETE Alumno;**

19

- Comando: **SELECT**
- Permite consultar datos de una o más tablas.
- **(SELECT [LISTA DEL SELECT] ---COLUMNAS O EXPRESIÓN A CONSULTAR**
- **FROM [LISTA DE TABLAS] -- TABLAS A CONSULTAR)**
- **WHERE [CONDICIÓN DE BÚSQUEDA] -- REGISTROS A CONSULTAR**
- **GROUP BY [LISTA DE CAMPOS] -- AGRUPACIÓN DE DATOS**
- **HAVING [EXPRESIÓN] -- FILTRO SOBRE LOS DATOS AGRUPADOS**
- **ORDER BY [LISTA DE CAMPOS] -- ORDENAR EL RESULTADO DE LA CONSULTA**

20

- **TABLA DE EJEMPLO:** EMPLEADO (NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, EDAD, TURNO, MUNICIPIO)
- **//LISTA DEL SELECT**
- **SELECT * FROM EMPLEADO;**
- **SELECT NOMBRE,APELLIDO_PAT, APELLIDO_MAT FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO + 100 FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO + 100 * 12 FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, (SALARIO + 100) * 12 FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 AS ANUAL FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 SALARIO_ANUAL FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 "SALARIO ANUAL" FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 "Salario Anual" FROM EMPLEADO;**
- **SELECT NOMBRE, APELLIDO_PAT, APELLIDO_MAT, SALARIO, SALARIO * 12 anual FROM EMPLEADO;**

21

- SELECT NOMBRE || APELLIDO_PAT || APELLIDO_MAT FROM EMPLEADO;
- SELECT NOMBRE || ' ' || APELLIDO_PAT || ' ' || APELLIDO_MAT FROM EMPLEADO;
- SELECT NOMBRE || ' ' || APELLIDO_PAT || ' ' || APELLIDO_MAT "NOMBRE COMPLETO" FROM EMPLEADO;
- SELECT 'NOMBRE COMPLETO: ' || NOMBRE || ' ' || APELLIDO_PAT || ' ' || APELLIDO_MAT "NOMBRE COMPLETO" FROM EMPLEADO;
- SELECT 'NOMBRE: ' || NOMBRE || ' ' || APELLIDO_PAT || ' ' || APELLIDO_MAT "NOMBRE COMPLETO" FROM EMPLEADO;
- SELECT Q'NOMBRE:S' || NOMBRE || ' ' || APELLIDO_PAT || ' ' || APELLIDO_MAT "NOMBRE COMPLETO" FROM EMPLEADO;
- --|| {} 0 <>
- SELECT EDAD FROM EMPLEADO;
- SELECT TURNO FROM EMPLEADO;
- SELECT MUNICIPIO FROM EMPLEADO;

22

- **//FUNCIONES SOBRE VALORES NULL**
- SELECT EDAD FROM EMPLEADO –para estos ejercicios se considera que la edad permite valor NULL, en su caso buscar una columna que permita valor NULL
- ~~SELECT NVL(EDAD,0) FROM EMPLEADO~~
- SELECT EDAD,NVL2(EDAD,EDAD+1,30) FROM EMPLEADO
- SELECT APELLIDO_PAT,APELLIDO_MAT,NULLIF(LENGTH(APELLIDO_PAT),LENGTH(APELLIDO_MAT))
- FROM EMPLEADO
- SELECT COALESCE(TELEFONO,APELLIDO_PAT) FROM EMPLEADO;

23

- **//CASE**
- SELECT * FROM EMPLEADO
- SELECT ID_EMPLEADO,SALARIO,
- CASE WHEN SALARIO >=2000 AND SALARIO <=3000 THEN 'GANA MAL'
- WHEN SALARIO >3000 AND SALARIO <=4000 THEN 'GANA REGULAR'
- ELSE 'GANA BIEN' END AS NIVEL_SALARIO
- FROM EMPLEADO

24

- **//FUNCIONES DE RENGLÓN SIMPLE**
-
- SELECT LOWER(APELLIDO_PAT) FROM EMPLEADO
- SELECT UPPER(APELLIDO_PAT) FROM EMPLEADO
- SELECT INITCAP(APELLIDO_PAT) FROM EMPLEADO
- SELECT SUBSTR(APELLIDO_PAT,3) FROM EMPLEADO
- SELECT SUBSTR(APELLIDO_PAT,3,2) FROM EMPLEADO
- SELECT APELLIDO_PAT, SUBSTR(APELLIDO_PAT,-4,1) FROM EMPLEADO
- SELECT APELLIDO_PAT, SUBSTR(APELLIDO_PAT,-2,1) FROM EMPLEADO
- SELECT APELLIDO_PAT, SUBSTR(APELLIDO_PAT,-2,2) FROM EMPLEADO
- SELECT APELLIDO_PAT, SUBSTR(APELLIDO_PAT,-2) FROM EMPLEADO
- SELECT LENGTH(APELLIDO_PAT) FROM EMPLEADO
- SELECT INSTR(APELLIDO_PAT,7) FROM EMPLEADO
- SELECT APELLIDO_PAT, INSTR(APELLIDO_PAT,'U',2,1) FROM EMPLEADO
- SELECT APELLIDO_PAT, INSTR(APELLIDO_PAT,'E') FROM EMPLEADO
- SELECT APELLIDO_PAT, INSTR(APELLIDO_PAT,'E',1,2) FROM EMPLEADO
- SELECT APELLIDO_PAT, INSTR(APELLIDO_PAT,'E',1,1) FROM EMPLEADO
- SELECT LPAD(APELLIDO_PAT,10,' ') FROM EMPLEADO
- SELECT RPAD(APELLIDO_PAT,10,' ') FROM EMPLEADO

25

- SELECT TRIM(APELLIDO_PAT) FROM EMPLEADO
- SELECT LTRIM(APELLIDO_PAT) FROM EMPLEADO
- SELECT RTRIM(APELLIDO_PAT) FROM EMPLEADO
- SELECT TRIM('L' FROM APELLIDO_PAT) FROM EMPLEADO
- SELECT TRIM(LEADING 'L' FROM APELLIDO_PAT) FROM EMPLEADO
- SELECT TRIM(TRAILING 'A' FROM APELLIDO_PAT) FROM EMPLEADO
- SELECT TRIM(BOTH 'A' FROM APELLIDO_PAT) FROM EMPLEADO
- SELECT REPLACE(APELLIDO_PAT,'U','A') FROM EMPLEADO
- SELECT SALARIO FROM EMPLEADO
- SELECT SALARIO, ROUND(SALARIO,1) FROM EMPLEADO
- SELECT SALARIO, TRUNC(SALARIO,1) FROM EMPLEADO
- SELECT SALARIO, MOD(SALARIO,3) FROM EMPLEADO
- SELECT SALARIO, ROUND(SALARIO,-1) FROM EMPLEADO
- SELECT SALARIO, ROUND(SALARIO,-2) FROM EMPLEADO
- SELECT SALARIO, ROUND(SALARIO,-3) FROM EMPLEADO
- SELECT SALARIO, ROUND(SALARIO,-4) FROM EMPLEADO

26

- **//FUNCIONES DE MÚLTIPLES RENGLONES**
- SELECT SALARIO FROM EMPLEADO
- SELECT AVG(SALARIO) FROM EMPLEADO
- SELECT EDAD FROM EMPLEADO
- SELECT AVG(EDAD) FROM EMPLEADO
- SELECT AVG(NVL(EDAD,0)) FROM EMPLEADO
- SELECT COUNT(*) FROM EMPLEADO
- SELECT COUNT(EDAD) FROM EMPLEADO
- SELECT COUNT(DISTINCT EDAD) FROM EMPLEADO
- SELECT MAX(EDAD) FROM EMPLEADO
- SELECT MIN(EDAD) FROM EMPLEADO
- SELECT SUM(SALARIO) FROM EMPLEADO
- SELECT VARIANCE(EDAD) FROM EMPLEADO
- SELECT STDDEV(EDAD) FROM EMPLEADO

27

- **//CLAUSULA WHERE**
- SELECT * FROM EMPLEADO WHERE ID_EMPLEADO =1
- SELECT * FROM EMPLEADO WHERE SALARIO > 3000
- SELECT * FROM EMPLEADO WHERE SALARIO >= 3000 AND SALARIO <=5000
- SELECT * FROM EMPLEADO WHERE SALARIO BETWEEN 3000 AND 5000
- ~~SELECT * FROM EMPLEADO WHERE SALARIO NOT BETWEEN 3000 AND 5000~~
- SELECT * FROM EMPLEADO WHERE SALARIO IN (1000,2000,3000,4000,5000)
- SELECT * FROM EMPLEADO WHERE SALARIO = 1000 OR SALARIO = 2000 OR SALARIO =3000 OR SALARIO = 4000 OR SALARIO = 5000
- SELECT * FROM EMPLEADO WHERE SALARIO NOT IN (1000,2000,3000,4000,5000)
- SELECT * FROM EMPLEADO WHERE APELLIDO_PAT LIKE 'L%'
- SELECT * FROM EMPLEADO WHERE APELLIDO_PAT LIKE '%A'
- SELECT * FROM EMPLEADO WHERE APELLIDO_PAT LIKE '%A%'
- SELECT * FROM EMPLEADO WHERE APELLIDO_PAT LIKE '_U%'
- SELECT * FROM EMPLEADO WHERE APELLIDO_PAT NOT LIKE '%A%'
- SELECT * FROM EMPLEADO WHERE APELLIDO_MAT IS NULL
- SELECT * FROM EMPLEADO WHERE APELLIDO_MAT IS NOT NULL

28

- **//ORDER BY**
- SELECT * FROM EMPLEADO ORDER BY APELLIDO_PAT
- SELECT * FROM EMPLEADO ORDER BY APELLIDO_PAT DESC
- ~~SELECT * FROM EMPLEADO ORDER BY 2~~ _____
- SELECT ID_EMPLEADO, APELLIDO_PAT AS PATERNO FROM EMPLEADO ORDER BY PATERNO
- SELECT * FROM EMPLEADO ORDER BY APELLIDO_PAT, APELLIDO_MAT, NOMBRE
- SELECT * FROM EMPLEADO ORDER BY APELLIDO_MAT NULLS FIRST
- SELECT * FROM EMPLEADO ORDER BY APELLIDO_MAT NULLS LAST

29

- **//VARIABLES DE SUSTITUCIÓN**
- SELECT * FROM EMPLEADO ORDER BY '&ORDEN'
- SELECT * FROM EMPLEADO WHERE &CONDICION ORDER BY '&COLUMNA' _____
- SELECT ID_EMPLEADO, APELLIDO_PAT, APELLIDO_MAT, NOMBRE, &&COLUMNA
- FROM EMPLEADO ORDER BY &COLUMNA
- DEFINE SALARIO =5000
- SELECT * FROM EMPLEADO WHERE SALARIO =&SALARIO
- UNDEFINE SALARIO;

30

- **//GROUP BY Y HAVING**
- SELECT EDAD, COUNT(*) FROM EMPLEADO
GROUP BY EDAD
- SELECT MUNICIPIO, COUNT(*) FROM EMPLEADO
GROUP BY MUNICIPIO
- SELECT MUNICIPIO, COUNT(*) FROM EMPLEADO
GROUP BY MUNICIPIO HAVING COUNT(*) > 1
- SELECT MUNICIPIO, EDAD, COUNT(*) FROM
EMPLEADO GROUP BY MUNICIPIO, EDAD ORDER
BY MUNICIPIO, EDAD
- SELECT EDAD, MUNICIPIO, COUNT(*) FROM
EMPLEADO GROUP BY EDAD, MUNICIPIO ORDER
BY EDAD, MUNICIPIO

31

- **//FROM –UNIONES**
- SELECT * FROM EMPLEADO;
- SELECT * FROM VENTA
- SELECT DISTINCT APELLIDO_PAT, APELLIDO_MAT, NOMBRE FROM EMPLEADO
NATURAL JOIN VENTA
- SELECT DISTINCT APELLIDO_PAT, APELLIDO_MAT, NOMBRE FROM EMPLEADO
JOIN VENTA USING(ID_EMPLEADO)
- SELECT DISTINCT APELLIDO_PAT, APELLIDO_MAT, NOMBRE
FROM EMPLEADO JOIN VENTA ON EMPLEADO.ID_EMPLEADO =
VENTA.ID_EMPLEADO
- SELECT DISTINCT APELLIDO_PAT, APELLIDO_MAT, NOMBRE
FROM EMPLEADO E JOIN VENTA V ON E.ID_EMPLEADO = VID_EMPLEADO
- SELECT * FROM EMPLEADO E LEFT OUTER JOIN VENTA V ON E.ID_EMPLEADO
= VID_EMPLEADO
- SELECT * FROM EMPLEADO E RIGHT OUTER JOIN VENTA V ON
E.ID_EMPLEADO = VID_EMPLEADO
- SELECT * FROM EMPLEADO E FULL OUTER JOIN VENTA V ON E.ID_EMPLEADO
= VID_EMPLEADO
- SELECT * FROM EMPLEADO E CROSS JOIN VENTA

32
