**Oracle**

-- Crear la tabla ListadeEsperaUCI (RB40) en Oracle

CREATE TABLE ListadeEsperaUCI (

idListadeEsperaUCI NUMBER PRIMARY KEY,

cedulaPac VARCHAR2(10),

FechaIngreso DATE,

HoraIngreso TIMESTAMP,

Prioridad NUMBER

);

-- Crear la tabla ListadeEsperaUCIespecial (RB41) en Oracle

CREATE TABLE ListadeEsperaUCIespecial (

idListadeEsperaUCIespecial NUMBER PRIMARY KEY,

cedulaPac VARCHAR2(10),

cedulaMed VARCHAR2(10),

FechaIngreso DATE,

HoraIngreso TIMESTAMP,

Prioridad NUMBER

);

-- Crear la tabla ListadeEsperaEventual (RB42) en Oracle

CREATE TABLE ListadeEsperaEventual (

idListadeEsperaEventual NUMBER PRIMARY KEY,

cedulaPac VARCHAR2(10),

FechaIngreso DATE,

HoraIngreso TIMESTAMP,

Prioridad NUMBER

);

///////////////////////////////////////////////////////////////////////////////////////////////

-- Insertar datos en la tabla ListadeEsperaUCI (RB40) en Oracle

INSERT INTO ListadeEsperaUCI (idListadeEsperaUCI, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', SYSDATE, CURRENT\_TIMESTAMP, 1);

-- Insertar datos en la tabla ListadeEsperaUCIespecial (RB41) en Oracle

INSERT INTO ListadeEsperaUCIespecial (idListadeEsperaUCIespecial, cedulaPac, cedulaMed, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', '9876543210', SYSDATE, CURRENT\_TIMESTAMP, 2);

-- Insertar datos en la tabla ListadeEsperaEventual (RB42) en Oracle

INSERT INTO ListadeEsperaEventual (idListadeEsperaEventual, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', SYSDATE, CURRENT\_TIMESTAMP, 3);

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Consulta para contar el número de camas disponibles en la entidad 'UCI' en Oracle

SELECT COUNT(\*) AS UCIdisponibles

FROM UCI

WHERE estadoUCI = 'Si';

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**CONSULTAS (ESTADÍSTICAS)**

-- Camas ocupadas en un día

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= TRUNC(SYSDATE)

AND fechaOcupacion < TRUNC(SYSDATE) + 1;

-- Camas ocupadas en una semana

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= TRUNC(SYSDATE) - 6

AND fechaOcupacion < TRUNC(SYSDATE) + 1;

-- Camas ocupadas en un mes

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'MONTH')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'MONTH'), 1);

-- Camas ocupadas en un año

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'YEAR')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'YEAR'), 12);

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones ocupadas en un día

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= TRUNC(SYSDATE)

AND fechaOcupacion < TRUNC(SYSDATE) + 1;

-- Habitaciones ocupadas en una semana

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= TRUNC(SYSDATE) - 6

AND fechaOcupacion < TRUNC(SYSDATE) + 1;

-- Habitaciones ocupadas en un mes

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'MM')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), 1);

-- Habitaciones ocupadas en un año

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'YEAR')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'YEAR'), 12);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs ocupadas en un día

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= TRUNC(SYSDATE)

AND fechaOcupacion < TRUNC(SYSDATE) + 1

-- UCIs ocupadas en una semana

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= TRUNC(SYSDATE) - 6

AND fechaOcupacion < TRUNC(SYSDATE) + 1

-- UCIs ocupadas en un mes

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'MM')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), 1)

-- UCIs ocupadas en un año

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= TRUNC(SYSDATE, 'YEAR')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'YEAR'), 12)

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Camas disponibles en un día

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= TRUNC(SYSDATE)

AND fechaOcupacion < TRUNC(SYSDATE) + 1

-- Camas disponibles en una semana

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= TRUNC(SYSDATE) - 6

AND fechaOcupacion < TRUNC(SYSDATE) + 1

-- Camas disponibles en un mes

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= TRUNC(SYSDATE, 'MM')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), 1)

-- Camas disponibles en un semestre

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= TRUNC(SYSDATE, 'Q')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'Q'), 3)

-- Camas disponibles en un año

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= TRUNC(SYSDATE, 'YYYY')

AND fechaOcupacion < ADD\_MONTHS(TRUNC(SYSDATE, 'YYYY'), 12)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones disponibles en un día

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(SYSDATE)

AND fechaocupacion < TRUNC(SYSDATE) + 1

-- Habitaciones disponibles en una semana

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(SYSDATE) - 6

AND fechaocupacion < TRUNC(SYSDATE) + 1

-- Habitaciones disponibles en un mes

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'MM')

AND fechaocupacion < TRUNC(SYSDATE, 'MM') + INTERVAL '1' MONTH

-- Habitaciones disponibles en un semestre

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'Q')

AND fechaocupacion < TRUNC(SYSDATE, 'Q') + INTERVAL '6' MONTH

-- Habitaciones disponibles en un año

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'YYYY')

AND fechaocupacion < TRUNC(SYSDATE, 'YYYY') + INTERVAL '1' YEAR

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs disponibles en un día

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(SYSDATE)

AND fechaocupacion < TRUNC(SYSDATE) + 1

-- UCIs disponibles en una semana

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(SYSDATE) - 6

AND fechaocupacion < TRUNC(SYSDATE) + 1

-- UCIs disponibles en un mes

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'MM')

AND fechaocupacion < TRUNC(SYSDATE, 'MM') + INTERVAL '1' MONTH

-- UCIs disponibles en un semestre

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'Q')

AND fechaocupacion < TRUNC(SYSDATE, 'Q') + INTERVAL '6' MONTH

-- UCIs disponibles en un año

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(SYSDATE, 'YYYY')

AND fechaocupacion < TRUNC(SYSDATE, 'YYYY') + INTERVAL '1' YEAR

**Microsoft SQL Server**

-- Crear la tabla ListadeEsperaUCI (RB40) en Microsoft SQL Server

CREATE TABLE ListadeEsperaUCI (

idListadeEsperaUCI INT PRIMARY KEY,

cedulaPac VARCHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

-- Crear la tabla ListadeEsperaUCIespecial (RB41) en Microsoft SQL Server

CREATE TABLE ListadeEsperaUCIespecial (

idListadeEsperaUCIespecial INT PRIMARY KEY,

cedulaPac VARCHAR(10),

cedulaMed VARCHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

-- Crear la tabla ListadeEsperaEventual (RB42) en Microsoft SQL Server

CREATE TABLE ListadeEsperaEventual (

idListadeEsperaEventual INT PRIMARY KEY,

cedulaPac VARCHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

/////////////////////////////////////////////////////////////////////////////////////////////////////

-- Insertar datos en la tabla ListadeEsperaUCI (RB40) en Microsoft SQL Server

INSERT INTO ListadeEsperaUCI (idListadeEsperaUCI, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', GETDATE(), CURRENT\_TIMESTAMP, 1);

-- Insertar datos en la tabla ListadeEsperaUCIespecial (RB41) en Microsoft SQL Server

INSERT INTO ListadeEsperaUCIespecial (idListadeEsperaUCIespecial, cedulaPac, cedulaMed, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', '9876543210', GETDATE(), CURRENT\_TIMESTAMP, 2);

-- Insertar datos en la tabla ListadeEsperaEventual (RB42) en Microsoft SQL Server

INSERT INTO ListadeEsperaEventual (idListadeEsperaEventual, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', GETDATE(), CURRENT\_TIMESTAMP, 3);

////////////////////////////////////////////////////////////////////////////////////////////

-- Consulta para contar el número de camas disponibles en la entidad 'UCI' en Microsoft SQL Server

SELECT COUNT(\*) AS UCIdisponibles

FROM UCI

WHERE estadoUCI = 'Si';

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**CONSULTAS (ESTADÍSTICAS)**

-- Camas ocupadas en un día

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= CAST(GETDATE() AS DATE)

AND fechaOcupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE));

-- Camas ocupadas en una semana

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEADD(DAY, -6, CAST(GETDATE() AS DATE))

AND fechaOcupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE));

-- Camas ocupadas en un mes

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1)

AND fechaOcupacion < DATEADD(MONTH, 1, DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1));

-- Camas ocupadas en un año

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1)

AND fechaOcupacion < DATEADD(YEAR, 1, DATEFROMPARTS(YEAR(GETDATE()), 1, 1));

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones ocupadas en un día

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= CONVERT(DATE, GETDATE())

AND fechaOcupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()));

-- Habitaciones ocupadas en una semana

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(DAY, -6, CONVERT(DATE, GETDATE()))

AND fechaOcupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()));

-- Habitaciones ocupadas en un mes

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) + 1, 0);

-- Habitaciones ocupadas en un año

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) + 1, 0);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs ocupadas en un día

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= CONVERT(DATE, GETDATE())

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATE, GETDATE()))

-- UCIs ocupadas en una semana

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(dd, -6, CONVERT(DATE, GETDATE()))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATE, GETDATE()))

-- UCIs ocupadas en un mes

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(mm, DATEDIFF(mm, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(mm, DATEDIFF(mm, 0, GETDATE()) + 1, 0)

-- UCIs ocupadas en un año

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(yy, DATEDIFF(yy, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(yy, DATEDIFF(yy, 0, GETDATE()) + 1, 0)

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Camas disponibles en un día

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaocupacion >= CONVERT(DATE, GETDATE())

AND fechaocupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()))

-- Camas disponibles en una semana

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaocupacion >= DATEADD(DAY, -6, CONVERT(DATE, GETDATE()))

AND fechaocupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()))

-- Camas disponibles en un mes

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1)

AND fechaocupacion < DATEADD(MONTH, 1, DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1))

-- Camas disponibles en un semestre

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1)

AND fechaocupacion < DATEADD(MONTH, 6, DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1))

-- Camas disponibles en un año

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1)

AND fechaocupacion < DATEADD(YEAR, 1, DATEFROMPARTS(YEAR(GETDATE()), 1, 1))

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones disponibles en un día

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= CAST(GETDATE() AS DATE)

AND fechaocupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE))

-- Habitaciones disponibles en una semana

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEADD(DAY, -6, CAST(GETDATE() AS DATE))

AND fechaocupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE))

-- Habitaciones disponibles en un mes

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1)

AND fechaocupacion < DATEADD(MONTH, 1, DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1))

-- Habitaciones disponibles en un semestre

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1)

AND fechaocupacion < DATEADD(MONTH, 6, DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1))

-- Habitaciones disponibles en un año

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1)

AND fechaocupacion < DATEADD(YEAR, 1, DATEFROMPARTS(YEAR(GETDATE()), 1, 1))

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs disponibles en un día

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= CAST(GETDATE() AS DATE)

AND fechaocupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE))

-- UCIs disponibles en una semana

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= DATEADD(DAY, -6, CAST(GETDATE() AS DATE))

AND fechaocupacion < DATEADD(DAY, 1, CAST(GETDATE() AS DATE))

-- UCIs disponibles en un mes

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1)

AND fechaocupacion < DATEADD(MONTH, 1, DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1))

-- UCIs disponibles en un semestre

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1)

AND fechaocupacion < DATEADD(MONTH, 6, DATEFROMPARTS(YEAR(GETDATE()), ((MONTH(GETDATE()) - 1) / 6) \* 6 + 1, 1))

-- UCIs disponibles en un año

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1)

AND fechaocupacion < DATEADD(YEAR, 1, DATEFROMPARTS(YEAR(GETDATE()), 1, 1))

**Sybase**

-- Crear la tabla ListadeEsperaUCI (RB40) en Sybase

CREATE TABLE ListadeEsperaUCI (

idListadeEsperaUCI INT PRIMARY KEY,

cedulaPac CHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

-- Crear la tabla ListadeEsperaUCIespecial (RB41) en Sybase

CREATE TABLE ListadeEsperaUCIespecial (

idListadeEsperaUCIespecial INT PRIMARY KEY,

cedulaPac CHAR(10),

cedulaMed CHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

-- Crear la tabla ListadeEsperaEventual (RB42) en Sybase

CREATE TABLE ListadeEsperaEventual (

idListadeEsperaEventual INT PRIMARY KEY,

cedulaPac CHAR(10),

FechaIngreso DATE,

HoraIngreso DATETIME,

Prioridad INT

);

//////////////////////////////////////////////////////////////////////////////////////////////////////

-- Insertar datos en la tabla ListadeEsperaUCI (RB40) en Sybase

INSERT INTO ListadeEsperaUCI (idListadeEsperaUCI, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', GETDATE(), CURRENT TIMESTAMP, 1);

-- Insertar datos en la tabla ListadeEsperaUCIespecial (RB41) en Sybase

INSERT INTO ListadeEsperaUCIespecial (idListadeEsperaUCIespecial, cedulaPac, cedulaMed, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', '9876543210', GETDATE(), CURRENT TIMESTAMP, 2);

-- Insertar datos en la tabla ListadeEsperaEventual (RB42) en Sybase

INSERT INTO ListadeEsperaEventual (idListadeEsperaEventual, cedulaPac, FechaIngreso, HoraIngreso, Prioridad)

VALUES (1, '1234567890', GETDATE(), CURRENT TIMESTAMP, 3);

//////////////////////////////////////////////////////////////////////////////////////////////////////

-- Consulta para contar el número de camas disponibles en la entidad 'UCI' en Sybase

SELECT COUNT(\*) AS UCIdisponibles

FROM UCI

WHERE estadoUCI = 'Si';

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**CONSULTAS (ESTADÍSTICAS)**

-- Camas ocupadas en un día

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= CONVERT(DATE, GETDATE())

AND fechaOcupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()));

-- Camas ocupadas en una semana

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEADD(DAY, -6, CONVERT(DATE, GETDATE()))

AND fechaOcupacion < DATEADD(DAY, 1, CONVERT(DATE, GETDATE()));

-- Camas ocupadas en un mes

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) + 1, 0);

-- Camas ocupadas en un año

SELECT COUNT(\*) AS CamasOcupadas

FROM camas

WHERE estadocama = 0

AND fechaOcupacion >= DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) + 1, 0);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones ocupadas en un día

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= CONVERT(DATETIME, CONVERT(DATE, GETDATE()))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(DATE, GETDATE())))

-- Habitaciones ocupadas en una semana

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(dd, -6, CONVERT(DATETIME, CONVERT(DATE, GETDATE())))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(DATE, GETDATE())))

-- Habitaciones ocupadas en un mes

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(mm, DATEDIFF(mm, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(mm, DATEDIFF(mm, 0, GETDATE()) + 1, 0)

-- Habitaciones ocupadas en un año

SELECT COUNT(\*) AS HabitacionesOcupadas

FROM habitacion

WHERE estadoHabitacion = 0

AND fechaOcupacion >= DATEADD(yy, DATEDIFF(yy, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(yy, DATEDIFF(yy, 0, GETDATE()) + 1, 0)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs ocupadas en un día

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= CONVERT(DATETIME, CONVERT(DATE, GETDATE()))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(DATE, GETDATE())))

-- UCIs ocupadas en una semana

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(dd, -6, CONVERT(DATETIME, CONVERT(DATE, GETDATE()))))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(DATE, GETDATE()))))

-- UCIs ocupadas en un mes

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(mm, DATEDIFF(mm, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(mm, DATEDIFF(mm, 0, GETDATE()) + 1, 0)

-- UCIs ocupadas en un año

SELECT COUNT(\*) AS UCIsOcupadas

FROM uci

WHERE estadoUCI = 0

AND fechaOcupacion >= DATEADD(yy, DATEDIFF(yy, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(yy, DATEDIFF(yy, 0, GETDATE()) + 1, 0)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Camas disponibles en un día

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= CONVERT(DATETIME, CONVERT(VARCHAR(10), GETDATE(), 111))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(VARCHAR(10), GETDATE(), 111)))

-- Camas disponibles en una semana

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= DATEADD(dd, -6, CONVERT(DATETIME, CONVERT(VARCHAR(10), GETDATE(), 111)))

AND fechaOcupacion < DATEADD(dd, 1, CONVERT(DATETIME, CONVERT(VARCHAR(10), GETDATE(), 111)))

-- Camas disponibles en un mes

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= DATEADD(mm, DATEDIFF(mm, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(mm, DATEDIFF(mm, 0, GETDATE()) + 1, 0)

-- Camas disponibles en un semestre

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= DATEADD(qq, DATEDIFF(qq, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(qq, DATEDIFF(qq, 0, GETDATE()) + 3, 0)

-- Camas disponibles en un año

SELECT COUNT(\*) AS CamasDisponibles

FROM camas

WHERE estadocama = 1

AND fechaOcupacion >= DATEADD(yy, DATEDIFF(yy, 0, GETDATE()), 0)

AND fechaOcupacion < DATEADD(yy, DATEDIFF(yy, 0, GETDATE()) + 1, 0)

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- Habitaciones disponibles en un día

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(GETDATE())

AND fechaocupacion < TRUNC(GETDATE()) + 1

-- Habitaciones disponibles en una semana

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= TRUNC(GETDATE()) - 6

AND fechaocupacion < TRUNC(GETDATE()) + 1

-- Habitaciones disponibles en un mes

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEADD(mm, DATEDIFF(mm, 0, GETDATE()), 0)

AND fechaocupacion < DATEADD(mm, DATEDIFF(mm, 0, GETDATE()) + 1, 0)

-- Habitaciones disponibles en un semestre

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEADD(qq, DATEDIFF(qq, 0, GETDATE()), 0)

AND fechaocupacion < DATEADD(qq, DATEDIFF(qq, 0, GETDATE()) + 2, 0)

-- Habitaciones disponibles en un año

SELECT COUNT(\*) AS HabitacionesDisponibles

FROM habitacion

WHERE estadohabitacion = 1

AND fechaocupacion >= DATEADD(yy, DATEDIFF(yy, 0, GETDATE()), 0)

AND fechaocupacion < DATEADD(yy, DATEDIFF(yy, 0, GETDATE()) + 1, 0)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

-- UCIs disponibles en un día

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(CURRENT\_DATE, 'DD')

AND fechaocupacion < DATEADD(DAY, 1, TRUNC(CURRENT\_DATE, 'DD'))

-- UCIs disponibles en una semana

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= DATEADD(DAY, -6, TRUNC(CURRENT\_DATE, 'DD'))

AND fechaocupacion < DATEADD(DAY, 1, TRUNC(CURRENT\_DATE, 'DD'))

-- UCIs disponibles en un mes

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(CURRENT\_DATE, 'MM')

AND fechaocupacion < DATEADD(MONTH, 1, TRUNC(CURRENT\_DATE, 'MM'))

-- UCIs disponibles en un semestre

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(CURRENT\_DATE, 'QQ')

AND fechaocupacion < DATEADD(MONTH, 6, TRUNC(CURRENT\_DATE, 'QQ'))

-- UCIs disponibles en un año

SELECT COUNT(\*) AS UCIsDisponibles

FROM uci

WHERE estadouci = 1

AND fechaocupacion >= TRUNC(CURRENT\_DATE, 'YYYY')

AND fechaocupacion < DATEADD(YEAR, 1, TRUNC(CURRENT\_DATE, 'YYYY'))

Marco Teórico (extended)

* **Conceptos básicos:**

*Para comprender completamente el mundo de las bases de datos, es fundamental tener un conocimiento sólido de los conceptos básicos. Esto implica comprender la definición de datos y los elementos fundamentales de una base de datos, como las tablas, los registros y los campos. Además, es esencial familiarizarse con los diferentes tipos de datos y las restricciones que se pueden aplicar a ellos. Al comprender estos conceptos, se sienta una base sólida para trabajar con bases de datos de manera efectiva.*

**Modelos de datos:** *Existen varios modelos de datos, cada uno con su propia forma de representar la estructura y las relaciones de los datos. Entre los modelos más comunes se encuentran el modelo relacional, el modelo jerárquico y el modelo de red. Conocer estos modelos y entender cómo se utilizan en la práctica es crucial para el diseño y la gestión de bases de datos. Cada modelo tiene sus fortalezas y debilidades, y comprenderlos en detalle permite seleccionar el modelo más adecuado para un sistema de bases de datos específico.[2]*

**Normalización:** *La normalización es un proceso importante en el diseño de bases de datos. Consiste en aplicar una serie de reglas y principios para eliminar redundancias y garantizar la eficiencia y la integridad de los datos. A medida que una base de datos crece y se desarrolla, es esencial normalizar adecuadamente para evitar problemas como la duplicación de datos y la inconsistencia. Comprender los principios de la normalización y cómo aplicarlos correctamente es fundamental para diseñar y mantener bases de datos eficientes y fiables.[2]*

**Consultas y manipulación de datos:** *La consulta y manipulación de datos es una habilidad fundamental para trabajar con bases de datos. SQL (Structured Query Language) es un lenguaje comúnmente utilizado para interactuar con las bases de datos y realizar operaciones como la selección, inserción, actualización y eliminación de registros. Aprender las operaciones básicas de consulta y manipulación de datos utilizando SQL es esencial para extraer información útil de una base de datos y realizar cambios en los datos de manera segura y eficiente.[2]*

* **Sistemas Gestores de Bases de Datos (DBMS):**

*Un Sistema Gestor de Bases de Datos (DBMS) es una herramienta de software diseñada para administrar y organizar grandes volúmenes de datos de manera eficiente. Los DBMS proporcionan una interfaz para interactuar con la base de datos, lo que permite realizar tareas como la creación, modificación y consulta de datos de manera estructurada. Hay varios DBMS ampliamente utilizados en la industria, y cada uno tiene sus características y fortalezas específicas.[3]*

***SQL Server Management Studio*** *es un ejemplo de DBMS desarrollado por Microsoft, diseñado específicamente para administrar bases de datos SQL Server. Proporciona una interfaz gráfica que permite a los usuarios realizar tareas de administración, consultar bases de datos utilizando SQL y monitorear el rendimiento de la base de datos. Esta herramienta es ampliamente utilizada en entornos que utilizan Microsoft SQL Server como su sistema de gestión de bases de datos.[4]*

***SQL Developer*** *es otro DBMS popular desarrollado por Oracle Corporation. Está diseñado para trabajar con bases de datos Oracle y ofrece una amplia gama de capacidades. SQL Developer permite a los usuarios desarrollar, administrar y mantener bases de datos Oracle, incluyendo la capacidad de escribir consultas SQL, diseñar objetos de base de datos y realizar tareas de administración. Es una herramienta muy valorada por los desarrolladores y administradores de bases de datos que trabajan con Oracle.*

***ASE SAP*** *(Sybase Adaptive Server Enterprise) es un DBMS utilizado en entornos empresariales que requieren funcionalidades avanzadas para administrar grandes volúmenes de datos. Se utiliza en sectores como la banca, las telecomunicaciones y la logística, donde la fiabilidad y el rendimiento son aspectos cruciales. ASE SAP ofrece características robustas para garantizar la integridad y la disponibilidad de la información, lo que lo convierte en una opción popular para empresas que manejan grandes volúmenes de datos críticos.*

* **Modelado de datos:**

***Modelado conceptual:*** *Antes de comenzar a construir una base de datos, es esencial comprender los requisitos y las entidades clave del sistema. El modelado conceptual implica representar conceptualmente estas entidades y sus relaciones utilizando técnicas como los diagramas de entidad-relación (DER). Estos diagramas permiten visualizar la estructura lógica de la base de datos y brindan una base sólida para el diseño posterior.*

***Modelado lógico****: Una vez que se ha realizado el modelado conceptual, es necesario transformar el modelo en un formato más estructurado y detallado. El modelado lógico implica definir las tablas, los atributos y las relaciones entre las entidades de manera más precisa. Esta etapa del proceso de modelado se centra en traducir los conceptos abstractos en una estructura lógica que pueda implementarse en una base de datos real.*

***Modelado físico****: Una vez que se ha realizado el modelado lógico, es necesario diseñar la estructura física de la base de datos. Esto implica tomar decisiones sobre aspectos como la definición de índices, las claves primarias y foráneas, y otras consideraciones de rendimiento. El modelado físico tiene como objetivo optimizar el rendimiento y la eficiencia de la base de datos en función de las necesidades específicas del sistema y los requisitos de los usuarios.*

* **Diagramas de flujo:**

***Símbolos y notación****: Los diagramas de flujo utilizan una variedad de símbolos y notaciones para representar las diferentes acciones y procesos en un sistema. Estos símbolos incluyen formas geométricas como rectángulos, óvalos, diamantes y flechas, que se utilizan para representar acciones, decisiones, entrada/salida de datos, bucles y conexiones entre diferentes partes del flujo. Familiarizarse con estos símbolos y notaciones es esencial para comprender y crear diagramas de flujo efectivos.*

***Procesos y flujos****: Los diagramas de flujo permiten visualizar gráficamente los procesos y flujos de información dentro de un sistema. Estos diagramas muestran el orden y la secuencia de las actividades, así como las decisiones que se toman en el camino. Al seguir el flujo de un diagrama, es posible comprender cómo se mueven los datos y cómo interactúan diferentes componentes del sistema. Los diagramas de flujo son una herramienta invaluable para la planificación, el análisis y la documentación de los procesos de un sistema.*

* **Lenguajes de programación:**

***SQL****: El lenguaje SQL es una parte integral de trabajar con bases de datos. Adquirir conocimientos de SQL permite realizar consultas y manipulación de datos en la base de datos. Esto incluye la capacidad de crear tablas, definir restricciones y realizar operaciones avanzadas como la combinación de tablas, la agregación de datos y la creación de vistas. SQL es un lenguaje declarativo que permite interactuar con las bases de datos de manera eficiente y precisa.*

***Lenguaje de programación****: Además de SQL, es común utilizar un lenguaje de programación adecuado para implementar la lógica del sistema y la interacción con la base de datos. Los lenguajes de programación como Python, Java, C# o PHP ofrecen una amplia gama de funcionalidades y bibliotecas que facilitan el desarrollo de aplicaciones y la interacción con bases de datos. Estos lenguajes permiten realizar operaciones más complejas, como la manipulación de datos, el procesamiento de lógica empresarial y la creación de interfaces de usuario interactivas.*

* **PowerDesigner:**

*PowerDesigner, desarrollado por SAP, es una herramienta utilizada en la ingeniería de software y la gestión de datos. Permite el modelado y diseño de bases de datos, procesos empresariales y arquitecturas de sistemas. En el modelado de datos, el PowerDesigner ofrece la creación y gestión de modelos en diferentes niveles de abstracción. Facilita la representación visual de la estructura de los datos, incluyendo tablas y relaciones [5].*

*En el modelado de procesos empresariales, el PowerDesigner utiliza la notación BPMN para diseñar y documentar flujos de trabajo. Proporciona una amplia gama de símbolos y elementos gráficos para representar los procesos de manera visual. En la arquitectura de sistemas, el PowerDesigner permite diseñar y documentar la estructura y relaciones entre los componentes de un sistema. También ofrece la generación automática de diagramas y documentación [5].*

*El PowerDesigner se destaca por su capacidad de integración con otras herramientas y tecnologías. Puede conectarse a bases de datos y herramientas de administración de bases de datos, y permite importar y exportar modelos en diferentes formatos. Esto facilita la colaboración y la integración con otros flujos de trabajo. En resumen, el PowerDesigner es una herramienta versátil y potente que mejora la eficiencia y precisión en el desarrollo de sistemas de información [5].*

* **SQL:**

*SQL (Structured Query Language) es un lenguaje de consulta utilizado para interactuar con sistemas de gestión de bases de datos relacionales. Proporciona una sintaxis estándar y poderosa para realizar diversas operaciones en una base de datos. El marco teórico de SQL se centra en los conceptos fundamentales y las características clave del lenguaje. Incluye el estudio de la sintaxis y semántica de las consultas SQL, así como la comprensión de las cláusulas SELECT, INSERT, UPDATE y DELETE, que permiten realizar operaciones de consulta, inserción, actualización y eliminación de datos en una base de datos relacional. Además, el marco teórico de SQL abarca el conocimiento de las cláusulas JOIN, que facilitan la combinación de datos de varias tablas, y las subconsultas, que permiten realizar consultas anidadas para obtener resultados más complejos [6].*

* **Diseño de Bases de Datos:**

*El diseño de bases de datos es un proceso fundamental para crear estructuras de almacenamiento eficientes y organizadas que permitan gestionar y acceder a los datos de manera óptima. El diseño de bases de datos abarca varios conceptos clave. En primer lugar, se examina la normalización, que es el proceso de dividir una base de datos en estructuras más pequeñas y coherentes, eliminando la redundancia y garantizando la integridad de los datos. La normalización se basa en reglas específicas, como las formas normales, que ayudan a identificar y resolver problemas de diseño [7].*

* ***Modelo Entidad / Relación (ER)***

*El modelo Entidad-Relación (ER) es una técnica ampliamente utilizada en el diseño conceptual de bases de datos. Su objetivo principal es representar de manera gráfica las entidades, los atributos y las relaciones entre ellos. Las entidades representan objetos del mundo real, los atributos describen las características y las relaciones establecen las asociaciones entre las entidades. Este modelo proporciona una representación visual clara y concisa de la estructura de los datos, facilitando el análisis, diseño y comunicación de las bases de datos [8].*

*El modelo ER sirve como base para la creación de esquemas de base de datos y la implementación de sistemas de gestión de bases de datos relacionales. Permite definir de forma precisa las estructuras de datos, establecer las restricciones de integridad y comprender la forma en que los datos se relacionan entre sí. Además, el modelo ER es independiente del sistema de gestión de bases de datos (SGBD), lo que proporciona flexibilidad y portabilidad al diseñar bases de datos [8].*

*El modelo Entidad-Relación se divide en tres niveles: conceptual, lógico y físico. En el nivel conceptual, se captura la visión global y abstracta de los datos, centrándose en las entidades, atributos y relaciones entre ellas. El nivel lógico se encarga de transformar el modelo conceptual al modelo lógico, definiendo tablas, relaciones y restricciones de integridad. Por último, en el nivel físico se realiza la implementación física del modelo lógico en un SGBD específico, definiendo los tipos de datos, índices y configuraciones físicas de almacenamiento [8].*

* **Reglas COOD**

*Las reglas de Codd, establecidas por Edgar F. Codd, son un conjunto de principios fundamentales para garantizar la integridad y confiabilidad de los datos en las bases de datos relacionales. Estas reglas sirven como guía en el diseño y desarrollo de sistemas de bases de datos relacionales, asegurando la consistencia y calidad de los datos [10].*

1. *Regla de la Información: Se establece que toda la información en una base de datos debe estar representada de manera explícita y en una forma lógica. Esto implica que los datos deben ser estructurados y organizados de manera coherente, evitando la redundancia y la ambigüedad [10].*
2. *Regla del Acceso Garantizado: Cada dato individual debe ser accesible lógicamente mediante una combinación de nombre de tabla, clave primaria y nombre de columna. Esta regla garantiza que los datos puedan ser accedidos de manera precisa y consistente, facilitando las operaciones de consulta y manipulación [10].*
3. *Regla del Tratamiento Sistemático de Nulos: Esta regla establece que los nulos deben ser tratados de manera sistemática y consistente. Los nulos representan valores desconocidos o inexistentes, y su manejo adecuado es crucial para mantener la integridad de los datos y evitar inconsistencias [10].*
4. *Regla del Catálogo en Línea: Según esta regla, la descripción completa de la base de datos debe estar almacenada en el catálogo de la base de datos y ser accesible por los usuarios autorizados. El catálogo contiene información sobre las tablas, relaciones, restricciones y otros elementos de la base de datos, lo que facilita la administración y comprensión de la estructura de los datos [10].*
5. *Regla de la Sublenguaje de Datos Completo: Esta regla establece que la base de datos debe admitir un lenguaje completo para la definición de datos, manipulación de datos, restricciones de integridad y seguridad. Esto implica que el lenguaje utilizado para interactuar con la base de datos debe ser lo suficientemente expresivo y poderoso como para realizar todas las operaciones necesarias de manera eficiente [10].*
6. *Regla de la Vista Actualizada: Según esta regla, las vistas deben reflejar siempre los datos actualizados de la base de datos. Las vistas son representaciones personalizadas de los datos almacenados en la base de datos y deben mantenerse actualizadas para brindar información precisa y coherente a los usuarios [10].*
7. *Regla de Operaciones de Alta Cantidad: Esta regla establece que las operaciones de la base de datos deben ser capaces de manipular conjuntos completos de registros, además de registros individuales. Esto permite realizar operaciones eficientes y consistentes en grandes volúmenes de datos, mejorando el rendimiento y la productividad [10].*
8. *Regla de Independencia Física: Según esta regla, la base de datos debe ser independiente de la representación física de los datos. Esto significa que los cambios en la forma en que se almacenan los datos no deben afectar la forma en que se accede y manipula la información. La independencia física brinda flexibilidad y adaptabilidad a los sistemas de bases de datos [10].*
9. *Regla de Independencia Lógica: Esta regla establece que las aplicaciones deben estar separadas de los detalles de almacenamiento físico y la estructura lógica de la base de datos. Los cambios en la estructura interna de la base de datos no deben requerir modificaciones en las aplicaciones que la utilizan, lo que proporciona modularidad y facilita el mantenimiento [10].*
10. *Regla de Integridad Independiente: Según esta regla, todas las restricciones de integridad deben ser almacenadas en la base de datos y ser independientes de las aplicaciones que acceden a ella. Esto garantiza que las reglas de integridad se apliquen de manera consistente y no dependan de implementaciones externas [10].*
11. *Regla de la Distribución: Esta regla establece que la base de datos debe ser capaz de distribuirse en varios sistemas de procesamiento y mantener la integridad y seguridad de los datos. La distribución permite escalabilidad y rendimiento en entornos de bases de datos distribuidas [10].*
12. *Regla de la No-Subversión: Según esta regla, no debe ser posible modificar las reglas de integridad mediante operaciones normales de la base de datos. Esto evita que las reglas de integridad sean comprometidas o evitadas, garantizando la validez y consistencia de los datos almacenados [10].*

* **Restricciones**

*Las restricciones en bases de datos son reglas que aseguran la integridad y consistencia de los datos. Existen varios tipos de restricciones:*

* *Restricciones de integridad: Incluyen restricciones de clave primaria para garantizar la unicidad de los registros, restricciones de integridad referencial para mantener la coherencia entre las tablas y otras restricciones que garantizan la consistencia de los datos [11].*
* *Restricciones de dominio: Establecen los valores permitidos para un atributo. Por ejemplo, se puede definir que un atributo sólo acepte valores numéricos positivos [11].*
* *Restricciones de comprobación: Verifican condiciones específicas que deben cumplirse para que los datos sean válidos. Por ejemplo, se puede verificar que la fecha de nacimiento de una persona sea anterior a la fecha actual [11].*
* *Restricciones de exclusión: Protegen datos importantes evitando su eliminación o modificación accidental. Por ejemplo, se puede establecer una restricción que impida eliminar un registro de un empleado activo [11].*
* *Restricciones de reglas de negocio: Son reglas específicas establecidas por una organización para garantizar que los datos cumplan con políticas y requisitos particulares. Por ejemplo, una regla de negocio podría especificar que el descuento máximo aplicable a un producto no puede superar el 50% [11].*

*Las restricciones desempeñan un papel crucial en la garantía de la calidad de los datos y su conformidad con las reglas establecidas. Ayudan a prevenir inconsistencias y errores en la base de datos, asegurando que los datos sean confiables y coherentes [11].*

* **Índices**

*Los índices en bases de datos son estructuras que mejoran la eficiencia de las consultas y búsquedas. Funcionan como guías para acceder rápidamente a los datos en una tabla. Se basan en claves de índice y utilizan estructuras de árbol. Existen diferentes tipos de índices, como primarios, secundarios, únicos y de texto completo. Mantener los índices actualizados es importante para un rendimiento óptimo [12].*

* **Batch**

*Un batch es un conjunto de tareas o comandos que se ejecutan secuencialmente sin intervención manual. Se utilizan para automatizar tareas repetitivas o secuenciales, como procesar archivos o realizar operaciones en una base de datos. Los batches ahorran tiempo y recursos al ejecutar tareas en grupo, mejorando la eficiencia y la automatización en entornos informáticos [13].*

* **Trigger**

*Los triggers son objetos de base de datos que responden automáticamente a eventos específicos, como la inserción, actualización o eliminación de datos. Se utilizan para automatizar tareas y aplicar reglas de negocio en la base de datos. Los triggers constan de un evento, una condición y una acción, y se activan cuando se cumple la condición establecida. Pueden utilizarse para mantener la integridad de los datos, auditar cambios, notificar eventos, entre otros. Los triggers son una herramienta esencial para mejorar la automatización y la consistencia de los datos en una base de datos [14].*

* **Transacción**

*Las transacciones en bases de datos son unidades de trabajo que agrupan operaciones para asegurar la consistencia de los datos. Siguiendo el principio ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), las transacciones se realizan de manera completa y segura. En caso de fallos, las transacciones pueden ser deshechas, manteniendo así la integridad de los datos [15].*

* **Candados**

*Los candados en bases de datos son mecanismos de bloqueo que evitan el acceso simultáneo y conflictivo a los recursos compartidos. Los candados compartidos permiten el acceso para lectura mientras que los candados exclusivos bloquean el acceso tanto para lectura como para escritura. Además, los candados de actualización se utilizan para mantener la consistencia en operaciones de actualización [16].*

* **Cursores**

*Los cursores en bases de datos son punteros que permiten recorrer y manipular registros de forma secuencial. Se utilizan para procesar conjuntos de datos uno por uno, realizando operaciones personalizadas en cada fila. Los cursores proporcionan flexibilidad y control, pero pueden afectar el rendimiento en conjuntos de datos grandes. Es importante utilizarlos adecuadamente y considerar alternativas más eficientes, como consultas de conjunto o funciones de agregación, para maximizar el rendimiento de las operaciones en bases de datos [17].*

* **NoSQL**

*Las bases de datos NoSQL son sistemas de almacenamiento de datos diseñados para manejar grandes volúmenes de información no estructurada o semiestructurada. A diferencia de las bases de datos relacionales, no siguen un esquema fijo y no utilizan tablas con relaciones definidas. En cambio, utilizan modelos flexibles que se adaptan a los datos, lo que permite una mayor escalabilidad horizontal y un rendimiento mejorado en entornos distribuidos. Las bases de datos NoSQL son ampliamente utilizadas en aplicaciones web, móviles y en análisis de big data, ya que ofrecen alta disponibilidad, rendimiento escalable y facilidad para trabajar con datos no estructurados. Algunos ejemplos populares incluyen MongoDB, Cassandra, Redis y Neo4j [18].*