BASES DE DATOS AVANZADAS, TRABAJO PRÁCTICO

Bases de Datos Temporales

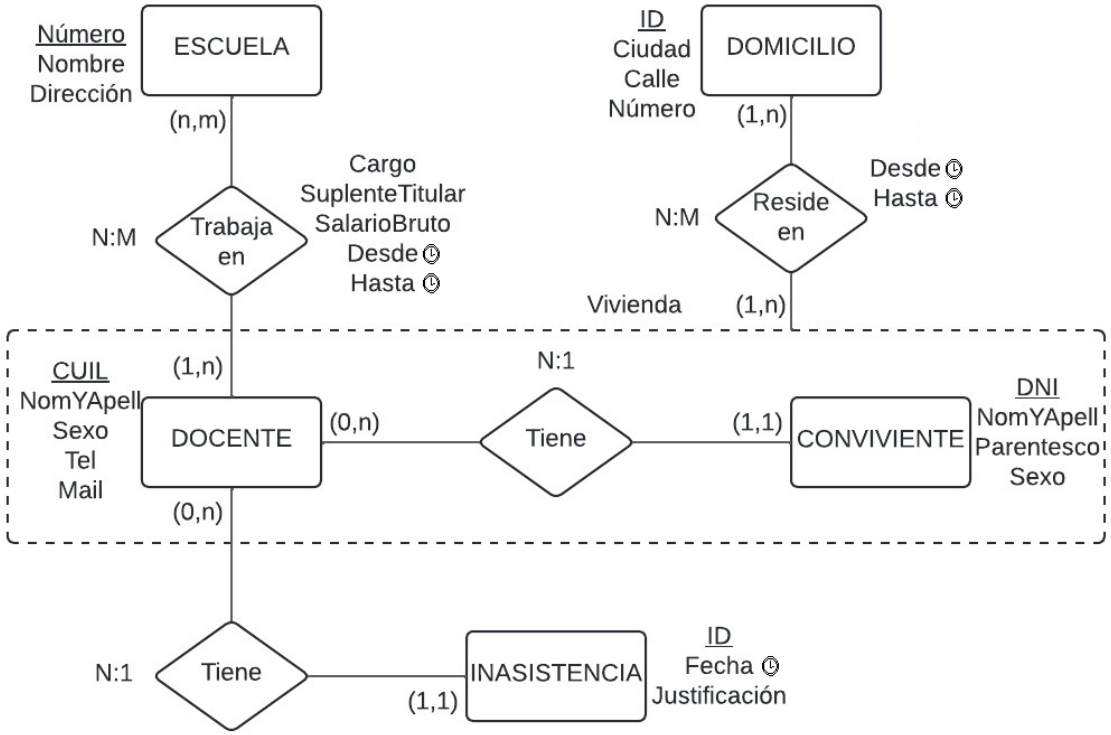
UADER FCyT
Profesores Schmukler Jorge, Trossero Sebastián

Pablo Santángelo, Alex Marioni

Contenido

1. Diagrama Entidad-Relación 2
2. Informe Tratamiento de Datos Temporales 2
3. Tablas Normalizadas 3
4. Scripts de Creación de Bases de Datos 3
5. Inserts de Datos Utilizados Durante el Trabajo 5
6. Scripts de Consultas (Querys) 6
   a) Información del salario de un docente entre dos fechas. 6
   b) Promedio anual del salario de un docente (sumatoria de sus salarios mensuales / 12) 6
   c) Porcentaje de aumento de salario entre dos fechas 7
   d) Información del domicilio de un empleado entre dos fechas. 7
   e) Información de los empleados que cambiaron su salario entre dos fechas 8
   f) Información de días de asistencia de un empleado entre dos fechas 8
   g) Información de costo asociado por inasistencias en general por año. 9
   h) Promedio anual de días de vacaciones tomados por cargo y por escuela. 10
   i) Promedio anual de inasistencias (sin considerar licencia por enfermedad, licencia por cuidado de conviviente). 10
   j) Porcentaje de distribución de asistencia, inasistencia por año y escuela. 11
   k) Porcentaje de distribución de inasistencias por año y titular / suplentes. 11
   l) Ranking de ausencias de docentes por año y escuela. 12
7. Demostración Requisito “El salario de un docente puede cambiar temporalmente” 12
8. Comparativa de características temporales entre SQL:2011 y SQL Server 2017 13

1. Diagrama Entidad-Relación



2. Informe Tratamiento de Datos Temporales

Los atributos marcados con este símbolo (reloj), son aquellos identificados como de tratamiento temporal.

La tabla intermedia entre Docente y Escuela se llamó Trayectoria y correspondería asignarle dos atributos de **tiempo de validez** que establezcan el **período** tiempo en que el docente trabajó en una escuela, con un mismo cargo, con la misma titularidad (titular o suplente), con un mismo sueldo. Cualquier cambio de estas características ameritaría un nuevo registro con distinto período de validez. De esta forma, Trayectoria guardaría no sólo dónde o cuándo trabajó un docente, sinó cada cambio de salario, titularidad, cargo o interrupción de período.

La tabla Inasistencia debe contener registros únicos para cada inasistencia y, entre sus atributos, la fecha a la cual corresponde la misma. Esta fecha sería de **tiempo de validez** de tipo **punto.**

Entre la tabla Domicilio y la agregación Vivienda debería existir una tabla intermedia con columnas de **período** en que esta última residió en un domicilio, siendo esto, también, **tiempo de validez**.

Debido a una cláusula de la consigna de este trabajo, que nos pedía implementar todo como tiempo de transacción, todas las tablas fueron adaptadas con este propósito, recurriendo a sintaxis propia de tiempo de transacción para reemplazar un tratamiento de tiempo de validez, cuando fue necesario.

3. Tablas Normalizadas

Docente

CUIL	DNI	NomyApell	Sexo	FechaNac	TelFijo	TelCelular	Mail	*idDomicilio*

Conviviente

DNI	Nombre	Parentesco	Sexo	FechaNac	*idDomicilio*

Trayectoria

CUIL	***EscuelaNro***	Cargo	Suplente_titular	SalarioBruto

Escuela

EscuelaNro	Nombre	Direccion

Domicilio

idDomicilio	Ciudad	Calle	Número

Inasistencia

idInasistencia	Fecha	*CUIL*	*idJustificacion*

Vivienda

idVivienda	*CUILDocente*	*DNIConviviente*	*idDomicilio*

Justificacion

idJustificacion	Descripcion

PrimaryKey *ForeignKey*

4. Scripts de Creación de Bases de Datos

```
-- Crear esquema para tablas temporales

CREATE SCHEMA TemporalHistory;

-- Tabla Docente
CREATE TABLE Docente (
    CUIL VARCHAR(11) PRIMARY KEY,
    DNI VARCHAR(8) NOT NULL,
    NomyApell VARCHAR(100) NOT NULL,
    Sexo CHAR(1),
    FechaNac DATE,
    TelFijo VARCHAR(15),
    TelCelular VARCHAR(15),
    Mail VARCHAR(100),
    idDomicilio INT,
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Docente_History));


-- Tabla Conviviente
CREATE TABLE Conviviente (
    DNI VARCHAR(8) PRIMARY KEY,
    Nombre VARCHAR(100),
    Parentesco VARCHAR(50),
    Sexo CHAR(1),
    FechaNac DATE,
    idDomicilio INT,
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Conviviente_History));
```

```
--Tabla Escuela
CREATE TABLE Escuela (
    EscuelaNro INT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Direccion VARCHAR(200)
);
--Tabla Domicilio
CREATE TABLE Domicilio (
    idDomicilio INT PRIMARY KEY IDENTITY(1,1),
    Ciudad VARCHAR(100) NOT NULL,
    Calle VARCHAR(100) NOT NULL,
    Número VARCHAR(10) NOT NULL
);
--Tabla Vivienda
CREATE TABLE Vivienda (
    idVivienda INT PRIMARY KEY IDENTITY(1,1),
    CUILDocente VARCHAR(11),
       DNIConviviente VARCHAR(8),
       idDomicilio INT,
       SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
       FOREIGN KEY (CUILDocente) REFERENCES Docente (CUIL),
       FOREIGN KEY (DNIConviviente) REFERENCES Conviviente (DNI),
       FOREIGN KEY (idDomicilio) REFERENCES Domicilio (idDomicilio)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Vivienda_History));
--Tabla Trayectoria
CREATE TABLE Trayectoria (
    CUIL VARCHAR(11) NOT NULL,
       EscuelaNro INT NOT NULL,
    Cargo VARCHAR(100) NOT NULL,
    Suplente_titular CHAR(1) NOT NULL, -- 'S' para suplente, 'T' para titular
    SalarioBruto DECIMAL(10,2) NOT NULL,
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
    CONSTRAINT PK_Trayectoria PRIMARY KEY (CUIL, EscuelaNro, SysStartTime),
    FOREIGN KEY (CUIL) REFERENCES Docente(CUIL),
    FOREIGN KEY (EscuelaNro) REFERENCES Escuela(EscuelaNro)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Trayectoria_History));
-- Tabla Justificación de inasistencias
CREATE TABLE Justificacion (
       idJustificacion INT PRIMARY KEY,
       Descripcion VARCHAR (100)
);
--Tabla Inasistencia
CREATE TABLE Inasistencia (
    idInasistencia INT PRIMARY KEY IDENTITY(1,1),
    Fecha DATE NOT NULL,
    CUIL VARCHAR(11) NOT NULL,
       idJustificacion INT,
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
    FOREIGN KEY (CUIL) REFERENCES Docente(CUIL),
       FOREIGN KEY (idJustificacion) REFERENCES Justificacion (idJustificacion)
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Inasistencia_History));
```

5. Inserts de Datos Utilizados Durante el Trabajo

```
INSERT INTO Docente (CUIL, DNI, NomyApell, Sexo, FechaNac, TelFijo, TelCelular,
Mail, idDomicilio)
VALUES ('20444444445', '12345678', 'Juan Pérez', 'M', '1980-04-15', '011-1234-
5678', '011-9876-5432', 'juan.perez@example.com', 1),
('27367654324',      '36765432',   'María García', 'F', '1975-08-20', '011-2233-
4455', '011-5566-7788', 'maria.garcia@example.com', 2),
('20333333333', '33333333', 'Carlos López', 'M', '1990-12-10', '011-3344-5566',
'011-6677-8899', 'carlos.lopez@example.com', 3);

INSERT INTO Escuela (EscuelaNro, Nombre, Direccion)
VALUES (101, 'Escuela Secundaria Nº 1', 'Av. Siempreviva 742, Ciudad A'),
(102, 'Escuela Técnica Nº 2', 'Calle Falsa 123, Ciudad B'),
(103, 'Colegio Nacional Nº 3', 'Boulevard Principal 456, Ciudad C'),
(104, 'Escuela de Artes Nº 4', 'Ruta 1 Km 23, Ciudad D');

INSERT INTO Trayectoria (CUIL, EscuelaNro, Cargo, Suplente_titular, SalarioBruto)
VALUES
('20444444445', 102, 'Profesora de Quimioterapia', 'T', 110000.00),
('27367654324', 101, 'Profesor de Historia del LoL', 'S', 4000.00),
('20333333333', 104, 'Profesora de Recreo', 'T', 999999.00);

INSERT INTO Domicilio (Ciudad, Calle, Número)
VALUES
('Buenos Aires', 'Avenida Siempre Viva', '742'),
('Córdoba', 'Calle Falsa', '123'),
('Rosario', 'Boulevard Oroño', '2156'),
('Mendoza', 'San Martín', '4321'),
('Salta', 'Mitre', '987'),
('Neuquén', 'Sarmiento', '654'),
('Tucumán', 'Rivadavia', '1050'),
('Mar del Plata', 'Colón', '789');

INSERT INTO Justificacion(idJustificacion, Descripcion) VALUES
(1, 'Enfermedad'),
(2, 'Licencia por cuidado de conviviente'),
(3, 'Licencia por enfermedad'),
(4, NULL),
(5, 'Reunión en la AFA para quitar descensos'),
(6, 'Trámite personal'),
(7, 'Tratamiento médico'),
(8, 'Vacaciones'),
(9, 'Me dormí');

INSERT INTO Inasistencia (Fecha, CUIL, idJustificacion) VALUES
('2022-01-16', '20333333333', 1),
('2022-01-17', '20333333333', 1),
('2022-04-07', '20444444445', 3),
('2022-04-08', '20444444445', 3),
('2022-04-09', '20444444445', 3),
('2022-04-10', '20444444445', 3),
('2022-04-11', '20444444445', 3),
('2022-04-12', '20444444445', 3),
('2023-09-03', '20444444445', 2),
('2023-09-04', '20444444445', 2),
('2023-09-15', '20333333333', 6),
('2023-11-20', '27367654324', 4),
('2023-12-17', '20333333333', 8),
('2023-12-18', '20333333333', 8),
('2023-12-19', '20333333333', 8),
('2023-12-20', '20333333333', 8),
… --Omitido por exceso de renglones, el INSERT completo está en el script
```

6. Scripts de Consultas (Querys)

a) Información del salario de un docente entre dos fechas.

```
DECLARE @fechaUno DATETIME2 = '2024-01-01';
DECLARE @fechaDos DATETIME2 = '2027-12-31';
DECLARE @cuilDado VARCHAR(11) = '20444444445';

SELECT CUIL, SalarioBruto
FROM Trayectoria
FOR SYSTEM_TIME BETWEEN @fechaUno AND @fechaDos
       WHERE CUIL = @cuilDado;
```

b) Promedio anual del salario de un docente (sumatoria de sus salarios mensuales / 12)

```
CREATE FUNCTION PromedioAnual(@CUIL VARCHAR(11), @AÑO INT)
RETURNS TABLE
AS
RETURN
(
    WITH Salarios AS (
        SELECT
            CUIL,
            SalarioBruto,
            CASE
                -- Si todo el registro existe dentro del año @AÑO
                WHEN YEAR(SysStartTime) = @AÑO AND YEAR(SysEndTime) = @AÑO THEN
DATEDIFF(MONTH, SysStartTime, SysEndTime)
                -- Si el registro empieza en el año @AÑO pero termina después
                WHEN YEAR(SysStartTime) = @AÑO AND YEAR(SysEndTime) > @AÑO THEN
DATEDIFF(MONTH, SysStartTime, CAST(@AÑO AS VARCHAR) + '-12-31') + 1
                -- Si el registro empieza antes del año @AÑO pero termina en @AÑO
                WHEN YEAR(SysStartTime) < @AÑO AND YEAR(SysEndTime) = @AÑO THEN
DATEDIFF(MONTH, CAST(@AÑO AS VARCHAR) + '-01-01', SysEndTime) + 1
                -- Si el registro no pertenece al año @AÑO
                WHEN YEAR(SysEndTime) < @AÑO OR YEAR(SysStartTime) > @AÑO THEN 0
                -- Si el registro cubre todo el año
                ELSE 12
            END AS MesesTrabajados
        FROM Trayectoria
        FOR SYSTEM_TIME ALL
        WHERE CUIL = @CUIL
    )
    SELECT
        CUIL,
        @AÑO AS Año,
             SUM(MesesTrabajados) AS MesesTrabajados,
        CAST (SUM(SalarioBruto * MesesTrabajados) / 12.0 AS DECIMAL(10,2)) AS
PromedioAnual
    FROM Salarios
    GROUP BY CUIL
);

SELECT *
FROM PromedioAnual('20444444445', 2024);
```

c) Porcentaje de aumento de salario entre dos fechas

```
--Declaraciones
DECLARE @cuilDado VARCHAR(11);
DECLARE @fechaInicialDada DATETIME2;
DECLARE @fechaFinalDada DATETIME2;

--Inicializaciones
SET @cuilDado = '20444444445';
SET @fechaInicialDada = '2024-06-02';
SET @fechaFinalDada = '2024-11-20';

--Obtener Salario total de la fecha Desde
DECLARE @salarioInicial DECIMAL(10,2) = (
    SELECT
        SUM(SalarioBruto)
    FROM
        Trayectoria
    FOR SYSTEM_TIME ALL
    WHERE
        CUIL = @cuilDado
        AND @fechaInicialDada BETWEEN SysStartTime AND SysEndTime
    GROUP BY CUIL
);
SELECT @salarioInicial as SalarioInicial;

-- Obtener Salario total de la fecha Hasta
DECLARE @salarioFinal DECIMAL(10,2) = (
    SELECT
        SUM(SalarioBruto)
    FROM
        Trayectoria
    FOR SYSTEM_TIME ALL
    WHERE
        CUIL = @cuilDado
        AND @fechaFinalDada BETWEEN SysStartTime AND SysEndTime
    GROUP BY CUIL
);
SELECT @salarioFinal as SalarioFinal;

--Calcular porcentaje de aumento
SELECT CAST(ROUND((((COALESCE(@salarioFinal, 0) - COALESCE(@salarioInicial, 0)) *
100) / @salarioInicial), 2, 1) AS DECIMAL(10, 2)) AS PorcentajeAumento;
```

d) Información del domicilio de un empleado entre dos fechas.

```
DECLARE @fechaDesdeDada DATETIME2 = '2024-09-05';
DECLARE @fechaHastaDada DATETIME2 = '2024-11-04';

SELECT
       Docente.idDomicilio,
       Docente.CUIL,
       Domicilio.ciudad,
       Domicilio.calle,
       Domicilio.Número
FROM Docente FOR SYSTEM_TIME BETWEEN @fechaDesdeDada AND @fechaHastaDada
INNER JOIN Domicilio
ON Domicilio.idDomicilio = Docente.idDomicilio;
```

e) Información de los empleados que cambiaron su salario entre dos fechas

```
DECLARE @fechaInicial DATETIME2 = '2024-10-16';
DECLARE @fechaFin DATETIME2 = '2025-06-11';

SELECT
       Docente.CUIL,
       Docente.NomyApell,
       Docente.FechaNac,
       Docente.TelCelular,
       Trayectoria.SysStartTime,
       Trayectoria.SysEndTime,
       SalarioBruto
FROM Trayectoria INNER JOIN Docente ON Docente.CUIL = Trayectoria.CUIL
WHERE
Trayectoria.CUIL = Docente.CUIL
AND (
    (Trayectoria.SysStartTime <= @fechaInicial AND (@fechaInicial <
Trayectoria.SysEndTime OR Trayectoria.SysEndTime IS NULL)
        AND NOT (Trayectoria.SysStartTime <= @fechaFin AND (@fechaFin <
Trayectoria.SysEndTime OR Trayectoria.SysEndTime IS NULL)))
    OR
    (Trayectoria.SysStartTime <= @fechaFin AND (@fechaFin <
Trayectoria.SysEndTime OR Trayectoria.SysEndTime IS NULL)
        AND NOT (Trayectoria.SysStartTime <= @fechaInicial AND (@fechaInicial <
Trayectoria.SysEndTime OR Trayectoria.SysEndTime IS NULL)))
       );
```

f) Información de días de asistencia de un empleado entre dos fechas

```
DECLARE @cuilDado VARCHAR(11);
DECLARE @fechaInicial DATE;
DECLARE @fechaFinal DATE;
DECLARE @diasTotal INT;
DECLARE @diasInasistencia INT;
DECLARE @diasAsistencia INT;

-- Inicializaciones
SET @cuilDado = '20444444445';
SET @fechaInicial = '2024-10-01';
SET @fechaFinal = '2024-11-30';

-- Inicializaciones de valores para obtener resultado
SET @diasTotal = DATEDIFF(DAY, @fechaInicial, @fechaFinal) + 1;

SET @diasInasistencia = (
       SELECT
         COUNT(*) AS TotalInasistencias
    FROM
        Inasistencia
    WHERE
        CUIL = @cuilDado
        AND Fecha BETWEEN @fechaInicial AND @fechaFinal
);

-- Resta para obtener resultado
SET @diasAsistencia = @diasTotal - @diasInasistencia;

SELECT @diasAsistencia;
```

g) Información de costo asociado por inasistencias en general por año.

```
WITH InasistenciasMensuales AS (
    SELECT
        YEAR(I.Fecha) AS Año,
        I.CUIL,
        COUNT(I.idInasistencia) AS InasistenciasEnMes
    FROM
        Inasistencia I
    WHERE
        I.Fecha BETWEEN '2022-01-01' AND '2024-12-31'
    GROUP BY
        YEAR(I.Fecha),
        I.CUIL
),
SalarioMensual AS (
    SELECT
        T.CUIL,
        SUM(T.SalarioBruto) AS SalarioBrutoTotal
    FROM
        Trayectoria T
    WHERE
        T.SysStartTime <= '2024-12-31' AND T.SysEndTime >= '2022-01-01'
    GROUP BY
        T.CUIL
)
SELECT
    IM.Año,
    IM.CUIL,
    SUM((SM.SalarioBrutoTotal / 30) * IM.InasistenciasEnMes) AS
CostoAnualAsociado
FROM
    InasistenciasMensuales IM
INNER JOIN
    SalarioMensual SM ON SM.CUIL = IM.CUIL
GROUP BY
    IM.Año,
    IM.CUIL
ORDER BY
    IM.Año,
    IM.CUIL;
```

h) Promedio anual de días de vacaciones tomados por cargo y por escuela.

```
SELECT
    T.EscuelaNro,
    T.Cargo,
    YEAR(I.Fecha) AS Año,
    COUNT(I.idInasistencia) AS TotalVacaciones,
    COUNT(DISTINCT I.CUIL) AS TotalDocentes,
    CAST((COUNT(I.idInasistencia) * 100.0) / (COUNT(DISTINCT I.CUIL) * 160.0) AS
DECIMAL(10,2)) AS PromedioVacaciones
FROM
    Inasistencia I
INNER JOIN
    Justificacion J ON I.idJustificacion = J.idJustificacion
INNER JOIN
    Trayectoria T ON I.CUIL = T.CUIL
WHERE
    J.Descripcion = 'Vacaciones'
GROUP BY
    T.EscuelaNro,
    T.Cargo,
    YEAR(I.Fecha)
ORDER BY
    Año,
    EscuelaNro,
    Cargo;
```

i) Promedio anual de inasistencias (sin considerar licencia por enfermedad, licencia por cuidado de conviviente).

```
SELECT * FROM Docente;
SELECT * FROM Inasistencia;
SELECT * FROM Justificacion;
--
SELECT
    CUIL,
    YEAR(Fecha) AS Año,
    COUNT(*) AS Total_Inasistencias,
    (COUNT(*) / 160.0) AS Promedio_Inasistencias
FROM
    Inasistencia I
INNER JOIN
       Justificacion J ON I.idJustificacion = J.idJustificacion
WHERE
    J.Descripcion IS NULL OR J.Descripcion NOT IN ('Licencia por enfermedad',
'Licencia por cuidado de conviviente')
GROUP BY
    CUIL,
    YEAR(Fecha);
```

j) Porcentaje de distribución de asistencia, inasistencia por año y escuela.

```
SELECT
	YEAR(I.Fecha) AS Año,
	T.EscuelaNro,
	COUNT(I.idInasistencia) AS TotalInasistencia,
	CAST((COUNT(I.idInasistencia) * 100.0 / (COUNT(DISTINCT I.Cuil) * 160.0))
AS DECIMAL(10,2)) AS porcentajeInasistencias,
	(100.0 - (CAST((COUNT(I.idInasistencia) * 100.0 / (COUNT(DISTINCT I.Cuil)
* 160.0)) AS DECIMAL(10,2)))) AS porcentajeAsistencias
FROM Trayectoria T INNER JOIN Inasistencia I
ON T.CUIL = I.CUIL
GROUP BY
	YEAR(I.Fecha),
	T.EscuelaNro
ORDER BY
	YEAR(I.Fecha),
	T.EscuelaNro;
```

k) Porcentaje de distribución de inasistencias por año y titular / suplentes.

```
SELECT
    YEAR(I.Fecha) AS Año,
    T.Suplente_titular,
    COUNT(I.idInasistencia) AS TotalInasistencias,
       (
       (COUNT(I.idInasistencia) * 100.0) /
       (
              SELECT COUNT(I2.idInasistencia)
              FROM Inasistencia I2 INNER JOIN Trayectoria T2 ON I2.CUIL = T2.CUIL
              WHERE YEAR(I2.Fecha) = YEAR(I.Fecha)
       ))AS PorcentajeInasistencias
FROM
    Inasistencia I
INNER JOIN
    Trayectoria T ON I.CUIL = T.CUIL
GROUP BY
    YEAR(I.Fecha),
    T.Suplente_titular
ORDER BY
    Año,
    T.Suplente_titular;
```

I) Ranking de ausencias de docentes por año y escuela.

```
SELECT
    YEAR(I.Fecha) AS Año,
    T.EscuelaNro,
    D.CUIL,
    COUNT(I.idInasistencia) AS TotalInasistencias,
    RANK() OVER (PARTITION BY YEAR(I.Fecha), T.EscuelaNro ORDER BY
COUNT(I.idInasistencia) DESC) AS RankingInasistencias
FROM
    Inasistencia I
INNER JOIN
    Docente D ON I.CUIL = D.CUIL
INNER JOIN
    Trayectoria T ON D.CUIL = T.CUIL
GROUP BY
    YEAR(I.Fecha),
    T.EscuelaNro,
    D.CUIL
ORDER BY
    Año,
    EscuelaNro,
    RankingInasistencias;
```

7. Demostración Requisito “El salario de un docente puede cambiar temporalmente”

Considerando los siguientes datos inicialmente insertados:

```
INSERT INTO Trayectoria (CUIL, EscuelaNro, Cargo, Suplente_titular, SalarioBruto)
VALUES
('20444444445', 101, 'Profesor de Matemáticas', 'T', 50000.00), -- Juan Pérez
('27367654324', 102, 'Profesora de Historia', 'S', 45000.00), -- María García
('20333333333', 103, 'Profesor de Física', 'S', 48000.00); -- Carlos López
```

Supongamos que pasó un tiempo y ahora Juan Pérez recibió un aumento de sueldo, María García adquirió la titularidad y Carlos López ahora trabaja en la misma escuela que Juan.

Vamos a actualizar los datos, lo que pasará es que se crearán nuevos registros, que pasarán a la tabla actual y los registros anteriores pasarán a la tabla histórica.

```
UPDATE Trayectoria
SET SalarioBruto = 85000.00
WHERE CUIL = '20444444445'
  AND EscuelaNro = 101
  AND Suplente_titular = 'T';

UPDATE Trayectoria
SET Suplente_titular = 'T',
    SalarioBruto = 55000.00
WHERE CUIL = '27367654324'
  AND EscuelaNro = 102
  AND Suplente_titular = 'S';
```

```
UPDATE Trayectoria
SET EscuelaNro = 101
WHERE CUIL = '20333333333'
        AND EscuelaNro = 103;
```

Tabla actual:

```
select CUIL, EscuelaNro, Cargo, Suplente_titular, SalarioBruto, SysStartTime,
SysEndTime from Trayectoria;
```

	CUIL	EscuelaNro	Cargo	Suplente_titular	SalarioBruto	SysStartTime	SysEndTime
1	20333333333	101	Profesor de Física	S	48000.00	2024-10-20 22:21:30.6545837	9999-12-31 23:59:59.9999999
2	20444444445	101	Profesor de Matemáticas	T	85000.00	2024-10-20 22:21:24.7905064	9999-12-31 23:59:59.9999999
3	27367654324	102	Profesora de Historia	T	55000.00	2024-10-20 22:21:28.7341089	9999-12-31 23:59:59.9999999

Tabla histórica:

```
SELECT * FROM TemporalHistory.Trayectoria_History;
```

	CUIL	EscuelaNro	Cargo	Suplente_titular	SalarioBruto	SysStartTime	SysEndTime
1	20444444445	101	Profesor de Matemáticas	T	50000.00	2024-10-20 22:11:03.1810588	2024-10-20 22:21:24.7905064
2	27367654324	102	Profesora de Historia	S	45000.00	2024-10-20 22:11:03.1810588	2024-10-20 22:21:28.7341089
3	20333333333	103	Profesor de Física	S	48000.00	2024-10-20 22:11:03.1810588	2024-10-20 22:21:30.6545837

8. Comparativa de características temporales entre SQL:2011 y SQL Server 2017

1. Soporte de tiempos: bitemporalidad vs. tiempo de transacción

- **SQL:2011**:
    - Dos tipos de periodos temporales:
        1. Tiempo de validez (Valid-time period): Representa el tiempo en el que los datos son válidos en el mundo real.
        2. Tiempo de transacción (Transaction-time period): Representa el tiempo en que las transacciones en la base de datos fueron realizadas.
    - Permite que los datos tengan tanto un periodo de validez como un periodo de transacción de forma simultánea, lo que se conoce como bitemporalidad.
- **SQL Server 2017**:
    - No soporta nativamente el tiempo de validez ni la bitemporalidad de forma directa.
    - Enfocado principalmente en el tiempo de transacción a través de las System-Versioned Tables (tablas con versionado del sistema), que guardan automáticamente el historial de los datos sin intervención manual.

2. Sintaxis para definir tablas temporales

- **SQL:2011**:
    - Soporta la definición explícita de periodos de tiempo mediante la cláusula PERIOD FOR, que permite definir tanto el tiempo de validez como el tiempo de transacción en una tabla.

Ejemplo:

```
CREATE TABLE Trayectoria (
    …
    PeriodoValidezInicio DATE,
    PeriodoValidezFin DATE,
    PeriodoTransaccionInicio TIMESTAMP,
    PeriodoTransaccionFin TIMESTAMP,
    PERIOD FOR periodo_de_validez (PeriodoValidezInicio,
PeriodoValidezFin),
    PERIOD FOR periodo_de_transaccion (PeriodoTransaccionInicio,
PeriodoTransaccionFin)
);
```

SQL Server 2017:

- Se enfoca en System-Versioned Tables. Se utiliza la cláusula PERIOD FOR SYSTEM_TIME para definir el tiempo de transacción. SQL Server 2017 administra automáticamente las columnas SysStartTime y SysEndTime.

Ejemplo:

-- Definición de una tabla con versionado del sistema en SQL Server 2017

```
CREATE TABLE Trayectoria (
    CUIL VARCHAR(11) PRIMARY KEY,
    …
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime),
) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
TemporalHistory.Trayectoria_History));
```

3. Tipos de datos temporales

- **SQL:2011**:
    - No define tipos de datos temporales específicos, sino que permite trabajar con periodos definidos por el usuario, los cuales se asocian a columnas que contienen tipos de datos como DATE o TIMESTAMP.
    - Los periodos están vinculados a intervalos de tiempo basados en las columnas de tipo fecha o timestamp.

- **SQL Server 2017**:
    - Introduce el uso de las columnas SysStartTime y SysEndTime, que son de tipo DATETIME2.
    - Estos tipos de datos son generados automáticamente cuando se utiliza el versionado del sistema, y no requieren ser gestionados manualmente.

4. Consultas temporales

- **SQL:2011**:
    - Permite realizar consultas en puntos específicos de tiempo o rangos, tanto para el tiempo de validez como para el tiempo de transacción.
    - Utiliza las cláusulas FOR PORTION OF para definir periodos de consulta para las columnas de validez o transacción.

Ejemplos:

```
SELECT * FROM Trayectoria FOR VALID TIME FOR PORTION OF '2024-01-01' TO '2024-12-31' FROM '2024-03-01' TO '2024-03-31';
```

```
SELECT * FROM Trayectoria FOR VALID_TIME FROM '2024-01-01' TO '2024-12-31' FOR TRANSACTION TIME AS OF '2024-06-30';
```

- **SQL Server 2017**:
    - Las consultas temporales se realizan usando la cláusula FOR SYSTEM_TIME, que permite consultar las versiones históricas de los datos en un punto de tiempo específico o en un rango de tiempo.
    - Soporta las opciones:
        - FOR SYSTEM_TIME AS OF: Recupera la versión de los datos en un momento específico.
        - FOR SYSTEM_TIME BETWEEN: Recupera datos dentro de un rango de tiempo.
        - FOR SYSTEM_TIME CONTAINED IN: Recupera datos cuya vigencia cae dentro del rango especificado.

Ejemplo:

```
SELECT CUIL, SalarioBruto
FROM Trayectoria
FOR SYSTEM_TIME BETWEEN @fechaUno AND @fechaDos
WHERE CUIL = @cuilDado;
```

5. Manejo de historial y versiones

- **SQL:2011**:

El manejo del historial y las versiones debe ser implementado manualmente o depender de la implementación del sistema de bases de datos. SQL:2011 no ofrece una solución automatizada para almacenar versiones anteriores de los datos. Cuando un registro deja de ser válido o cambia, tú mismo debes actualizar las fechas o insertar una nueva versión del registro.

- **SQL Server 2017**:

Las System-Versioned Tables gestionan el historial de los datos automáticamente. Cada vez que se actualiza o elimina un registro, SQL Server mueve la versión anterior a una tabla de historial. Esto permite consultar las versiones anteriores sin necesidad de implementar la lógica manualmente.