Métodos de Junta en SQL Server

SQL Server utiliza 3 tipos de métodos de junta
- 1) Nested loops join
- 2) Merge join
- 3) Hash join

Para poder probar los distintos ejemplos que se muestran a lo largo de este documento, ejecutar el Script que se incluye en el Anexo (ubicado al final de este documento).

1) Nested Loops

Es el método de junta más simple de los tres.

Su nombre es Bucles Anidados y funciona recorriendo primero una de las dos tablas (las más pequeña o la que no tenga índice) y para cada fila de esa tabla, busca en la otra tabla (mediante el índice, Index Seek) las filas que tienen igual valor en el atributo de junta, para combinarlas.

Este método se utiliza generalmente cuando una tabla es muy pequeña y la otra es muy grande y tiene índice en el atributo de junta.

Aplicable a joins por igualdad, comparación de mayor, menor y diferencia.

Veamos algunos ejemplos:

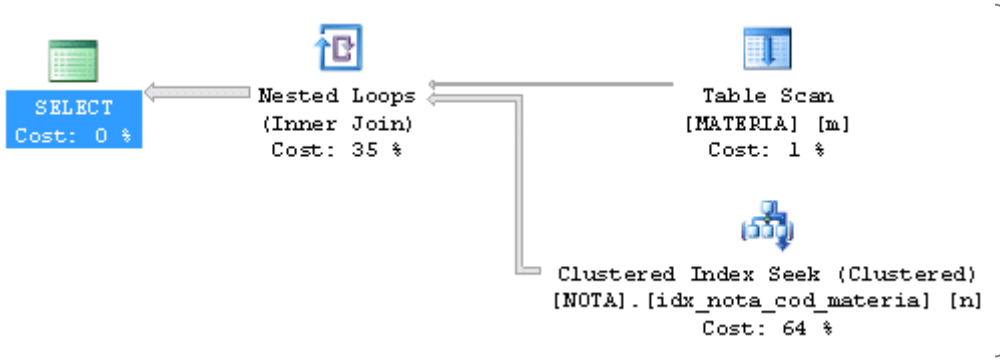
Caso 1.1 – Con un índice

Este es el caso más frecuente de Nested Loops.

```
CREATE CLUSTERED INDEX idx_nota_cod_materia ON NOTA (cod_materia)
```

Agregamos una condición sobre la tabla Materia para que el motor de base de datos prefiera resolver la junta con Nested Loops. Otra forma, seria forzar el uso de Nested Loop con el Hint “INNER LOOP JOIN”.

```
SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
AND    m.carrera='7 - Comunicación Social'
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```



El orden en el que se muestran las tablas en el árbol es importante.

La tabla que se muestra arriba es siempre la que se lee primero y la que se muestra debajo es la que se lee en segundo lugar.

```
drop index nota.idx_nota_cod_materia
```

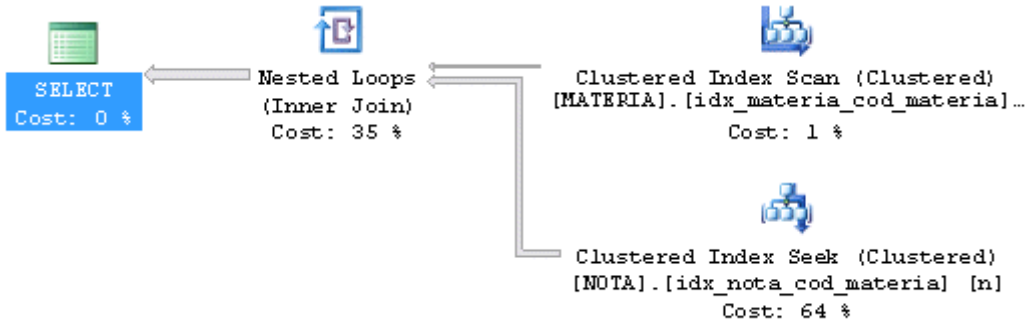
La tabla MATERIA es la más pequeña y se lee primero y para cada fila de ella se buscan (mediante el Índice Cluster) en la tabla NOTA las filas para combinar.

Otros casos menos frecuentes:

Caso 1.2 – Con 2 índices.

```
CREATE CLUSTERED INDEX idx_nota_cod_materia ON NOTA (cod_materia)
CREATE CLUSTERED INDEX idx_materia_cod_materia ON MATERIA (cod_materia)

SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
AND    m.carrera='7 - Comunicación Social'
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```

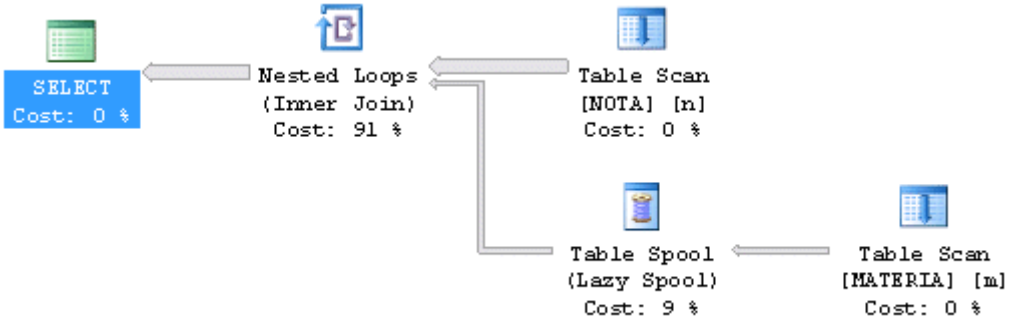



```
drop index nota.idx_nota_cod_materia
drop index materia.idx_materia_cod_materia
```

La tabla que figura como **Clustered Index Scan** es la tabla que se lee primero (mediante su índice) y la que figura debajo como **Clustered Index Seek** es la que lee en segundo lugar, y es donde se hace una búsqueda con índice por cada fila de la primer tabla.

Caso 1.3 – Sin índices

```
SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n INNER LOOP JOIN MATERIA m
    ON     (n.cod_materia=m.cod_materia)
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```

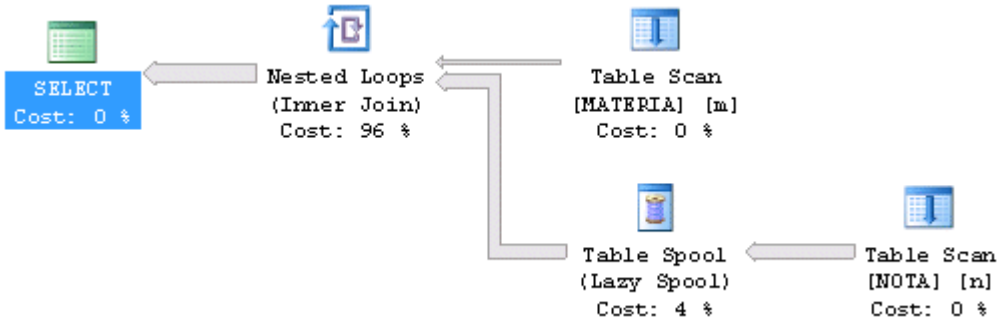


Normalmente, cuando las tablas no tienen índices el Optimizador decide usar otro método, pero en este caso lo forzamos con el Hint “INNER LOOP JOIN” solo para ver cómo sería el Árbol del Plan de Ejecución.

La operación que figura como “Table Spool” es un almacenamiento temporal de datos que van a ser leídos más de una vez. Es como una Tabla Temporal y sirve para mejorar la performance.

Caso 1.4 – Con una condición de desigualdad

```
SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia<>m.cod_materia
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```



El único método de junta que permite condiciones de desigualdad (mayor, menor o distinto) es el Nested Loops. Los otros métodos necesitan al menos una condición de igualdad (equijoin).

2) Merge Join 

También conocido como Sort Join o Sort-Merge Join.

Este método se utiliza cuando ambas tablas se encuentran ordenadas físicamente por el atributo de junta (con índice cluster) o bien cuando solo una de ellas esta ordenada y la otra es relativamente pequeña y se puede ordenar rápidamente antes de hacer la junta (también se pueden ordenar ambas tablas antes de hacer la junta, pero en ese caso este método suele no ser tan conveniente).
La gran ventaja de este método es que es muy eficiente y solo necesita leer una vez cada tabla y por lo tanto su costo es muy bajo.

El Merge Join lee simultáneamente una fila de cada tabla y las compara por el atributo de junta. Si el valor coincide, esas filas son incluidas en el resultado. Si no coinciden, la fila con el menor valor es descartada, ya que como ambas tablas están ordenadas, esa fila no podrá combinarse con ninguna otra fila de las restantes.
De esta forma se va avanzando y recorriendo ambas tablas desde la primer fila hasta la última, sin necesidad de retroceder.

Como resultado, el costo total de aplicar este método es el correspondiente a leer una vez cada tabla.

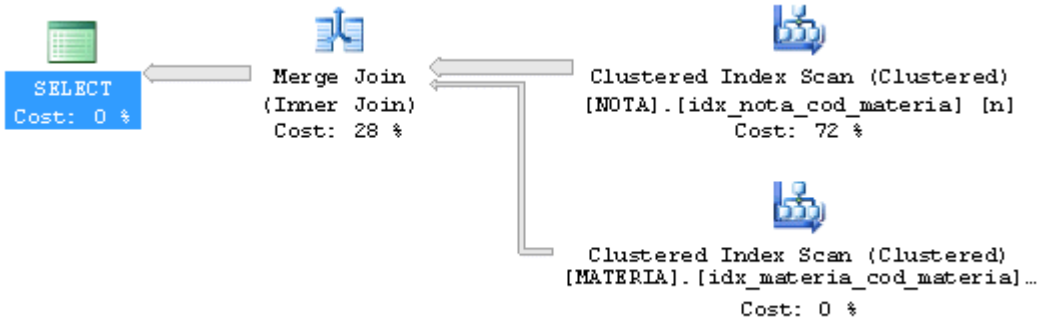
A diferencia del Nested Loops que soporta cualquier condición de junta (igualdad, mayor, menor, distinto, etc.), este método requiere al menos una condición de igualdad en las condiciones de junta (equijoin).
Si tiene más de una condición de junta, resuelve la junta con la condición de igualdad y las otras condiciones las resuelve luego en memoria, por ejemplo: “T1.a = T2.a and T1.b > T2.b.”

Veamos algunos ejemplos:

Caso 2.1
Veamos un ejemplo con ambas tablas ordenadas (con índice cluster en el atributo de junta).

```
CREATE CLUSTERED INDEX idx_nota_cod_materia ON NOTA (cod_materia)
CREATE CLUSTERED INDEX idx_materia_cod_materia ON MATERIA (cod_materia)

SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```



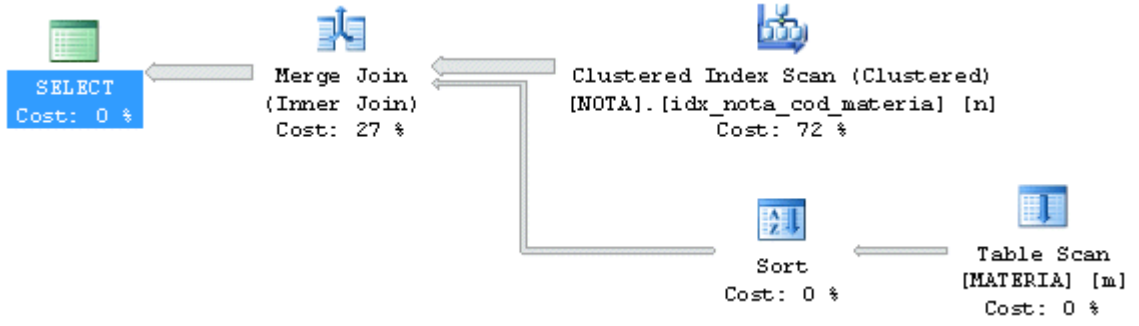

```
drop index nota.idx_nota_cod_materia
drop index materia.idx_materia_cod_materia
```

Ambas tablas se encuentran ordenadas físicamente por el atributo de junta (tienen índice cluster) así que el Optimizador de Consultas decide aprovechar esto y aplicar directamente el método de junta MERGE JOIN.

Caso 2.2
Veamos un ejemplo con una Tabla ordenada (con índice cluster) y la otra no ordenada (sin índices).

```
CREATE CLUSTERED INDEX idx_nota_cod_materia ON NOTA (cod_materia)

SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```




```
drop index nota.idx_nota_cod_materia
```

La tabla más grande es NOTA y ya está ordenada físicamente por el atributo de junta (tiene índice Cluster), luego la tabla MATERIA es muy pequeña y no esta ordenada, así que el Optimizador de Consultas decide que lo mejor es ordenarla rápidamente y luego que ambas tablas quedan ordenadas aplicar el método de junta MERGE JOIN.

3) Hash Match

Este método se usa generalmente cuando las tablas no están ordenadas ni tienen índices. Es decir, que este método se suele aplicar cuando no es posible utilizar ninguno de los dos métodos anteriores.
Cuando SQL Server elije el método Hash join pueden ser una mala señal porque indica que probablemente algo se podría mejorar (por ejemplo, crear un índice).

En este método consiste en:
1. Construir una tabla hash en memoria con el contenido de la tabla menor
2. Recorrer la segunda tabla, comparando con el hash de memoria.

Para que este método sea útil debe haber memoria suficiente para construir el hash.

El costo es el de leer una vez cada tabla (una para construir el hash y la otra para buscar coincidencias).

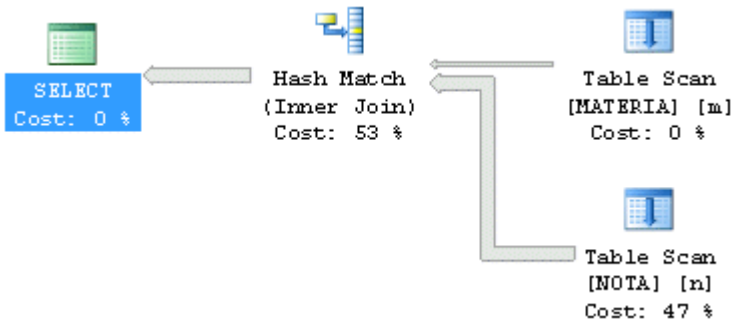
Al igual que el Merge, este método requiere al menos una condición de igualdad en las condiciones de junta (equijoin).
Si tiene más de una condición de junta, resuelve la junta con la condición de igualdad y las otras condiciones las resuelve luego en memoria, por ejemplo: “T1.a = T2.a and T1.b > T2.b.”

Veamos algunos ejemplos:

Caso 3.1

Las tablas no tienen índices.

```
SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```



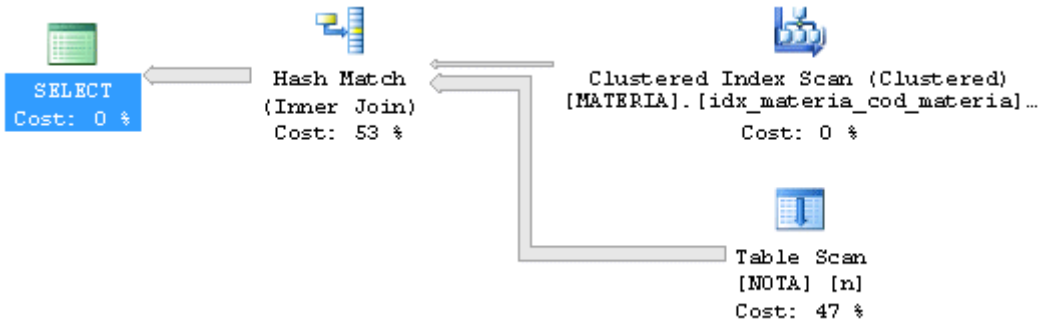
Como las tablas no tienen índices, el Optimizador de Consultas decide utilizar la junta Hash.

Caso 3.2

La tabla más chica (Materia) tiene índice Cluster.

```
CREATE CLUSTERED INDEX idx_materia_cod_materia ON MATERIA (cod_materia)

SELECT n.legajo, m.descripcion, n.nota
FROM   NOTA n, MATERIA m
WHERE  n.cod_materia=m.cod_materia
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```




```
drop index materia.idx_materia_cod_materia
```

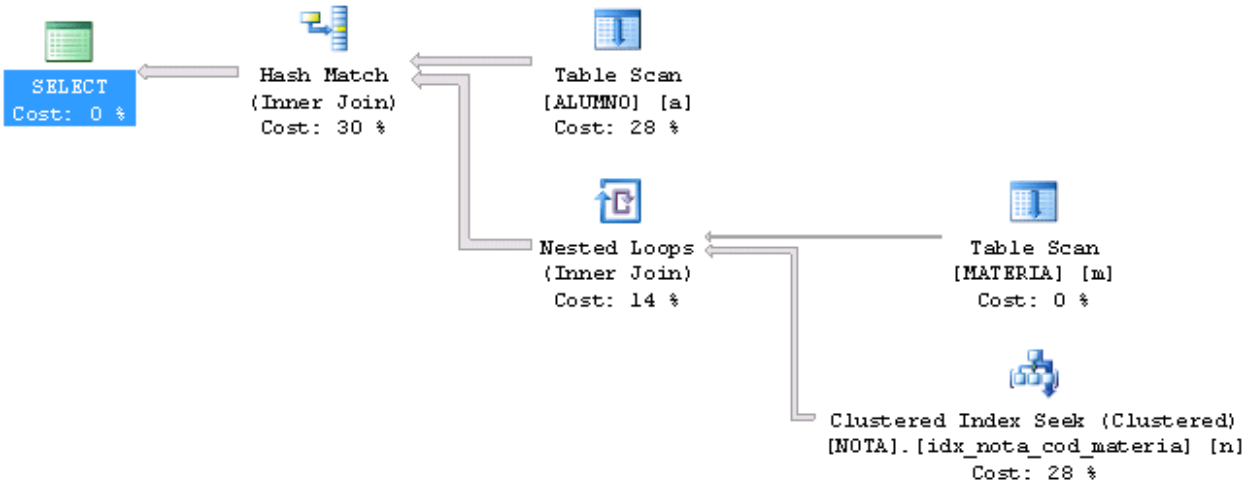
Si el índice Cluster hubiese estado en la tabla más grande (Nota) seguramente el Optimizador hubiese escogido el método de junta Nested Loops, tal como vimos en el Caso 1.1

Combinaciones con múltiples juntas

Por último y solo a modo ilustrativo, vemos un ejemplo de una consulta más compleja con múltiples juntas en la que se combinan dos métodos distintos.

```
CREATE CLUSTERED INDEX idx_nota_cod_materia ON NOTA (cod_materia)

SELECT a.legajo, a.apellido
             ,n.nota
             ,m.descripcion
FROM   ALUMNO a, NOTA n, MATERIA m
WHERE  a.legajo=n.legajo
AND    n.cod_materia=m.cod_materia
AND    n.nota>=4
AND    a.sexo='M'
AND    a.ciudad='Haedo'
AND    m.carrera='7 - Comunicación Social'
OPTION (MAXDOP 1) --Para evitar el Paralelismo y simplificar el árbol
```



Primero realiza la junta con la tabla más chica (Materia). En esa primer junta entre Materia y Nota, la tabla más grande tiene un índice y entonces el Optimizador decide usar el método Nested Loops. Luego, hace la junta entre el resultado de la primera y la tabla Alumno, en este caso decide usar el método Hash porque no tiene índices para usar.

Notas:

1) Para armar el siguiente documento se utilizó SQL Server 2012.

2) Se pueden consultar los indices que tiene cada tabla, utilizando el siguiente Stored Procedure:

```
exec sp_helpindex NOTA
exec sp_helpindex MATERIA
exec sp_helpindex ALUMNO
```

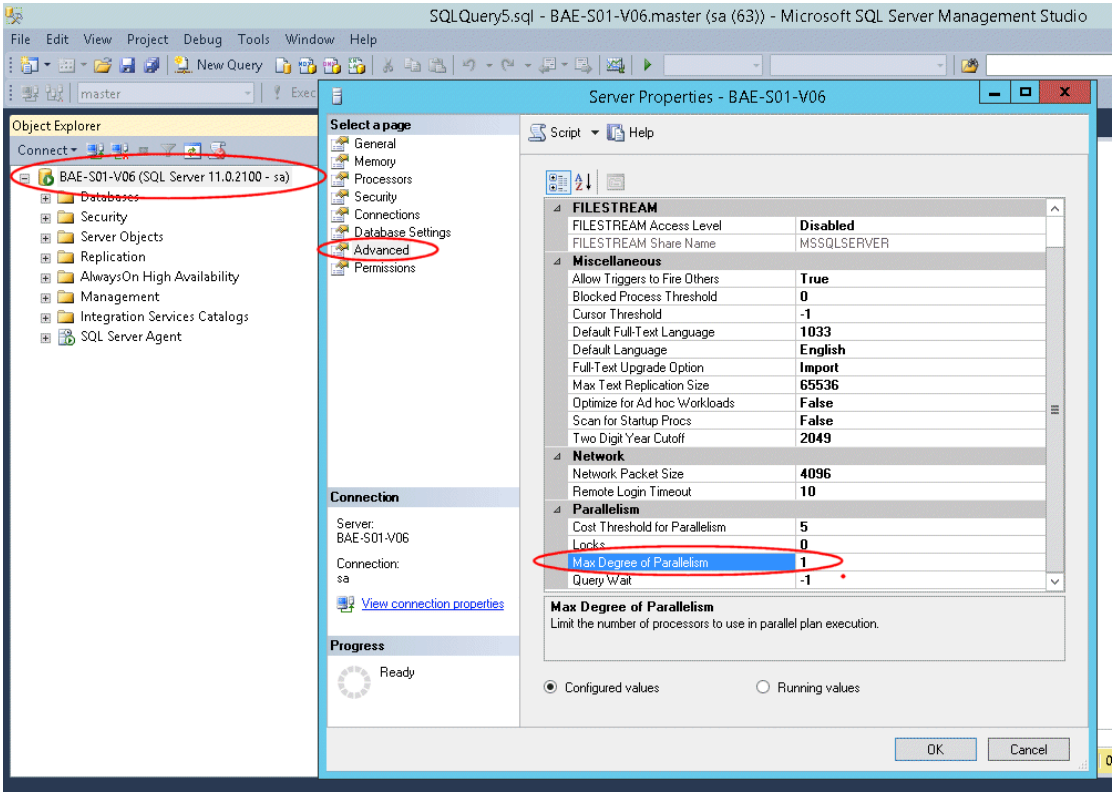
3) Para forzar un determinado tipo de junta, se pueden usar los JOIN HINTS.
Por ejemplo: INNER LOOP/HASH/MERGE JOIN

4) A fin de simplificar el Árbol del Plan de ejecución, es conveniente desactivar el Paralelismo. Esto se puede hacer de 2 formas:

a) Usando el Hint "`OPTION (MAXDOP 1)`" al final de la consulta SQL.
Esto solo afecta a la consulta SQL que estamos ejecutando y no modifica la configuración del servidor.

b) Modificando un parámetro a nivel Servidor (click sobre el Servidor -> Properties -> Advanced -> Buscar la variable "Max Degree of Parallelism" y cambiar el valor de 0 a 1 (esto indica que se usará un solo procesador)

IMPORTANTE: Este cambio puede reducir la performance del servidor.



Anexo

Script de creación de tablas (crea 4 tablas):

```
--=========================================
--Tabla TEMPORAL
--=========================================

--Primero se genera una tabla con nros Secuenciales,
--esta tabla nos ayudará a llenar con datos las otras Tablas.
CREATE TABLE TEMPORAL
(
        SECUENCIAL [int] IDENTITY
        ,AUX [varchar] (40)
);

--Insertamos 3 filas manualmente
INSERT INTO TEMPORAL VALUES ('A');

INSERT INTO TEMPORAL VALUES ('B');

INSERT INTO TEMPORAL VALUES ('C');

--Ahora se generan las filas artificialmente haciendo productos cartesianos
INSERT INTO TEMPORAL
SELECT  t1.aux
FROM    TEMPORAL t1, TEMPORAL t2;
--9 filas creadas

INSERT INTO TEMPORAL
SELECT  t1.aux
FROM    TEMPORAL t1, TEMPORAL t2;
--144 filas creadas

INSERT INTO TEMPORAL
SELECT  t1.aux
FROM    TEMPORAL t1, TEMPORAL t2;
--24336 filas creadas

--SELECT count(*)
--FROM   TEMPORAL
--24492

--Se crea un valor auxiliar para cada Nro Secuencial
--el cual servirá luego como número Aleatorio
UPDATE  TEMPORAL
SET     aux=NEWID();

--=========================================
--Tabla ALUMNO
--=========================================

--Ahora se crea la tabla de ALUMNOS.
CREATE TABLE ALUMNO
(
        LEGAJO          [numeric](12, 0),
        APELLIDO [char] (128),
        NOMBRE          [char] (128),
        FECHA_NACIMIENTO [datetime],
        SEXO            [char] (1) CHECK (SEXO IN ('F','M')),
        CIUDAD          [char] (128)
);
```

```
INSERT INTO ALUMNO
SELECT  SECUENCIAL
        --10000 Apellidos distintos, luego se repiten.
        ,'Apellido'+replace(right(str(SECUENCIAL),4),' ','0') APELLIDO
        --100 Nombres distintos, luego se repiten.
        ,'Nombre'+replace(right(str(SECUENCIAL),2),' ','0') NOMBRE
        --Fecha de Nacimiento aleatoria
        ,Convert(datetime
,'199'+RIGHT(ASCII(SUBSTRING(aux,1,1)),1)+replace('0'+RIGHT(ASCII(SUBSTRING(aux,2,1)),1),'00','10')+replac
e('0'+RIGHT(ASCII(SUBSTRING(aux,3,1)),1),'00','10'),112) FECHA_NACIMIENTO
        --Sexo basado en un nro aleatorio, si es impar ponemos 'F', sino 'M'.
        ,(CASE ASCII(SUBSTRING(aux,4,1))%2 WHEN 1 THEN 'F' ELSE 'M' END) SEXO
        --Ciudad basada en un nro aleatorio (del 0 al 9)
        ,(CASE right(ASCII(SUBSTRING(aux,5,1)),1)
                WHEN '0' THEN 'Haedo'
                WHEN '1' THEN 'Morón'
                WHEN '2' THEN 'San Justo'
                WHEN '3' THEN 'Ramos Mejía'
                WHEN '4' THEN 'Castelar'
                WHEN '5' THEN 'Palomar'
                WHEN '6' THEN 'Merlo'
                WHEN '7' THEN 'Flores'
                WHEN '8' THEN 'Haedo'
                WHEN '9' THEN 'Castelar'
                        END) CIUDAD
FROM    TEMPORAL;

--=========================================
--Tabla MATERIA
--=========================================

CREATE TABLE MATERIA
(
        COD_MATERIA     [numeric](12, 0),
        DESCRIPCION     [char] (128),
        CARRERA         [char] (128)
);

INSERT INTO MATERIA
SELECT  SECUENCIAL
        ,'Materia'+replace(right(str(SECUENCIAL),3),' ','0') DESCRIPCION
        --Carrera basada en un nro aleatorio (del 0 al 9)
        ,(CASE right(ASCII(SUBSTRING(aux,6,1)),1)
                WHEN '0' THEN '0 - Ingeniería en Informática'
                WHEN '1' THEN '1 - Ingeniería Electrónica'
                WHEN '2' THEN '2 - Ingeniería Industrial'
                WHEN '3' THEN '3 - Contador Público'
                WHEN '4' THEN '4 - Licenciatura en Administración de Empresas'
                WHEN '5' THEN '5 - Licenciatura en Comercio Exterior'
                WHEN '6' THEN '6 - Abogacía'
                WHEN '7' THEN '7 - Comunicación Social'
                WHEN '8' THEN '8 - Trabajo Social'
                WHEN '9' THEN '9 - Profesorado en Educación Física'
                        END) CARRERA
FROM    TEMPORAL
--Creamos solo 999 Materias
WHERE   secuencial<=999
--40 Materias por Carrera
AND     right(secuencial,2)<40;


--=========================================
--Tabla NOTA
--=========================================

CREATE TABLE NOTA
(
        LEGAJO              [numeric](12, 0),
        COD_MATERIA         [numeric](12, 0),
        NOTA                [int],
        FECHA               [datetime],
        FINAL               [int] CHECK (FINAL IN (0,1)),
        NRO_DE_PARCIAL      [int] CHECK (NRO_DE_PARCIAL IN (0,1,2)),
        NRO_DE_RECUPERATORIO [int] CHECK (NRO_DE_RECUPERATORIO IN (0,1,2))
);
```

```
INSERT INTO NOTA
SELECT  a.legajo
        ,m.cod_materia
        --La Nota es una cuenta cualquiera
        ,replace(replace(right(day(fecha_nacimiento)+cod_materia,1),'0','4'),'1','4') NOTA
        ,Convert(datetime
,'200'+replace(replace(right(year(fecha_nacimiento),1),'8','0'),'9','1')+replace('0'+right(cod_materia+4,1
),'00','10')+replace('0'+right(cod_materia,1),'00','10'),112) FECHA
        ,0 FINAL
        ,0 NRO_DE_PARCIAL
        ,0 NRO_DE_RECUPERATORIO
FROM    ALUMNO a, MATERIA m
--Para cada Alumno, tomamos las Materias de una sola Carrera (segun su mes de nacimiento)
WHERE   month(fecha_nacimiento)+3=left(m.carrera,1);


--Si el Examen es en Enero, Febrero o Diciembre => lo seteamos como Final
UPDATE  NOTA
SET     FINAL=1
WHERE   month(fecha) IN (1,2,12);

--Para los restantes, si el mes es <= a 7 es 1er parcial
--y según el día es recuperatorio o no
UPDATE  NOTA
SET     NRO_DE_PARCIAL=1
WHERE   month(fecha)<=7
AND     final<>1;

UPDATE  NOTA
SET     NRO_DE_PARCIAL=2
WHERE   month(fecha)>7
AND     final<>1;

UPDATE  NOTA
SET     NRO_DE_RECUPERATORIO=1
WHERE   day(fecha) IN (5,7,9)
AND     final<>1;

UPDATE  NOTA
SET     NRO_DE_RECUPERATORIO=2
WHERE   day(fecha) IN (6,8,10)
AND     final<>1;
```