CI-0127 Bases de Datos
Grupo 01-I-2023
Estudiantes (del equipo LosOsitos):
Esteban Castañeda Blanco: C01795
Daniel Lizano Morales: C04285
Israel López Vallecillo: C04396
Carlos Quesada Estrada: C06133
Dylan Gabriel Tenorio Rojas: C07802
Quiz 7

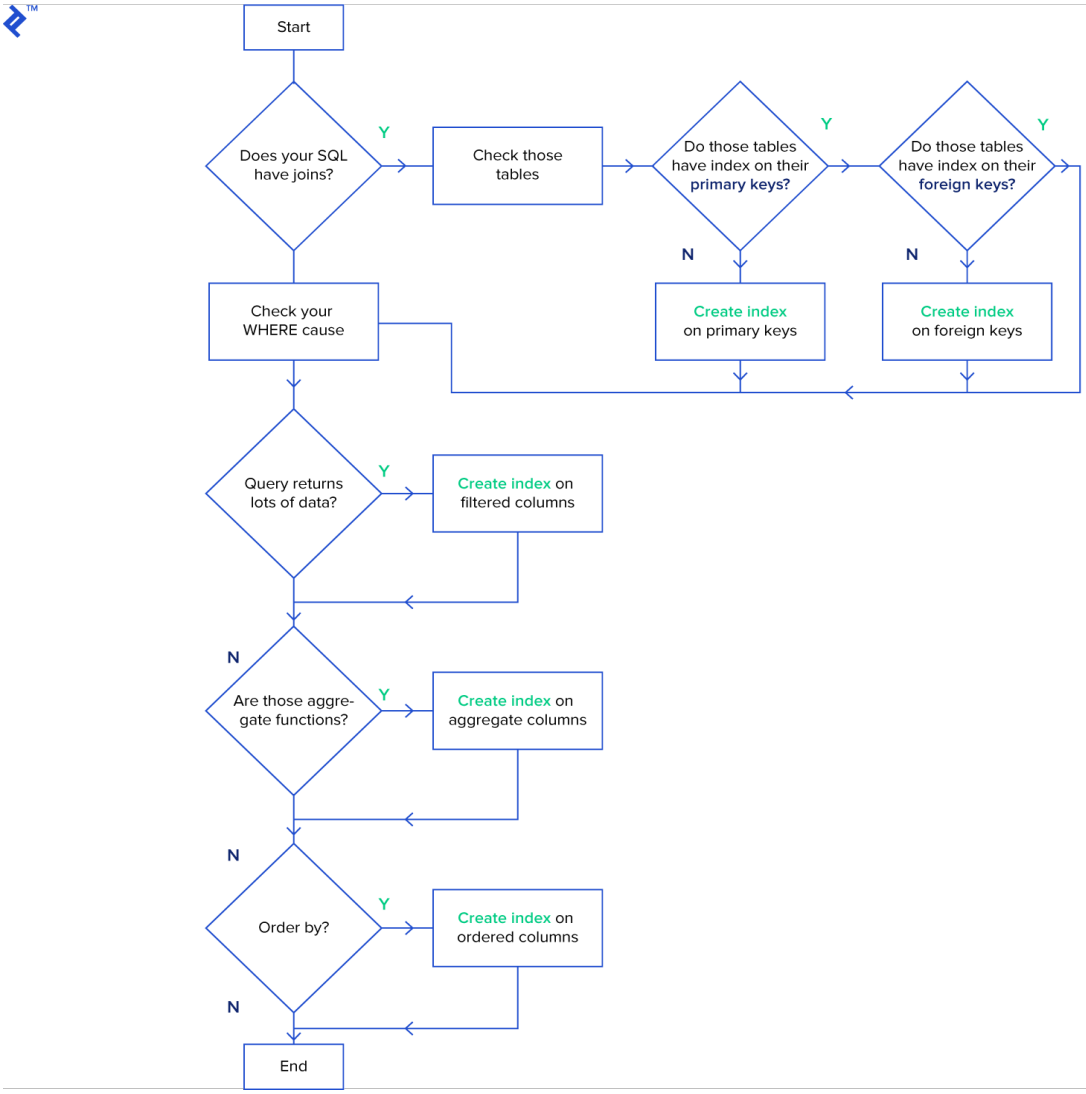

Imagen 1: Diagrama para implementar índices
Obtenido de: https://www.toptal.com/sql-server/sql-database-tuning-for-developers

Nota: Este diagrama se utilizó como guía para la creación de los índices

Índice 1

Realizado por:
Esteban Castañeda Blanco: C01795
Daniel Lizano Morales: C04285
Carlos Quesada Estrada: C06133

Nombre del índice: *Camping_Index*

Tabla sobre la cual se implementó: *Camping*

Justificación del por qué se decidió implementar:

La tabla de *Camping* es utilizada tanto en los *procedures* que se crearon para el PI, como para consultas realizadas en el *backend*. Los *procedures* son usados constantemente para determnar si aún queda espacio suficiente en el parque y la consulta que trae de la base de datos las reservas utiliza tanto la tabla de *Camping* como la de *Picnic* (sobre la cual se creó el otro índice). Además, se aprovechó las dos

Script para su creación:

```
CREATE INDEX Camping_Index ON Camping (Start_Date, End_Date)
```

* Este índice es de tipo *nonclustered*

Consulta sobre cual se probó el índice*

Nota*: Este índice, para este casó se probó sobre una de las consultas realizadas en el *procedure* llamado *RemainingCampingCapacity*, específicamente en :

```
SELECT Amount, Reservation_Method

FROM Ticket_Reservation JOIN Camping ON Ticket_Reservation.ID_Client =
Camping.ID_Client AND Camping.Reservation_Date = Ticket_Reservation.Reservation_Date
JOIN Reservation ON Reservation.ID_Client = Ticket_Reservation.ID_Client AND
Reservation.Reservation_Date = Ticket_Reservation.Reservation_Date

WHERE Reservation_Type = 1 AND @date BETWEEN Camping.Start_Date AND Camping.End_Date
```

En este caso, para realizar la prueba que el índice estaba siendo utilizado, se realizó lo siguiente:

```
DECLARE @date DATE;
SET @date = '2023-05-30';

SELECT Amount, Reservation_Method
             FROM Ticket_Reservation JOIN Camping ON Ticket_Reservation.ID_Client =
Camping.ID_Client AND Camping.Reservation_Date = Ticket_Reservation.Reservation_Date
             JOIN Reservation ON Reservation.ID_Client =
Ticket_Reservation.ID_Client AND Reservation.Reservation_Date =
Ticket_Reservation.Reservation_Date
```

```
            WHERE Reservation_Type = 1 AND @date BETWEEN Camping.Start_Date AND
Camping.End_Date
```

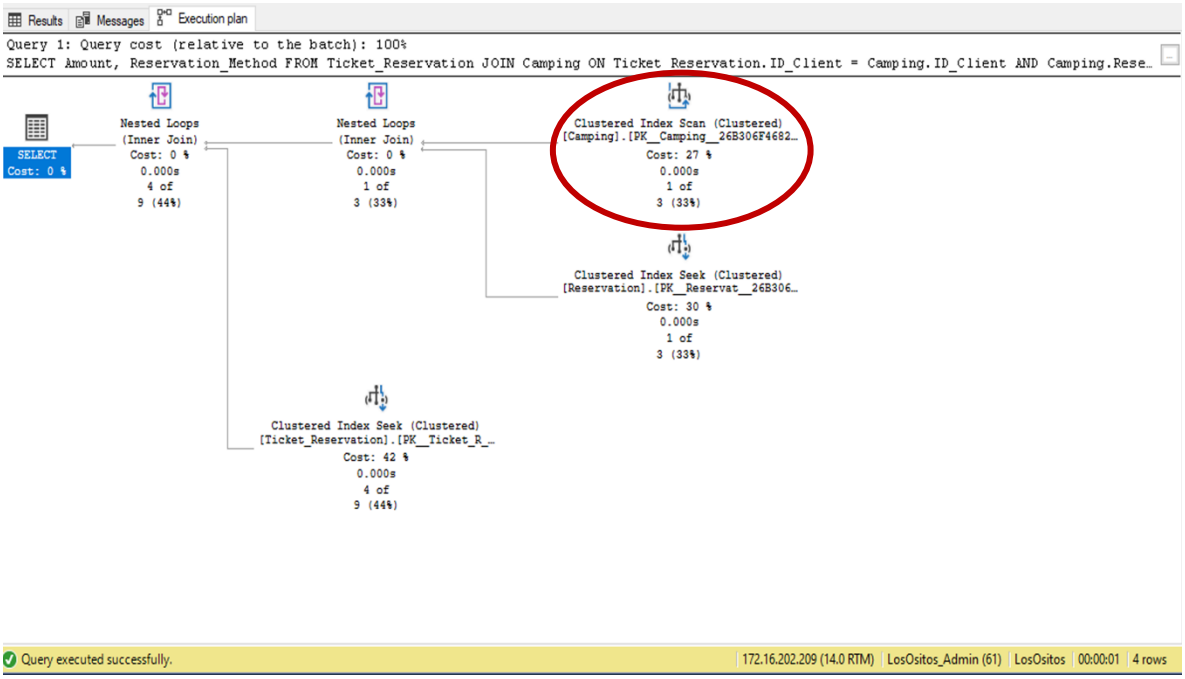

Imagen 2: Plan de implementación de la consulta sobre el *procedure RemainingCampingCapacity*

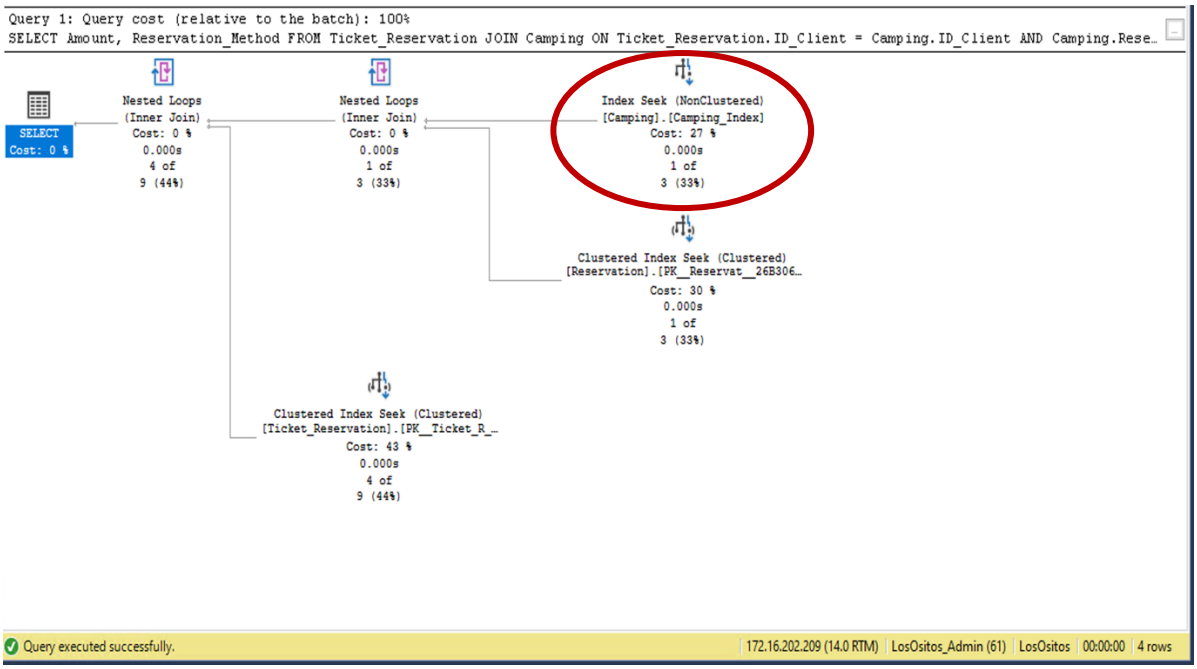

Imagen 3: Plan de implementación de la consulta sobre el *procedure RemainingCampingCapacity* con el nuevo índice creado

Como se puede observar, el índice creado sí es utilizado ahora en la consulta. En la imagen 2, se utilizaba el índice llamado *PK_Camping* (el índice *clustered* de *Camping*) para realizar un *Clustered Index Scan*, sin embargo, como se puede observar en la imagen 3, ahora se utiliza el *Camping_Index* para realizar un *Index_Seek*. Para entender la diferencia entre estos, hay que tener en cuenta que el índice *PK_Camping* está conformado por *ID_Client* y *Reservation_Date*. Por lo tanto, este no cuenta con *Start_Date* y *End_Date*. Al no tener estos atributos en el *clustered index*, el DBMS escaneaba todo el índice para poder leer las páginas que contuvieran los datos que cumplieran con la especificicación de la *query*. En cambio, como ahora se agregó *Camping_Index*, este tiene una mayor selectividad debido a que contiene *Start_Date* y *End_Date* (recordar que la consulta tiene lo siguiente: `WHERE Reservation_Type = 1 AND @date BETWEEN Camping.Start_Date AND Camping.End_Date`). Entonces, se utiliza, en vez de un *clustered index scan*, un *index seek*, el cual resulta más eficiente ya que recupera directamente las filas/tuplas del índice, sin necesidad de ir al archivo de la tabla.

Índice 2

Realizado por:
Dylan Gabriel Tenorio Rojas: C07802
Israel López Vallecillo: C04396

Nombre del índice: *Picnic_Index*

Tabla sobre la cual se implementó: *Picnic*

Justificación del por qué se decidió implementar:

Como se mencionó con la tabla de *Camping*, la tabla de *Picnic* también es utilizada tanto en los *procedures* que se crearon para el PI, como para consultas realizadas en el *backend*. Los *procedures* son usados constantemente para determnar si aún queda espacio suficiente en el parque y la consulta que trae de la base de datos las reservas utiliza tanto la tabla de *Camping* como la de *Picnic*.

Script para su creación:

```
CREATE INDEX Picnic_Index ON Picnic (Picnic_Date)
```

* Este índice es de tipo *nonclustered*

Consulta sobre cual se probó el índice*

Nota*: Este índice, para este casó se probó sobre una de las consultas realizadas en el *procedure* llamado *RemainingPicnicCapacity*, específicamente en :

```
SELECT Amount, Reservation_Method

FROM Ticket_Reservation JOIN Picnic ON Ticket_Reservation.ID_Client =
Picnic.ID_Client AND Ticket_Reservation.Reservation_Date = Picnic.Reservation_Date
JOIN Reservation ON Reservation.ID_Client = Picnic.ID_Client AND
Reservation.Reservation_Date = Picnic.Reservation_Date

WHERE Reservation_Type = 0 AND Picnic_Date >= @date AND Picnic_Date <
DATEADD(DAY, 1, @date)
```

En este caso, para realizar la prueba que el índice estaba siendo utilizado, se realizó lo siguiente:

```
DECLARE @date DATE;
SET @date = '2023-06-03'

SELECT Amount, Reservation_Method
	FROM Ticket_Reservation JOIN Picnic ON Ticket_Reservation.ID_Client =
Picnic.ID_Client AND Ticket_Reservation.Reservation_Date = Picnic.Reservation_Date
	JOIN Reservation ON Reservation.ID_Client = Picnic.ID_Client AND
Reservation.Reservation_Date = Picnic.Reservation_Date
	WHERE Reservation_Type = 0 AND Picnic_Date >= @date AND Picnic_Date <
DATEADD(DAY, 1, @date)
```

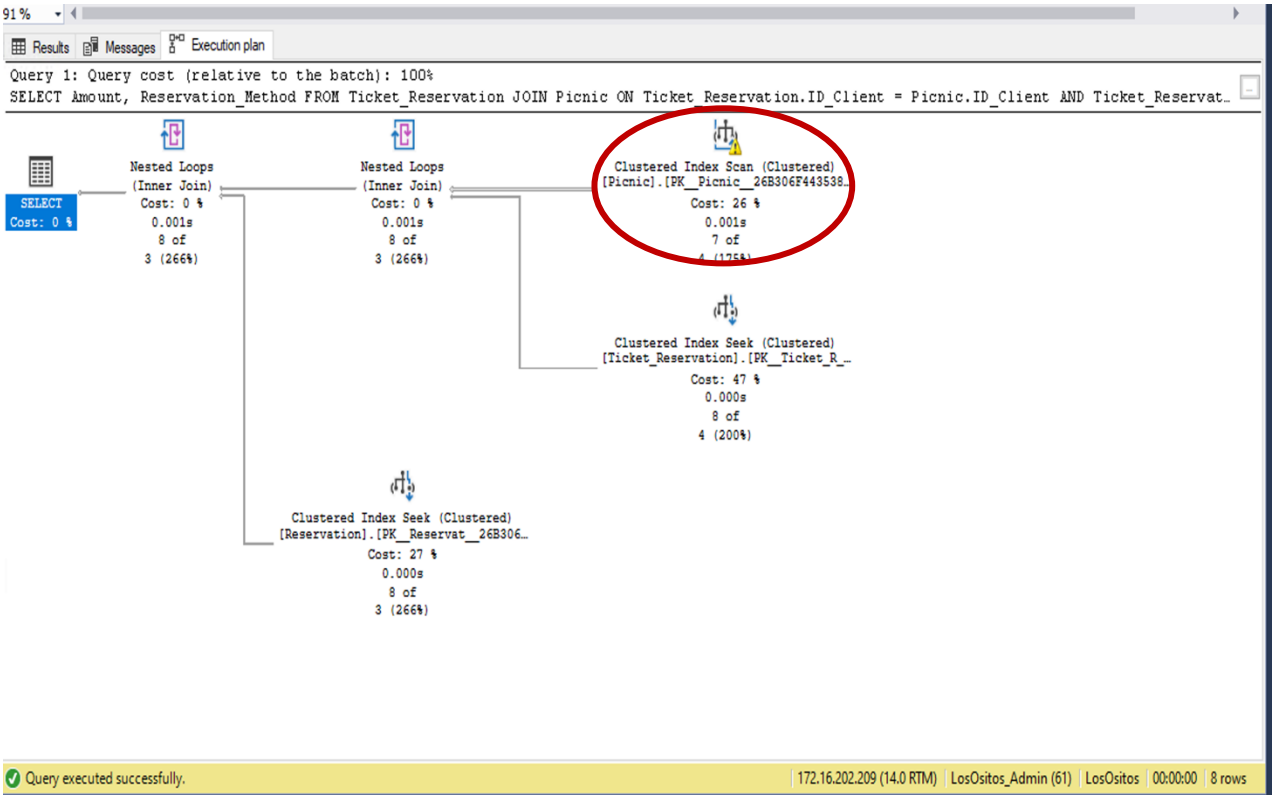

Imagen 4: Plan de implementación de la consulta sobre el *procedure RemainingPicnicCapacity*

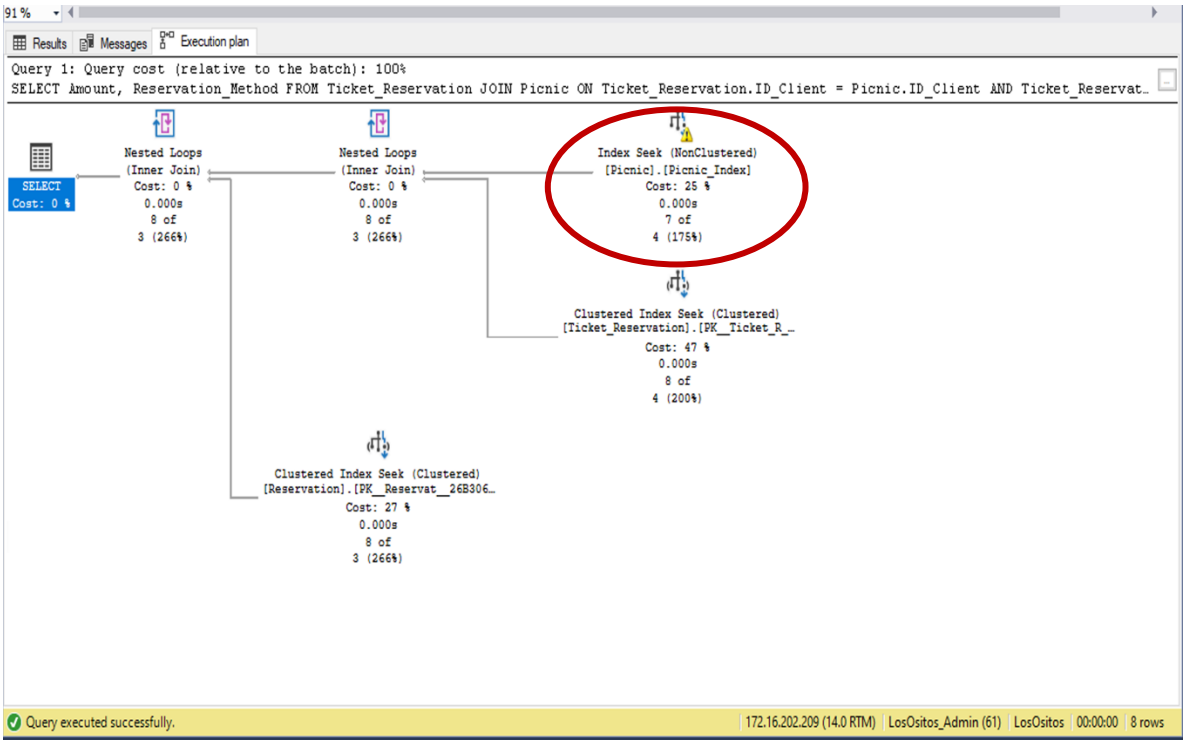

Imagen 5: Plan de implementación de la consulta sobre el *procedure* *RemainingPicnicCapacity* con el nuevo índice creado

Como se puede observar, en este caso, como en el pasado, el índice creado sí es utilizado ahora en la consulta. En la imagen 4, se utilizaba el índice llamado *PK_Picnic* (el índice *clustered* de *Pincnic*) para realizar un *Clustered Index Scan*, sin embargo, como se puede observar en la imagen 3, ahora se utiliza el *Picnic_Index* para realizar un *Index_Seek*. Para entender la diferencia entre estos, hay que tener en cuenta que el índice *PK_Camping* está conformado por *ID_Client* y *Reservation_Date*. Por lo tanto, este no cuenta con *Picnic_Date*. Al no tener este atributo en el *clustered index*, el DBMS escaneaba todo el índice para poder leer las páginas que contuvieran los datos que cumplieran con la especificicación de la *query*. En cambio, como ahora se agregó *Picnic_Index*, este tiene una mayor selectividad debido a que contiene *Picnic_Date* (recordar que la consulta tiene lo siguiente: `WHERE Reservation_Type = 0 AND Picnic_Date >= @date AND Picnic_Date < DATEADD(DAY, 1, @date)`). Entonces, se utiliza, en vez de un *clustered index scan*, un *index seek*, el cual resulta más eficiente ya que recupera directamente las filas/tuplas del índice, sin necesidad de ir al archivo de la tabla.

Extra

Como se ha mencionado, no solo es en los *procedures* que se utilizan las tablas de *Camping* y *Picnic*, sino que también en *queries* que se realizan en el *backend* para traer datos al *frontend*. Una de las que se mencionó es para traer las reservas realizadas por medio de la aplicación. Para demostrar el uso del índice, adjuntamos la *query* junto a los planes de ejecución antes y después de implementar los índices.

Consulta sobre cual se probaron los índices creados:

```
SELECT DISTINCT Person.ID, Person.Name, Person.Birth_Date, Person.State,
Person.Gender, Person.LastName1, Person.LastName2, Person.Email,
Person.Country_Name, Reservation_Method, Reservation.Status,
Reservation.Reservation_Date, Ticket_Reservation.Reservation_Type,
Camping.Start_Date, Camping.End_Date, Picnic.Picnic_Date
          FROM Reservation JOIN Person ON Reservation.ID_Client = Person.ID
          JOIN Ticket_Reservation ON Reservation.ID_Client =
Ticket_Reservation.ID_Client AND Reservation.Reservation_Date =
Ticket_Reservation.Reservation_Date
          FULL OUTER JOIN Camping ON Reservation.ID_Client = Camping.ID_Client AND
Reservation.Reservation_Date = Camping.Reservation_Date
          FULL OUTER JOIN Picnic on Reservation.ID_Client = Picnic.ID_Client AND
Reservation.Reservation_Date = Picnic.Reservation_Date
```

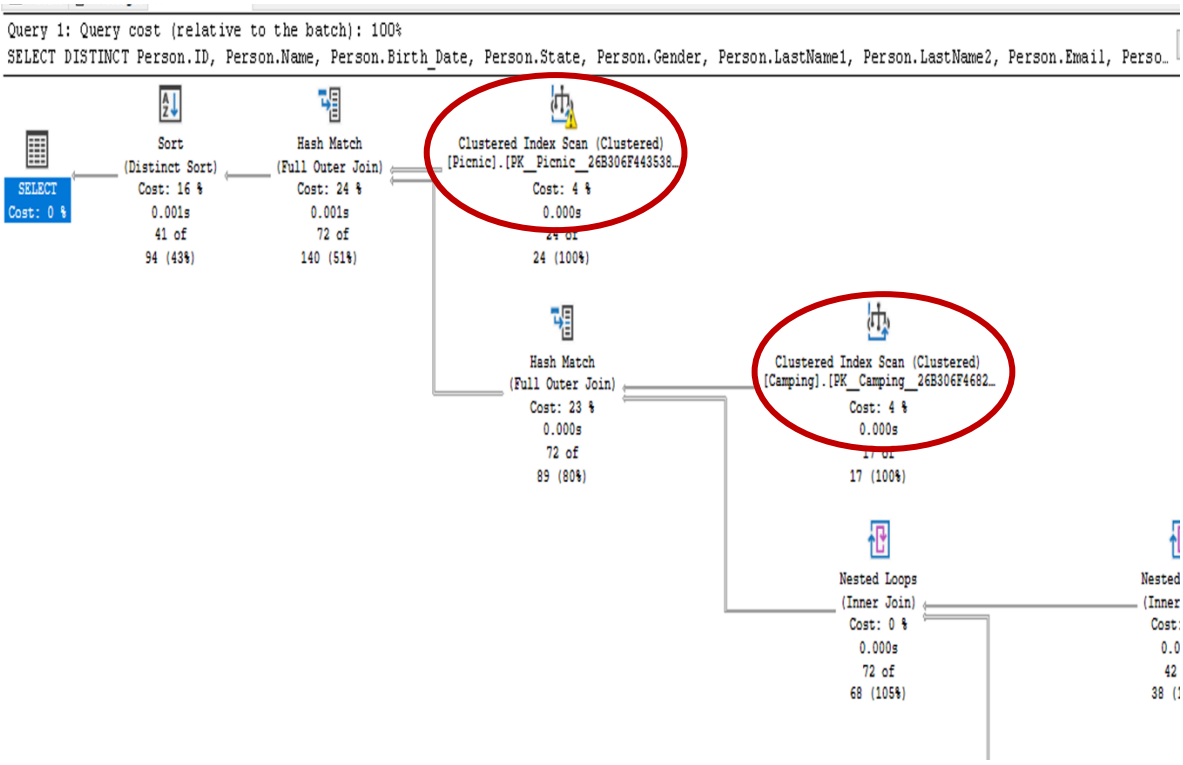

Imagen 6: Plan de implementación de la consulta antes de implementar los índices
Nota: debido a que la consulta es grande, solo se incluye la parte que hace referencia a los índices

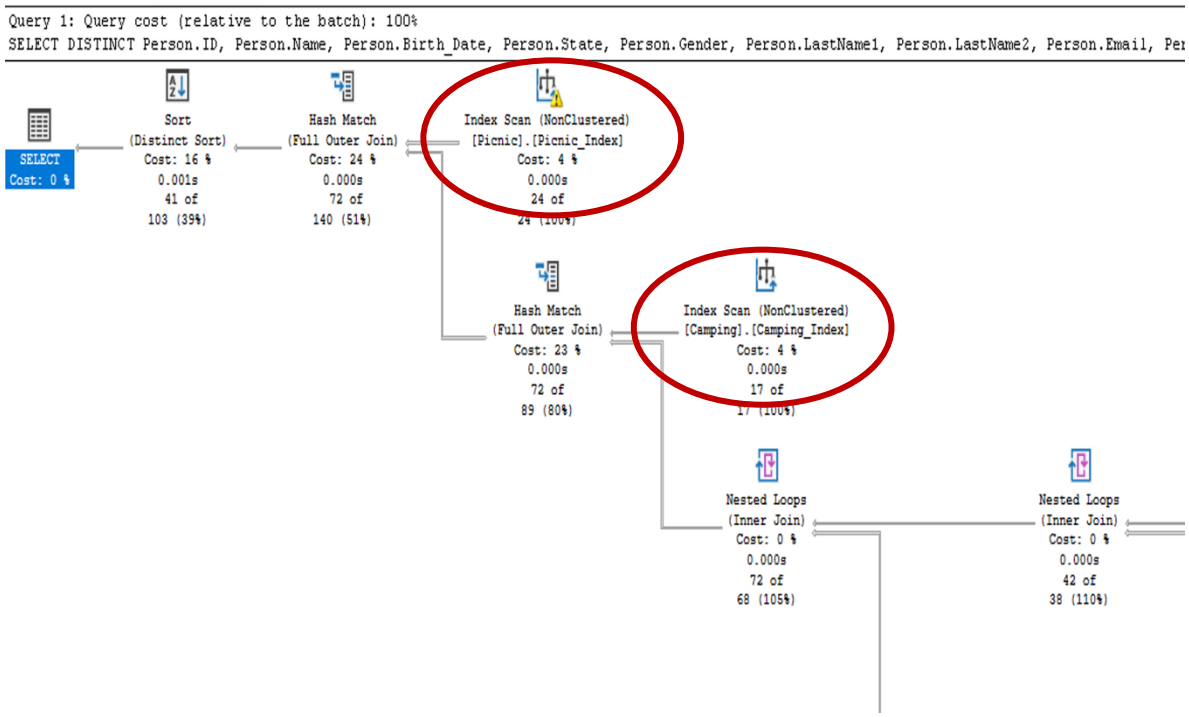

Imagen 7: Plan de implementación de la consulta después de implementar los índices
Nota: debido a que la consulta es grande, solo se incluye la parte que hace referencia a los índices

Como se puede observar en las imágenes 6 y 7, ahora se utiliza tanto *Camping_Index* como *Picnic_Index*. Sin embargo, se sigue realizando un *Index_Scan*.