



Universitat
de les Illes Balears

GRAU EN ENGINYERIA INFORMÀTICA

Sistemes de Gestió de Bases de Dades

Pràctica 2 EFL

Lluís Barca Pons
lluis.barca1@estudiant.uib.es

18 de setembre de 2024

0. Tasca prèvia

A continuació s'ha implementat un script amb SQL que ens permet crear tants de registres com necessitem, de forma aleatòria, per a cada taula. En aquest cas, se'ns ha demanat crear-ne 100.

Deixar constància que tot el codi executat consta de la següent sentència, per referir-se a la base de dades en concret:

```
1 SET search_path TO botiga;
```

A continuació els diferents scripts per a la creació dels diversos inserts que ens permeten introduir els registres a les diverses taules.

0.1. Taula ALIMENT

```
1 INSERT INTO ALIMENT (REFERENCIA, NOM, PREU)
2 SELECT
3     substring(md5(random()::text) from 1 for 13),
4     'Aliment ' || generate_series(1, 100),
5     floor(random() * 100)
6 FROM
7     generate_series(1, 100)
8 LIMIT 100;
```

0.2. Taula PROVEIDOR

```
1 INSERT INTO PROVEIDOR (NIF, NOM)
2 SELECT
3     substring(md5(random()::text) from 1 for 11),
4     'Proveidor ' || generate_series(1, 100)
5 FROM
6     generate_series(1, 100)
7 LIMIT 100;
```

0.3. Taula ALBARA

```
1  INSERT INTO ALBARA (CODI, NIF_PRO, DATA, FACTURAT)
2  SELECT
3      (floor(random() * 1000000)::integer) AS CODI,
4      (SELECT NIF FROM PROVEIDOR ORDER BY random() LIMIT 1),
5      current_date - (floor(random() * 3650) || ' days')::interval,
6      CASE floor(random() * 2) WHEN 0 THEN 'S' ELSE 'N' END
7  FROM
8      generate_series(1, 100);
```

0.4. Taula LINIA_ALBARA

```
1  INSERT INTO LINIA_ALBARA (CODI_ALB, REFERENCIA, QUILOGRAMS, PREU)
2  SELECT
3      ALBARA.CODI,
4      ALIMENT.REFERENCIA,
5      floor(random() * 100),
6      floor(random() * 1000)
7  FROM
8      ALBARA
9  CROSS JOIN
10     ALIMENT
11  JOIN
12     PROVEIDOR ON ALBARA.NIF_PRO = PROVEIDOR.NIF
13  LEFT JOIN
14     LINIA_ALBARA ON ALBARA.CODI = LINIA_ALBARA.CODI_ALB AND
15     ↪ ALIMENT.REFERENCIA = LINIA_ALBARA.REFERENCIA
16  WHERE
17     LINIA_ALBARA.CODI_ALB IS NULL; -- Garantir que REFERENCIA no estigui a
18     ↪ LINIA_ALBARA per al mateix CODI_ALB
```

0.5. Taula VENDA

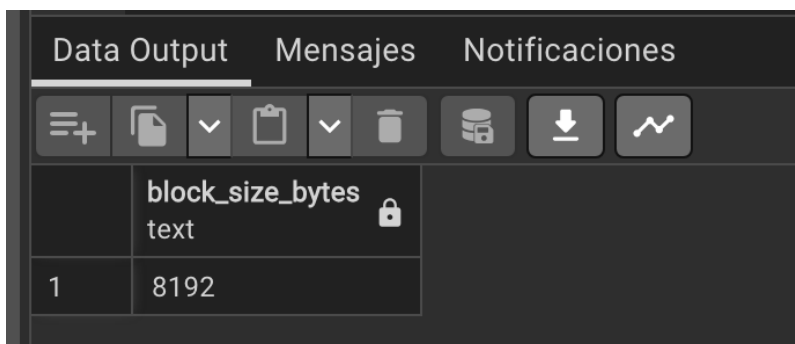
```
1  -- Obtenir una llista de totes les referències dels aliments existents a
   ↳ la taula ALIMENT.
2  WITH referencias AS (
3      SELECT REFERENCIA FROM ALIMENT
4  ),
5
6  -- Obtenir una llista de tots els codis d'albarans existents en la taula
   ↳ ALBARA.
7  codigos_albaran AS (
8      SELECT CODI FROM ALBARA
9  )
10
11 INSERT INTO VENDA (CODI, REFERENCIA, DATA, QUILOGRAMS, PREU)
12 SELECT
13     ROW_NUMBER() OVER () AS CODI,
14     referencias.REFERENCIA,
15     ALBARA.DATA,
16     FLOOR(RANDOM() * LINIA_ALBARA.QUILOGRAMS) + 1 AS QUILOGRAMS,
17     FLOOR(RANDOM() * (ALIMENT.PREU - 1)) + 1 AS PREU
18 FROM
19     referencias
20     JOIN ALIMENT ON referencias.REFERENCIA = ALIMENT.REFERENCIA
21     JOIN LINIA_ALBARA ON ALIMENT.REFERENCIA = LINIA_ALBARA.REFERENCIA
22     JOIN ALBARA ON LINIA_ALBARA.CODI_ALB = ALBARA.CODI
23     JOIN codigos_albaran ON ALBARA.CODI = codigos_albaran.CODI;
```

1. Grandària del bloc

Per a obtenir la grandària del bloc, es pot utilitzar la següent comanda:

```
1 SHOW block_size;
```

Que ens retorna la següent imatge:



	block_size_bytes
1	8192

El que ens indica que tenim una grandària de bloc de 8192 bytes, la grandària per defecte de PostgreSQL.

2. Anàlisi de la taula Aliment i extracció de registres per bloc

La taula Aliment consta de tres columnes (Referència, Preu i Nom). Per tant, haurem de calcular la grandària per fila (registre) tenint en compte aquestes columnes.

```
1 SELECT
2     pg_column_size('referencia') + pg_column_size('nom') +
3     ↪ pg_column_size('preu') AS tamany_fila
4 FROM
5     Aliment
6 LIMIT 1;
```

Que retorna un total de 20 bytes per fila. Llavors ara podem procedir a calcular els registres que poden caber dins un bloc:

```
1 SELECT
2     floor(block_size / 20) AS registres_bloc
3 FROM
4     (SELECT current_setting('block_size')::int AS block_size FROM Aliment
5     ↪ LIMIT 1) AS subquery;
```

$$\text{Nombre de registres per bloc} = \frac{\text{Tamany del bloc}}{\text{Tamany de la fila}} = \frac{8192 \text{ bytes}}{20 \text{ bytes}} = 409,6 \text{ registre per fila}$$

Que ens retorna un total de 409,6 registres per bloc, que arrodonint a la baixa ens dona 409 registres/bloc.

3. Creació d'un nou *tablespace*

Des de la PSQL Tool podem crear el nou *tablespace*, però abans haurem d'assignar els permisos corresponents a la ruta on volem emmagatzemar-lo (assignant l'usuari que vulguem que accedeixi).

```
1 chown -R postgres:postgres /Users/luisbarcap/pgsql/data
```

Ara si podem crear el nou *tablespace*:

```
1 CREATE TABLESPACE tb1 LOCATION /Users/luisbarcap/pgsql/data
```

4. Trasllat de la base de dades al nou *tablespace*

Primer hem de modificar el *tablespace* per defecte de la nostra base de dades.

```
1 ALTER DATABASE botiga SET TABLESPACE tb1;
```

A continuació traslladem totes les taules a la base de dades ubicada al nou *tablespace*.

```
1 ALTER TABLE Aliment SET TABLESPACE tb1;
2 ALTER TABLE Albara SET TABLESPACE tb1;
3 ALTER TABLE Proveidor SET TABLESPACE tb1;
4 ALTER TABLE Linia_Albara SET TABLESPACE tb1;
5 ALTER TABLE Venda SET TABLESPACE tb1;
```

5. Anàlisi de la taula Albarà i extracció de registres per bloc

La taula Albarà consta de quatre columnes (Codi, NIF_Pro, Data i Facturat). Per tant, haurem de calcular la grandària per fila (registre) tenint en compte aquestes columnes.

```
1  SELECT
2      pg_column_size('codi') + pg_column_size('nif_pro') +
      ↪ pg_column_size('data') + pg_column_size('facturat') AS tamany_fila
3  FROM
4      Albara
5  LIMIT 1;
```

Que retorna un total de 27 bytes per fila. Llavors ara podem procedir a calcular els registres que poden caber dins un bloc:

```
1  SELECT
2      floor(block_size / 27) AS registres_bloc
3  FROM
4      (SELECT current_setting('block_size')::int AS block_size FROM Albara
      ↪ LIMIT 1) AS subquery;
```

$$\text{Nombre de registres per bloc} = \frac{\text{Tamany del bloc}}{\text{Tamany de la fila}} = \frac{8192 \text{ bytes}}{27 \text{ bytes}} = 303,4 \text{ registre per fila}$$

Que ens retorna un total de 303,4 registres per bloc, que arrodonint a la baixa ens dona 303 registres/bloc.

5. Modificació de la grandària del bloc a 64K

Per a dur a terme la correcta modificació de la grandària del bloc, hem d'instal·lar de nou la versió de PostgreSQL que desitgem, però utilitzant el codi font i no un instal·lador per defecte.

En aquest cas el que s'ha de fer és baixar des de la pàgina oficial el codi font i accedir des de la terminal al directori concret per a executar la següent comanda:

```
1  ./configure --with-blocksize=64
```

Això ens permetrà configurar la compilació de PostgreSQL amb una grandària de bloc modificada. El problema és que aquesta grandària no és acceptada tal com podem veure en la següent imatge de la consola:

```
~/Downloads (0.024s)
cd postgresql-14.7

~/Downloads/postgresql-14.7 (4.946s)
./configure --with-blocksize=64
checking build system type... x86_64-apple-darwin23.0.0
checking host system type... x86_64-apple-darwin23.0.0
checking which template to use... darwin
checking whether NLS is wanted... no
checking for default port number... 5432
checking for block size... configure: error: Invalid block size. Allowed values are 1,2,4,8,16,32.
```

Per això es realitzarà la modificació a 32 K; utilitzant la mateixa comanda. Una vegada executada, haurem de compilar el programa.

```
1 make
2 sudo make install
```

A continuació podem inicialitzar un nou clúster amb la nova grandària de bloc

```
1 /usr/local/pgsql/bin/initdb -D /Users/luisbarcap/pgsql/data
```

Iniciem el servidor de PostgreSQL des del nou clúster:

```
1 /usr/local/pgsql/bin/pg_ctl -D /Users/luisbarcap/pgsql/data start
```

Ara ja, tindriem la nova instal·lació de PostgreSQL amb les modificacions pertinents.

6. Anàlisi de la taula Aliment i Albarà, i extracció de registres per bloc

Les comandes SQL en aquest apartat s'han obviat perquè són les mateixes que a l'apartat 2 i 5, modificant les dades corresponents. A més, les taules Aliment i Albarà tenen les mateixes columnes, l'únic que canvia és la grandària del bloc que a continuació es calcularà.

6.1. Taula Aliment

$$\text{Nombre de registres per bloc} = \frac{\text{Tamany del bloc}}{\text{Tamany de la fila}} = \frac{32.768 \text{ bytes}}{20 \text{ bytes}} = 1638,4 \text{ registre per fila}$$

Que arrodonit a la baixa ens dona un total de 1638 registres/bloc.

6.2. Taula Albarà

$$\text{Nombre de registres per bloc} = \frac{\text{Tamany del bloc}}{\text{Tamany de la fila}} = \frac{32.768 \text{ bytes}}{27 \text{ bytes}} = 1213,6 \text{ registre per fila}$$

Que arrodonit a la baixa ens dona un total de 1213 registres/bloc.

7. Anàlisi i explicació de les diferències

Clarament, podem observar com amb una grandària de bloc més gran, podem emmagatzemar més registres/bloc. Però, el que realment ens interessa és saber en quines situacions és més eficient una grandària o una altra.

- **Mida de Bloc Més Petita (per exemple, 8K):**

- Avantatges:

- * **Menor Ús de Memòria:** Es poden carregar més blocs en memòria.
- * **Optimització per OLTP:** Les aplicacions OLTP (processament de transaccions en línia) que involucren operacions de lectura i escriptura freqüents poden beneficiar-se de blocs més petits, ja que redueixen l'impacte de les operacions d'E/S.

- Desavantatges:

- * **Major Sobrecàrrega d'E/S:** Com que hi ha més blocs per llegir i escriure per a la mateixa quantitat de dades, pot haver-hi una major sobrecàrrega d'E/S, especialment per a operacions que impliquen grans volums de dades.
- * **Fragmentació:** Pot haver-hi més fragmentació interna al disc, la qual cosa porta a un ús menys eficient de l'espai en disc.

- **Mida de Bloc Més Gran (per exemple, 32K o 64K):**

- Avantatges:

- * **Menor Sobrecàrrega d'E/S:** Com que hi ha menys blocs per llegir i escriure per a la mateixa quantitat de dades, pot haver-hi una menor sobrecàrrega d'E/S, especialment per a operacions que impliquen grans volums de dades.
- * **Menor Fragmentació:** Hi ha menys fragmentació interna, la qual cosa porta a un ús més eficient de l'espai en disc.

- Desavantatges:

- * **Major Ús de Memòria:** Els blocs més grans poden ocupar més memòria, que pot ser una consideració en sistemes amb recursos limitats.
- * **Optimització per OLAP (processament analític en línia):** Impliquen anàlisis complexes i consultes agregades poden beneficiar-se de blocs més grans, ja que redueixen la quantitat d'E/S necessària per accedir a grans volums de dades.

En resum, l'elecció de la mida del bloc ha de basar-se en les característiques específiques de l'aplicació i els requisits de rendiment del sistema. Per a aplicacions OLTP amb moltes operacions de lectura i escriptura petites, els blocs més petits poden ser adequats. Per a aplicacions OLAP que impliquen anàlisis complexes i grans volums de dades, els blocs més grans poden oferir un millor rendiment. És important dur a terme proves i ajustos específics del sistema per trobar la mida del bloc que millor s'adapti a les necessitats de l'aplicació.