Sperimentare

- Abbiamo visto sulla carta, ma possiamo toccare con mano

- Questa presentazione mostra
  - l'esempio visto in dettaglio a lezione sull'esercizio già svolto
  - una piccola riflessione sui buffer (da riprendere nella prossima lezione)

Esempio reale con Postgres 09/04/2018 domanda 3

- Creiamo le strutture

```
---drop  schema "20180409dom3" cascade;
create schema "20180409dom3";

set search_path to "20180409dom3";

drop table if exists r1;
create table r1 ( a numeric not null primary key,
                  b numeric,
                  c numeric);
drop table if exists r2;
create table r2 ( d numeric not null primary key,
                  e numeric,
                  f numeric);

---alter table r1 add constraint fk1 foreign key (c)
  references r2 (d);
```

Esempio reale con Postgres (2)

- popoliamo le strutture
  - con insert (un po' noioso)
  - con copy da file excel
  - con script, esempio (guardare il manuale per capire il significato e per applicare varianti)

Esempio reale con Postgres: script

```
delete from r2;
create or replace function loadr2() returns
  void as $$
begin
  for i in 1..4000000 loop
    insert into r2 (d,e,f)
    values(i, floor(random()*100+1),
              floor(random()*1000)+1);
  end loop;
end;
$$ language plpgsql;
select loadr2();
```

Esempio reale con Postgres: script (2)

```
delete from r1;
create or replace function loadr1() returns
  void as $$
begin
  for i in 1..2000000 loop
    insert into r1(a,b,c)
    values(i, floor(random()*200000+1),
              floor(random()*4000000)+1);
  end loop;
end;
$$ language plpgsql;
select loadr1();
```

Vediamo le dimensioni

- Dimensione del blocco
  SHOW block_size;
  SELECT current_setting('block_size');
- Spazio occupato da un valore di un certo tipo
  SELECT pg_column_size(1::numeric);
- Spazio occupato da una tabella in MB
  SELECT pg_size_pretty (pg_table_size('r1'));
- Spazio occupato dagli indici di una tabella in MB
  SELECT pg_size_pretty (pg_indexes_size('r2'));
- Numero di ennuple e numero di blocchi di una tabella
  ANALYZE VERBOSE r1

https://www.postgresql.org/docs/11/functions-admin.html
https://www.postgresql.org/docs/11/runtime-config-preset.html
https://www.postgresql.org/docs/11/sql-show.html
(https://www.postgresqltutorial.com/postgresql-database-indexes-table-size/)
(https://www.postgresql.org/docs/11/sql-analyze.html)

Nell'esempio

- Dimensione del blocco 8 KB
- Dimensione di un attributo di tipo numerico 8 B
- Dimensione di una ennupla (tre attributi numerici) 24 B
- Fattore di blocco (calcolato da noi): 8192 / 24 = ca 333
- Con ANALYZE VERBOSE
  - Numero di ennuple 4.000.000
  - Numero di blocchi 25.469

– Il fattore di blocco è quindi molto più piccolo (157), perché un blocco contiene molte informazioni di servizio

(http://www.interdb.jp/pg/pgsql01.html)

Vediamo quale join viene scelto

```
set search_path to "20180409dom3"

select *
from r1 join r2 on C=D
```

Possiamo forzare il metodo di join (inibendo gli altri); eseguire varie prove a conferma di quanto scritto nel compito

```
set enable_hashjoin=0; -- 0 disabilita 1 abilita
set enable_mergejoin=0;
set enable_nestloop=0
set max_parallel_workers = 0
       ---disabila parallelismoi (in un certo senso)

explain analyze
select * from r1 join r2 on c=d;
```

Interrogazioni 2 e 2bis

```
select *
from r1 join r2 on C=D
where b >= 41 and b <= 45

create index i12 on r1(B);

select *
from r1 join r2 on C=D
where b >= 41 and b <= 45
```

Risultati

- Conferma di quanto visto sulla carta
  - interrogazione 1
    - conviene hash-join, poi merge-join, ultimo nested-loop
  - Interrogazione 2
    - conviene nested-loop, poi hash-join, ultimo merge-join
  - Interrogazione 2bis
    - conviene di gran lunga nested-loop, poi hash-join, ultimo merge-join

Statistiche disponibili per monitorare letture e uso dei buffer

- Qualche riflessione sui buffer, pin e read
- Tabella con statistiche, sull'intero db

  `pg_stat_database`
- Il dettaglio non è rilevante, usiamo i due campi che indicano il numero di letture fisiche (blks_read) e il numero di record trovati nel buffer (blks_hit) – la somma è il numero di pin richieste

```
3  select blks_hit, blks_read
4  from pg_stat_database
5  where datname like 'postgres';
```

Data Output | Explain | Messages | Notifications

	blks_hit bigint	blks_read bigint
1	16785308	506977

Scansione sequenziale

```
drop table  if exists  prima cascade;
select blks_hit+blks_read as numPin, blks_read as numRead into
prima
from pg_stat_database where datname like 'postgres';

select * from r2;

select ((dopo.blks_hit+dopo.blks_read) -
        prima.numPin) as numPin,
       ((dopo.blks_read)-prima.numRead) as numRead
from pg_stat_database as dopo , prima
where datname like 'postgres'
```

	numpin bigint	numread bigint
1	26503	24553

Scansione sequenziale più piccola

```
select * from r2 where d < 100000;

drop table  if exists  prima cascade;
select blks_hit+blks_read as numPin, blks_read as numRead into
prima
from pg_stat_database where datname like 'postgres';

select * from r2 where d < 100000;

select ((dopo.blks_hit+dopo.blks_read) -
        prima.numPin) as numPin,
       ((dopo.blks_read)-prima.numRead) as numRead
from pg_stat_database as dopo , prima
where datname like 'postgres'
```

	numpin bigint	numread bigint
1	1225	1

Perché il numero è cambiato così drasticamente?

- Nel secondo caso il numero di blocchi visitati dall'interrogazione entra tutto nel buffer e quindi la seconda esecuzione trova tutti i blocchi che le servono nel buffer