HOMEWORK 1

Sono definite due relazioni R1($\underline{A}, B, C$) e R2($\underline{D}, E, F$) con vincolo di integrità referenziale tra su C di R1 e D di R2 e caricate su PostgresSQL.

Le relazioni vengono create e popolate:

```
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);

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();

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();
```

0.1. **Prima interrogazione: join.**
SELECT * from r1 join r2 on C = D

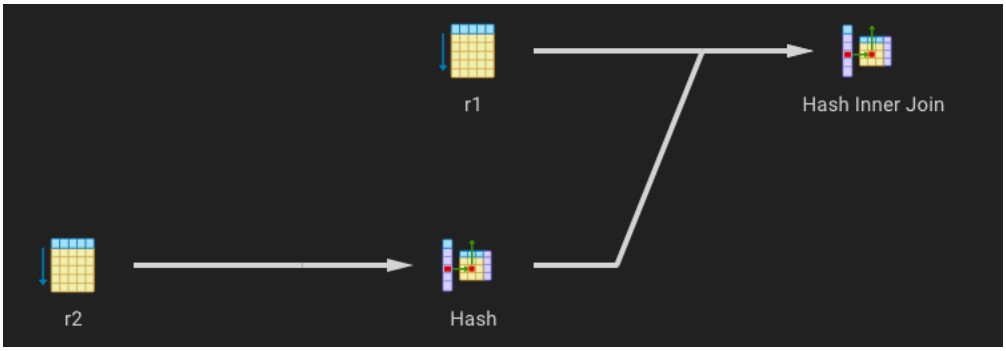

FIGURE 1. Per fare il join viene scelto l'hash join

0.2. **Seconda interrogazione: join con selezione.**
SELECT * from r1 join r2 on C = D where b = 100

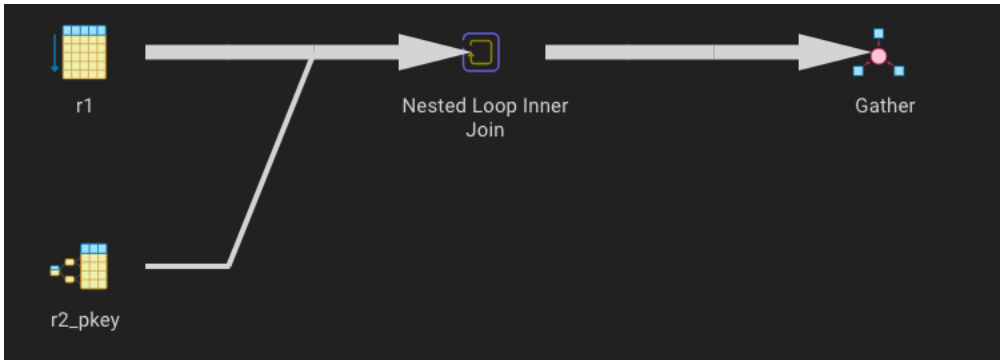

FIGURE 2. Per fare il join viene scelto il nested loop con indice

0.3. **Terza interrogazione: join con selezione e proiezione.**
SELECT a, b from r1 join r2 on C = D where b = 100

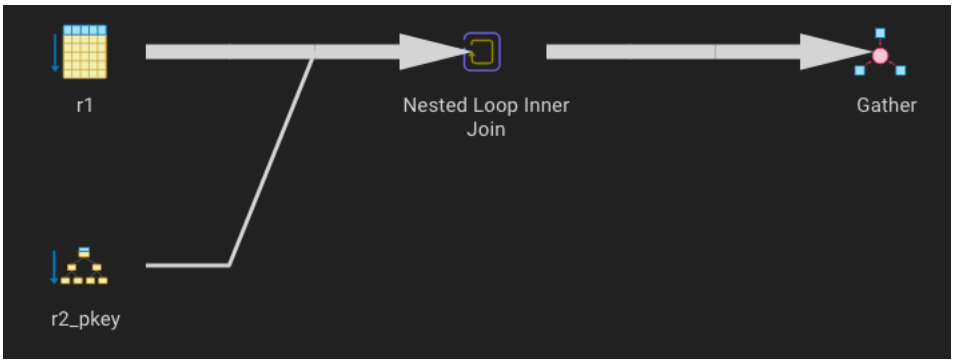

FIGURE 3. Per fare il join viene scelto il nested loop con indice

0.4. **Tentativo: impedire a PostgresSQL di usare il Nested Loop.**
Si ripete la Query precedente (la 0.3) con l'opzione: SET ENABLE_NESTLOOP TO FALSE;

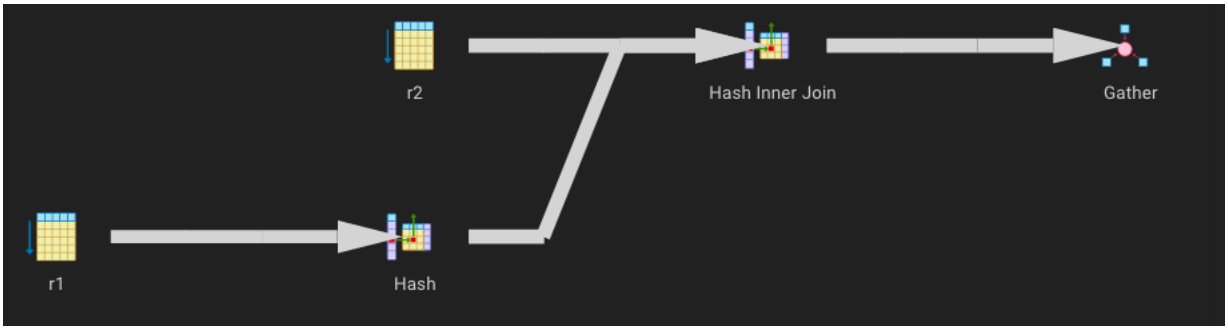

FIGURE 4. Per fare il join viene scelto di nuovo l'hash join

0.5. **Conclusioni.**

Date le relazioni R_1 e R_2 e i risultati delle query si può osservare che quando il join è selettivo conviene utilizzare il Nested loop (con indice sulla relazione interna), quando il join riguarda un numero considerevole di record conviene utilizzare l'Hash join. L'Hash join viene utilizzato con una funzione hash che comprende un numero di valori distinti paragonabile al numero di pagine P di buffer. Indicando rispettivamente con B_1 e B_2 il numero di blocchi delle relazioni R_1 e R_2, se $P^2 > \min(B_1, B_2)$ il costo del join è $3(B_1+B_2)$. Il Nested loop con indice sulla relazione interna invece ha un costo di circa $B_1+L_1*(I_2-2+1)$ (indicando con L_1 il numero di record di R_1 e con I_2 la profondità dell'indice di R_2).

(1) Il costo del Nested loop è alto se il numero L_1 è molto grande (e quindi va fatto un numero considerevole di accessi all'indice di R_2).
(2) Al contrario, se il join viene fatto su un campo molto selettivo, il numero di accessi all'indice di R_2 diminuisce ed il costo complessivo del Nested loop diminuisce.