# ***Progetto Basi Di Dati – Social market***

# ***Anno accademico 2021-2022***

**PARTE 3**

Simone Lutero 4801326

Ilaria Bruzzone 4844842

Debora Rustemaj 4684484

**9)Progetto fisico e validazione, articolato in:**

**Carico di lavoro composto da 3 interrogazioni:**

|  | **Gestione** | **Operazione** | **Frequenza giornaliera** |
| --- | --- | --- | --- |
| **Query 1** | Famiglia | Selezionare le famiglie con un saldopunti inferiore a 100 |  |
| **Query 2** | Prodotto | Selezionare l’elenco dei prodotti deperibili |  |
| **Query 3** | Turno | Seleziona nome del volontario e il tipo di servizio offerto per ogni turno |  |

1. **Descrizione in linguaggio naturale e codice SQL implementato per implementare le interrogazioni del carico di lavoro (Visionare file sql3parte.sql)**
   1. La Query 1 restituisce le famiglie con un saldopunti maggiore di 0. Potrebbe essere utile per avvisare le famiglie sul fatto che stanno esaurendo i punti.

SELECT nome, cognome FROM famiglia WHERE saldoPunti < 100;

* 1. La Query 2 restituisce un elenco dei prodotti deperibili

SELECT nome FROM prodotto WHERE deperibile = true;

* 1. La Query 3 stampa per ogni volontario, il loro nome e il tipo di servizio effettuato per ogni turno

SELECT v.nome, t.tipoServizio FROM volontario v JOIN turno t ON v.cf = t.volontario;

1. **Progetto fisico, contenente l’elenco degli indici che si intendono creare  
   per le interrogazioni contenute nel carico di lavoro (specificando relazione  
   di riferimento e chiave di ricerca), il loro tipo (ordinato/hash, clusterizzato/  
   non clusterizzato) e la motivazione che ha portato alla loro creazione. (Visionare file sql3parte.sql)**

| **Indici** | **Tipo indice** | **Relazione** | **Chiave di ricerca** | **Clusterizzato**? |
| --- | --- | --- | --- | --- |
| 1 | Ordinato | Famiglia | saldopunti | No |
| 2 | Ordinato | Prodotto | deperibile | No |
| 3 | Ordinato | Turno | volontario | No |

1. Per la Query 1 abbiamo creato un indice sulla colonna saldopunti che consentirà di accelerare la ricerca delle famiglie con un saldo di punti inferiore a 100, poiché creerà una struttura di dati ottimizzata per questa colonna.

CREATE INDEX idx\_saldoPunti ON famiglia (saldoPunti);

1. Per quanto riguarda la Query 2 abbiamo creato un indice sulla colonna deperibile in prodotto, che realizzerà una struttura dati per accelerare la ricerca dei prodotti deperibili.

CREATE INDEX idx\_deperibile ON prodotto (deperibile);

1. Anche per la Query 3 abbiamo creato un indice ordinato, sulla colonna volontario di Turno.

CREATE INDEX idx\_appuntamento\_cfvol ON turno (volontario);

1. **Una tabella che, per ogni tabella coinvolta nelle operazioni del carico di lavoro,  
   riporti il numero di tuple inserite e la dimensione in blocchi della tabella. (Visionare file sql3parte.sql)**

| **Relazione** | **Qta tuple** | **Dim blocchi** |
| --- | --- | --- |
| Volontario |  |  |
| Prodotto |  |  |
| Turno |  |  |

/\*DATI TABELLA DA INSERIRE\*/

SELECT RelName AS Nome\_Tabella, RelPages AS Num\_Pagine

FROM pg\_class

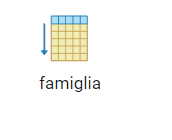
WHERE Nome\_Tabella IN('cliente','famiglia','appuntamento','prodotto');

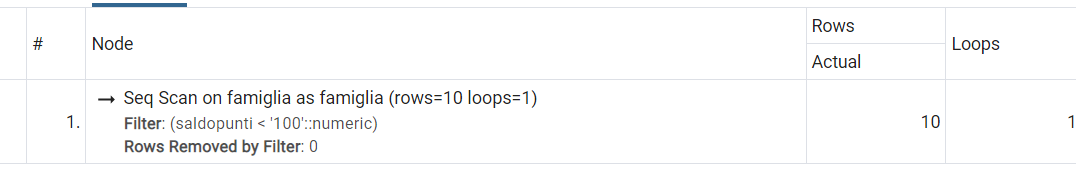
SELECT COUNT(\*) FROM cliente/famiglia/appuntamento/prodotto;

1. **Descrizione dei piani di esecuzione scelti dal sistema (prima e dopo la creazione  
   dello schema fisico) per le interrogazioni contenute nel carico di lavoro,  
   riportando la visualizzazione del piano di esecuzione prodotta da PostgreSQL e i  
   tempi di esecuzione (prima e dopo la creazione dello schema fisico); confrontare  
   i piani e i tempi ottenuti nei due casi, fornendo una giustificazione per i risultati  
   ottenuti.**

**QUERY 1:** piano di esecuzione prima di creare lo schema fisico:

Totally query runtime= 83 msec





Piano di esecuzione dopo la creazione dello schema fisico:

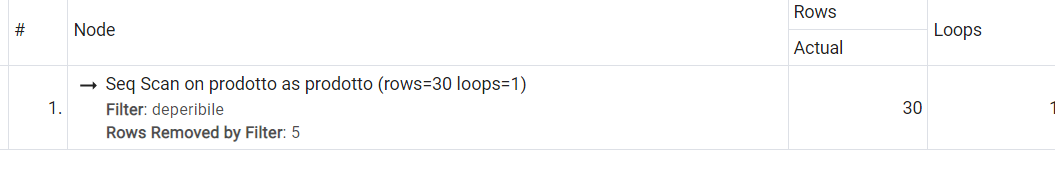
Totally query runtime: 49 msec

La prima query si basa sull’accesso ad una unica tabella, e come cammino di accesso da utilizzare verrà scelta la scansione sequenziale. La scansione sequenziale è l’unico cammino di accesso disponibile prima di creare lo schema fisico. Il cammino scelto, visto che deve essere scelto il cammino di costo minore, prenderà come riferimento la chiave di ricerca “saldopunti”, come specificato nell’indice ordinato.

**QUERY 2:** piano di esecuzione prima di creare lo schema fisico:

Totally query runtime= 89 msec





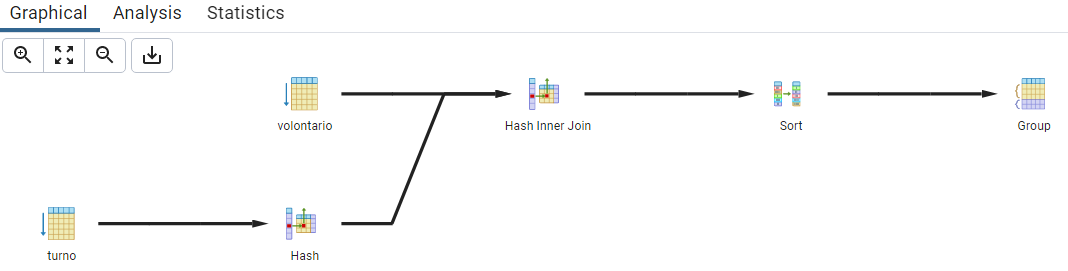
Piano di esecuzione dopo la creazione dello schema fisico:

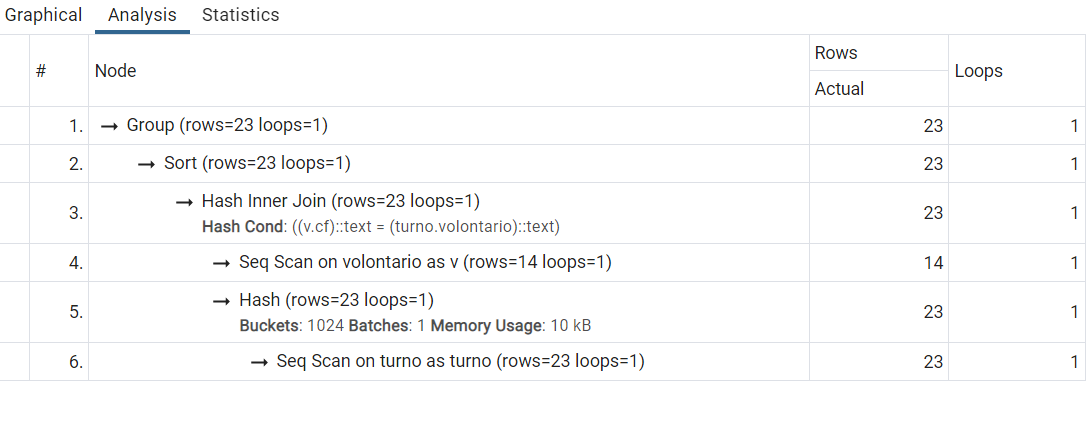
Totally query runtime = 75 msec

La seconda query si basa sull’accesso ad una unica tabella, e come cammino di accesso da utilizzare verrà scelta la scansione sequenziale. Anche qui la scansione sequenziale è l’unico cammino di accesso disponibile prima di creare lo schema fisico. Il cammino scelto prenderà come riferimento la chiave di ricerca “deperibile”, come specificato nell’indice ordinato al fine di velocizzare la ricerca e prendere direttamente i prodotti deperibili.

**QUERY 3:** piano di esecuzione prima di creare lo schema fisico:

Totally query runtime = 137 msec





Piano di esecuzione dopo aver creato lo schema fisico:

Totally query runtime = 122 msec

La terza query si basa sull’accesso a due relazioni (Volontario e Turno), e come cammino di accesso da utilizzare verranno usate delle ricerche sequenziali, su entrambe le relazioni. La scansione sequenziale è l’unico cammino di accesso disponibile prima di creare lo schema fisico. Il cammino scelto prenderà come riferimento la chiave di ricerca “volontario”, come specificato nell’indice ordinato.

**10) Descrizione della transazione considerata e  
giustificazione per il livello di isolamento  
prescelto. (Visionare file sql3parte.sql)**

Specificare in linguaggio naturale e in PL/pgSQL una transazione con almeno una operazione di scrittura e almeno due operazioni di lettura indicando quale livello di isolamento si ritiene opportuno.

BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Interrogazione di lettura 1: Recupera il saldo punti di una famiglia

SELECT saldoPunti

FROM famiglia

WHERE codfam = 2222;

-- Interrogazione di scrittura: Aggiorna il saldo punti della famiglia

UPDATE famiglia

SET saldoPunti = saldoPunti + 50

WHERE codfam = 2222;

-- Interrogazione di lettura 2: Verifica il saldo punti aggiornato

SELECT saldoPunti

FROM famiglia

WHERE codfam = 2222;

COMMIT;

Per quanto riguarda la transazione indicata qui sopra abbiamo realizzato:

1. Operazione di lettura -> recupero saldo punti di una famiglia
2. Operazione di scrittura -> aggiornamento saldo punti della famiglia
3. Operazione di lettura -> verifica saldo punti aggiornato

Le due interrogazioni di lettura recuperano il saldo punti di una famiglia prima e dopo l'aggiornamento, rispettivamente. L'interrogazione di scrittura aggiorna il saldo punti della famiglia aumentandolo di 50.

Come livello di isolamento verrà utilizzato il livello di isolamento READ COMMITTED, che garantisce che i dati letti durante la transazione siano sempre aggiornati al momento della lettura (così la seconda operazione di lettura stamperà il valore aggiornato e corretto). Con l’utilizzo di questo livello di isolamento possiamo evitare anomalie DIRTY READ (letture sporche per modifiche effettuate da altre transazioni).

**11)Descrizione della politica di controllo dell’accesso scelta, motivando le scelte effettuate. Si suggerisce di riassumere la politica di controllo dell’accesso con una tabella contenente una riga per ogni tabella, una colonna per ogni utente. Ogni cella (i,j) dovrà contenere i privilegi che si intendono assegnare all’utente j sulla tabella i.**

| Relazione | Alice | Bob (Volontario) |
| --- | --- | --- |
| Cliente | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |
| Prodotto | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |
| Appuntamento | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |
| Turno | SELECT, INSERT, UPDATE, DELETE | SELECT |
| Donatore | SELECT, INSERT, UPDATE, DELETE | 0 |
| Famiglia | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |
| Volontario | SELECT, INSERT, UPDATE, DELETE | SELECT |
| Importo | SELECT, INSERT, UPDATE, DELETE | 0 |
| Inventario | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |
| Trasporto | SELECT, INSERT, UPDATE, DELETE | SELECT, INSERT, UPDATE, DELETE |

Supponiamo che Bob sia un volontario presso il social market e si occupa di più attività, come ad esempio gestione appuntamenti, accoglienza clienti, trasporto prodotti. Considerando le tipologie di attività che un volontario potrebbe effettuare, avrà i privilegi di lettura e modifica delle tabelle che riguardano la gestione dei prodotti (ovvero Prodotto, Trasporto, Inventario), la gestione dei clienti (Cliente, Famiglia, Appuntamento). Per quanto riguarda i privilegi di sola lettura avremo le tabelle Turno e Volontario per consentire a Bob di poter aggiungere appuntamenti visualizzando i dati relativi ai turni ed ai volontari. Bob non avrà invece accesso ai dati relativi alle donazioni (quindi Donatore e Importo), siccome verranno gestite soltanto dal gestore del socialmarket (ovvero Alice).

Supponiamo che Alice sia il gestore del social market. Essendo il gestore, potrà accedere a tutte le relazioni/tabelle e quindi avrà tutti i permessi su ognuna di essa. Si può considerare Alice come se fosse un “utente master” e avrà il potere di dare privilegi con “GRANT OPTION” ad altri utenti, se fosse necessario.