Cazzola Michele s323270

# Data Science e Tecnologie per le Basi di Dati
# Homework 3

Le tabelle sono rappresentate dalle seguenti abbreviazioni:

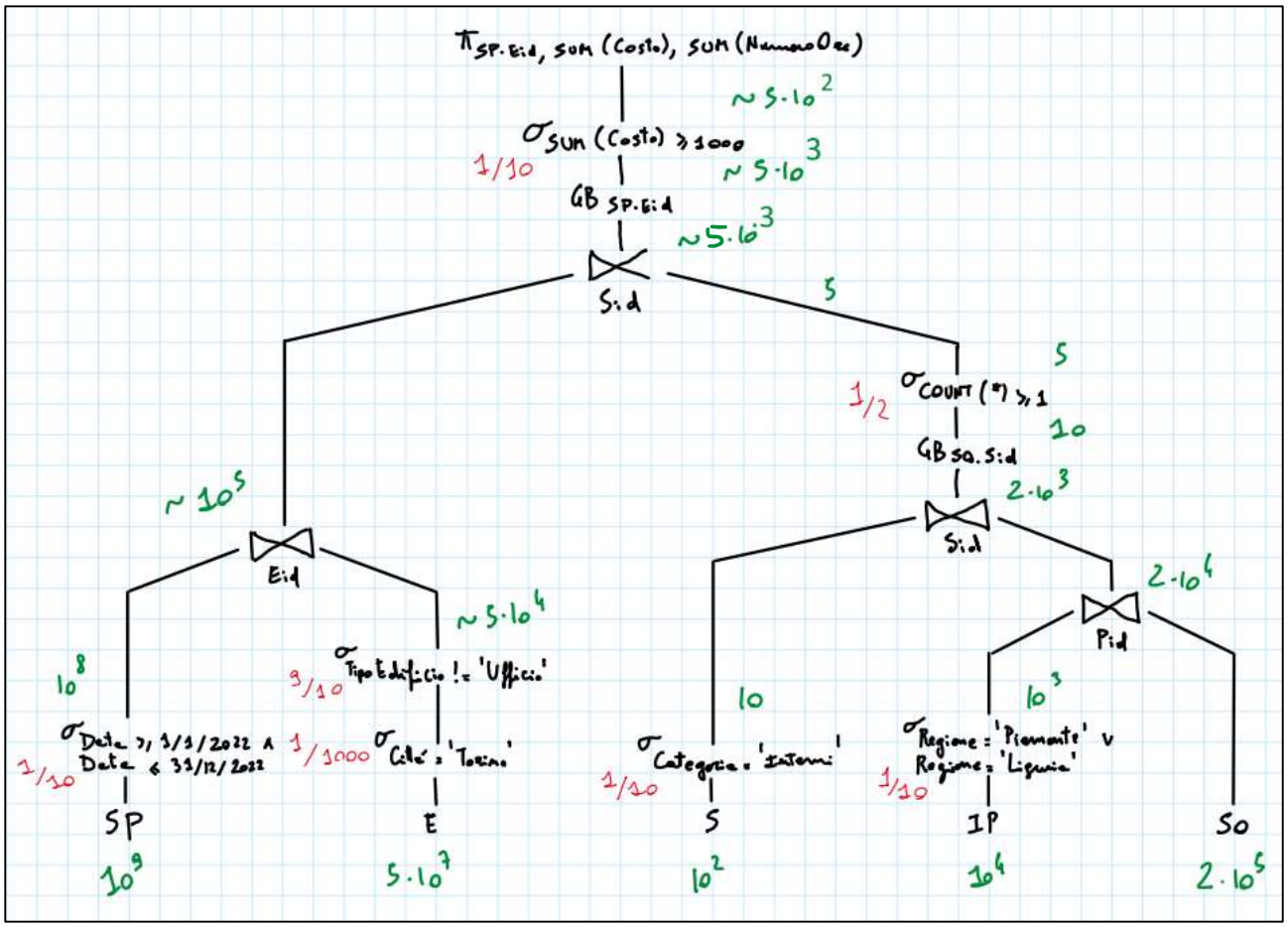
- SP: Servizi-Pulizia;
- E: Edificio;
- S: Servizio;
- IP: Impresa-Pulizie;
- SO: Servizi-Offerti.

## Esercizio 1

Si scriva l'espressione algebrica corrispondente, indicando le operazioni svolte, la cardinalità e la selettività di ogni operazione. Dove necessario, si ipotizzi la distribuzione dei dati. Discutere la possibilità di anticipare l'operatore GROUP BY.

### Albero relazionale

Ipotizzando distribuzione uniforme dei dati nelle tabelle, l'albero relazionale corrispondente alla query è il seguente:



La scelta degli ordini di join è effettuata indipendentemente per le due query coinvolte nell'interrogazione; le motivazioni sono le seguenti:

- query interna:
  - consente un anticipo della clausola GROUP BY (SO.Sid) sul ramo destro della join
  - poiché la prima operazione di join può essere effettuata con *hash join*, mentre la seconda viene sicuramente effettuata con *nested loop join*, l'ordine scelto permette di ridurre la cardinalità della *outer table* di un fattore 400, rispetto all'altra configurazione possibile, in quanto si può beneficiare dell'anticipo della clausola GROUP BY; inoltre, la *hash join* effettuata come prima operazione, ha un aumento di cardinalità della tabella di sinistra (in questo caso IP, nell'altro possibile S) di un fattore 100: globalmente, secondo le stime effettuate, la configurazione scelta risulta più efficiente.
- query esterna: in entrambi i casi possibili, il semi join viene effettuato con *nested loop join*, mentre il full join viene effettuato con *hash join*; in questa configurazione, il semi join ha la outer table ridotta di un fattore 1000 rispetto all'altra possibile configurazione, mentre il full join, pur beneficiando di un eventuale anticipo della clausola GROUP BY (SP.Eid), sarebbe effettuato su tabelle aventi cardinalità $5 \cdot 10^4$ (comune ad entrambi i casi) e $5 \cdot 10^5$, ovvero una dimensione solo 200 volte inferiore a quella della configurazione corrente (tabella SP): alla luce di tali stime, la configurazione scelta risulta essere quella più efficiente.

## Anticipo GROUP BY

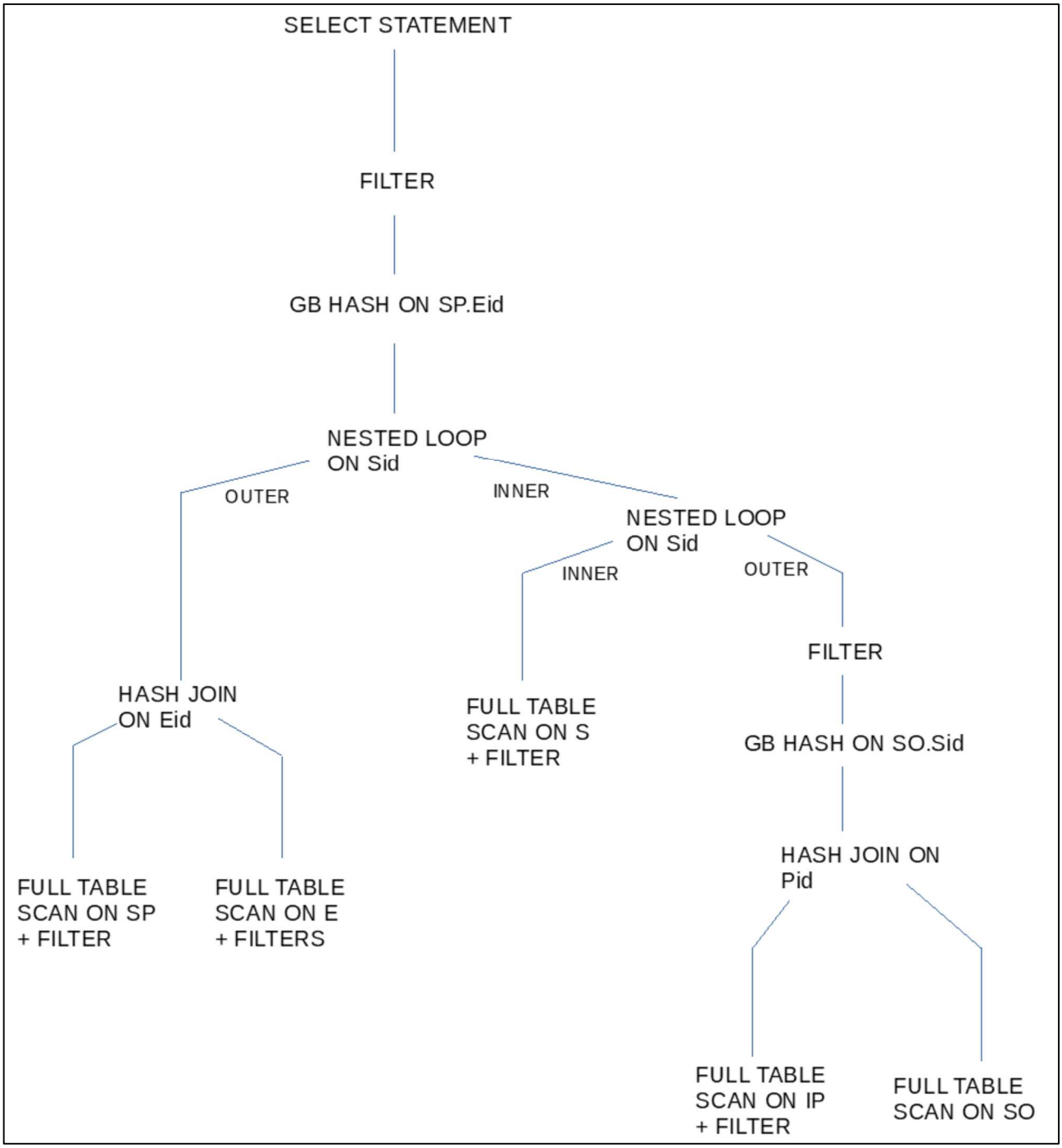
L'anticipo delle clausole di GROUP BY, insieme alle rispettive clausole HAVING che le seguono, risulta possibile (ed efficiente) solo nei casi in cui sia immediatamente preceduto da un'operazione di join effettuata sullo stesso attributo.

Di conseguenza:

- GROUP BY (SO.Sid): si può anticipare sul ramo destro dell'operazione di join che la precede; è conveniente in quanto riduce la cardinalità della *inner table* di un fattore 400, in quanto:
  - GROUP BY (SO.Sid) riduce la cardinalità a 100, essendo 100 il numero di servizi diversi presenti;
  - HAVING COUNT (*) >= 1 riduce la cardinalità di un fattore 2, dalle statistiche inizialmente fornite;
- GROUP BY (SP.Eid):
  - non si può anticipare, in quanto è preceduta da una join sull'attributo *Sid*;
  - un'eventuale realizzazione, sul ramo sinistro, di una GROUP BY (SP.Eid, SP.Sid) non sarebbe conveniente: la cardinalità generata sarebbe pari al numero totale di servizi moltiplicata per il numero (filtrato) di edifici, ovvero $5 \cdot 10^6$; esso è però un numero superiore a quello già presente a valle dell'operazione di join su *Eid* (pari a $10^5$), quindi renderebbe più lenta l'esecuzione della query, essendo a tutti gli effetti un'operazione aggiuntiva rispetto al caso iniziale.

## Piano esecuzione base

Il piano di esecuzione, senza l'utilizzo di strutture fisiche di supporto, ma già considerando gli anticipi della clausola GROUP BY, ove possibile, è il seguente:




Caratteristiche:

- metodi di accesso: tutte le tabelle sono lette mediante *full table scan* che, dove necessario (ovvero in presenza di uno o più predicati di selezione), effettuano direttamente l'operazione di FILTER durante la scansione della tabella;

- metodi di join:
  - le operazioni $SP \bowtie E$ e $IP \bowtie SO$ sono effettuate con *hash join*, avendo entrambe le tabelle con una cardinalità superiore (o uguale, nel caso di IP) a $10^3$;
  - le altre due operazioni di join sono effettuate mediante *nested loop join*, avendo almeno una delle due tabelle coinvolte una cardinalità inferiore a $10^3$: come *inner table* si sceglie sempre la tabella avente cardinalità inferiore;
- group by: in entrambi i casi, le operazioni di GROUP BY sono seguite da un'operazione di FILTER, dovuta alla presenza della clausola HAVING, e sono svolte mediante *group by hash*, in quanto:
  - la cardinalità delle tabelle su cui vengono effettuate è superiore a $10^3$;
  - non sono precedute immediatamente da un'operazione di *hash join*, effettuata sullo stesso attributo di raggruppamento.

# Esercizio 2

Si scelgano le strutture fisiche accessorie per migliorare le prestazioni dell'interrogazione. Si motivi la scelta e si definisca il piano di esecuzione (ordine e tipo dei join, accesso alle tabelle e/o indici, etc.).

## Indici

Date le tabelle presenti, si valuta l'inserimento di indici sui loro attributi:

- SP:
  - Data: si introduce un indice (di tipo B-tree essendo un predicato di range), poiché la tabella è grande e il predicato ha selettività 1/10;
- E:
  - Città: si introduce un indice (di tipo hash essendo il predicato di uguaglianza su una stringa), poiché la tabella è grande e il predicato ha selettività 1/1000;
  - TipoEdificio: non si introducono indici, avendo il predicato selettività 9/10;
- S: non si inseriscono indici, essendo la tabella piccola
- IP:
  - Regione: si introduce un indice (di tipo hash essendo il predicato di uguaglianza su una stringa), poiché il predicato ha selettività 1/10;
- SO:
  - no

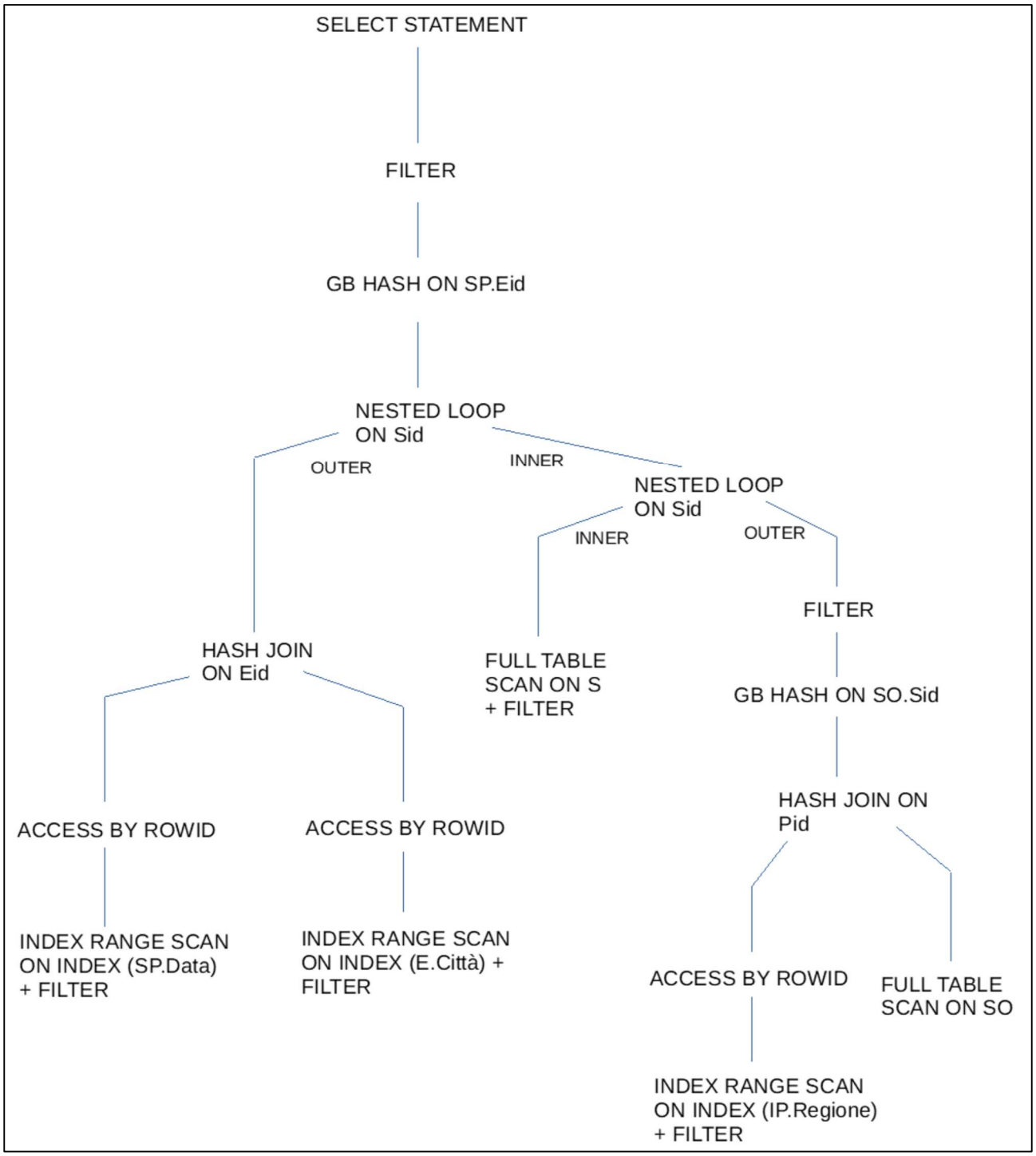
In ogni caso, non si considerano indici sulle chiavi primarie in quanto Oracle inserisce di default indici di sistema su di esse.

## Piano esecuzione ottimizzato

Il piano di esecuzione ottimizzato ha le operazioni seguenti:

- accesso alle tabelle:
  - *index range scan* e *access by rowid* per le tabelle SP, E, IP, in quanto hanno indici definiti rispettivamente sui loro attributi *Data*, *Città*, *Regione*;
  - *full table scan* e *access by rowid* per le tabelle S, SO, in quanto non ci sono indici definiti su di esse;
- metodi di join: rimangono invariati rispetto al caso non ottimizzato;
- metodi di group by: rimangono invariati rispetto al caso non ottimizzato.

Il piano di esecuzione ottimizzato è quindi il seguente:


SELECT STATEMENT
FILTER
GB HASH ON SP.Eid
NESTED LOOP
ON Sid
OUTER
INNER
NESTED LOOP
ON Sid
INNER
OUTER
FILTER
HASH JOIN
ON Eid
FULL TABLE
SCAN ON S
+ FILTER
GB HASH ON SO.Sid
HASH JOIN ON
Pid
ACCESS BY ROWID
ACCESS BY ROWID
INDEX RANGE SCAN
ON INDEX (SP.Data)
+ FILTER
INDEX RANGE SCAN
ON INDEX (E.Città) +
FILTER
ACCESS BY ROWID
FULL TABLE
SCAN ON SO
INDEX RANGE SCAN
ON INDEX (IP.Regione)
+ FILTER