Orti Scolastici & Citizen Science

Progetto fisico e validazione

a) Descrizione in linguaggio naturale e il codice SQL sviluppato per implementare le interrogazioni del carico di lavoro

Abbiamo deciso di implementare le seguenti query:

QUERY 1

Per ogni scuola di Milano, restituire le specie studiate.

Osservazione: non abbiamo fatto natural join in quanto veniva fatta una join sbagliata, non sull'attributo codiceMecc.

```
SELECT Scuola.codiceMecc, nome, StudiaSpecie.nomeS
FROM Scuola
JOIN StudiaSpecie ON Scuola.codiceMecc = StudiaSpecie.codiceMecc
WHERE Scuola.provincia = 'MI';
```

QUERY 2

Per ogni scuola, restituire gli orti la cui tipo orto è 'campo' e la condizione è 'fitobonifica'.

Osservazione: non abbiamo fatto natural join in quanto veniva fatta una join sbagliata, non sull'attributo codiceMecc.

```
SELECT Scuola.codiceMecc, Scuola.nome, Orto.nome, Orto.gps, Orto.tipo,
Orto.condizione
FROM Scuola
JOIN Orto ON Orto.codiceMecc=Scuola.codiceMecc
WHERE Orto.tipo = 'campo' AND Orto.condizione = 'fitobonifica';
```

QUERY 3

Per ogni scuola, restituire i gruppi osservati e le relative specie.

Osservazione: non abbiamo fatto natural join in quanto venivano fatte join sbagliate, non sugli attributi codiceMecc e gps.

```
SELECT Scuola.codiceMecc, Scuola.nome, Gruppo.idGruppo, Gruppo.nomeS
FROM Scuola
JOIN Orto ON Scuola.codiceMecc = Orto.codiceMecc
JOIN Gruppo ON Orto.gps = Gruppo.gps;
```

b) 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.

Abbiamo inserito i seguenti indici:

QUERY 1:

```
create index codiceMecc on Scuola (codiceMecc);
create index codiceMecc_S on StudiaSpecie (codiceMecc);
create index provincia_scuola on Scuola using hash (provincia);
cluster Scuola using codiceMecc;
```

Per la query 1 abbiamo deciso di utilizzare gli indici ordinati sull'attributo **codiceMecc** sia di Scuola che di StudiaSpecie per favorire l'operazione di join.

Abbiamo inoltre usato un indice di hash per l'attributo **provincia** di Scuola, in quanto quest'attributo non assume valori unici ma possono quindi ripetersi e pertanto abbiamo pensato fosse più sensato raggrupparli con chiave hash.

A seguito del popolamento abbiamo visto che la tabella Scuola era quella contenente più dati, e abbiamo quindi deciso di utilizzare un indice clusterizzato per ottimizzare i tempi di risposta delle query che riferiscono la tabella Scuola (in quanto essendo la tabella più utilizzata e con il maggior numero di tuple, abbiamo la necessità di recuperare più velocemente i dati).

QUERY 2:

```
create index orto_tipo on Orto using hash (tipo);
create index orto_condizione on Orto using hash (condizione);
```

Per la query 2 l'indice per la relazione Scuola era già stato creato precedentemente, quindi abbiamo solo aggiunto due indici di tipo hash per gli attributi **tipo** e **condizione** della relazione Orto, per motivazioni analoghe al caso precedente poiché i valori assunti possono ripetersi e quindi avendo una query con condizione di uguaglianza volevamo recuperare i dati più velocemente.

QUERY 3:

```
create index gps_gruppo on Gruppo (gps);
create index gps_orto on Orto (gps);
```

Per la query 3 l'indice su **codiceMecc** di Scuola e Orto era già stato creato, quindi abbiamo solo aggiunto due indici ordinati su **gps** delle relazioni Gruppo e Orto per poter eseguire l'operazione di join più velocemente.

c) 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.

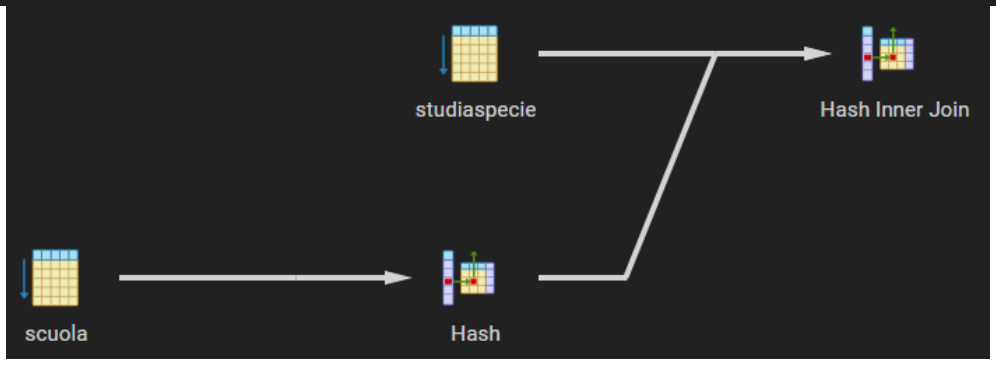
	N° TUPLE INSERITE	DIMENSIONE IN BLOCCHI
SCUOLA	**1020**	**15**
ORTO	**216**	**10**
GRUPPO	**186**	**8**
SPECIE	**198**	**9**
STUDIASPECIE	**211**	**9**

d) 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:

Prima della creazione degli indici:

	QUERY PLAN text
1	Hash Join (cost=27.79..39.45 rows=1 width=38) (actual time=0.336..0.390 rows=7 loops=1)
2	Hash Cond: (studiaspecie.codicemecc = scuola.codicemecc)
3	-> Seq Scan on studiaspecie (cost=0.00..11.11 rows=211 width=29) (actual time=0.021..0.040 rows=211 loops...
4	-> Hash (cost=27.75..27.75 rows=3 width=20) (actual time=0.302..0.303 rows=3 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Seq Scan on scuola (cost=0.00..27.75 rows=3 width=20) (actual time=0.291..0.296 rows=3 loops=1)
7	Filter: (provincia = 'MI'::bpchar)
8	Rows Removed by Filter: 1017
9	Planning Time: 3.793 ms
10	Execution Time: 0.649 ms



Prima della creazione degli indici, nella query 1, il sistema ha scelto come piano di esecuzione fisica una scansione sequenziale sulla relazione StudiaSpecie e una scansione sequenziale su Scuola.

Dopo la creazione degli indici:

	QUERY PLAN text
1	Hash Join (cost=12.07..23.74 rows=1 width=38) (actual time=0.064..0.135 rows=7 loops=1)
2	Hash Cond: (studiaspecie.codicemecc = scuola.codicemecc)
3	-> Seq Scan on studiaspecie (cost=0.00..11.11 rows=211 width=29) (actual time=0.017..0.043 rows=211 loops=1)
4	-> Hash (cost=12.04..12.04 rows=3 width=20) (actual time=0.032..0.032 rows=3 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 9kB
6	-> Bitmap Heap Scan on scuola (cost=4.02..12.04 rows=3 width=20) (actual time=0.026..0.029 rows=3 loops=1)
7	Recheck Cond: (provincia = 'MI'::bpchar)
8	Heap Blocks: exact=2
9	-> Bitmap Index Scan on provincia_scuola (cost=0.00..4.02 rows=3 width=0) (actual time=0.018..0.018 rows=3 loop...
10	Index Cond: (provincia = 'MI'::bpchar)
11	Planning Time: 9.688 ms
12	Execution Time: 0.193 ms

Anche dopo la creazione degli indici il sistema sceglie una scansione sequenziale su StudiaSpecie, mentre sceglie un Heap Scan su Scuola e un Index Scan sull'attributo **provincia** di Scuola, il primo dovuto alla creazione dell'indice ordinato, il secondo dovuto alla creazione dell'indice hash sull'attributo.

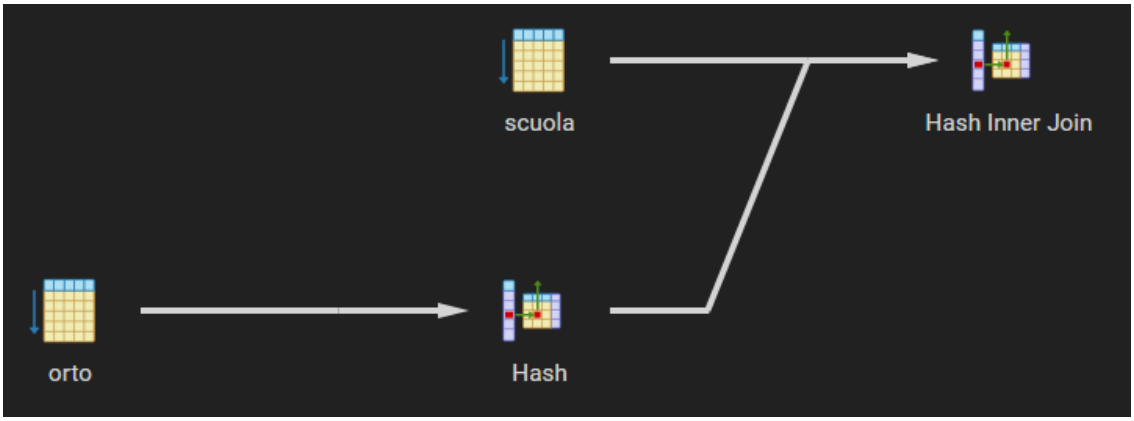
Per quanto riguarda i tempi di esecuzione, notiamo un miglioramento nelle prestazioni, difatti prima della creazione degli indici il tempo di esecuzione era di 0,649ms mentre dopo la creazione si è ridotto a 0,193ms.
I tempi di esecuzione diminuiscono grazie alla presenza degli indici, e in particolare quello clusterizzato, che riduce notevolmente il tempo di risposta della query.

Notiamo inoltre una diminuzione del costo della funzione di Hash Join dovuta alla creazione degli indici ordinati su **codiceMecc**.

QUERY 2:

Prima della creazione degli indici:

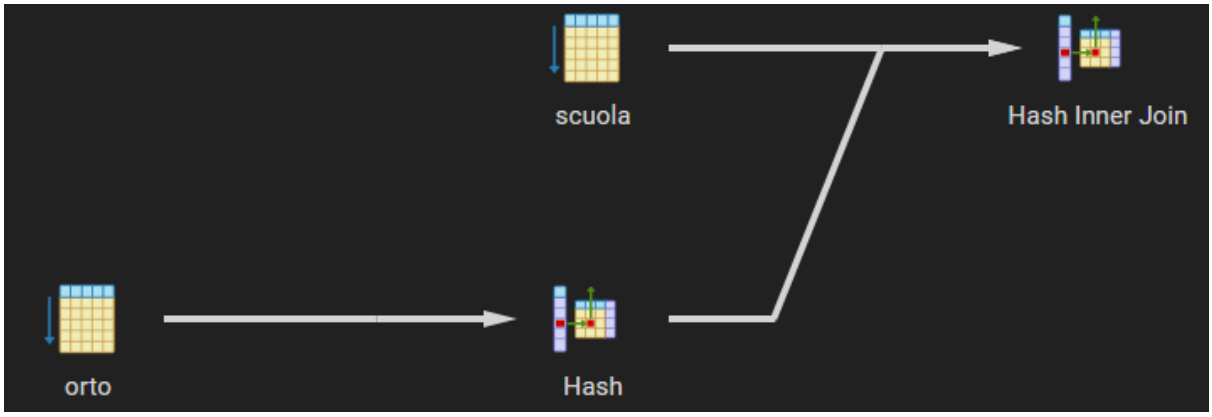
	QUERY PLAN text
1	Hash Join (cost=12.79..58.80 rows=44 width=72) (actual time=0.158..0.458 rows=50 loops=1)
2	Hash Cond: (scuola.codicemecc = orto.codicemecc)
3	-> Seq Scan on scuola (cost=0.00..39.20 rows=1020 width=20) (actual time=0.029..0.139 rows=1020 l...
4	-> Hash (cost=12.24..12.24 rows=44 width=63) (actual time=0.111..0.111 rows=50 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 13kB
6	-> Seq Scan on orto (cost=0.00..12.24 rows=44 width=63) (actual time=0.018..0.084 rows=50 loops...
7	Filter: (((tipo)::text = 'campo'::text) AND ((condizione)::text = 'fitobonifica'::text))
8	Rows Removed by Filter: 166
9	Planning Time: 0.647 ms
10	Execution Time: 0.495 ms



Anche per la query 2 il sistema sceglie una scansione sequenziale su Scuola, e una scansione sequenziale su Orto.

Dopo della creazione degli indici:

	QUERY PLAN text
1	Hash Join (cost=12.79..44.80 rows=44 width=72) (actual time=0.107..0.356 rows=50 loops=1)
2	Hash Cond: (scuola.codicemecc = orto.codicemecc)
3	-> Seq Scan on scuola (cost=0.00..25.20 rows=1020 width=20) (actual time=0.013..0.115 rows=1020 l...
4	-> Hash (cost=12.24..12.24 rows=44 width=63) (actual time=0.085..0.085 rows=50 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 13kB
6	-> Seq Scan on orto (cost=0.00..12.24 rows=44 width=63) (actual time=0.011..0.066 rows=50 loops...
7	Filter: (((tipo)::text = 'campo'::text) AND ((condizione)::text = 'fitobonifica'::text))
8	Rows Removed by Filter: 166
9	Planning Time: 0.422 ms
10	Execution Time: 0.381 ms



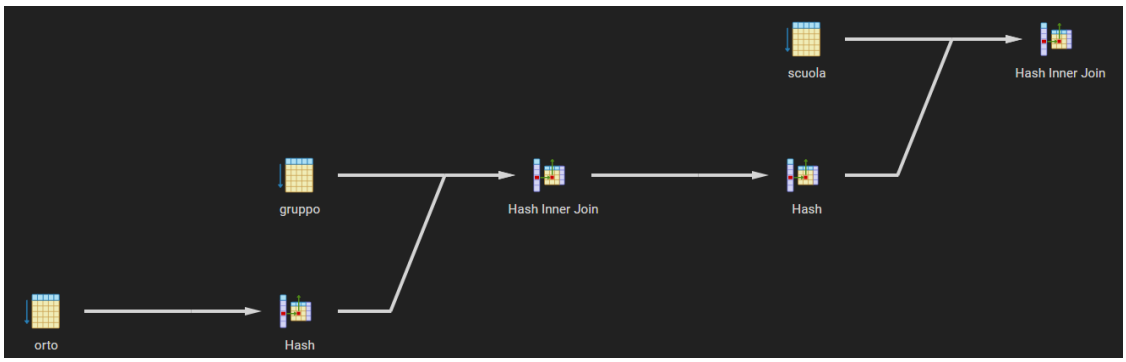
Nonostante la creazione degli indici hash il sistema sceglie comunque di utilizzare delle scansioni sequenziali.

Notiamo solo un miglioramento nel tempo di esecuzione, che passa da 0,495ms a 0,381ms.

QUERY 3:

Prima della creazione degli indici:

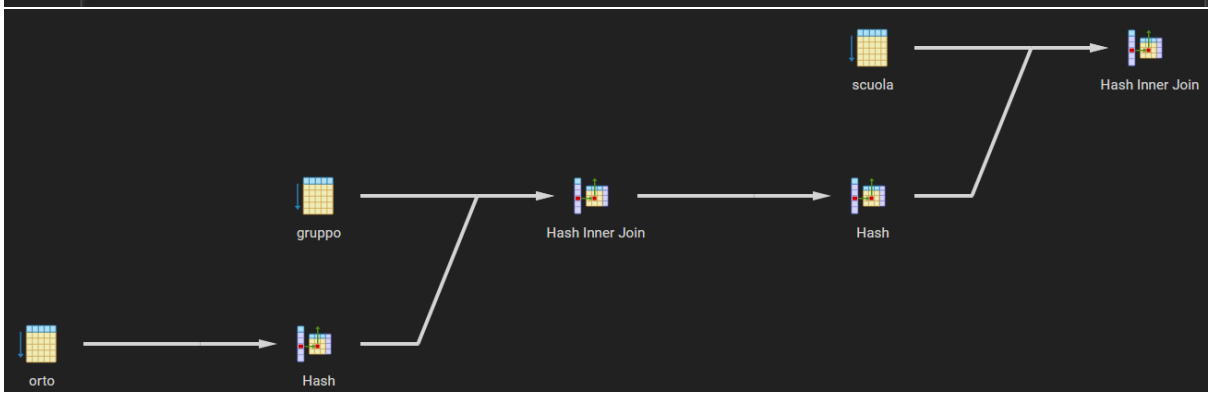
	QUERY PLAN text
1	Hash Join (cost=26.54..73.98 rows=186 width=42) (actual time=0.365..0.646 rows=186 loops=1)
2	Hash Cond: (scuola.codicemecc = orto.codicemecc)
3	-> Seq Scan on scuola (cost=0.00..39.20 rows=1020 width=20) (actual time=0.017..0.108 rows=1020 loops=1)
4	-> Hash (cost=24.22..24.22 rows=186 width=33) (actual time=0.328..0.329 rows=186 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 21kB
6	-> Hash Join (cost=13.86..24.22 rows=186 width=33) (actual time=0.157..0.264 rows=186 loops=1)
7	Hash Cond: ((gruppo.gps)::text = (orto.gps)::text)
8	-> Seq Scan on gruppo (cost=0.00..9.86 rows=186 width=35) (actual time=0.008..0.028 rows=186 loops=...
9	-> Hash (cost=11.16..11.16 rows=216 width=24) (actual time=0.141..0.142 rows=216 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 20kB
11	-> Seq Scan on orto (cost=0.00..11.16 rows=216 width=24) (actual time=0.011..0.066 rows=216 loop...
12	Planning Time: 0.724 ms
13	Execution Time: 0.686 ms



Nella query 3 il sistema ha utilizzato una scansione sequenziale su Scuola, una scansione sequenziale su Gruppo e una scansione sequenziale su Orto.

Dopo della creazione degli indici:

	QUERY PLAN text
1	Hash Join (cost=26.54..59.98 rows=186 width=42) (actual time=0.231..0.440 rows=186 loops=1)
2	Hash Cond: (scuola.codicemecc = orto.codicemecc)
3	-> Seq Scan on scuola (cost=0.00..25.20 rows=1020 width=20) (actual time=0.017..0.091 rows=1020 loops=1)
4	-> Hash (cost=24.22..24.22 rows=186 width=33) (actual time=0.206..0.207 rows=186 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 21kB
6	-> Hash Join (cost=13.86..24.22 rows=186 width=33) (actual time=0.095..0.171 rows=186 loops=1)
7	Hash Cond: ((gruppo.gps)::text = (orto.gps)::text)
8	-> Seq Scan on gruppo (cost=0.00..9.86 rows=186 width=35) (actual time=0.008..0.021 rows=186 loops=...
9	-> Hash (cost=11.16..11.16 rows=216 width=24) (actual time=0.082..0.083 rows=216 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 20kB
11	-> Seq Scan on orto (cost=0.00..11.16 rows=216 width=24) (actual time=0.009..0.046 rows=216 loop...
12	Planning Time: 0.674 ms
13	Execution Time: 0.470 ms



Anche in questo caso il sistema ha deciso di adottare gli stessi piani di esecuzione.

Notiamo comunque anche in questo caso una diminuzione del tempo di esecuzione, che passa da 0,686ms a 0,470ms.

Politica di controllo dell'accesso

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 e una colonna per ogni ruolo. Ogni cella (i,j) dovrà contenere i privilegi che si intendono assegnare all'utente j sulla tabella i.

Innanzitutto noi non abbiamo creato il ruolo **referente di istituto**, in quanto abbiamo inteso Scuola e Istituto come sinonimi nella creazione del nostro schema, e quindi i due ruoli sarebbero stati la medesima cosa.

	Gestore globale del progetto	Referente scuola	Insegnante	Studente
Partecipante	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT**
Finanziamento	**ALL PRIVILEGES**	**SELECT**	**SELECT**	**-**
Scuola	**ALL PRIVILEGES**	**SELECT**	**SELECT**	**SELECT**
Classe	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT**
Specie	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
Gruppo	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
Controllo	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
Orto	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT**	**SELECT**
Dispositivo	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
Pianta	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
StudiaSpecie	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT**	**SELECT**

Osservazione	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
RespIns	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
RespOss	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**
DatiOss	**ALL PRIVILEGES**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**	**SELECT, INSERT, UPDATE, DELETE**

Abbiamo deciso di dare tutti i privilegi al **gestore globale del progetto,** in quanto è il soggetto più in alto nella gerarchia e può avere l'accesso a tutto il sistema.

Al **referente della scuola** abbiamo dato la possibilità di leggere, inserire, modificare ed eliminare i dati di quasi tutte le relazioni, ad eccezione di Finanziamento e Scuola, perché essendo il referente di una singola scuola, non gli diamo il permesso di inserire, modificare ed eliminare i dati di altre scuole e/o finanziamenti.

All'**insegnante** abbiamo dato i permessi di SELECT, INSERT, UPDATE e DELETE, su tutte le tabelle tranne Finanziamento, Scuola, Orto e StudiaSpecie; per ovvi motivi (si veda anche ruolo precedente) garantiamo solo la SELECT in Finanziamento e Scuola e abbiamo deciso di consentire solo la SELECT anche su Orto e StudiaSpecie poiché non ci sembrava opportuno permettere di inserire, modificare e eliminare tali dati, in non è il responsabile del progetto e quindi non decide quali orti e specie può gestire la scuola.

Infine allo **studente** abbiamo il minor numero di permessi possibili, nessuno su Finanziamento, in quanto uno studente non può accedere a questi dati "sensibili", fornendo la SELECT su tutte le altre tabelle, con la possibilità di inserire, modificare ed eliminare i dati di tabelle che contengono informazioni strettamente legate al progetto, e quindi accessibili dallo studente.