Transazione considerata

```
-- dato un volontario (id = 1) otteniamo il numero di turni a cui e' assegnato
-- dopodiche', inseriamo un nuovo turno e lo assegnamo al dato volontario,
-- infine ricalcoliamo il dato
BEGIN
      SERIALIZABLE;

      SELECT v.id AS id_volontario, COUNT(*) AS n_turni
      FROM volontari v
      JOIN volontari_turni vt ON vt.volontario = v.id
      GROUP BY v.id;

      -- Assegnamo un nuovo turno al volontario
      INSERT INTO volontari_turni(volontario, turno) VALUES
                  (1, 1000);

      SELECT v.id AS id_volontario, COUNT(*) AS n_turni
      FROM volontari v
      JOIN volontari_turni vt ON vt.volontario = v.id
      GROUP BY v.id;
END;
```

Considerato il funzionamento di questa transazione, le due anomalie possibili sono:

- **phantom row**

Nel momento in cui vengono contati i turni la prima volta e inserito il turno, e' possibile che una transazione concorrente stia a sua volta assegnando un turno al volontario 1, pero' la nostra transazione rilegge il numero dei turni e fa commit: in questo caso ci siamo persi il turno inserito dalla transazione concorrente.

- **dirty read**

Fatte le letture e l'inserimento del turno, magari una transazione concorrette vuole a sua volta visualizzare i turni assegnati al volontario 1, pero', per esempio, la nostra transazione esegue un ROLLBACK a causa di una system failure, lasciando un valore inconsistente alla transazione concorrente.

L'unico livello di isolamento che si occupa delle phandom row e' il livello **SERIALIZABLE**, percio' lo abbiamo scelto.