CORSO DI BASI DI DATI

PROGETTAZIONE DATABASE PER LA SOCIETÀ DI TAXI

Rossi Luca – Matricola 598217 Data di Consegna: 2023/07/04

Descrizione del Dominio

Un autista, durante il proprio turno di lavoro, guida un determinato taxi con quale effettua delle corse.

Ogni autista deve fornire i dati riguardanti la propria patente.

Solo un autista può essere un referente presso la società.

Un cliente richiede il servizio taxi di una corsa tramite prenotazione.

Un cliente può accordarsi con la società per attivare una delle convenzioni proposte dalla società.

Un proprietario immette le revisioni che ogni taxi deve fare.

Un addetto alla manutenzione effettua le revisioni e ne inserisce i dati.

Un taxi, durante turni diversi, può essere guidati da autisti diversi, oppure non essere "in servizio".

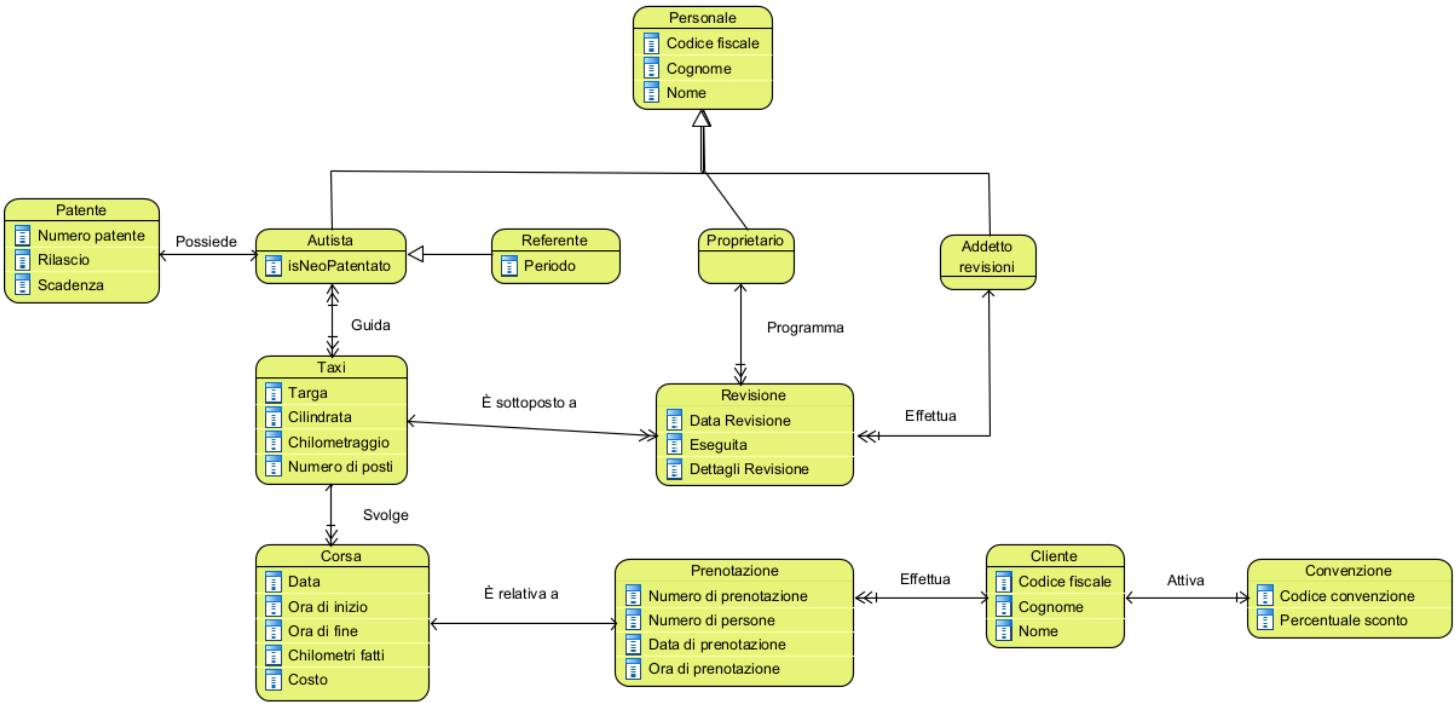
Di ogni taxi si vogliono registrare le informazioni relative al veicolo (identificazione, cilindrata, numero massimo di passeggeri trasportabili).

Una corsa rappresenta quando un cliente effettua un viaggio tramite taxi.

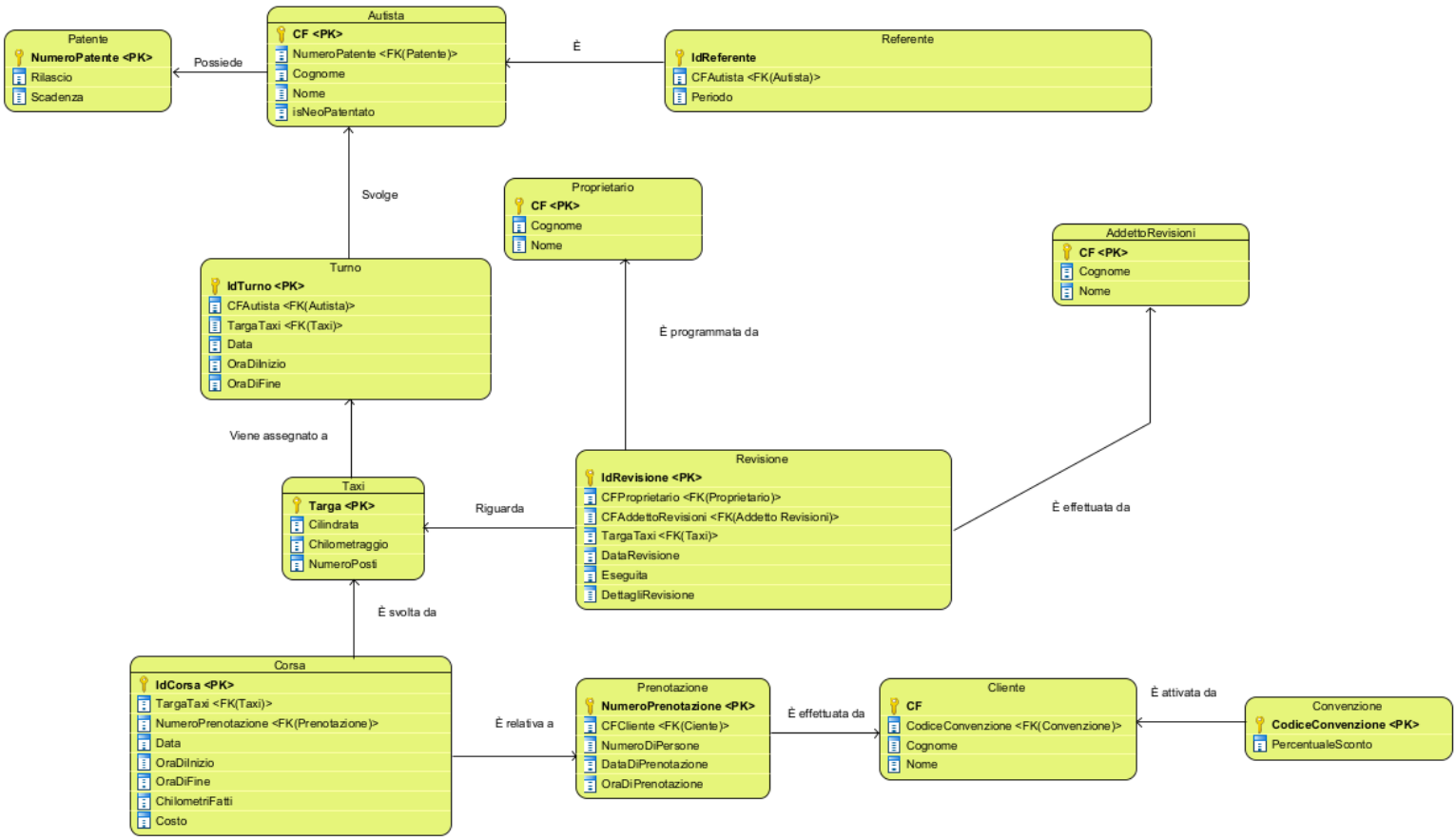
Di seguito le informazioni sulle gerachie:

- Autista, referente, proprietario ed addetto alla manutenzione sono tutti membri del personale.
- Un referente è un autista che fa da referente degli altri autisti presso la società.

Schema concettuale



Schema logico relazionale



Vincoli**:**

Autista: (NumeroPatente IS UNIQUE NOT NULL AND IsNeoPatentato IS NOT NULL AND Nome IS NOT NULL AND Cognome IS NOT NULL)

Patente: ((Rilascio, Scadenza) IS NOT NULL)

Referente: ((CFAutista, Periodo) IS UNIQUE NOT NULL)

Proprietario: (Nome IS NOT NULL AND Cognome IS NOT NULL)

AddettoRevisioni: (Nome IS NOT NULL AND Cognome IS NOT NULL)

Turno: ((TargaTaxi, Data, OraDiInizio, OraDiFine) IS UNIQUE NOT NULL AND (CFAutista, Data, OraDiInizio, OraDiFine) IS UNIQUE NOT NULL) AND OraDiInizio < OraDiFine)

Taxi: ((Cilindrata, Chilometraggio, NumeroPosti) IS NOT NULL)

Corsa: (NumeroPrenotazione IS UNIQUE NOT NULL AND (TargaTaxi, Data, OraDiInizio, OraDiFine) IS UNIQUE NOT NULL AND OraDiInizio < OraDiFine AND (ChilometriFatti, Costo) IS NOT NULL)

Prenotazione: ((CFCliente, DataDiPrenotazione, OraDiPrenotazione) IS UNIQUE NOT NULL AND NumeroDiPersone IS NOT NULL)

Cliente: ((Nome, Cognome) IS NOT NULL)

Convenzione: (PercentualeSconto IS UNIQUE NOT NULL)

Revisione((TargaTaxi, DataRevisione) IS UNIQUE NOT NULL AND Eseguita IS NOT NULL AND CHECK (Eseguita = FALSE OR DettagliRevisione IS NOT NULL))

Dipendenze funzionali:

Autista:

{CF→Nome, Cognome

NumeroPatente→IsNeoPatentato}

Referente:

{IdReferente→CFAutista, Periodo}

Proprietario:

{CF→Nome, Cognome}

AddettoRevisioni:

{CF→Nome, Cognome}

Patente:

{NumeroPatente→Rilascio, Scadenza}

Taxi:

{Targa→Cilindrata, Chilometraggio, NumeroPosti}

Turno:

{IdTurno→CFAutista, TargaTaxi, Data, OraDiInizio, OraDiFine}

Corsa:

{IdCorsa→TargaTaxi, NumeroDiPrenotazione, Data, OraDiInizio, OraDiFine, ChilometriFatti, Costo}

Prenotazione:

{NumeroPrenotazione→CFCliente, NumeroDiPersone, DataDiPrenotazione, OraDiPrenotazione}

Revisione:

{IdRevisione→CFProprietario, CFAddettoRevisioni, TargaTaxi, DataRevisione, Eseguita, DettagliRevisione}

Cliente:

{CF→CodiceConvenzione, Cognome, Nome}

Convenzione:

{CodiceConvenzione→PercentualeSconto}

Ogni dipendenza rispetta la forma normale di Boyce Codd ad esclusione della dipendenza “NumeroPatente → isNeoPatentato”, rendendo quindi lo schema normalizzato secondo la terza forma normale.

Interrogazioni SQL

A. Elenco dei turni degli autisti neo patentati:

```
SELECT Turno.*
FROM Turno
INNER JOIN Autista
ON (Turno.CFAutista = Autista.CF)
WHERE Autista.isNeoPatentato = TRUE
```

B. Numero di turni maggiori di 1 per taxi fatti da ogni autista neopatentato ordinati per CF dell’autista:

```
SELECT t.TargaTaxi, t.CFAutista, COUNT(DISTINCT t.TargaTaxi, t.CFAutista) AS NumeroTurni
FROM Turno t, Autista a
```

WHERE a.IsNeoPatentato = TRUE AND t.CFAutista = a.CF

GROUP BY t.TargaTaxi, t.CFAutista

HAVING COUNT(DISTINCT t.TargaTaxi, t.CFAutista) > 1

ORDER BY t.CFAutista ASC

C. Numero di prenotazioni per cliente che ha eseguito più di una prenotazione relative a più di una persona:

SELECT Cliente.CF, COUNT(Prenotazione.CFCliente) AS NumeroPrenotazioniCliente

FROM Prenotazione

INNER JOIN Cliente

ON (Prenotazione.CFCliente = Cliente.CF)

WHERE Prenotazione.NumeroDiPersone > 1

GROUP BY Prenotazione.CFCliente

HAVING COUNT(Prenotazione.CFCliente) > 1

D. Informazioni di ogni cliente che ha effettuato almeno una prenotazione:

SELECT c.CF, c.Nome, c.Cognome

FROM Cliente c

WHERE EXISTS (SELECT p.NumeroPrenotazione FROM Prenotazione p WHERE p.CFCliente = c.CF)

E. Informazioni di ogni cliente che non ha ancora effetuato una prenotazione:

SELECT *

FROM Cliente

WHERE CF <> ALL (SELECT p.CFCliente FROM Prenotazione p)

F. Informazioni sulla prenotazione che corrispondono a corse che hanno fatto più di 50 km:

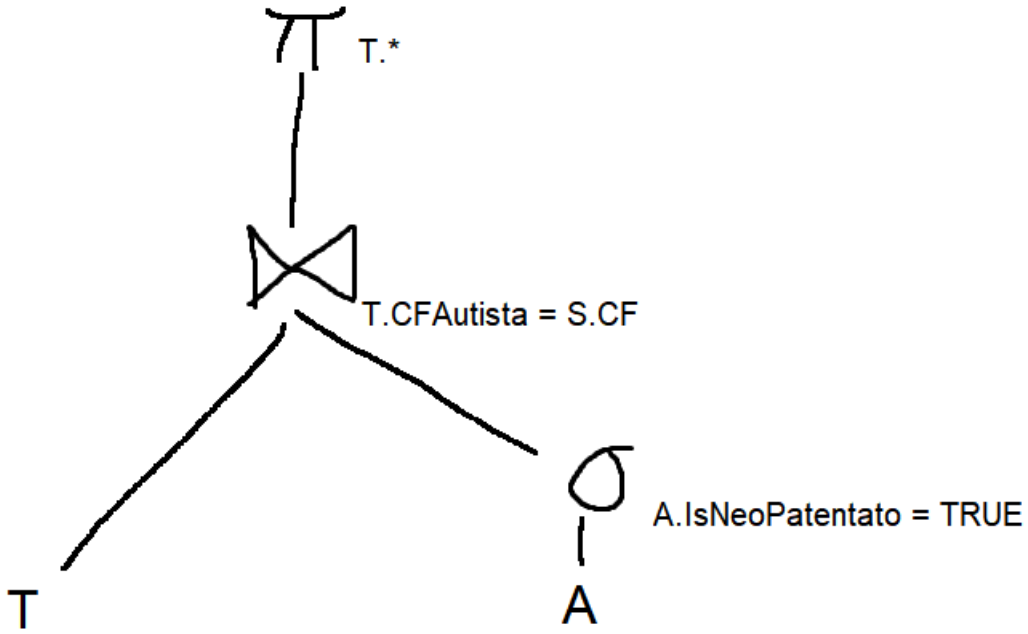
SELECT *

FROM Prenotazione

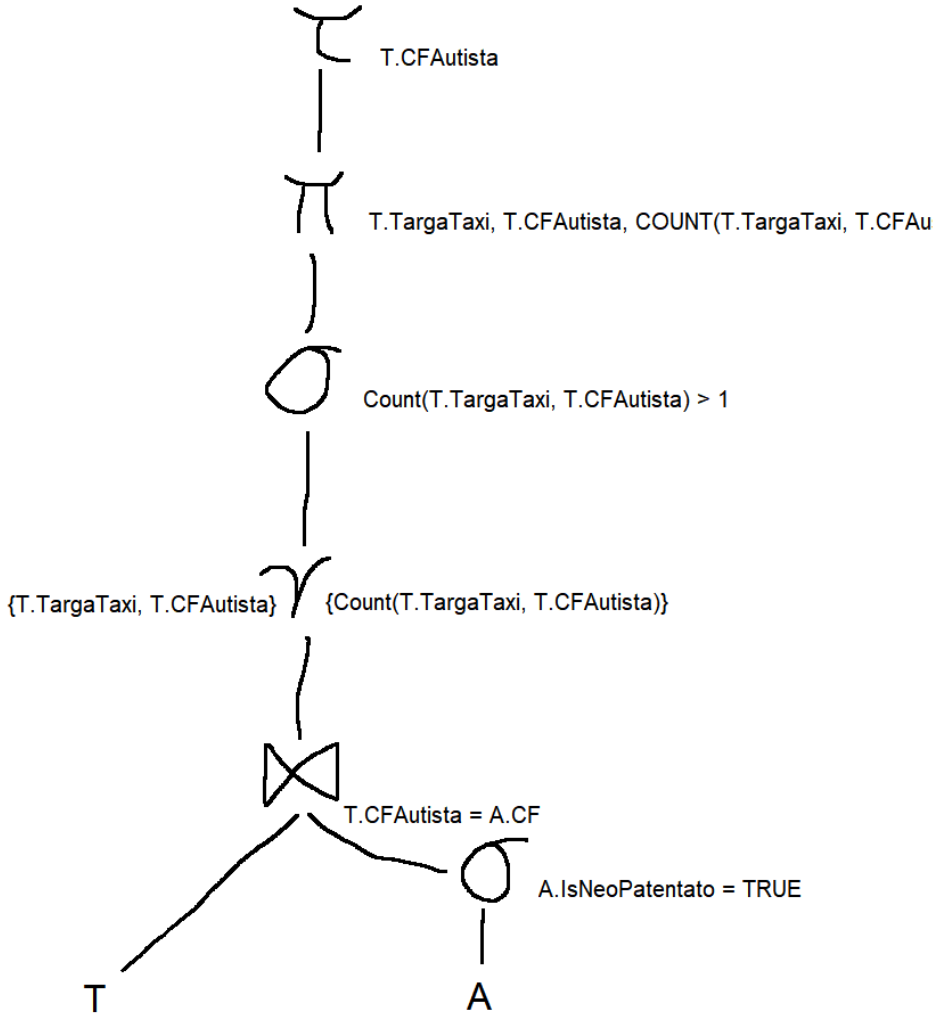
WHERE NumeroPrenotazione = (SELECT NumeroPrenotazione FROM Corsa WHERE ChilometriFatti > 50)

Piani di accesso logici

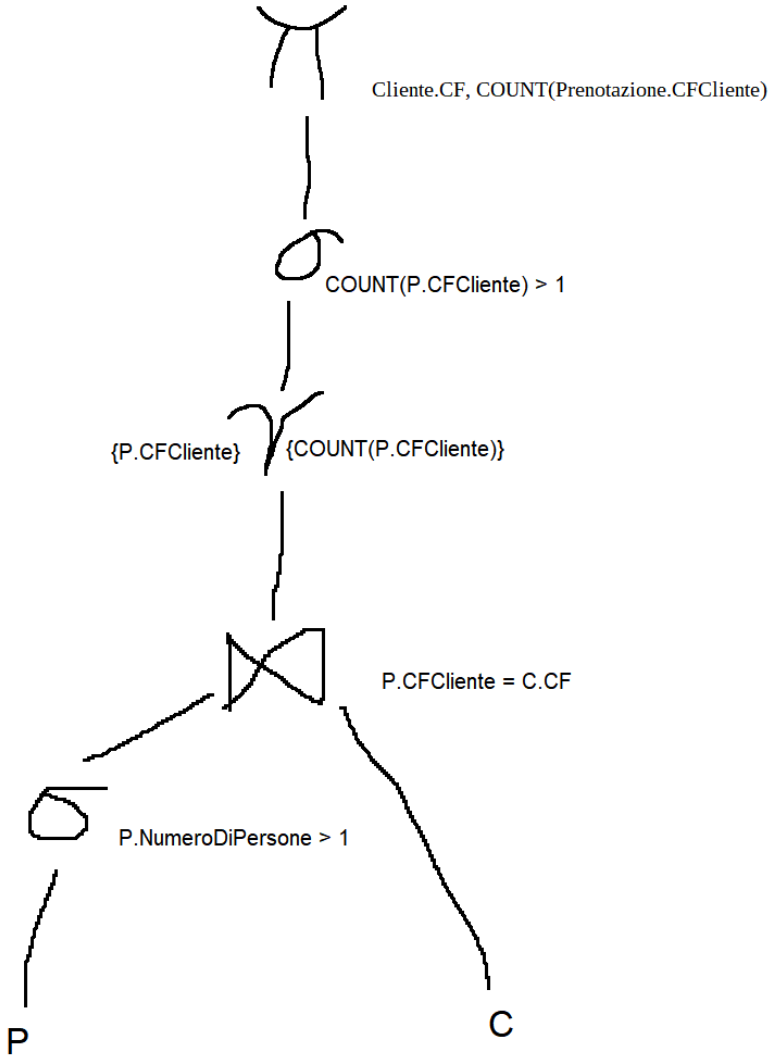
a)



b)

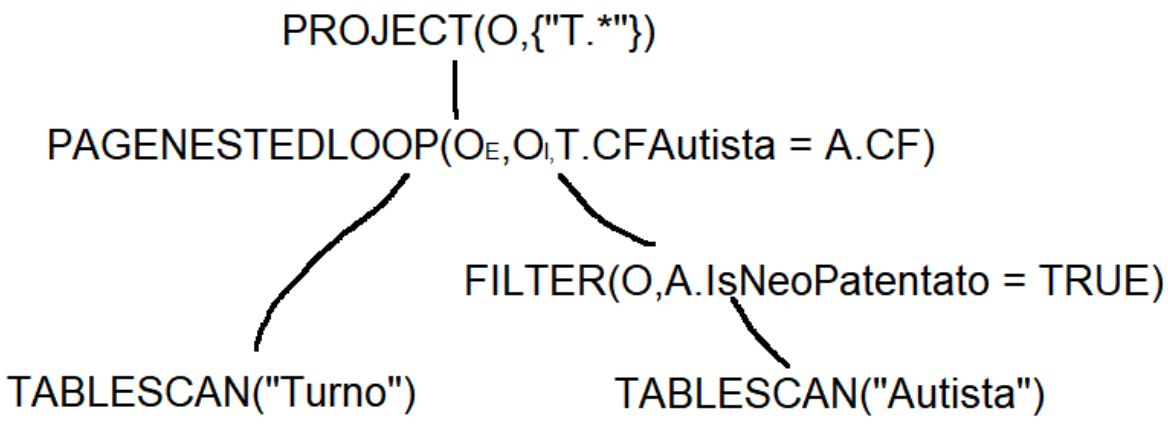


c)

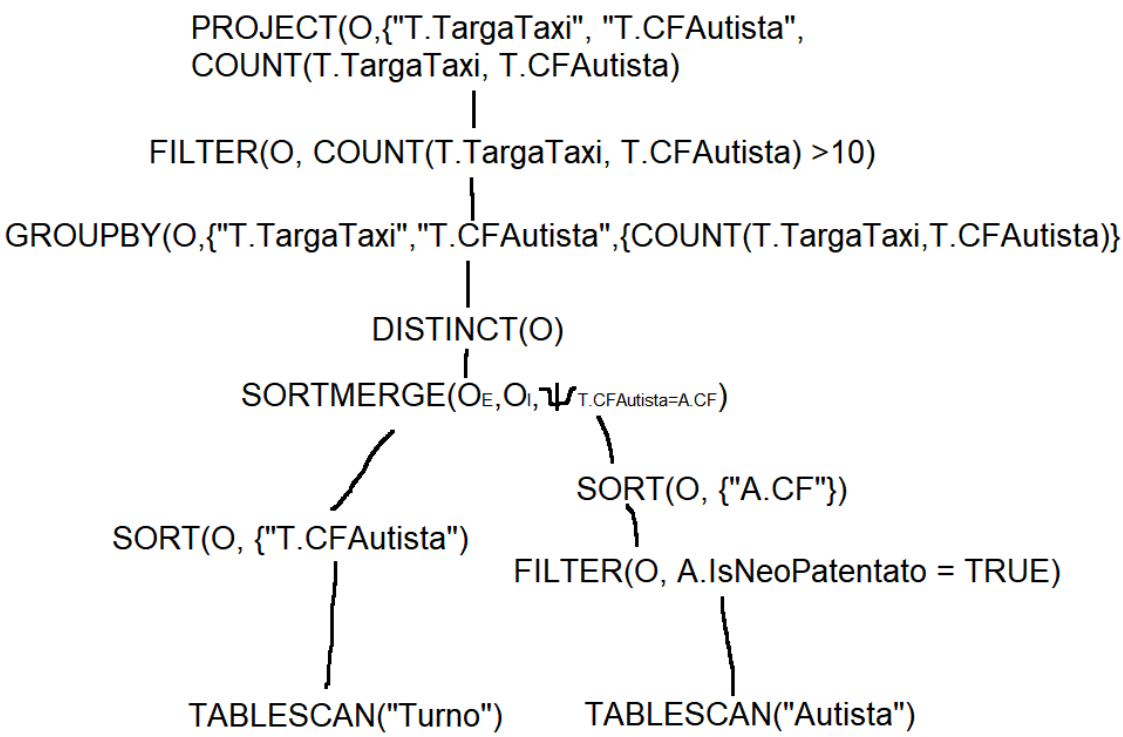


Piani di accesso fisici senza uso di indici

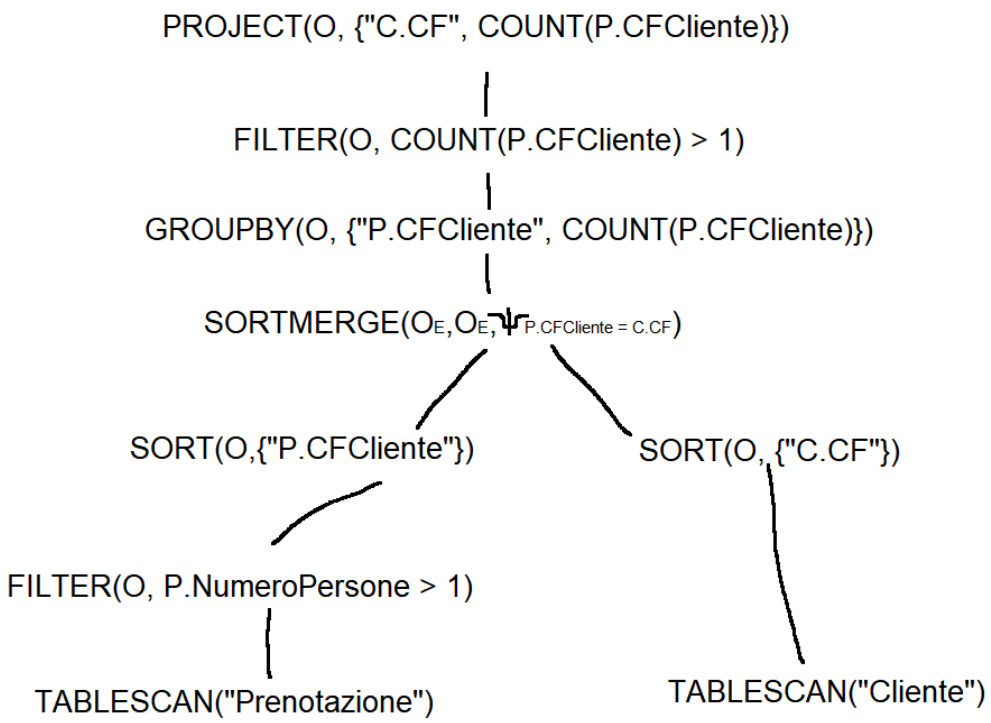
a)



b)

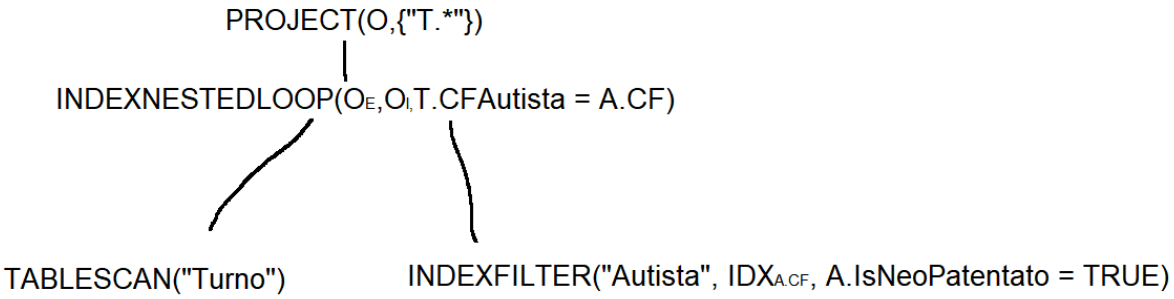

PROJECT(O,{"T.TargaTaxi", "T.CFAutista",
COUNT(T.TargaTaxi, T.CFAutista)
FILTER(O, COUNT(T.TargaTaxi, T.CFAutista) >10)
GROUPBY(O,{"T.TargaTaxi","T.CFAutista",{COUNT(T.TargaTaxi,T.CFAutista)}
DISTINCT(O)
SORTMERGE(OE,OI,⋈T.CFAutista=A.CF)
SORT(O, {"T.CFAutista")
SORT(O, {"A.CF"})
FILTER(O, A.IsNeoPatentato = TRUE)
TABLESCAN("Turno")
TABLESCAN("Autista")

c)

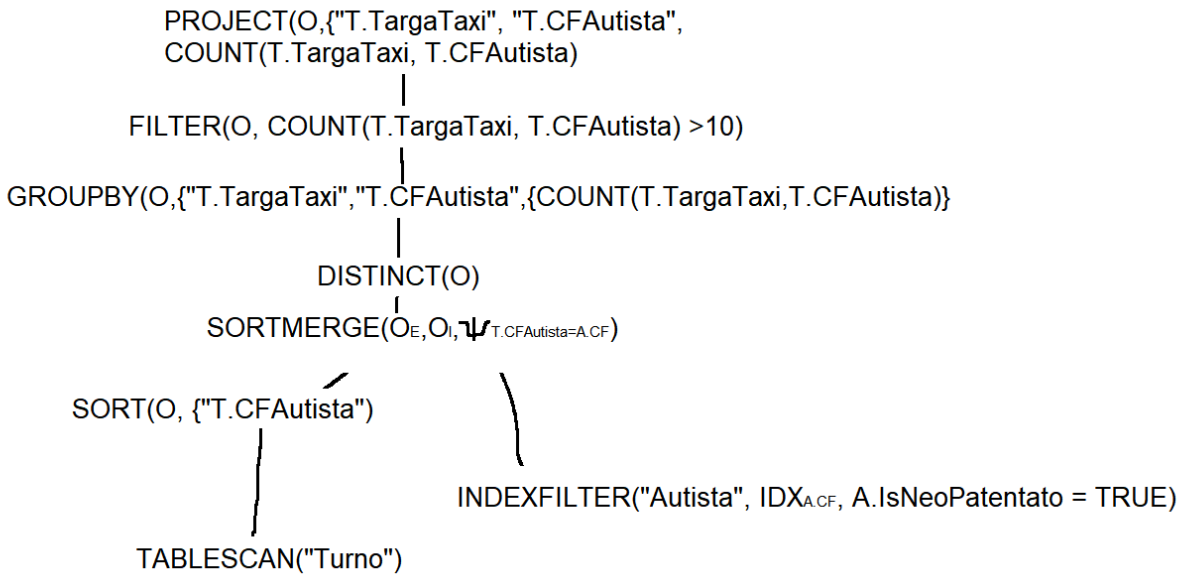

PROJECT(O, {"C.CF", COUNT(P.CFCliente)})
FILTER(O, COUNT(P.CFCliente) > 1)
GROUPBY(O, {"P.CFCliente", COUNT(P.CFCliente)})
SORTMERGE(OE,OE,⋈P.CFCliente = C.CF)
SORT(O,{"P.CFCliente"})
SORT(O, {"C.CF"})
FILTER(O, P.NumeroPersone > 1)
TABLESCAN("Prenotazione")
TABLESCAN("Cliente")

Piani di accesso fisici con uso di indici

a)



b)



c)