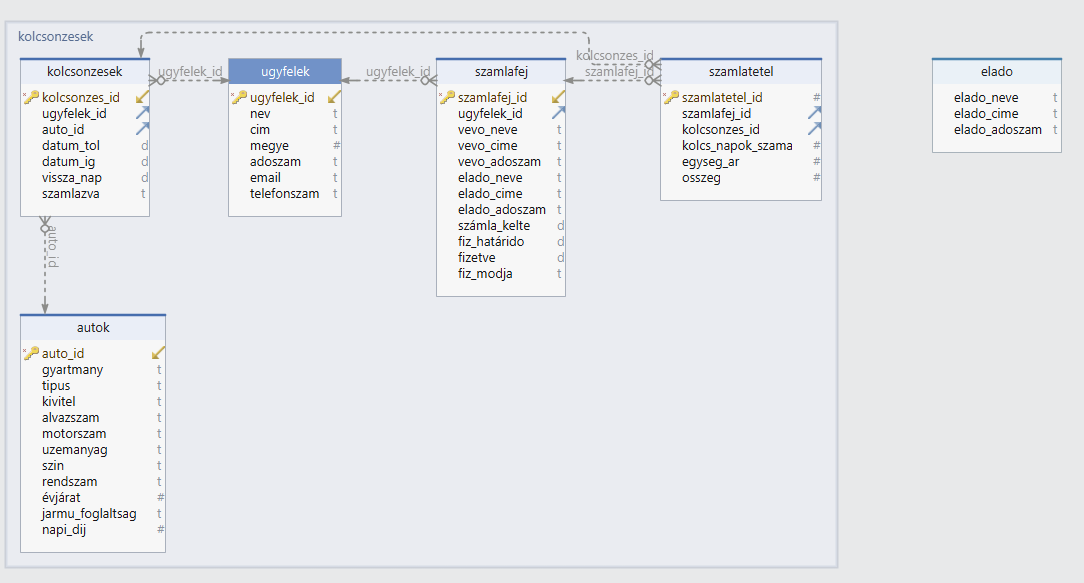
Autókölcsönző

# Táblaszerkezet



1. Ábra - Táblaszerkezet diagram

Az autókölcsönző adatbázis egy hat táblából álló adatbázis, ahol a KOLCSONZESEK tábla kapcsoló táblaként funkcionál, ezzel biztosítva azt, hogy az adatbázis 3NF-ben van. A számlázás két táblára bontva került az adatbázisba, azért hogy kezelni lehessen azt, ha az ügyfélnek több kölcsönzött járműve van. Az eladó tábla egy rekordot tartalmaz, a kölcsönző adatait.

Az adatbázis az alábbi táblákat tartalmazza, melyben a következőképpen kerülnek definiálásra a kulcsok, külsőkulcsok és mezők:

**ÜGYFELEK** (***ugyfelek\_id***, nev, cím, megye, adoszam, email, telefonszam)

Megye

* megyekódot tartalmaz
* értéke (1-20 ) 🡪 19 megye + Budapest

**AUTOK**(***auto\_id***, gyartmany, tipus, kivitel, alvazszam, motorszam, uzemanyag, szin, rendszam, évjárat, jarmu\_foglaltsag, napi\_dij)

Jarmu\_foglaltsag:

* jármű foglaltságát jelző flag
* értéke alapértelmezetten ’N’ – tehát nem foglalt
* amennyiben a járművet kiadták a fogaltságjelző értéke ’Y’-ra változik

Napi\_dij:

* forintban értendő

**KOLCSONZESEK** (***kolcsonzes\_id***, *ugyfelek\_id,* *auto\_id*, datum\_tol, datum\_ig, vissza\_nap, szamlazva)

* a kölcsönzések tábla kapcsoló tábla
  + ki , melyik autót kölcsönözte
* mennyi ideig kölcsönözte:
  + tól dátum: kölcsönzés első napja
  + ig dátum: kölcsönzés utolsó napja
* visszavétel napja
  + nem fix dátum
  + az a nap amikor ténylegesen visszahozta
  + az autok.jarmu\_foglaltsag mellett abból is lehet tudni, hogy melyik auto foglalt, hogyha a visszavétel napja értéke NULL - tehát a foglaltság kereshető a visszavétel napja töltöttsége alapján is

A számla azért került megbontásra szamlafej és számlatetel táblára, hogy kezelni lehesen azt a lehetőséget, hogy egy ügyfélnek több kölcsönzött járműve van.

**SZAMLAFEJ**(***szamlafej\_id****, ugyfelek\_id*, vevo\_neve,vevo\_cime, vevo\_adoszam, elado\_neve, elado\_cime, elado\_adoszam,szamla\_kelte, fiz\_határido, fizetve, fiz\_modja)

* az eladó és a vevő adatait tartalmazza

**SZAMLATETEL**(***szamlatetel\_id,*** *szamlafej\_id, kolcsonzes\_id*, kolcs\_napok\_szama, egyseg\_ar,osszeg)

* egyseg\_ar az autok.napi\_dij
* összeg: kolcsonzesek.kolcs\_napok\_szama \* autok.napi\_dij

**ELADO**(elado\_neve, elado\_cime, elado\_adoszama)

* a tábla az autokölcsönző vagyis, aki kiadja, az autókat annak adatait tartalmazza
* ez a tábla egy rekordot tartalmaz
* ezen adatok jellegzetessége, hogy minden kiállított számlán szerepelnek és minden számlán ugyanazok
* ha a cég adataiban változás áll be, az insertben szükséges módosítani azt

Particionálás:

A particionálás megyénként történt*,* ***partition by range*** segítségével.

Teljesítményanalízis:

Particionálás során a nagyméretű adattáblákat kisebb részekre osztunk fel, ezzel is gyorsítva a lekérdezés sebességét, mert így a lekérdezésnek kevesebb adatot kell egyszerre feldolgoznia. Így az adatbázis teljesítményét fokozhatja. Bár kisméretű adatbázis és táblák esetén ez a teljesítményfokozó hatás elképzelhető, hogy nem számottevő.

Teljesítményanalízis az ***explain (analyze)*** segítségével készült. Ahol az ***explain*** megadja a sorok becsült számát, az átlagos eredménysor becsült méretét és a becsült lekérdezés költségét, az analyze pedig a tényleges végrehajtási időt milliszekundumban.

A teljesítményanalízis során a lekérdezések vizsgálata történik indexelés nélkül illetve indexeléssel.

Az indexelés az adatok keresésének gyorsaságát segíti, azáltal hogy csökkenti az adatbázis által kezelendő –vizsgálat, leválogatás, összekapcsolás- adatok mennyiségét.

Teljesítmény analízis a három alábbi lekérdezésen készült:

## Lekérdezés 1

--Ki melyik autót mettől meddig kölcsönözte?

**explain** (**analyze**) **select** u.nev,a.rendszam, datum\_tol, datum\_ig **from** kolcsonzesek k

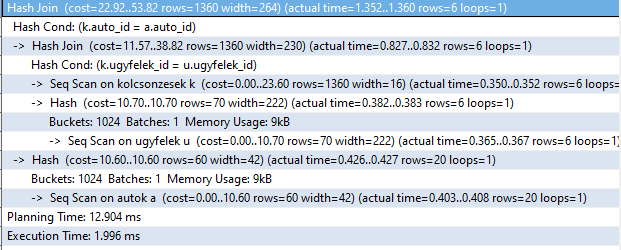
**join** ugyfelek u

**on** u.ugyfelek\_id = k.ugyfelek\_id

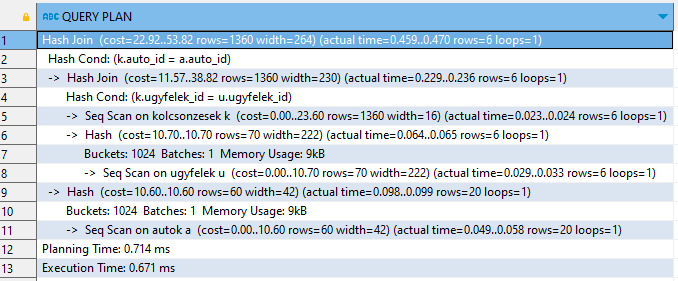
**join** autok a

**on** a.auto\_id = k.auto\_id

Kapott teljesítmény indexelés nélkül:



Kapott teljesítmény indexeléssel:



A két eredmény közül az indexeléssel történt végrehajtás jelentősen gyorsabb.

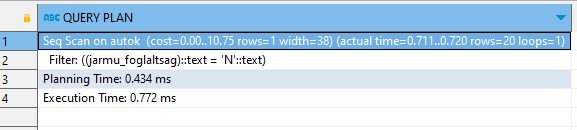
## Lekérdezés 2

-- Melyik auto nincs kiadva?

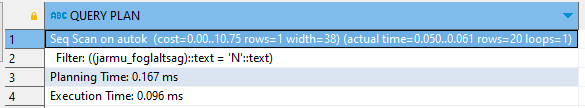
**explain** (**analyze**) **select** rendszam **from** autok

**where** jarmu\_foglaltsag = 'N'

Kapott teljesítmény indexelés nélkül:



Kapott teljesítmény indexeléssel:



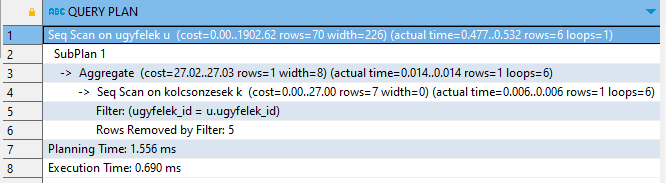
A két eredmény közül az indexeléssel történt végrehajtás jelentősen gyorsabb.

## Lekérdezés 3

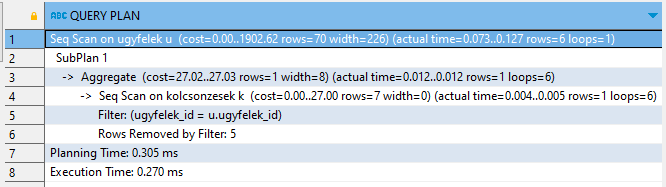
--Melyik ügyfél hányszor bérelt autót?

**explain** (**analyze**) **select** u.nev, (**select** **count**(\*) **from** kolcsonzesek k **where** k.ugyfelek\_id = u.ugyfelek\_id) **from** ugyfelek u

Kapott teljesítmény indexelés nélkül:



Kapott teljesítmény indexeléssel:



A két eredmény közül az indexeléssel történt végrehajtás jelentősen gyorsabb.

Mindhárom lekérdezésnél látható, hogy az indexelés ténylegesen gyorsított a lekérdezések valós végrahajtási idején.

Trigger:

A trigger a járműfoglaltságot állítja, kölcsönvételkor a flag értéke ’Y’-ra változik, míg visszavételkor ’N’-re .

A trigger működőképessége, egy kolcsonzesek táblába történő update-el került ellenőrzésre.