Objektum elnevezési szabályok:

- Betűvel kezdődik, legfeljebb 30 karakterből áll
  - Kivétel: adatbázisnév 8 karakter
    adatbázis kapcsolat 128, tartalmazhat @ és . karaktereket
- Csak az A–Z, a–z, 0–9, _ (aláhúzás), $ és # karaktereket tartalmazhatják (az utóbbiak használata nem javasolt)
- Nem különböztetjük meg a kis- és nagybetűket
- Nem egyezhetnek meg az Oracle Szerver fenntartott szavaival
- Egy felhasználónak nem lehet két azonos nevű objektuma, ha azok azonos névterületen vannak

I. ADATDEFINÍCIÓS NYELV (DDL)

Az adatbázis és az adatbázis objektumok létrehozására, kezelésére szolgál

Parancsai:

Létrehozás:	CREATE
Módosítás:	ALTER
Törlés:	DROP
Átnevezés:	RENAME

SQL szabványos adatbázis objektumok

- **TABLE (tábla)**: felhasználói és rendszeradatok tárolására szolgál, sorokból és oszlopokból áll
- **VIEW (nézettábla)**: más táblák, nézettáblák alapján létrehozott 'virtuális' tábla, adatokat nem tárol
- **INDEX (index):** lekérdezéseket gyorsítja, DML műveleteket lassítja, karbantartásuk időigényes
- **SYNONYM (szinonima)**: objektumok alternatív neve

Egyéb adatbázis objektumok

- **SEQUENCE (szekvencia)**: egyedi értéket generáló objektum
- **TYPE (típus)**: felhasználói adattípus
- **MATERIALIZED VIEW (materializált nézet)**: más táblákból, nézettáblákból származtatott adatok tárolására szolgáló tábla. Adatainak frissítése állítható.
- **Programegységek: (PL/SQL vagy más procedurális nyelven készíthetők)**
  FUNCTION (függvény) PROCEDURE (eljárás)
  PACKAGE (csomag) TRIGGER (trigger)
- **DATABASE, TABLESPACE, DATABASE LINK**: az adatbázisnak és logikai szerkezetének kialakítására szolgálnak
- **SYSTEM (instance), SESSION (felhasználó adatbázishoz való kapcsolódása)**. Működési egységek paramétereinek kezelésére szolgáló objektumok
- **USER, PROFILE, SCHEMA, ROLE:** Felhasználók és jogosultságok kezelésére szolgálnak

1. Táblák az Oracle adatbázisban

- **Adatszótár:**
  - Táblák és nézetek gyűjteménye, az adatbázis létrehozásakor jön létre, az Oracle rendszer kezeli, az adatbázisról tartalmaznak információt
  - A táblák tulajdonosa a SYS felhasználó, a felhasználók az adatszótár nézeteit használják
  - Tartalma: felhasználók nevei és jogosultságai, az adatbázis-objektumok nevei, a táblákra vonatkozó megszorítások, stb.

USER_	azon objektumok adatai, amelyeknek a felhasználó a tulajdonosa USER_TABLES, USER_OBJECTS
ALL_	azon objektumok adatai, amelyekhez a felhasználónak hozzáférési jogosultsága van, de azok más tulajdonában is lehetnek ALL_TABLES, ALL_OBJECTS
DBA_	az adatbázisban lévő összes objektum adata DBA_TABLES

- **Felhasználói táblák**: felhasználók hozzák létre

a) Tábla létrehozása:

```
CREATE TABLE tábla
      (oszlop adattípus [DEFAULT kif][oszlopmegszorítás], ...
            [táblamegszorítás],…)
      [tárolási előírások]
      [AS alselect];
```

Adattípusok:

CHAR[(m)]	Rögzített hosszúságú karakteres adat (1<=**m**<=2000)
VARCHAR2(m)	Változó hosszúságú karakteres adat (1<=**m**<=4000) A maximális *méret* megadása kötelező CHAR: 'A '='A' igaz VARCHAR2:'A '='A' hamis 'A '>'A' igaz
NUMBER[(p [,s])]	**p** pontosságú, **s** tizedes jegyet tartalmazó szám p: összes számjegy 1<=p<=38 s: a tizedesjeltől jobbra levő számjegyek száma -84<=s<=127
DATE	I. e. 4712.01.01 és i. sz. 9999. 12.31. közé eső dátum- és időérték

DEFAULT: alapértelmezett érték az oszlop számára, amelyet akkor kap meg, ha új sor beszúrásakor (INSERT) nem adunk értéket neki. Lehet literál, kifejezés vagy SQL függvény (SYSDATE, USER).

```
CREATE TABLE tabla
     (kod          VARCHAR2(4)        NOT NULL,
     nev           VARCHAR2(15)       NOT NULL,
     varos         VARCHAR2(15)       DEFAULT 'DEBRECEN'
     fizetes       NUMBER(8),
     datum         DATE);
```

AS alselect:

- A rendszer létrehozza a táblát, és beszúrja a táblába a SELECT utasítás által visszaadott sorokat
- Ha nem adunk meg oszlopokat, a létrejövő tábla oszlopainak neve megegyezik az alselect oszlopainak nevével
- Ha megadunk oszlopokat, azok számának meg kell egyeznie az alselect által visszaadott oszlopok számával
- Ha az alselectben kifejezés szerepel, adjunk másodlagos nevet az oszlophoz
- *Az oszlopok definíciójában csak az oszlop neve, alapértelmezett értéke és integritási megszorítás szerepelhet, adattípus nem*
- *Hivatkozási integritási megszorítás nem adható meg, csak az ALTER TABLE-ben*
- *Az új táblára az integritási szabályok nem vonatkoznak, csak az oszlop adattípusú definíciók: név, típus, méret, NOT NULL*

```
CREATE TABLE cikk1
     AS SELECT *
     FROM cikk
     WHERE afa=12;
```

Integritási megszorítások, kényszerek (CONSTRAINT):

Konzisztencia: Annak megkövetelése, hogy

- Összetartozó adatok módosítása együtt, megfelelő sorrendben történjen (pl. ha a telephely megszűnik, előbb az alkalmazottak törlődjenek, aztán a telephely)
- Ha vannak redundáns adatok, azoknak nem szabad egymással ellentmondásban lenni

Megszorítások:

- Szabályok, melyek biztosítják az adatbázis konzisztenciáját (ne kerüljenek hibás értékek a táblákba, használható legyen az adatbázis, logikai ellentmondásokat ne tartalmazzon)
- Betartásukat az RDBMS automatikusan biztosítja. Nem enged meg olyan adatbázis műveletek, amelyek révén az adatok a szabályokat megszegnék

Megjegyzések:

- Adjunk nevet a megszorításoknak, különben a rendszer generál egyet számukra SYS_C*n* formában Ez a név hibaüzenetekben jelenik meg, vagy az ALTER TABLE utasításban használhatjuk
- A rendszer az adatszótárban tárolja a megszorításokat:
  USER_CONSTRAINTS
  USER_CONS_COLUMNS
- Megszorításokat definiálhatunk a CREATE TABLE vagy ALTER TABLE parancsokban

Megszorítások általános alakja:

[**CONSTRAINT** megszorításnév] megszorítás [megszorítás_állapot]

Megszorítás:

1) **NOT NULL**: biztosítja, hogy az adott oszlopba ne kerüljön null érték

2) **UNIQUE(oszlop,…)**: megköveteli, hogy az oszlop(ok)ban szereplő értékek különbözőek legyenek a tábla különböző soraiban. A megadott oszlopot (vagy oszlopok halmazát) egyedi kulcsnak hívjuk, több oszlop megadása esetén összetett egyedi kulcsról beszélünk.
   Egyedi kulcs: tartalmazhat NULL értéket is, ha nem adtunk NOT NULL megszorítást ugyanazon oszlopokra. Ha nem adunk meg NOT NULL megszorítást, akkor tetszőleges számú sorban szerepelhet null érték, mert a rendszer a null értéket nem tekinti azonosnak semmi mással, tehát egy oszlopban (vagy akár az összes oszlopban) szereplő null érték mindig kielégíti a UNIQUE feltételt

3) **PRIMARY KEY(oszlop,...)**: elsődleges kulcsot hoz létre a táblához. Olyan oszlop(ok)at definiál, amelyben szereplő érték(ek) egyértelműen azonosítják a tábla minden sorát. A megszorítás egyrészt biztosítja a megadott oszlop(ok)ban szereplő érték(ek) egyediségét, másrészt azt, hogy az elsődleges kulcsot alkotó oszlopok egyike se tartalmazzon null értéket (UNIQUE és NOT NULL)

Az Oracle Szerver a UNIQUE és PRIMARY KEY kulcsmegszorítás betartatásához is létrehoz egy egyedi indexet a megfelelő oszlophoz.

4) **[FOREIGN KEY (oszlop,…)]**
   REFERENCES tábla[(oszlop,…)]
   [ON DELETE {CASCADE | SET NULL}]

oszlop(ok)at jelöl ki **külső kulcsk**ént (hivatkozási integritási megszorításként) és kapcsolatot hoz létre a külső kulcs és ugyanazon vagy egy másik tábla elsődleges vagy egyedi kulcsa között.
A külső kulcs értékének vagy meg kell egyeznie a szülőtáblában szereplő értékek egyikével, vagy null értéknek kell lennie.
A külső kulcsot a gyermektáblában kell megadnunk, a hivatkozott oszlopot tartalmazó tábla lesz a szülőtábla. A külső kulcsot a következő kulcsszavak használatával definiáljuk:

- FOREIGN KEY: táblaszintű megszorításnál megadja a gyermektábla oszlopát, oszlopszintű megszorításnál elhagyható a kulcsszó
- REFERENCES: azonosítja a szülőtáblát és annak oszlopát/oszlopait
- ON DELETE CASCADE: a szülőtábla egy sorának törlése esetén a rendszer a gyermektábla függő sorait is törölje
- ON DELETE SET NULL: a szülőtábla egy sorának törlése esetén a rendszer a gyerektábla megfelelő oszlopaiban a külső kulcs értékét nullára konvertálja
- Ha az utóbbi két opció egyikét sem adjuk meg, nem törölhetjük a szülőtábla azon sorait, amelyekre a gyermektábla hivatkozik

Ha az ALKALMAZOTT táblában a TKOD oszlopot az alábbi megszorítással hoztuk létre:

```
tkod VARCHAR2(2) CONSTRAINT alk_fk
                    REFERENCES telephely(tkod) ON DELETE CASCADE,
```

akkor a

```
DELETE FROM telephely WHERE tkod=30;
```

parancs kiadása esetén a

```
DELETE FROM alkalmazott WHERE tkod=30;
```

is végrehajtódik.

5) **CHECK(feltétel)**: a feltételt minden sornak ki kell elégítenie. A feltételben a lekérdezések összeállításakor alkalmazott feltételeket és szerkezeteket használhatjuk, a következő kivételekkel:
   - Nem hivatkozhatunk pszeudooszlopra, mint pl. CURRVAL, NEXTVAL, LEVEL vagy ROWNUM
   - Nem használhatjuk a SYSDATE, UID, USER, USERENV függvényeket
   - Nem alkalmazhatunk más sorok értékeire hivatkozó lekérdezéseket

Az egy oszlopra megadható CHECK megszorítások száma nincs korlátozva.

Megszorításokat definiálásának szintjei:
- **Oszlop**: Egy oszlopra vonatkozik, az adott oszlop definíciójának részeként adjuk meg. Bármilyen típusú megszorítást megadhatunk, közvetlenül az oszlop definíciója után írjuk, vesszőt nem kell tenni közéjük
- **Tábla:** Egy vagy több oszlopra vonatkozik, a NOT NULL kivételével bármilyen típusú lehet. A tábla oszlopainak definíciójától függetlenül adjuk meg, a több oszlopot érintő megszorításokat (pl. összetett kulcsok) csak tábla szintű megszorításként írhatjuk elő, vesszőt kell tenni közéjük.

Megszorítás_állapot:
- ENABLE VALIDATE: biztosítja, hogy minden új, a megszorítás alatt álló adatra vonatkozó DML művelet kielégítse a megszorítást. A régi adatokat is ellenőrzi. Alapértelmezés
- ENABLE NOVALIDATE: biztosítja, hogy minden új, a megszorítás alatt álló adatra vonatkozó DML művelet kielégítse a megszorítást. A régi adatokat nem ellenőrzi
- DISABLE: kikapcsolja/letiltja a megszorítást, az Oracle nem ellenőrzi azt

b) Táblaszerkezet módosítása

ALTER TABLE [séma.]táblanév

ADD {oszlop | megszorítás} …	--új oszlop a tábla végére új megszorítás megadása
MODIFY …	--meglévő oszlop megváltoztatása, NOT NULL megszorítás megadása, megszorítások engedélyezése, letiltása
DISABLE CONSTRAINT	--megszorítás kikapcsolása
DROP CONSTRAINT…	--megszorítás eltávolítása
DROP [COLUMN]…	--oszlop eltávolítása
RENAME TO…	--tábla átnevezése

Általános szabály: az utasítás végrehajtása akkor lesz sikeres, ha a tábla módosított definíciójának eleget tesznek a táblában már meglévő adatok és a hivatkozási integritási megszorítások sem sérülnek.

I. Tábla bővítése új oszloppal, DEFAULT megadása:

A meglevő sorok az új oszlopban NULL értéket kapnak:

```
ALTER TABLE olvaso
     ADD telefon VARCHAR2(20);
```

Ha van DEFAULT, minden eddigi sor megkapja az adott DEFAULT értéket:

```
ALTER TABLE konyv
     ADD megj varchar2(20) DEFAULT('megjegyzes');
SELECT * FROM konyv;
```

NOT NULL csak akkor adható meg új oszlophoz, ha még nem léteznek sorok a táblában vagy DEFAULT-al együtt:

```
ALTER TABLE konyv
     ADD megj2 varchar2(20) DEFAULT('masik megj.') NOT NULL;
```


II. Megszorítás hozzáadása táblához:

```
ALTER TABLE olvaso
          ADD CONSTRAINT nagybetu CHECK (nev=UPPER(nev));
SELECT * FROM USER_CONSTRAINTS
          WHERE TABLE_NAME LIKE 'OLVASO';
```

III. Oszlop módosítása, DEFAULT megadása:

Növelhetjük a számértéket vagy karaktereket tartalmazó oszlopok szélességét
Csökkenthetjük az oszlopok szélességét, ha csak null értéket tartalmaznak, vagy ha a táblának nincsenek sorai
Módosíthatunk CHAR típusú oszlopot VARCHAR2-re vagy fordítva, ha az oszlop csak null értékeket tartalmaz, vagy ha a méretet nem változtatjuk meg

```
ALTER TABLE olvaso
     MODIFY telefon CHAR(12); --típus módosítás és csökkentés (üres oszlop)
```

Módosíthatjuk az oszlopok adattípusát, ha csak null értékeket tartalmaznak
Az alapértelmezett érték módosítása csak a táblába a módosítás után beszúrt sorokra van hatással

```
ALTER TABLE olvaso
          MODIFY varos DEFAULT ('EGER');     --csak az új sorokra érvényes
INSERT INTO olvaso(okod,nev) VALUES (222,'OLVASO');
SELECT * FROM olvaso;
```

IV. NOT NULL megszorítás megadása:

Használható egy oszlophoz, ha minden létező sorban szerepel érték

```
ALTER TABLE olvaso
          MODIFY varos NOT NULL;
```

V. Megszorítás letiltása/engedélyezése:

```
ALTER TABLE olvaso
     DISABLE CONSTRAINT nagybetu [CASCADE];
SELECT * FROM USER_CONSTRAINTS
          WHERE TABLE_NAME LIKE 'OLVASO';
```

CASCADE a függő megszorításokat is kikapcsolja

VI. Oszlop törlése: (8i-től)
- Az oszlopok tartalmazhatnak adatot
- Legalább egy oszlopnak maradnia kell a táblában
- Törölt oszlopot nem tudunk helyreállítani
- Nem törölhetünk olyan oszlopot, amelyre más megszorítások hivatkoznak, de a CASCADE CONSTRAINTS opció megadásával az oszlopra hivatkozó megszorítások is törlődnek

```
ALTER TABLE konyv
     DROP (megj, megj2);
```

VII. Megszorítás törlése:

```
ALTER TABLE olvaso
     DROP CONSTRAINT nagybetu;
ALTER TABLE olvaso
     DROP CONSTRAINT nagybetu CASCADE;  -- törli a függő megszorításokat is
```

VIII. Tábla átnevezése:

```
ALTER TABLE kiado
        RENAME TO kiado2;
```

c) Táblaszerkezet törlése

DROP TABLE táblanév [CASCADE CONSTRAINTS];

Adatok, indexek, jogosultságok törlődnek, szinonimák, nézetek maradnak, de érvénytelenek lesznek. A parancsot a tábla tulajdonosa, vagy DROP ANY TABLE jogosultsággal bíró felhasználó adhatja ki.
CASCADE CONSTRAINTS: a táblával együtt a táblára hivatkozó megszorítások is automatikusan törlődnek

d) Tábla csonkolása

TRUNCATE TABLE táblanév;

- Törli a tábla összes sorát, felszabadíthatja az elfoglalt tárterületet
- A sorok törlése nem visszagörgethető, a parancs nem generál visszagörgetési információt, így gyorsabb, mint a DELETE
- A parancsot a tábla tulajdonosa, vagy DELETE TABLE objektumjogosultsággal rendelkező felhasználó adhatja ki
- Ha a tábla egy hivatkozási integritási megszorítás szülője, akkor a táblát nem lehet csonkolni, tiltsuk le a megszorítást, mielőtt kiadnánk az utasítást

DELETE (DML): törli a tábla sorait, de nem szabadítja fel az elfoglalt tárterületet, a sorok törlése visszagörgethető.

e) Megjegyzés fűzése táblához

COMMENT {ON TABLE tábla | COLUMN oszlop} IS 'szöveg';

f) Objektum átnevezése

Az objektum tulajdonosa átnevezheti a táblát, nézetet, szekvenciát vagy szinonimát.

RENAME réginév TO újnév;

2. Nézettábla

Más táblá(k)ból és/vagy nézet(ek)ből SELECT utasítással származtatott adatbázis objektum. Ablakhoz hasonlít, amelyen keresztül megtekinthetők és módosíthatók az alapjául szolgáló táblák adatai. Adatokat nem tartalmaz, a rendszer a nézeteket SELECT utasítás formájában tárolja az adatszótárban.

USER_VIEWS

Használatának előnyei:

- Adathozzáférés szabályozása: bizonyos felhasználók az alaptáblához nem kapnak hozzáférési jogot, hanem nézeten keresztül az adatok egy részéhez férhetnek hozzá
- Adatbázis komplexitásának elrejtése: egy vagy több táblára, nézettáblára támaszkodó bonyolult lekérdezés eredménye könnyebben kezelhető, ha nézetet alkalmazunk
- Elérhetjük, hogy a felhasználók különböző csoportjai saját igényeiknek megfelelően különféle adatokhoz férhessenek hozzá

a) Nézettábla létrehozása, létező nézet felülírása

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW név [(oszlopnév,...)]
AS szelekciós_utasítás
[WITH CHECK OPTION] [CONSTRAINT megszorítás_név]
[WITH READ ONLY] [CONSTRAINT megszorítás_név];

FORCE: akkor is létrehozza a nézetet, ha az alaptábla nem létezik
NOFORCE: csak akkor hozza létre a nézetet, ha az alaptábla létezik
Szelekciós utasítás: tetszőleges, ORDER BY nélküli lekérdezés
WITH READ ONLY: DML műveletek letiltása
WITH CHECK OPTION: biztosítja, hogy csak a nézet számára hozzáférhető sorok szúrhatók be vagy módosíthatók (amilyen adatokat lekérdezhetünk a nézettáblán keresztül, csak azokat tarthassuk karban)

Amikor nézettáblán hajtunk végre lekérdezést, a rendszer az alaptáblákból veszi az adatokat, ha az alaptáblák tartalma módosul, akkor módosul a nézettábla is.

- **Egyszerű nézet**:
  - Adatai egy táblából származnak
  - Nem tartalmaz oszlopkifejezést, függvényt vagy adatcsoportokat
  - A nézeten keresztül végrehajthatók DML utasítások, ha a nézet tartalmazza a tábla minden NOT NULL oszlopát (kivéve, ha van DEFAULT érték)
- **Az összetett nézet:**
  - Adatai több táblából származnak
  - Tartalmazhat függvényt vagy adatcsoportokat
  - Nem mindig hajthatók végre DML utasítások a nézeten keresztül

Egyszerű nézettáblán keresztül az alaptábla karbantartható az INSERT, UPDATE, DELETE műveletekkel

b) Nézettábla módosítása

ALTER VIEW nézettábla_név;

c) Nézettábla törlése

DROP VIEW nézettábla_név;

A parancsot a nézet létrehozója, vagy DROP ANY VIEW jogosultsággal rendelkező felhasználó adhatja ki

Karbantartás nézeten keresztül:

Ha nézettáblán keresztül kérdezünk le adatokat, a rendszer mindig az alaptáblából veszi azokat. Ha az alaptábla tartalma módosul, a nézettábla is módosul. Ha nézeten keresztül végzünk karbantartást, akkor is az alaptáblák adatait tartjuk karban. Nézeten keresztül történő karbantartás akkor végezhető el, ha minden szükséges adat megadható a nézeten keresztül és a karbantartandó sorok is egyértelműek.

Törölhetünk sorokat a nézetből, ha nem tartalmaz csoportfüggvényeket, GROUP BY utasításrészt, DISTINCT kulcsszót, a ROWNUM pszeudo-oszlop kulcsszavát
Módosíthatunk adatokat a nézetben, ha nem tartalmaz csoportfüggvényeket, GROUP BY utasításrészt, DISTINCT kulcsszót, a ROWNUM pszeudo-oszlop kulcsszavát vagy kifejezéssel definiált oszlopot
Felvehetünk adatokat a táblába, kivéve, ha a nézet nem tesz eleget az előző feltételeknek, illetve ha az alaptábla tartalmaz olyan alapértelmezett érték nélküli NOT NULL oszlopokat, amelyek nem szerepelnek a nézetben. Minden szükséges értéknek szerepelnie kell a nézetben. Ne felejtsük el, hogy a nézeten keresztül közvetlenül a nézet alaptáblájába írunk be új adatokat

3. Index

Sémaobjektum, amely
- mutató használatával felgyorsíthatja a sorok visszakeresését
- biztosíthatja egy vagy több oszlopon belül az értékek egyediségét
- ugyanakkor lassítja a táblával kapcsolatos DML műveleteket, mivel ezek végrehajtása után az összes indexet frissíteni kell

Egyedi index: értékei különbözők, az Oracle automatikusan létrehozza, ha egy tábla oszlopára PRIMARY KEY vagy UNIQUE megszorítást állítunk be, neve azonos lesz a megszorítás nevével
Felhasználói index: a felhasználó hozza létre, megfontolandó!!
Például egy FOREIGN KEY oszlopindex egy lekérdezésben megadott összekapcsoláshoz gyorsítja az adatok visszakeresését. Index egy vagy több oszlopra is létrehozható

Mikor hozzunk létre indexet?
Ha pl. van egy több ezer soros táblánk dátum típusú oszloppal és gyakran van szükségünk dátum vagy időszak szerinti lekérdezésre. Ha az oszlophoz nem tartozik index, akkor a rendszernek minden kereséskor a teljes táblát végig kell néznie.

Az index közvetlen és gyors hozzáférést biztosít a tábla soraihoz. Használatával csökkenthetjük a szükséges lemezműveletek számát, mert közvetlenül kijelöli a keresett adat helyét.

A rendszer automatikusan használja és karbantartja az indexeket, létrehozás után a felhasználónak nem kell foglalkoznia vele.
Az indexek logikailag és fizikailag is függetlenek a táblától, amelyet indexelnek. Ez azt jelenti, hogy bármikor létrehozhatók és törölhetők, ez nincs hatással az alaptáblára vagy más indexekre. Ha eldobunk egy táblát, a hozzá tartozó indexek is törlődnek.

a) Index létrehozása

```
CREATE [{UNIQUE | BITMAP}] INDEX indexnév
    ON táblanév (oszlopkif [ASC | DESC],...)    -- indexkulcsot alkotó oszlopok
    [tárolási és egyéb előírások];
```

UNIQUE: az oszlop(ok)nak, mely alapján indexelünk, egyedinek kell lenni
BITMAP: B-fa szerkezetű index helyett bitmap indexet hoz létre
ASC, DESC: kompatibilitás miatt van, az index létrehozása mindig az oszlopbeli értékek növekvő sorrendjében történik

b) Index módosítása

ALTER INDEX … utasítással csak a tárolási jellemzők változtathatók.

c) Index törlése

```
DROP INDEX indexnév;
```

Mikor célszerű indexet használni?
- Ha az oszlop értékei széles tartományba esnek
- Ha az oszlopban sok null érték szerepel
- Ha két vagy több oszlopot gyakran használunk együtt WHERE feltételben vagy összekapcsolásban
- Ha a tábla nagy, és a legtöbb lekérdezés csak a sorok legfeljebb 2–4 %-át adja vissza

Mikor ne használjunk indexet?
- Ha a tábla túl kicsi
- Ha az oszlopokat nem használjuk gyakran lekérdezések feltételeiben
- Ha a legtöbb lekérdezés a sorok több mint 2-4%-át adja vissza eredménytelenül
- Ha a tábla gyakran módosul
- Ha az indexszel ellátott oszlopokra kifejezés részeként hivatkozunk