Andmestruktuuride defineerimine SQL-is, DDL-laused(CREATE lause.) Andmete kontrollimehhanismide sissekirjutamine andmebaasi, relatsioonide kirjeldamine SQL-is.

Andmebaasi deklaratiivne terviklikkus.Andmesõnastik. Vaated (VIEW).

DDL (Data Definition Language) on SQL-I lausendite käskude alamhulk, mida kasutatakse andmebaasi struktuuride loomiseks, modifitseerimiseks ja kustutamiseks.

- **Tabelite loomine**

Tabelite nimed:
*peavad algama tähega, A-Z
*võivad sisaldada tähti, numbreid ja allkriipsu n. "KLIENDI_ARVED12"
*tabelite nimedes ei eristata suur ja väiketähti
*nimede pikkuseks võib olla kuni 18 märki (SQL-I standard)
*samanimelisi andmebaasi objekte ei tohi olla
*nimedeks ei tohi olla SQL-I reserveeritud sõnad.

Näiteks ei kõlba sellised nimed - "KLIENDI ARVED" -tühik vahel
"UPDATE" - SQL-I reserveeritud sõna
"12KLIENDI_ARVED" - algab numbriga

Nimedena on mõttekas kasutada antud andmetabelit(objekte) sisuliselt kirjeldavaid nimesid.

Tabeli loomise süntaks ja CREATE TABEL lause:

```
CREATE TABLE table-name
({column_name datatype [DEFAULT expr] [column_constraint] …
| table_constraint}
[,{column_name datatype [DEFAULT expr] [column_constraint] …
| table_constraint}  ] …)
```

Näiteks:

```
CREATE TABLE tudeng
      (tkood CHAR(10)  NOT NULL PRIMARY KEY,
        pnimi CHAR(10),
        enimi CHAR(15),
        synnip DATE,
        kursus NUMBER DEFAULT 1,
        suund CHAR(4));
```

Piirangud:
Constrait - piirang, mis kehtestatakse tervele tabelile või ühele selle tabeli veerule tabelis asuvate andmete terviklikkuse ja õigsuse tagamiseks. Relatsioonilisele andmebaasile on defineeritud kaks terviklikkuse reeglit:

1. **Olemi terviklikkus (*entity integrity*)** - mitte ükski primaarvõtmes sisalduv atribuut(veerg, column) ei või olla tühi.
2. **Viidete terviklikkus *(referential integrity)* -** kui mõnes tabelis on välisvõti, mis langeb kokku teise tabeli primaarvõtmega (aga välisvõti peab alati langema kokku mingi teise tabeli primaarvõtmega), siis see välisvõti peab olema kas täielikult võrdne teise tabeli võtmega või täiesti hull.

Piirangud laiemas mõttes hõlmavad ka andmete lubatavuse piiranguid, näit isikukood peab olema legaalne.

Tabeli ja veerupiirangute vahel ei ole põhimõttelist vahet, lihtsalt veerupiirangute abil on lihtsam ja loetavam kirjeldada ühte veergu puutuvaid tingimusi ja tabelipiirangutega mitut veergu korraga puudutavaid tingimusi.

Piirangute süntaks:
column_constraint:

```
[CONSTRAINT constraint_name]
  { [NOT] NULL |
     {UNIQUE | PRIMARY KEY}
         REFERENCES table_name [(column)]
    | CHECK (condition)}
```

table_constraint:

```
[CONSTRAINT constraint_name]
  { [NOT] NULL |
     {UNIQUE | PRIMARY KEY} (column [,column] …)
       | FOREIGN KEY ( column [, column] … )
         REFERENCES table_name [(column) [,column] … ]
    | CHECK (condition)}
```

Tähendused: **constraint_name** - piirangu identifikaator. Kui pole, siis genereeritakse. Piirangu identifikkator võib puududa siis, kui piirang kirjutatakse CREATE lauses kohe veeru definitsiooni järele, vt. eelmisi näiteid.

[NOT] NULL - antud veerg võib (või ei tohi) sisaldada NULL -väärtusi (NB! Ainult veerupiirangus)

UNIQUE - antud veerg või veergude kombinatsioon peab olema üle tabeli unikaalse väärtusega. (NB! Kui on kombinatsioon, siis peab olema unikaalne terve kombinatsioon, mitte iga väli eraldi)
PRIMARY KEY - antud veerg või veergude kombinatsioon on primaarvõti - see on alati unikaalne ja ei saa olla NULL.

FOREIGN KEY - antud veerg või veergude kombinatsioon on välisvõti (mingi teise tabeli primaarvõti)

REFERENCES - näitab ära , missuguse teise tabeli primaarvõtmega on tegemist

CHECK - näitab ära tingimuse, millele peab vastama iga rida

Näiteks:

```
        CREATE TABLE oppim
    (tudkood CHAR(10) CONSTRAINT td REFERENCES tudeng (tkood),
     akood CHAR(7),
      regaeg DATE DEFAULT SYSDATE,
      eksaeg DATE,
      hinne NUMBER DEFAULT 0 CHECK (hinne BETWEEN 0 AND 5)
      CONSTRAINT opvoti PRIMARY KEY (tudkood,akood),
      CONSTRAINT ainele FOREIGN KEY (akood) REFERENCES aine
(akood));
```

Terviklikkuse piirangud (*integrity constraints*) on kasutusel selleks, et andmebaasisüsteem järgiks teatud kindlaid suhteid ja reegleid, mis kehtivad tabelite vahel või tabelites endis. Piirangud on kasutusel selleks et :

*andmete sisestamisel, muutmisel ja kustutamisel andmebaasisüsteem teostaks kontrolli muudetavate andmete üle - selleks, et andmetega seotud operatsioon andmebaasis saaks toimuda, peab ta rahuldama piirangute tingimusi)

*et ära hoida tabeli või selle tabeli mingite kirjete kustutamist, kui sellest tabelist sõltuvad teised tabelid (nende kirjed)

- **Olemasolevate tabelite struktuuri muutmine ja andmesõnastik (Data Dictionary)**

DDL-i ülejäänud lausendid on olemasolevate struktuuride muutmiseks (ja ka kustutamiseks). Muuta võidakse nii tabeleid kui ka piiranguid.

Tabeli struktuuri muutmine:

```
ALTER TABLE table_name
 [ADD ( { column_name datatype [DEFAULT expr] [column_constraint] …
          | table_constraint}
         [,{ column_name datatype [DEFAULT expr] [column_constraint] …
          | table_constraint}…)]
[MODIFY ( { column_name (datatype) [DEFAULT expr] [column_constraint]..
             | table_constraint}
           [,{ column_name (datatype) [DEFAULT expr] [column_constraint].
            | table_constraint}…)]
    [ DROP  (column [,column] …)  ]


[DROP { PRIMARY KEY | UNIQUE (column [,column] …)
          CONSTRAINT constraint }]
```

Näide :

```
ALTER TABLE tudeng
    ADD (sugu CHAR(1) DEFAULT 'M')
    MODIFY (kursus DEFAULT 3,
                tkood PRIMARY KEY)
```

või

```
ALTER TABLE tudeng DROP kursus;
```

Tabeli eemaldamine :

```
DROP TABLE table_name;
```

Tabeli eemaldamisel kustutatakse tabel koos andmetega, kui need juhtuvad tabelis olema, enamuses andmebaasisüsteemides reeglina hoiatust kustutamisele ei eelne.

- **Andmebaasi deklaratiivne terviklikkus**

kujutab endast reeglite ja eeskirjade kogumit, millele andmebaasis sisalduvad andmed peavad vastama ja mida on võimalik defineerida (deklareerida) andmebaasi struktuuri kirjelduses(CREATE lauses)

Võib ette tulla olukordi, kus näiteks andmete sisestamisel on andmete õigsuse ja terviklikkuse kontrollil tõsisem ja keerukam, ning selle kontrolli läbiviimiseks tuleb kasutada eraldi algoritme ja protseduure (näiteks tabelisse andmete lisamisel tuleb kontrollida ja teha päringuid paljudesse teistesse andmebaasi tabelitesse , et kontrollida lisatavate andmete õigsust) . Selliste keerukamate seoste ja reeglite puhul ei saa kasutada deklaratiivset terviklikkust, vaid tuleb andmebaasi kirjutada eraldi andmete kontrolli ja manipuleerimise algoritmid kasutade **SPL- keelt** ja/või **andmebaasi trigereid** (nendest tuleb juttu edaspidi)

Üldiselt on võimaluse korral soovitatav kasutada deklaratiivset terviklikkust protseduuride kirjutamise asemel.

- **Andmesõnastik (Data Dictionary) :**

on andmebaasi tähtasim osa. Andmesõnastik sisaldab suure hulga nn. read-only tabeleid, mis annavad infot andmebaasi tabelite janende struktuuri kohta - veerud, andmetüübid, piirangud jne.)

Andmesõnastik sisaldab:

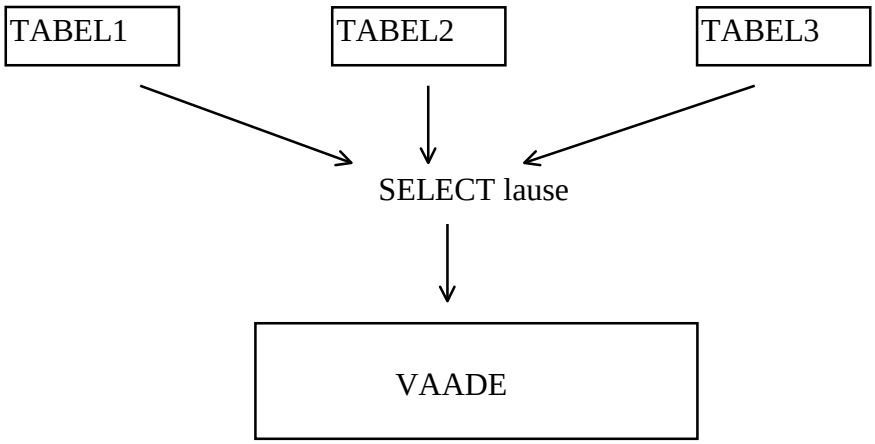
- andmebaasi kasutajate kasutajanimesid
- nende kasutajate õigusi ja privileege andmebaasi kasutamisel
- andmebaasi objektide nimesid (tabelid , vaated, indeksid, sünonüümid)
- andmebaasi tabelitele, veergudele rakendatud piiranguid
- muud, konkreetsest andmebaasisüsteemist sõltuvaid andmeid, nagu näiteks andmete kasutamise auditeerimisandmeid - kes ja millal on mingeid andmeid uuendanud, lisanud või kustutanud.

Andmesõnastikku sisaldavaid tabeleid nimetatakse andmebaasi **süsteemseteks tabeliteks**. Reeglina saab selliseid tabeleid ainult lugeda, kuid mõnesid tabeleid saavad vastavate kasutajaõigustega kasutajad (süsteemiadministraatorid) ka muuta. Süsteemsed tabelid luuakse andembaasisüsteemi poolt automaatselt andmebaasi loomisel, samuti toimub info uuendamine nendes automaatselt. Süsteemsete tabelite struktuuri teades on nendes asuvat infot lugeda tavaliste SELECT-lausete abil. Andmebaasis on olemas ka tabel(tabelid), milles sisaldub onformatsioon kõigi andmebaasi tabelite struktuuri (sealhulgas ka süsteemsete tabelite) kohta. Andmebaasi tabelite ettevaatamatu muutmine võib viia kogu andmebaasi riknemisele.

- **Vaated (*views*)**

Mis on vaade:

- vaated on otsekui “aknad” läbi mille on andme tabelites vaadatavad ja uuendatavad.
- vaade on tuletatud tabelist või teisest vaatest, millele vaates viiddatakse kui vaate **baastabelile** - “reaalsele” tabelile, kus andmeid tegelikult hoitakse.
- vaade on andmebaasi salvestatud andmebaasi kui SELECT lause, vaade on **virtuaalne tabel** , seda pole andmebaasis füüsiliselt olemas. Kasutajale “ilmub” tabel aga nii, nagu oleks tegemist tavalise tabeliga. Vaade on tegelikult vaate aluseks oleva päringu tulemusena saadud tabel.



- vaatel endal pole andmeid,ta manipuleerib baastabelite andmetega.
- läbi vaate nähtavaid andmeid on võimalik muuta ja enamusel juhtudest (lihtsamad vaated) on võimalik vaadetesse andmeid lisada. Sellisel juhul muutuvad vastavate baastabelite andmed.

Näide: luuakse vaade repsid tudengi tabeli esmakursuslastest

```
CREATE VIEW repsid
  AS SELECT * FROM tudeng WHERE kursus = 1
  WITH CHECK OPTION;
```

Läbi sellise vaate saab andmeid uuendada ainult juhul, kui lisatavatel või muudetavatel andmetel on tudengi kursuseks 1.

Läbi vaate ei saa andmeid tabelisse lisada siis, kui vaatesse ei ole lülitatud **kõiki** antud kirje jaoks vajalikke veerge (NOT NULL veerud peavad kõik olema esindatud)

Vaated on kasutatavad :

- andmebaasile juurdepääsu piiramiseks. Vaade võimaldab näidata ainult teatud osa andmebaasist.
- võimaldavad kasutajatel teha lihtsaid päringuid tulemustest, mis on omakorda saadud keerukaid päringuid kasutades.Keerukate päringute tegemine jääb seega andmebaasi projekteerijate , mitte lõppkasutajate hooleks. Näiteks võib kasutajale teha vaate mitmest tabelist, kus on kasutatud JOIN tingimusi.
- võimaldavad samu andmeid erinevatele kasutajatele näidata erineval viisil.

- **Vaate loomine , CREATE VIEW lause**

CREATE VIEW view_name [alias [,alias] …]
AS subquery
[WITH CHECK OPTION]

[alias [,alias] …] - vaates hõlmatud veergude uued nimed
WITH CHECK OPTION - andmeuuendused selle vaate abil peavad rahuldama vaadet moodustava alampäringu tingimusi.

Vaadet võib kasutada järgmistes lausetes:
SELECT,INSERT,UPDATE, DELETE

Muuta vaadet ei saa, puudub ALTER VIEW käsk.

- **Andmete muutmine läbi vaadete**

DELETE ei ole võimalik siis , kui vaade sisaldab

- JOIN-tingimust (kaks tabelit on päringus ühendatud WHERE klauslit kasutades)
- grupifunktsioone (AVG(), MIN() vms.)
- GROUP BY klauslit
- DISTINCT käsku

UPDATE on keelatud siis kui
- kõikudel ülalnimetatud juhtudel
- kui päringus on veerud tuletatud (n. PALK * 12)

INSERT on keelatud siis, kui:

- kõikidel ülalnimetatud juhtudel
- kui mõni NOT NULL veerg ei ole vaatesse valitud.

Vaate eemaldamiseks

DROP VIEW view_name;