BAB IV

SQL – Bahasa Basis Data Relasional

- SQL singkatan dari Structured Query Language
- Asal muasal disebut SEQUEL (Structured English Query Lanaguage) yang dirancang dan diimplemantasikan oleh Pusat Penelitian IBM untuk eksperimen system basis disebut system R
- Secara komersial telah digunakan sebagai high-level database Language untuk IBM DBZ
- Merupakan bahasa untuk memanipulasi basis data relasional yang distandarisasi oleh ANSI dean ISO :
  - SQL 1 (ANSI 1986)
  - SQL 2 (disebut juga sebagai SQL-92)
    Revisi + ekspansi terhadap SQL 1
  - SQL 3 (+object-oriented database)
- Sifat-sifat utama :
  - ❖ Declarative Language interface
    (Berisi statements yang menyatakan APA yang ingin dihasilkan)
  - ❖ Dapat digunakan baik sebagai DDL atau DML
  - ❖ Dapat di-embedded dalam general purpose high-level language seperti PASCAL & C

4.1 PENDEFINISIAN DATA DALAM SQL

- ➢ Dalam kuliah hanya dibahas konsep-konsep utama &penting (Detail dapat dilihat dalam dokumen khusus mengenai sintaks – sintaks dalam SQL)
- ➢ Istilah- Istilah TABEL,BARIS & KOLOM digunakan dalam SQL sebagai istilah – Istilah yang berkaitan dengan RELASI,TUPLE dan ATRTIBUTE

- ➢ Perintah-perintah SQL, untuk pendefinisian data meliputi :
  - CREATE
  - DROP
  - ALTER

4.1.1 KONSEP SCHEMA & CATALOG DALAM SQL

- ♣ Konsep 'SQL SCHEMA' digunakan untuk mengelompokan sejumlah table dan bentukan- bentukan lainnya yang dimiliki oleh satu aplikasi basis data yang sama
- ♣ Satu SQL SCHEMA diidentifikasi oleh:
  - Nama Schema
  - Pengenal otorisasi (user atau account yang memiliki Schema)
  - Elemen- Elemen Schema (table →tdk disimpan, view, domain→deklarasi data, dll)
- ♣ Catalog dlm SQL digunakan sbg kumpulan dari sejumlah Schema dlm lingkungan SQL
- ♣ Satu Schema dapat dibangun dengan menggunakan statement CREATE SCHEMA :

CREATE SCHEMA COMPANY AUTHORIZATION ADJUNAIDY;

- ♣ Satu catalog selalu berisikan skema khusus yang disebut INFORMATION-SCHEMA, yang di dalamnya menyediakan informasi mengenai :
  - semua element descriptors dari semua schema dalam catalog yang diberikan terhadap sejumlah user
  - definisi referential integrity constrains, bilamana terdapat relasi dalam schema pada catalog yang sama
  - definisi- definisi yang di-share oleh sejumlah schema dalam catalog yang sama (seperti domain definitions)

4.1.2 Perintah Create Table, Tipe-Tipe Data Dlm Sql Dan Constraints

- ♣ Perintah CREATE TABLE digunakan untuk mendefinisikan satu relasi baru yang berisikan :
  - nama relasi
  - nama attribut- attributnya disertai dengan tipe-tipe datanya dan domain constraint/values
  - key, entity integrity & referential integrity constraint
- ♣ Tabel yang dihasilkan disebut 'BASE TABLE' , yaitu table yang secara nyata dibuat dan disimpan dlm file oleh DBMS

Contoh :

CREATE TABLE EMPLOYEE ……..

(jika didefinisikan dlm lingkungan dimana CREATE TABLE di jalankan)

Atau

CREATE TABLE COMPANY. EMPLOYEE ……..

(deklarasi secara eksplisit terhadap schema)

☺ **TIPE-TIPE DATA DLM SQL**

- Tipe-tipe data untuk attribute meliputi :
  - ★ Numeric
  - ★ Character-string
  - ★ Bit-string (image)
  - ★ Date
  - ★ Time

★ **Tipe Data NUMERIC :**

- Bilangan- Bilangan integer : INTEGER (INT), SMALLINT
- Bilangan- Bilangan real : FLOAT, REAL, DOUBLE PRECISION

Formatted number di deklarasi :

DECIMAL(i,j)atau (DEC(i,j) atau NUMERIC(i,j))

i → presisi (jml. Decimal digit)

j → scale (jml. Digit pecahan)

★ **Tipe Data CHARACTER-STRING :**

- Fixed Length : CHAR(n) atau CHARACTER(n)
- Varying Length : VARCHAR(n) atau CHAR VARYING(n) Atau CHARACTER VARYING(n)

  n → jml. Maksimum dari CHARACTER

(default n = 1)

★ **Tipe Data BIT-STRING :**

- Fixed Length : BIT(n) , n = jml bit

- Varying Length : BIT VARYING(n) , n = jml bit maksimum
  (default n = 1)

★ **Tipe Data DATE & TIME :**

- DATE mempunyai 10 posisi dengan format untuk komponen YEAR-MONT-DAY : YYYY-MM-DD
- TIME paling sedikit mempunyai 8 posisi untuk komponen HOUR-MINUTE-SECOND dengan format : HH-MM-SS

Detail mengenai manipulasi dari tipe data DATE & TIME dapat mengacu pada Reference Manual SQL yang di pakai.

♣ 2 Cara mendefinisikan tipe data dari attribute :

- langsung di spesifikasikan pada setiap attribute dalam statement CREATE TABLE
- mendeklarasikan suatu tipe dari domain yang kemudian nama tipenya dapat digunakan dalam sejumlah attribute dengan domain yang sama:

  CREATE DOMAIN SSN_TYPE As CHAR(g)

  → memudahkan perubahan tipe data & menambah readability dari schema

CONTOH DEFINISI DATA UTK. SCHEMA BASIS DATA COMPANY :

```
CREATE TABLE EMPLOYEE
    ( FNAME          VARCHAR(9)        NOT NULL,
      MINIT          CHAR,
      LNAME          VARCHAR(15)       NOT NULL,
      SSN            CHAR(9)           NOT NULL,
      BDATE          DATE,
      ADDRESS        VARCHAR(30),
      SEX            CHAR,
      SALARY         DECIMAL(10,2),
      SUPERSSN       CHAR(9),
      DNO            INT               NOT NULL DEFAULT 1,

  CONSTRAINT EMPPK
      PRIMARY KEY (SSN),
```

```
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (SUPERSSN) REFERENCES EMLPOYEE (SSN)
                    ON DELETE SET NULL
                    ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)
                    ON DELETE SET DEFAULT
                    ON UPDATE CASCADE
);

CREATE TABLE DEPARTMENT
      ( DNAME          VARCHAR(15)  NOT NULL,
        DNUMBER        INT          NOT NULL,
        MGRSSN         CHAR(9)      NOT NULL DEFAULT"888665555",
        MGRSTARTDATE   DATE,
    CONSTRAINT DEPTPK
        PRIMARY KEY (DNUMBER),
    CONSTRAINT DEPTSK
        UNIQUE (DNAME),
    CONSTRAINT DEPTMGRFK
        FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE (SSN)
                    ON DELETE SET DEFAULT
                    ON UPDATE CASCADE

CREATE TABLE DEPT_LOCATIONS
      ( DNUMBER          INT             NOT NULL,
        DLOCATION        VARCHAR(15)     NOT NULL,
      PRIMARY KEY (DNUMBER, DLOCATION),
      FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT (DNUMBER)
                    ON DELETE CASCADE   ON UPDATE CASCADE
      );

CREATE TABLE PROJECT
      ( PNAME          VARCHAR(15)     NOT NULL,
```

```
  PNUMBER          INT              NOT NULL,
  PLOCATION        VARCHAR(15),
  DNUM             INT              NOT NULL,

  PRIMARY KEY (PNUMBER),
  UNIQUE (PNAME),
  FOREIGN KEY (DNUM) REFERENCES DEPARTMENT (DNUMBER)
  );

CREATE TABLE WORKS_ON
  ( ESSN           CHAR(9)          NOT NULL,
    PNO            INT              NOT NULL,
    HOURS          DECIMAL(3,1)     NOT NULL,

  PRIMARY KEY (ESSN, PNO),
  FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN),
  FOREIGN KEY (PNO) REFERENCES PROJECT (PNUMBER)
  );

CREATE TABLE DEPENDENT
  ( ESSN           CHAR(9)          NOT NULL,
    DEPENDENT_NAME VARCHAR(15)      NOT NULL,
    SEX            CHAR,
    BDATE          DATE,
    RELATIONSHIP   VARCHAR(8),

  PRIMARY KEY (ESSN, DEPENDENT_NAME),
  FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN)
  );
```

SYINTAX UMUM UTK SATEMENT “CREATE TABLE’ :

CREATE TABLE(table name)

(< coloumn name > < coloumn type > [< attribute constraint>]

{, < coloumn name > < coloumn type > [< attribute Constraint >]}

(EBNF = Extended Bacus Naur Form)

[< table constraint > { , < table constraint >}]

);

Dari syntax di atas , terdapat definisi yang sifatnya optional :

- < attribute constraint > : NOT NULL yang dapat diikuti dengan DEFAULT VALUE dari attribute.
- < table constraint > : untuk menspesifikasikan PRIMARY KEY & KEY CONSTRAINT dan FOREIGN KEY (Referential integrity constraint).

4.1.3 Perintah Drop Schema & Drop Table

♣ Syntax : DROP SCHEMA < schema name >

DROP TABLE < table name >

Digunakan untuk menghapus schema atau sejumlah table dari basis data. Keduanya boleh diikuti dengan option : CASCADE atau RESTRICT.

- contoh :

DROP SCHEMA COMPANY CASCADE ;

↳ Menghapus schema basis data company dan semua table , domain dan elemen-elemen lain di dalamnya.option RESTRICT digunakan apabila diinginkan untuk menghapus jika dan hanya jika schema TIDAK mempunyai elemen.

DROP TABLE DEPENDENT CASCADE ;

↳ Table DEPENDENT dihapus dari schema basis data COMPANY.
Dengan option CASCADE → semua table yang mengacu pada table yang di hapus , secara otomatis akan di hapus dari schema.
Dengan option RESTRICT → penghapusan dilakukan jika dan hanya jika TIDAK ADA table yang mengacu pada table yang akan di hapus.

4.1.4 Perintah Alter Table

♣ Syntax : ALTER < table name > ADD < col.name > < col.type >

♣ Digunakan untuk mengganti / merubah definisi dari ‘ base table ‘ yang telah dibuat.

♣ Tindakan (actions) yang diakibatkan oleh perintah ini dapat meliputi :

- adding atau dropping suatu kolom (attribute)
- mengganti definisi suatu kolom
- adding atau dropping table contrains

- contoh :

ALTER TABLE COMPANY.EMPLOYEE ADD JOB VARCHAR(12);

↳ menambah satu attribute baru. Jika default baru tidak disebutkan maka nilainya diset dengan NULL. Tidak boleh memp. NOT NULL CONSTRAINT. Nilai-nilai sebenarnya dari attribute baru HARUS tetap diisi untuk setiap tuple Employee.

ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS CASCADE ;

↳ menghapus satu kolom dari suatu table , harus diikuti oleh salah satu pilihan : CASCADE atau RESTRICT. Dengan CASCADE , maka semua constraint dan views yang mengacu pada kolom akan dihapus secara otomatis dari schema.

Jika RESTRICT diplih , maka perintah akan sukses bilamana tidak ada constraint atau view yang mengacu pada kolom yang akan dihapus.

ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN
DROP DEFAULT;
ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN
SET DEFAULT “333445555”;

↳ merubah definisi satu kolom dengan cara menghapus default clause yang ada, atau mengganti nilai dari default clause.

ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE ;

↳ mengganti Constraint yang telah didefinisikan dlm suatu table dengan menambah (adding) atau menghapus (dropping) constraint.Untuk tindakan dropping suatu constraint , maka constraint tersebut harus diberi nama pada

saat di definisikan. Contoh Constraint Foreign Key "EMPSUPERFK" dalam table EMPLOYEE.Constraint dapat di definisikan kembali dengan menambah (adding) suatu constraint baru ke dalam relasi.

4.2 MENDEFINISIKAN QUERY DALAM SQL

- SQL hanya mempunyai satu statement untuk melakukan 'Information retrieval' dari suatu basis data; yaitu : SELECT statement.

  (dengan segala pilihan dan variasinya)

  NOTE : SELECT statement dalam SQL TIDAK MEMPUNYAI KORELASI dengan SELECT OPERATION dalam aljabar relational.

- Dalam SQL , satu table BUKAN merupakan set dari tuple ; dimana dalam SQL suatu table di perbolehkan mempunyai dua atau lebih tuple yang sama.
  Jadi , jika diinginkan suatu hasil table yang berupa set , maka harus digunakan suatu option yang memungkinkan untuk ini (yaitu DISTINCT Option).
- Syntax umum :

```
SELECT [DISTINCT] < attribute list >
FROM ( < table name > { < alias > } | < joined table > )
        { , ( < table name > { < alias > } | < joined table > ) }
    [ WHERE <  condition >]
    [ GROUP BY < grouping attributes >
                    [ HAVING < group selection condition > ] ]
    [ ORDER BY < col.name > [ < order > ]
                    { , < col.name > [ < order > ] } ]
< attribute list > :: = ( * | ( < col.name > | < function >
                                ( ( [ DISTINCT ] < col.name > | * ) ) )
                            { , < col.name > | < function >
                                ( ( [ DICTINCT ] < col.name > | * ) ) } ) )
< grouping attributes > : = < col.name > { , < col.name > }
< order > : = ( ASC | DESC )
```

4.2.1 Basic Sql Queries

- Bentuk dasar dari select statement disebut sebagai suatu 'mapping' atau 'SELECT FROM WHERE block', yang mempunyai bentuk umum :

  SELECT < list dari attributes yang akan di_retrive >

FROM < list nama-nama relasi yang digunakan >

WHERE < kondisi berupa ekspresi Boolean >

- Dalam kuliah, features dari SQL ditunjukan melalui sejumlah contoh sebagai ilustrasi.

Q1 : Retrieve birthdate dan address dari employee yang bernama ‘ John B. Smith ‘

```
SELECT   BDATE , ADDRESS
FROM     EMPLOYEE
WHERE    FNAME = ‘John’ AND MINIT = ‘B’ NAD LNAME = ‘Smith’
```

⇨ satu Query SQL sederhana yang hanya melibatkan satu relasi dlm FROM CLAUSE serupa dengan operasi SELECT-PROJECT dalam aljabar relasional , dimana WHERE clause menyatakan kondisi selection. Perbedaannya , dlm SQL dapat dihasilkan suatu table dengan sejumlah tuple yang sama (redundant tuples).

Q2 : Retrieve name dan address dari semua employee yang bekerja pada Department ‘Research’.

```
SELECT      FNAME, LNAME, ADDRESS
FROM        EMPLOYEE, DEPARTMENT
WHERE       DNAME = ‘Research’ AND DNUMBER = DNO
```

⇨ Dalam aljabar relasional serupa dengan urutan SELECT-PROJECT-JOIN, sehingga bentuk query ini disebut sebagai SELECT-PROJECT-JOIN queries.

Q3 : Untuk setiap project yang berlokasi di ‘Stafford’, dapatkan daftar nomor project , nomor department pengendalinya , dan nama , alamat serta tanggal lahir dari manager department pangendali.

```
SELECT      PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM        PROJECT, DEPARTMENT, EMPLOYEE
WHERE       DNUM = DNUMBER AND MGRSSN = SSN AND
            PLOCATION = ‘Stafford’
```

4.2.2 Penanganan Attribute Yang Ambiguous Dan Penggunaan Alias

♣ Oleh karena sejumlah relasi yang berbeda mungkin punya nama-nama attribute yang sama ; maka untuk menghindari ambiguity (keracunan) , suatu Query yang mengacu ke dua atau lebih nama attribute yang sama harus menggunakan 'qualifier' nama attribute dan nama relasi yang diacu (dipisahkan oleh tanda '**.**').

Sebagai contoh , misalnya attribute LNAME dan DNO dalam relasi EMPLOYEE diganti dengan NAME dan DNUMBER ; dan attribute DNAME dalam relasi DEPARTMENT deganti dengan NAME , maka Query Q2 menjadi :

```
Q2A : SELECTF     FNAME, EMPLOYEE.NAME, ADDRESS
      FROM        EMPLOYEE, DEPARTMENT
      WHERE       DEPARTMENT.NAME = 'RESEARCH' AND
                  DEPARTMENT.DENUMBER=EMPLOYEE.DNUMBER
```

♣ Ambiguity juga dapat terjadi pada kasus dimana suatu Query mengacu pada relasi yang sama dua kali (recursive relationships), contoh :

Q4 : Untuk setiap employee, retrieve first dan last name dari employee, dan first dan last name dari supervisor langsung dari employee.

```
SELECT    E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM      EMPLOYEE E, EMPLOYEE S
WHERE     E.SUPERSSN = S.SSN
```

↳ Nama relasi alternative E dan S disebut ALIASES.

Nama alias dapat dituliskan langsung setelah nama relasi , atau dituliskan setelah keyword 'As' :

```
EMPLOYEE As E
```

Atau dapat juga digunakan untuk menamakan kembali (renaming) attribute-attribute suatu relasi dalam FROM clause :

```
EMPLOYEE As E (FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO)
```

♣ Aliases dapat juga digunakan untuk menyingkat penulisan qualifier suatu attribute dalam kasus untuk menghindari ambiguity, contoh berikut merupakan penulisan yang lain untuk Query yang sama dengan Q2A :

```
SELECT    FNAME, E.NAME, E.ADDRESS
FROM      EMPLOYEE E, DEPARTMENT D
```

```
WHERE       D.NAME = 'RESEARCH' AND
            D.DNUMBER = E.DNUMBER
```

4.2.3 Query Tanpa "Where" Clause Dan Penggunaan (*)

♣ Query SQL yang dispesifikasikan TANPA "WHERE" clause mengidnikasikan tidak adanya kondisi pemilihan tuple, sehingga SENUA TUPLE dari relasi yang dinyatakan dalam FROM clause dipilih sebagai hasil Query.

Q5 : Dapatkan semua SSN dari Employee

```
SELECT      SSN
FROM        EMPLOYEE
```

Jika relasi yang dinyatakan dalam FROM clause lebih dari satu , maka CROSS PRODUCT dari semua kombinasi yang mungkn dihasilkan :

Q6 : Dapatkan semua kombinasi dari SSN Employee dan DNAME Department

```
SELECT      SSN,DNAME
FROM        EMPLOYEE, DEPARTMENT
```

♣ Tanda asterisk (*) digunakan dalam SELECT clause bilamana diinginkan untuk melakukan retrieve dari semua attribute dari tuple-tuple terpilih.

Q2C : retrieve semua nilai-nilai attribute dari employee tuples yang bekerja pada department 5 :

```
SELECT      *
FROM        EMPLOYEE
WHERE       DNO = 5
```

Q2D : retrieve semua nilai-nilai attribute dari employee dan attribute Department dimana Employee tersebut bekerja bagi setiap employee yang bekerja pada Department 'RESEARCH' :

```
SELECT      *
FROM        EMPLOYEE,DEPARTMENT
WHERE       DNAME = 'RESEARCH' AND DNO = DNUMBER
```

Q6A : Dapatkan CROSS PRODUCT dari relasi EMPLOYEE dan DEPARTMENT

```
SELECT    *
FROM      EMPLOYEE, DEPARTMENT
```

4.2.4 Table Sebagai Set

♣ SQL tidak memperlakukan suatu relasi hasil / table sebagai suatu set, sehingga adanya tuple yang muncul lebih dari sekali TIDAK dieliminasi secara otomatis dalam hasil Query. Alasanya :

- Proses eliminasi ‘duplicate tuples’ merupakan operasi yang mahal, dimana satu cara untuk mengimplementasikan ini adalah dengan mensorting tuples dan kemudian mengeliminasi ‘duplicate tuples’ yang ada.
- User mungkin menginginkan untuk melihat duplicate tuples dalam hasil Query .
- Bilamana suatu fungus aggregate digunakan terhadap sejumlah tuple dalam SQL, hamper dalam semua kasus tidak dinginkan untuk mengeliminasi duplikasi tuple yang ada.

♣ Untuk mengeliminasi duplikasi tuple dari hasil suatu Query dalam SQL dilakukan dengan menggunakan keyword DISTINCT dalam SELECT clause.

Contoh :

Q7 : retrieve gaji yang diterima oleh setiap employee

```
SELECT    SALARY                SELECT    DISTINCT SALARY
FORM      EMPLOYEE              FROM      EMPLOYEE
(hasilnya mungkin ada           (hasil tanpa duplikasi)
duplikasi tuple)
```

❖ Operasi-operasi dalam SQL :

- UNION (operati union)
- INTERESECT (Operasi interse tion)
- EXCEPT (Operasi set difference)

Relasi yang dihasilkan oleh ketiga operasiini berupa SET OF TUPLES.

Oleh karena ketiga operasi set diatas hanya boleh dikenakan terhadap relasi – relasi yang Union-compatible, maka kedua relasi yang dikenakan operasi harus :

- Mempunyai attribute yang sama
- Mempunyai urutan attribute yang sama

QB : Dapatkan satu list dari semua nomor project yang melibatkan seorang employee yang bernama " smith" baik sebagai seorang pekerja biasa atau sebagai seorang manajer dari departement yang mengendalikan project.

```
(SELECT PNUMBER
 FROM PROJECT, DEPARTEMEN, EMPLOYEE
 WHERE DNUM = DNUMER AND MGRSSN = SSN AND LNAME = "SMITH"
)
UNION
(SELECT PENUMBER
 FROM PRJECT, WORKS-ON, EMPLOYEE
 WHERE PNUMBER = PNUMBER = PNO AND ESSN AND LNAME = " SMITH "
)
ATAU

SELECT PNUMBER
FROM WORKS-ON AS 10 (ESSN, PNUMBER, HOURS), EMPLOYEE
WHERE W. ESSN = AND LNAME = " SMITH"
```

4.2.5 Nested Queries & Set Comparations

❖ Nested Query sangat cocok digunakan dalam suatu query dimana nilai-nilai yang ada diambil dan digunakan dalam suatu kondisi perbandingan.

Dalam nested query, suatu query select yang lengkap digunakan dalam Where Clause dari query yang lain (Outer Query)

```
Q8A     : Spesifikasi Q 8 dengan menggunakan nested query
SELECT  DISTICT PNUMBER
FROM    PROJECT
WHERE   PNUMBER IN  (SELECT     PNUMBER
                     FROM       PROJECT, DEPARTEMENT, EMPLOYEE
```

```
                    WHERE   DNUM = PNUMBER AND LNAME    = “
                            SMITH”
                )
        OR
        PNUMBER IN  (SELECT PNO
                    FROM    WORKS-ON, EMPLOYEE
                    WHERE   ESSN = SSN AND LNAME = “ SMITH”
                    )
```

- Dalam contoh diatas, operator diatas, operator perbandingan “IN” di gunakan untuk membandingkan satu nilai V dengan satu set (atau multiset) nilai V dan menghasilkan evaluasi TRUE jika v EV
- Operator IN juga dapat digunakan untuk membandingkan satu tuple dari nilai-nilai yang dituliskan dalam tanda kurung _ dengan satu srt tuple yang union-compatible.

```
SELECT    DISTINCT ESSN
FROM      WORKS – ON        Tidak semua cocok
WHERE     (PNO, HOURS) IN  (SELECT PNO, HOUIR =
                             FROM WORKS-ON
                            WHERE  ESSN =  “ 123456789”
                           )
```

↳ Dapatkan social security number dari semua employee yang bekerja pada project dan jumlah jam yang sama dengan project dan jumlah yang sama dimana “smith” terlibat.

- Selain operator IN, terdapat sejumlah operator perbandingan yang dapat digunakan untuk membandingkan satui nilai V (biasanya satu atribut) terhadap satu set V (biasanya nested query)
  - ⇒ Operator = ANV (atau = some) ekivalen dengan operator IN, yang memberikan nilai TRUE jika nilai V sama dengan beberapa nilai dalam set V
  - ⇒ Operator –operator lain yang dapat dikombinasikan dengan ANY (atau SOME) adalah : >, >=, <, <= dan <>.Keyword ALL juga dapat dikombinasikan dengan salah satu operator –operator ini, misalnya :

Kondisi perbandingan (ν > ALL V) akan memberikan nilai TRUE jika ν lebih besar dari semua nilai dalam set V.

Contoh :

```
SELECT    LNAME
FROM      EMPLOYEE
WHERE     SALARY .> ALL  (SELECT  SALARY
                          FROM    EMPLOYEE
                          WHERE   1NO = 5)
```

Menghasilkan nama-nama employee yang gajinya lebih besar darpada gaji semua employee dalam departement S.

- Dalam nested Query yang melibatkan relasi dengan nama yang sama dengan outer query, maka acuan terhadapUnqualitified attribute hanya diperbolehkan dilakukan terhadap nested query yang terdalam sedang acuan terhadap outer query dilakukan dengan menggunakan nama alias dari relasi yang diacu

  Qg : Retrieve nama dari setiap employee yang mempunyai dependent dengan first name dan set yang sama dengan employee tersebut.

```
SELECT    E. NAME, E.LNAME
FROM      EMPLOYEE E
WHERE     E.SSN IN    (SELECT    ESSN
                       FROM      DEPENDENT
                       WHERE     ESSN = ESSN AND
                                 E.  FNAME=  DEPENDENT-NAME
                                 ASND SEX  =       E. SEX)
```

- Secara umum, suatu query dengan struktur nested select…..from…where… dan menggunakan operator perbandingan = atau In selalu dapat dinyatakan sebagai query dengan strukktur satu blok saja

```
Qga  : SELECT    ENAME, E. LNAME
       FROM      EMPLOYEE E, DEPENDENT D
       WHERE     E.SSN = D.ESSN   AND
                 E.SEX= D. SEX    AND.
                 E. FNAMAE = D. DEPENDENT-NAME
```

CATATAN :

Bilamana suatu kondisi dalam Where Clause dari suatu nested query mengacu beberapa attribute dari suatu relasi yang di deklarasi dalam outer query, maka kedua query dikatakn saling berkorelasi (corralated).

Suatu correlated query akan dapat di mengerti dengan lebih mudah dengan memperhatikan kenyataan bahwa "Nested Query di evaluasi sekali untuk setiap tuple (atau kombinasi dari sejumlah tuple) dalam outer query"

4.2.6 Fungsi Exists Dan Non Exists

❖ Digunakan untuk melakukan pengecekan apakah hasil dari suatu "correlated nested query" berisi tuple atau tidak.

EXISTS (Q) : Memberikan nilai return TRUE, jika dalam hasil query Q minimal terdiri satu tuple.

NOT EXISTS (Q) : Jika memberikan nilai return TRUE, jika tak satupun tuple yang dihasilkan dalam hasil query Q.

Q11 : Retrieve nama-nama employee yang tidak mempunyai dependent.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    NOT EXISTS (SELECT *
                     FROM DEPENDENT
                     WHERE SSN = ESSN)
```

Q12 : Dapatkan list nama-nama manajer yang paling tidak mempunyai satu dependent.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    EXISTS   (SELECT *
                   FROM DEPENDENT
                   WHERE SSN  = ESSN)
         AND
         EXISTS   (SELECT*
                   FROM DEPARTEMENT
                   WHERE SSN  =  MGRSSN)
```

Q10 A : Retrieve nama-nama employee yang bekerja pada semua proyek yang dikontrol aleh departement nomor 5

```
SELECT    NAME, FNAME
FROM      EMPLOYEE
WHERE     NOT EXISTS
          (SELECT *
          FROM WORKS-ON B
          WHERE (B. PNO IN      (SELECT PNUMBER
                                 FROM PROJECT
                                 WHERE DNUM  =  5
                                 )
                 )
          AND
          NOT EXISTS            (SELECT*
                                 FROM WORKS-ON C
                                 WHERE  C.ESSN  =  SSN  AND
                                        C.PNO = B. PNO
                                 )
)
```

⇒ Query di atas dapat dinyatakan kembali :

cari setiap employee sedemikian rupa sehingga tidak terdapat (suatu project yang dikendalikan oleh departement 5 dimana employeetersebut tidak bekerja di dalamnya.).

4.2.7 Eksplisit Set Dan Null

Digunakan untuk menyatakan nilai-nilai set yangeksplisit dalam suatu where clause

Q 13 : Retrieve SSN dari semua employee yang bekerja pada project nomor 4, 2, dan 3

```
SELECT    DISTICT ESSN
WORK      WORKS-ON
WHERE     PNO IN (1, 2, 3)
```

Q 14 : Retrieve nama-nama semua employee yang tidak mempunyai supervisor.

```
SELECT    ENAME, LNAME
FROM      EMPLOYEE
WHERE     SUPERSSN IS NULL
```

= NULL → IS NULL

≠ NULL → IS NOT NULL

4.2.8 Penanaman Kembali Attribute-Attribute Hasil Query Dan Tabel –Tabel Hasil Operasi Join

Penamaan kembali dari Attribute :

Q 4A : Untuk setiap employee, retrieve last name dari employee dan last name dari supervisor lansung dari employee tersebut.

```
SELECT    E.NAME      AS EMPLOYEE-NAME,
          S. LNAME    AS SUPERVISOR-NAME
FROM      EMPLOYEE    AS E, EMPLOYEE AS S
WHERE     E. SUPERSSN = S. SSN
```

Konsep “Joined Table” dalam SQL 2 di maksudkan untuk memungkinkan memungkinkan menyajikan spesifikasi yang dihasilkan dari operasi join dalam from –clause

Dalam SQL 2 terdapat 5 pilihan untuk menyajikan spesifikasi joined table :

- Inner Join (Join)
- Left Outer Join (Left Join)
- Right Outer Join (Right Join)
- Full Outer Join (Full Join)
- Natural Join

Q 4A : Untuk setiap employee, retrieve last name dari semua employee yang bekerja pada “research” dept.

```
SELECT    FNAME, LNAME, ADDRESS
FROM      (EMPLOYEE JOIN DEPARTEMENT ON DNO = DNUMBER)
WHERE     DNAME = “ RESEARCH”
```

```
Q2B : SELECT    FNAME, LNAME, ADDRESS
      FROM      (EMPLOYEE NATURAL JOIN (DEPARTEMENT AS DEPT
                (DNAME, DNO, MSSN, MSDATE)))
      WHERE     DNAME = RESEARCH
```

Q3A : SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM ((PROJECT JOIN DEPARTEMENT ON DNUM = D NUMBER)
JOIN EMPLOYEE ON MBRSSN =SSN)
WHERE DNAME = " STAFFORD"

4.2.9 Aggregate Functions & Grouping

Fungsi fungsi aggregate : – COUNT
– SUM
– MAX
– MIN
– AVG

dapat digunakan dalam select clause atau having clause

Q15 : Dapatkan jumlah salary dari semua employee, maksimum salary, minimum salary dan average salary dari semua employe.

```
SELECT SUM (SALARY), MAX (SALARY), MIN (SALARY), AVG
       (SALARY)
FROM EMPLOYEE
```

Q16 : sama seperti Q15, tetapi untuk semua employee dari Research Department.

```
SELECT SUM (SALARY), MAX (SALARY), MIN (SALARY), AVG
       (SALARY)
FROM 'EMPLOYEE' DEPARTMENT
WHERE DNO = DNUMBER = DNAME = 'RESEARCH'
```

Q17 & Q18 : Retrieve jumlah employee dalam Company (Q17) dan jumlah employee dalam 'Research' Department (Q18).

Q17 :
```
SELECT COUNT (*)
FROM EMPLOYEE
```

Q18 :
```
SELECT COUNT (*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO = DNUMBER = DNAME = 'RESEARCH'
```

⇒ Tanda * menyatakan baris

Q19 : Hitung jumlah nilai – nilai distinct salary dalam database :

```
SELECT    COUNT (DISTINCT SALARY)
FROM      EMPLOYEE
```

Q20 : Retrieve nama – nama employee yang mempunyai dependent sebanyak 2 atau lebih.

```
SELECT    LNAME, FNAME
FROM      EMPLOYEE
WHERE     (SELECT COUNT (*)
          FROM DEPENDENT
          WHERE SNN = ESSN
          ) ≥ 2
```

- **Penggunaan pengelompokan hasil berdasarkan attribute tertentu dapat menggunakan 'Group By Clause'. Untuk ini, attribute yang dijadikan grouping harus juga muncul dalam 'Select Clause'.**

  Q21 : Untuk setiap department, retrieve dept number, jumlah employee dalam department dan juga average salary dalam department tersebut.

```
SELECT DNO, COUNT (*), AVG (SALARY)
FROM EMPLOYEE
GROUP BY DNO
```

  Q22 : Untuk setiap project, retrieve nomor dan nama project serta jumlah employee yang bekerja pada projrct tersebut.

```
SELECT    PNUMBER, PNAME, COUNT (*)
FROM      PROJECT, WORKS – ON
WHERE     PNUMBER = DNO
GROUP BY  PNUMBER, PNAME
```

- **Pasangan clause "Group by" dan "Having" digunakan untuk melakukan retrieving pengelompokan yang memenuhi kondisi tertentu**.

  Q23 : Untuk setiap project yang melibatkan employee sebanyak 2 atau lebih, retrieve project number, project name, dan jumlah employee yang bekerja didalamnya.

```
SELECT      PNUMBER, PNAME, COUNT (*)
FROM        PROJECT, WORKS – ON
WHERE       PNUMBER = DNO
GROUP BY    PNUMBER, PNAME
HAVING      COUNT (*) > 2
```

Q24 : Hitunglah jumlah pegawai yang mempunyai salary > $ 40.000 dalam setiap department, tetapi hanya untuk department yang mempunyai jumlah pegawai >5.

```
SELECT      DNAME = DNO, EMPLOYEE
FROM        DEPARTMENT, EMPLOYEE
WHERE       DNUMBER = DNO AND SALARY > 40000 AND
            DNO IN (SELECT DNO
                    FROM EMPLOYEE
                    GROUP BY DNO
                    HAVING COUNT (*) > 5
                   )
GROUP BY    DNAME
```

4.2.10 Subtring Comparison Arithmetic Operator & Ordering

- **Subtring Comparison ➔ menggunakan <u>LIKE</u> comp. operator**
  - * % substring % : Menggantikan sembarang jumlah karakter
  - * – (underscore) : Menggantikan satu karakter.

Q25 : Retrieve semua employee yang mempunyai alamat di Hiuston

```
SELECT      FNAME, LNAME
FROM        EMPLOYEE
WHERE       ADDRESS LIKE  %HOUSTON%
```

Q26 : Retrieve semua employee yang dilahirkan dalam tahun 50 – an (1950 – 1959)

```
SELECT      FNAME, LNAME
FROM        EMPLOYEE
WHERE       BDATE LIKE '--5----------'
```

- **Arithmetic operator : +, – , * dan /**

  Q27 : Tunjukkan hasil perubahan gaji dari setiap pegawai yang bekerja pada project 'Productx" jika diberikan kenaikan sebesar 10%.

```
SELECT   FNAME, LNAME, 1.1*SALARY
FROM     EMPLOYEE, WORK – ON, PROJECT
WHERE    SSN = ESSN AND DNO = PNUMBER AND
         PNAME = 'Productx'
```

  ⇨ Untuk type data string : operator II digunakan untuk menggabungkan (contenate) 2 string.

  – Untuk tipe data DATE, TIME, TIMESTAMP dan INTERVAL

    Data types : operator yang dapat digunakan meliputi penambahan (+) dan pengurangan (-) oleh suatu interval dengan tipe yang compatible dari masing – masing tipe diatas.

- **Untuk mendapatkan hasil yang terurut berdasarkan satu atau lebih nilai – nilai attribute dapat digunakan : ORDER By Clause.**

  Q28 : Dapatkan suatu list dari employee dan project dimana ia bekerja; berturut berdasarkan nama department dan dalam setiap department diurut secara alphabetical order berdasarkan last name dan first name.

```
SELECT   FNAME, LNAME, FNAME, LNAME
FROM     DEPARTMENT EMPLOYEE, WORK – ON,
         PROJECT
WHERE    DNUMBER = DNO AND SNN = ESSN AND
         DNO = DNUMBER
DEFAULT ORDER  : Ascending Order
Pilihan eksplisit      : ASC (Ascending) | DESC (Descending)
ORDER BY  DNAME DEC, LNAME ASC, FNAME
```

4.3 UPDATE STATEMENT

– Digunakan untuk melakukan modifikasi database dan meliputi :

  - INSERT
  - DELETE

- UPDATE

– Suatu DBMS yang secara penuh mengimplementasikan SQL2 seharusnya mendukung dan memaksa semua integrity constraints yang dispesifikasikan dalam DLL.

4.3.1 Insert Command

- **BASIC :** digunakan untuk menyisipkan / menambahkan satu (single) tuple ke dalam suatu relasi.

  U1 : INSERT INTO EMPLOYEE
  VALUE (‘Ricard’, ‘K’, ‘Marini’, ‘653298653’, ’30-Dec-52’, ‘98 02K Forest’, Katy’, ‘M’, 37000’, ‘987654321’, ‘4’)
  VALUE (‘Ricard’, ‘Marini’, ‘653298653’)

- **VARIATION :**

  Menyisipkan multiple tuples ke dalam suatu relasi sehubungan dengan creating relation dan loading relasi yang dibuat dengan hasil suatu query dari relasi yang telah ada.

CONTOH :

Create temporary table dengan heading berupa nama, jumlah employee dan total salaries dari setiap department.

```
A2A : CREATE TABLE DEPTS – INFO (DEPT NAME VARCHAR (15)
                                 NO – OF – EMPS   INTEGER,
                                 TOTAL – SAL      INTEGER
                                 );
A2B : INSERT INTO DEPTS – INFO ( DEPT NAME, NO – OF – EMPS
                                 TOTAL – SAL)
                               SELECT  DNAME, COUNT (*) SUM (SALARY)
                               FROM    DEPARTMENT, EMPLOYEE
                               WHERE   DNUMBER = DNO
                               GROUP BY DNAME ;
```

NOTE : Untuk variasi ini, table sementara yang dihasilkan TIDAK BERUBAH akibat adanya perubahan – perubahan table DEPARTMENT atau EMPLOYEE.

4.3.2 Delete Command

- Digunakan untuk menghapus sejumlah tuple dari suatu relasi.
- Tergantung dari kondisi penghapusan yang dinyatakan dalam WHERE Clause, maka jumlah tuple yang dihapus akan bervariasi.
  Tanpa Where Clause, berarti menghapus semua tuple yang ada dalam tabel.

```
U3A  :  DELETE  FROM    EMPLOYEE
        WHERE           LNAME = 'Brown'

U3B  :  DELETE  FROM    EMPLOYEE
        WHERE           SNN = '123456789'

U3C  :  DELETE  FROM    EMPLOYEE
        WHERE           DNO IN      ( SELECT DNUMER
                                      FROM DEPARTMENT
                                      WHERE DNAME = 'Research'
                                    )
U3D  :  DELETE  FROM    EMPLOYEE
```

4.3.3 Delete Command

- Digunakan untuk memodifikasi nilai – nilai attribute dari satu atau lebih tuple yang dipilih.

```
U4  :  UPDATE   PROJECT
       SET      PLOCATION = 'Bellaire', DNUM = 5
       WHERE    PNUMBER = 10
U5  :  UPDATE   EMPLOYEE
       SET      SALARY *1.15
       WHERE    DNO IN      (  SELECT DNUMBER
                               FROM DEPARTMENT
                              WHERE DNAME = 'Research'
                             )
```

  Setiap perintah UPDATE secara eksplisit hanya boleh menspesifikasikan satu relasi saja.