SQL I

TPI4210
Sistem dan Teknologi Informasi

Summary of Basic SQL Retrievals

- Bentuk dasar perintah SELECT:

  SELECT <nama column atau * (semua kolom)>
  FROM <nama tabel atau nama alias>
  WHERE <syarat dan/atau ketentuan join>
  ORDER BY <nama kolom>

- Operasi ini mungkin melibatkan operator perbandingan <, <=, =, >, >=,<>

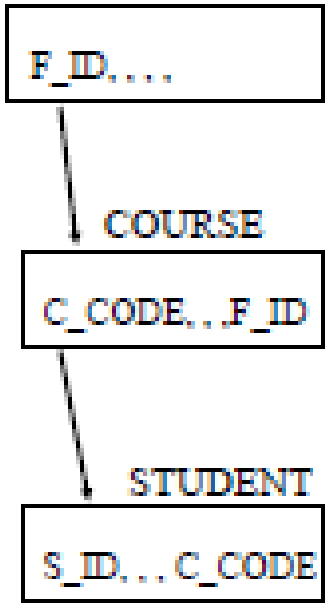
- Dalam pernyataan majemuk, digunakan operator Boolean AND, OR atau NOT

- ORDER BY digunakan untuk melakukan pengurutan

Subqueries

- Subquery adalah sebuah operasi subselect yang mengakomodasi query dalam query, sehingga hasil dari satu query dapat digunakan dalam syarat untuk query lain
- Bentuk dasar subquery SELECT-FROM-WHERE, sama seperti query lain. Hanya, operasi SELECT dalam subquery hanya boleh berisi satu pernyataan dan tidak boleh menghasilkan suatu nama kolom baru
- Subquery ditulis dalam kurung

Subquery Example




```
select surname, initials
from student
where course_code in
(select coursecode
from course
where faculty_id = ‘FTP');
```

Query diatas akan menampilkan daftar surname dan initial dari setiap siswa yang mengambil course dari FTP, tanpa memperhatikan jurusannya

Predicates and Search Conditions

- Syarat pencarian digunakan dengan WHERE dan HAVING untuk menyatakan data yang akan ditampilkan
- Syarat pencarian berisi predicates yang bisa diletakkan dalam kurung dan terhubung melalui operator Boolean
- Predicate adalah pernyataan benar-salah, yang mengevaluasi:
  - "true" atau "false" (dua nilai)
  - "true", "false" atau "unknown (tiga nilai)
  - .....
- ada tujuh jenis predicates

Boolean Truth Tables for Three-valued Logic

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

Predicates 1 - Comparison Predicates

- Perbandingan:

  = sama dengan

  != <> tidak sama dengan

  >, <, <=, >= ……

- Bentuk umum:

  <pernyataan> operator perbandingan <pernyataan>

- Contoh: **customer_id = “C34”**

Predicates 2 - The LIKE Predicate

- Digunakan untuk operasi pencocokan pola (*pattern matching*) dari tipe karakter data, bentuk umumnya:

  columnname [not] like pattern

- Operator yang digunakan:
  - % (percent) untuk nol atau lebih karakter apapun.
  - _ (underscore) untuk satu sembarang karakter

- Contoh: untuk mencocokkan data tipe string dalam kolom surname, yang dimulai dengan "Ari" (misalnya Ari, Arik, Arifianto) → **surname like "Ari%"**

Predicates 3 - The BETWEEN Predicate

Untuk mengantisipasi penggunaan predicate perbandingan:

x >= y AND x <= z

Digunakan:

x between y and z

Predicates 4 - The IN Predicate

- Bentuk umum:

  x IN (l,m,n,...z)

  Berarti x = l atau x = m atau .....x = z

- Juga bisa dinyatakan dalam

  expression [NOT] IN (subquery)

Consider the Base Table pupil …

Select * from pupil;

Surname	Initials	Suburb	Year_mark
Green	GG	Oakwood	56
Brown	BB	Elmwood	87
Gold	GG	Ashwood	68
White	WW	Elmwood	59
Violet	VV	Ashwood	77
Blue	BB	Elmwood	76
Black	BB	Oakwood	80
Red	RR	Oakwood	89
Orange	OO	Ashwood	75
Yellow	YY	Oakwood	64

Example of the IN Predicate

```
select * from pupil
where suburb in
(select suburb from pupil
where surname = 'Brown') ;
```

Surname	Initials	Suburb	Year_mark
Brown	BB	Elmwood	87
White	WW	Elmwood	59
Blue	BB	Elmwood	76

Predicates 5: The Any/All Predicate

- Bentuk umum:

  any/all operator <(subquery)>

- Operator yang bisa digunakan:
  =any, =all
  !=any, !=all
  <any, <all
  >any, >all
  <=any, <=all
  >=any, >=all

- Operator any (subquery) bernilai benar jika satu atau lebih nilai hasil subquery menyebabkannya bernilai benar

- Operator all (subquery) bernilai benar hanya jika semua nilai hasil subquery menyebabkannya bernilai benar

Predicates 6 - The EXISTS Predicate

- Bentuk umum:

  [NOT] EXISTS (subquery)

Bernilai benar jika hasil subquery tidak kosong

```
select surname, initials
from student
where exists
(select *
from course
where coursecode = student.course_code
and faculty = 'FTP');
```

Akan menampilkan daftar surname dan initial semua siswa yang mengukuti mata kuliah dalam FTP

Predicates 7 - The IS NULL Predicate

Bentuk umum:

IS [NOT] NULL

Predicate ini diperlukan karena operator = tidak bisa digunakan untuk menguji nilai yang bernilai null

The Set Functions

- Fungsi yang beroperasi dalam seluruh nilai kolom
- Biasanya disebut built-in function
- Format: set_function([distinct | all]exspression)
- Set functions adalah count, sum, avg, max, min
- Distinct, mengeliminasi duplikasi nilai dalam fungsi
- All, mengakomodasi seluruh nilai
- Distinct tidak perlu digunakan dalam fungsi min dan max

MAX( ) and MIN( )

```
select max(year_mark)
from student
where suburb = 'OAKWOOD';
```

dan

```
select min(year_mark)
from student
where suburb = 'OAKWOOD';
```

Akan menampilkan nilai tertinggi dan terendah bagi siswa yang tinggal di OAKWOOD

The COUNT( ) Function

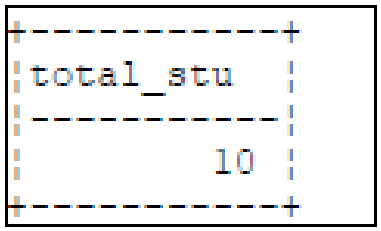
Fungsi count menampilkan jumlah entry dalam sebuah kolom, setiap kolom atau * bisa digunakan

```
select count(*)
from pupil
where suburb = 'OAKWOOD' ;
```

Query diatas akan menampilkan jumlah dari baris dimana kolom suburb memiliki nilai yang disyaratkan

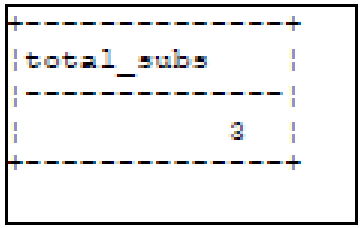
The Effect of DISTINCT

select
count(suburb) as total_stu
from pupil ;



total_stu
10

Select count(distinct suburb) as total_subs
from pupil ;



total_subs
3

Total_stu (jumlah siswa) dan total_subs (jumlah suburb) adalah contoh tampilan nama kolom baru

<expression> as <assigned column name>

AVG( ) with, and without, Assigned Column Names

Select avg(year_mark)
as avg_mark from pupil ;

```
+-----------+
|avg_mark   |
|-----------|
|     73.100|
+-----------+
```

select avg(year_mark)
from pupil ;

```
+-----------+
|sql1       |
|-----------|
|     73.100|
+-----------+
```

AVG( ) Used in a Subquery

```
select * from pupil
where year_mark > (select avg(year_mark)
from pupil) ;
```

Surname	Initials	Suburb	Year_mark
Black	BB	Oakwood	80
Blue	BB	Elmwood	76
Brown	BB	Elmwood	87
Orange	OO	Ashwood	75
Red	RR	Oakwood	89
Violet	VV	Ashwood	77

User:
tampilkan siswa yang nilainya lebih baik dari rata-rata kelas

Add a RESTRICTION Condition in both the query and the subquery . . .

```
select * from pupil
where suburb = 'OAKWOOD'
and year_mark >
(select avg(year_mark)
from pupil
where suburb = 'OAKWOOD');
```

Surname	Initials	Suburb	Year_mark
Black	BB	Oakwood	80
Red	RR	Oakwood	89

User:
tampilkan siswa dari OAKWOOD yang nilainya lebih baik dari rata-rata seluruh siswa dari OAKWOOD

A Computed Column:
When the built-in functions are not enough

select surname, (year_mark * 100 / 120) as percent_mark
from pupil ;

Surname	Percent_mark
Green	46
Brown	72
Gold	56
White	49
Violet	64
Blue	63
Black	66
Red	74
Orange	62
Yellow	53

User:
Jika nilai maksimal adalah 120, nyatakan nilai siswa dalam persentase

Joining as Table to Itself - correlation names required.

```
select *
from pupil p1, pupil p2
where p1.suburb = p2.suburb
and p1.surname != p2.surname
and p1.year_mark > 80 ;
```

Surname	Initials	Suburb	Year_m	Surname	Initials	Suburb	Year_m
Brown	BB	Elmwood	87	White	WW	Elmwood	59
Brown	BB	Elmwood	87	Blue	BB	Elmwood	76
Red	RR	Oakwood	89	Green	GG	Oakwood	56
Red	RR	Oakwood	89	Black	BB	Oakwood	80
Red	RR	Oakwood	89	Yellow	YY	Oakwood	64

User:
Tampilkan dalam pasangan siswa dari suburb yang sama. Dalam setiap pasangan harus ada siswa dengan nilai lebih dari 80

Consider the TEACHER Table

select * from teacher ;

Fam_name	Givname	Suburb	Quals
Rose	Rose	Elmwood	BA
Camelia	Camelia	Oakwood	BSc
Liliy	Liliy	Ashwood	BBus
Daisy	Daisy	Oakwood	BBus
Poppy	Poppy	Ashwood	BA
Violet	Violet	Ashwood	BSc

The PUPIL and TEACHER tables are not union-compatible, but vertical subsets from each are . . .

```
select fam_name, suburb
from teacher
union
select surname, suburb
from pupil ;
```

Tabel pupil berisi 10 baris, teacher 6 baris. Tabel yang dihasilkan dari operasi UNION berisi 15 baris karena ada seorang guru dan siswa yang sama-sama bernama VIOLET dan tinggal di ASHWOOD

Fam_name	Suburb
Black	OAKWOOD
Blue	ELMWOOD
Brown	ELMWOOD
Camelia	OAKWOOD
Daisy	OAKWOOD
Gold	ASHWOOD
Green	OAKWOOD
Lily	ASHWOOD
Orange	ASHWOOD
Poppy	ASHWOOD
Red	OAKWOOD
Rose	ELMWOOD
Violet	ASHWOOD
White	ELMWOOD
yellow	OAKWOOD

SQL queries need to be adequately tested, just like any other program . . .

```
select fam_name, suburb
from teacher
where suburb = 'ASHWOOD'
union
select surname, suburb
from pupil
where suburb = 'ASHWOOD';
```

Fam_name	Suburb
Gold	ASHWOOD
Lily	ASHWOOD
Orange	ASHWOOD
Poppy	ASHWOOD
Violet	ASHWOOD

User:
Tampilkan semua guru dan murid yang tinggal di ASHWOOD.

Operasi UNION yang menghilangkan duplikasi, tidak menampilkan siswa maupun guru bernama VIOLET yang tinggal di ASHWOOD