

SISTEM MANAJEMEN BASIS DATA

03. REVIEW DDL

04. Review DML

05. Function

07. View dan User Authorisation

08. Perulangan dan Keputusan

10. Stored Procedure

Doni Abdul Fatah

github.com/donifaft

Universitas Trunojoyo Madura

Pokok Bahasan

01. Overview SMBD- Entity Diagram

02. Tipe & Model Data

03. Review DDL

04. Review DML

05. Function

06. Transactional SQL

07. View dan User Authorisation

08. Perulangan dan Keputusan

09. Trigger

10. Stored Procedure

11. System Catalog

12. Embedded SQL

13. Basis Data NoSQL

14. UAS

01. SMBD

- 1) Review DDL
- 2) Review DML
- 3) Function
- 4) View dan User Authorisation
- 5) Stored Procedure
- 6) Perulangan dan Keputusan
- 7) Kontrak Perkuliahan
- 8) Kebutuhan Software
- 9) Contact
- 10) Referensi

4) Review DDL

1. Struktur SQL
2. CREATE/DROP basisdata
3. CREATE tabel
4. DROP tabel
5. RENAME tabel
6. ALTER tabel

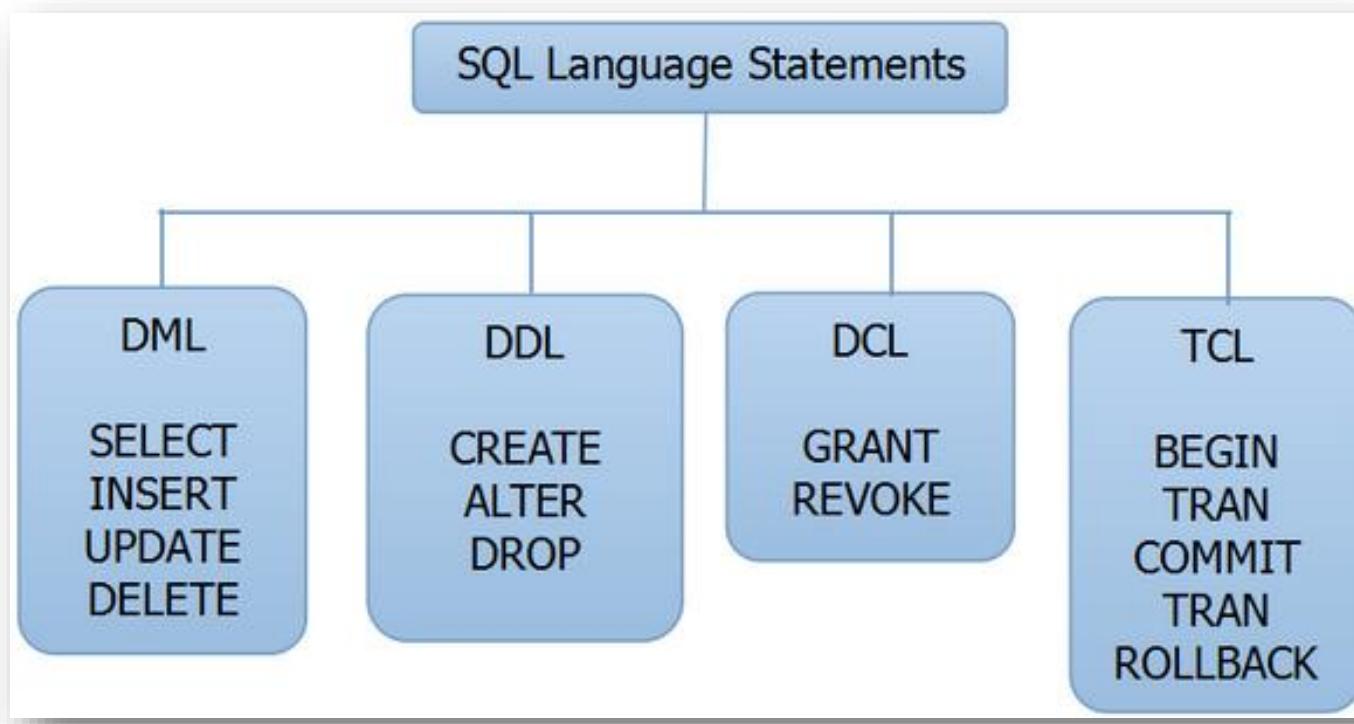
SQL - Structured Language Query

Adalah bahasa standar yang digunakan untuk **memanipulasi basisdata relasional**

- Terdiri dari:
 - **Data Definition Language (DDL):**
 - CREATE tables, indexes, views, Establish primary / foreign keys, DROP / ALTER tables etc
 - **Data Manipulation Language (DML):**
 - INSERT / UPDATE / DELETE, SELECT etc.
 - **Data Control Language (DCL):**
 - COMMIT / ROLLBACK work, GRANT / REVOKE etc

SQL

b) DDL DML

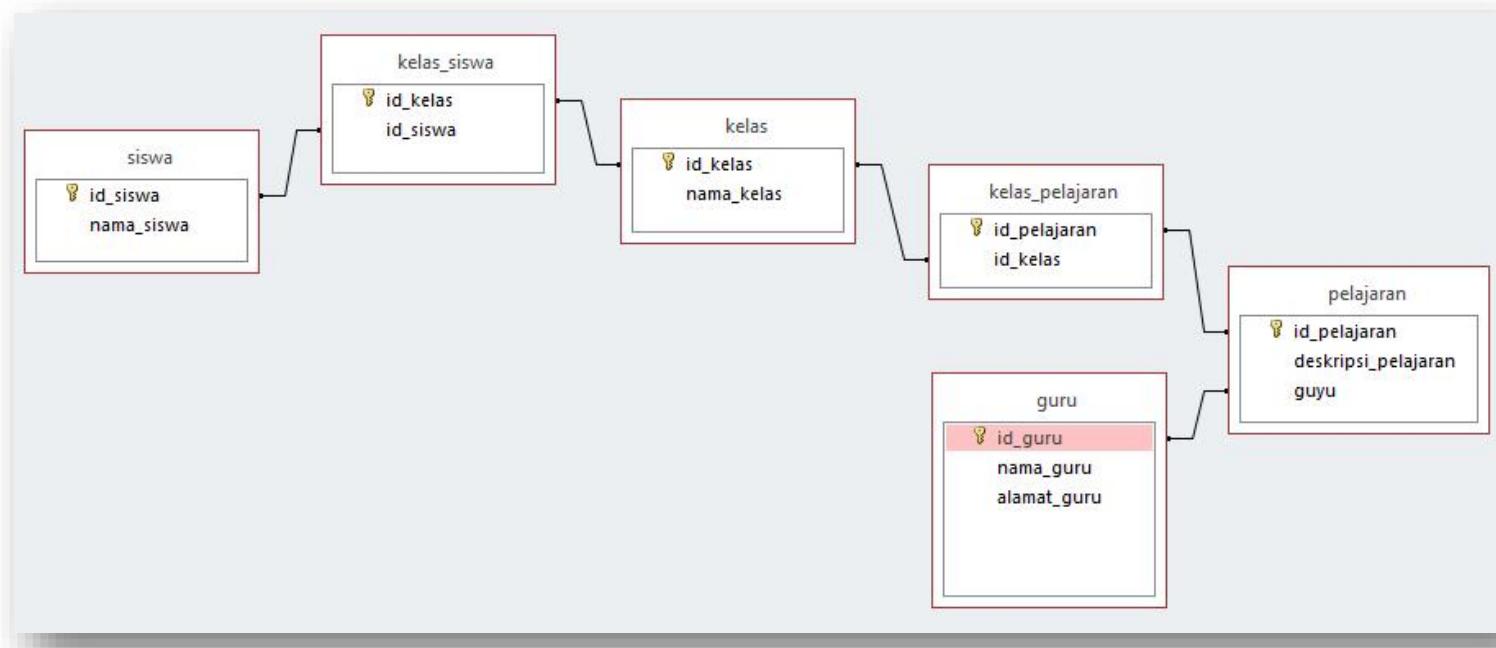


- Data Definition Language (DDL)
- Data Manipulation Language (DML)

SQL

b) DDL

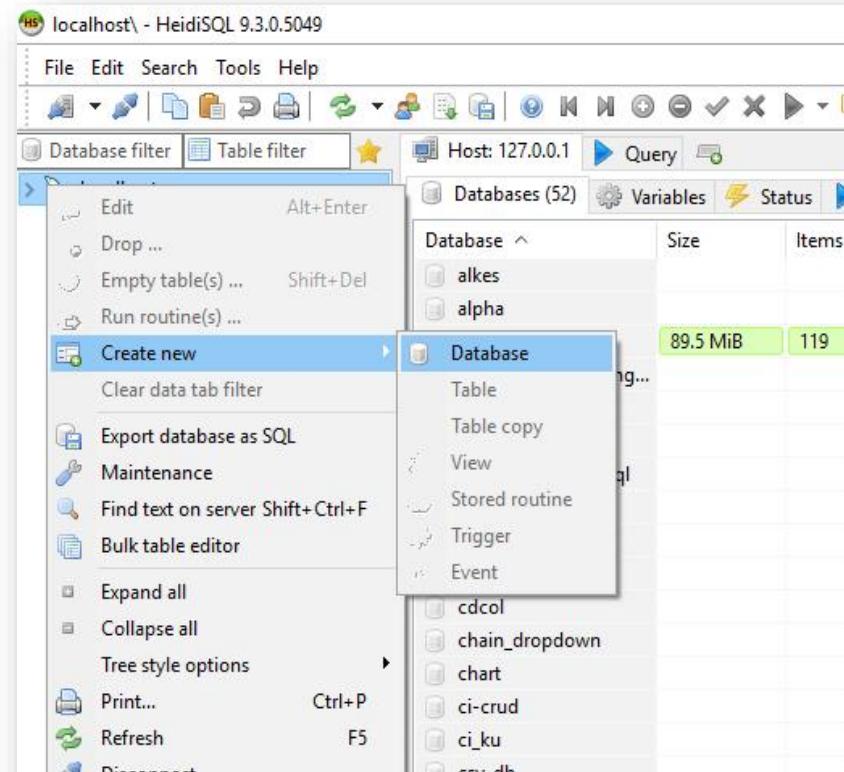
- Pengelolaan pembuatan database dan tabel.
- Dengan berdasar pada relasi table seperti gambar berikut, (nama database = **sekolah**)



SQL

b) DDL Script – buat database

GUI



CLI

```
CREATE DATABASE  
`sekolah`;
```

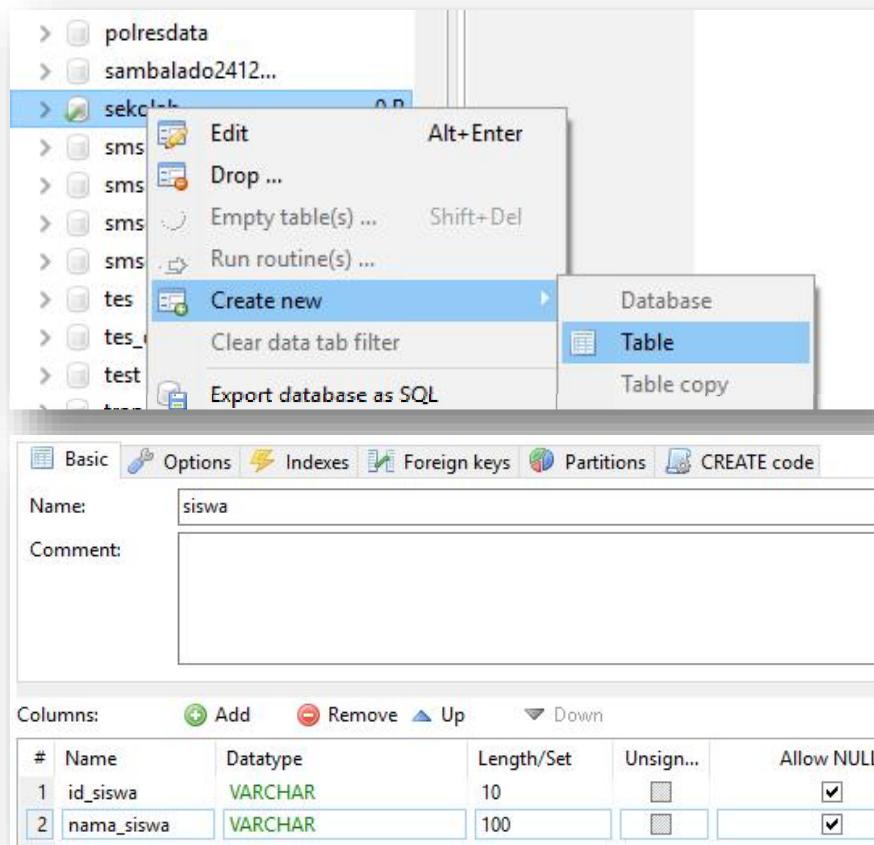
```
USE `sekolah`;
```

```
SHOW TABLES;
```

SQL

b) DDL Script - buat tabel

GUI



CLI

```
CREATE TABLE `siswa` (
  `id_siswa` VARCHAR(10)
NULL,
  `nama_siswa` VARCHAR(100)
NULL
);
```

```
SHOW TABLES;
```

Uji Coba - *CREATE/DROP*basisdata

- **CREATE DATABASE db_name**

Contoh:

- CREATE DATABASE tennis
- USE tennis
- CREATE DATABASE president
- USE president

- **DROP DATABASE db_name**

Contoh:

- DROP DATABASE tennis
- DROP DATABASE president

Uji Coba - *CREATE*tabel

```
CREATE TABLE tbl_name (  
    column_name data_type [DEFAULT  
expr]  
    [column_constraint] , ...  
    [table_constraint] );
```

Uji Coba - *CREATE*tabel (contd-1)

Contoh: Buat tabel Players!

Statement SQL:

```
CREATE TABLE PLAYERS  
  (PLAYERNO SMALLINT NOT NULL,  
   NAME CHAR(15) NOT NULL,  
   INITIALS CHAR(3) NOT NULL,  
   BIRTH_DATE DATE ,  
   SEX CHAR(1) NOT NULL,  
   JOINED SMALLINT NOT NULL,  
   STREET CHAR(15) NOT NULL,  
   HOUSENO CHAR(4) ,  
   POSTCODE CHAR(6) ,  
   TOWN CHAR(10) NOT NULL,  
   PHONENO CHAR(10) ,  
   LEAGUENO CHAR(4) ,  
   PRIMARY KEY (PLAYERNO));
```

Uji Coba - *CREATE*tabel (contd-2)

Contoh: Buat tabel Committee_Members!

Statement SQL:

```
CREATE TABLE COMMITTEE_MEMBERS  
  (PLAYERNO SMALLINT NOT NULL,  
   BEGIN_DATE DATE NOT NULL,  
   END_DATE DATE ,  
   POSITION CHAR(20) ,  
   PRIMARY KEY (PLAYERNO, BEGIN_DATE) );
```

Uji Coba - Skema Tabel (I)

PLAYERS

PLAYERNO (PLAYERNO) NOT NULL	NAME (NAME) NOT NULL	INITIALS (INITIALS) NOT NULL	BIRTH_DATE (DATE)	SEX (SEXCODE) NOT NULL	JOINED (DATE) NOT NULL
------------------------------------	----------------------------	------------------------------------	----------------------	------------------------------	------------------------------

<—PK—>

STREET (STREETNAME) NOT NULL	HOUSENO (HOUSENO)	POSTCODE (POSTCODE)	TOWN (TOWNNAME) NOT NULL	PHONENO (PHONENO)	LEAGUENO (LEAGUENO)
------------------------------------	----------------------	------------------------	--------------------------------	----------------------	------------------------

<—————>

TEAMS

TEAMNO (TEAMNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	DIVISION (DIVISIONNAME) NOT NULL
--------------------------------	------------------------------------	----------------------------------------

<—PK—> <—————>

Uji Coba - Skema Tabel (2)

MATCHES

MATCHENO (MATCHENO) NOT NULL	TEAMNO (TEAMNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	WON (NR_OF_SETS) NOT NULL	LOST (NR_OF_SETS) NOT NULL
------------------------------------	--------------------------------	------------------------------------	---------------------------------	----------------------------------

<—PK—>

PENALTIES

PAYMENTNO (PAYMENTNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	PAYMENT_DATE (DATE) NOT NULL	AMOUNT (AMOUNT) NOT NULL
--------------------------------------	------------------------------------	------------------------------------	--------------------------------

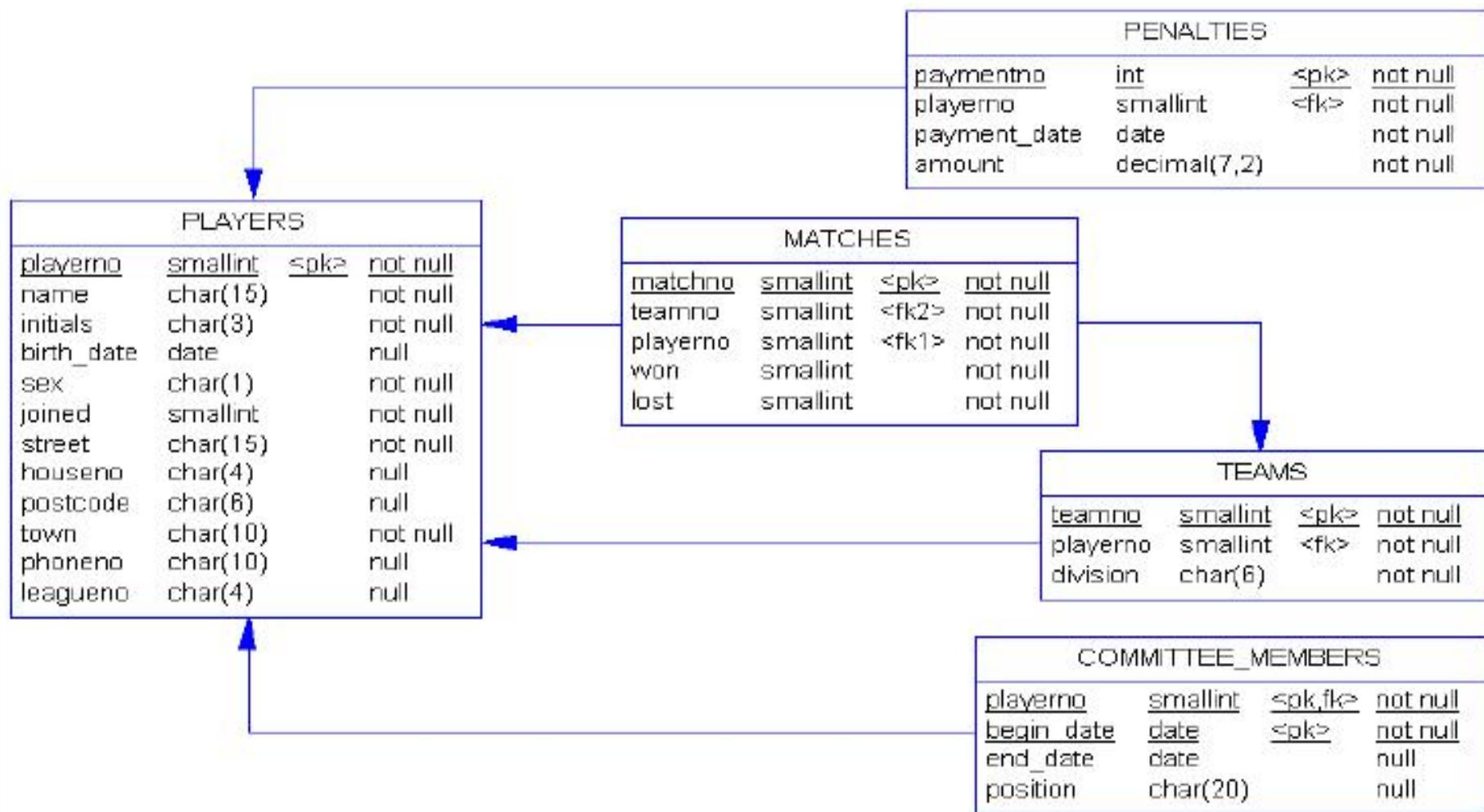
<—PK—>

COMMITTEE_MEMBERS

PLAYERNO (PLAYERNO) NOT NULL	BEGIN_DATE (DATE) NOT NULL	END_DATE (DATE)	POSITION (POSITIONNAME) NOT NULL
------------------------------------	----------------------------------	--------------------	----------------------------------------

<—PK—>

Uji Coba - Skema Basisdata



Uji Coba - Domains & domain constraints

- **PLAYERNO** = NUMERIC (4,0) values
 ≥ 1
- **NAME** = CHARACTER (15)
- **INITIALS** = CHARACTER (3)
- **DATE** = DATE
- **YEARNO** = NUMERIC (4,0)
- **SEXCODE** = CHARACTER (1) set of values = {'M','F'}
- **STREETNAME** = CHARACTER (15)
- **HOUSENO** = CHARACTER (4)
- **POSTCODE** = CHARACTER (6)
- **TOWNNAME** = CHARACTER (10)
- **PHONENO** = CHARACTER (10)
- **LEAGUENO** = CHARACTER (4)
- **TEAMNO** = NUMERIC (2,0) values
 ≥ 1
- **DIVISIONNAME** = CHARACTER (6)
- **MATCHENO** = NUMERIC (4,0) values
 ≥ 1
- **NR_OF_SETS** = NUMERIC (1,0) set of values = {0,1,2,3}
- **PAYMENTNO** = NUMERIC (8,0) values ≥ 1
- **AMOUNT** = NUMERIC (7,2)
- **POSITIONNAME** = CHARACTER (20) set of values = { 'Chairman','Secretary','Treasurer','General member'}

Uji Coba - *Intra table constraints*

Primary keys:

Indicated in the table schema's: <---PK--->

Note: Unique and all columns NOT NULL

Alternative keys:

Indicated in the table schema's:<----->

Note: Unique, not necessary NOT NULL

NOT NULL constraints:

Indicated in the table schema's:NOT NULL

Column value constraints:

PLAYERS (JOINED) values \geq 1970

PENALTIES (PAYMENT_DATE) values \geq '1970-01-01'

PENALTIES (AMOUNT) values $>$ 0.00

COMMITTEE_MEMBERS (BEGIN_DATE) values $>$ '1990-01-01'

Row constraints:

PLAYERS: YEAR (BIRTH_DATE) \leq JOINED

COMMITTEE_MEMBERS: END_DATE \geq BEGIN_DATE

Other intra table constraints:

Note: Not present in this database.

Uji Coba - Inter table constraints

- *Foreign key references:*

TEAMS (PLAYERNO) ---> PLAYERS (PLAYERNO)

MATCHES (TEAMNO) ---> TEAMS (TEAMNO)

MATCHES (PLAYERNO) ---> PLAYERS (PLAYERNO)

PENALTIES (PLAYERNO) ---> PLAYERS (PLAYERNO)

COMMITTEE_MEMBERS (PLAYERNO) ---> PLAYERS
(PLAYERNO)

- *Other inter table constraints:*

For rows of table PLAYERS that are referenced by a foreign key from tables TEAMS, MATCHES and PENALTIES: LEAGUENO IS NOT NULL

For table PENALTIES the value of PAYMENT_DATE must be greater or

equal than the value of JOINED in table PLAYERS for the same player.

Uji Coba - CREATE Tabel (contd-3)

- Contoh: Buat tabel Players!
- Statemen SQL:

```
CREATE TABLE PLAYERS
(PLAYERNO      SMALLINT      NOT NULL,
 NAME          CHAR(15)      NOT NULL,
 INITIALS       CHAR(3)      NOT NULL,
 BIRTH_DATE    DATE         ,
 SEX           CHAR(1)      NOT NULL,
 JOINED        SMALLINT      NOT NULL,
 STREET         CHAR(15)      NOT NULL,
 HOUSENO        CHAR(4)      ,
 POSTCODE       CHAR(6)      ,
 TOWN          CHAR(10)      NOT NULL,
 PHONENO        CHAR(10)      ,
 LEAGUENO        CHAR(4)      ,
 PRIMARY KEY   (PLAYERNO),
 CHECK (PLAYERNO >= 1),
 CHECK (SEX IN ('M', 'F')),
 CHECK (JOINED >= 1970),
 CHECK (YEAR (BIRTH_DATE) <= JOINED);
```

Uji Coba - CREATE Tabel (contd-4)

Contoh: Buat tabel TEAMS!

Statemen SQL:

```
CREATE TABLE TEAMS  
  (TEAMNO SMALLINT NOT NULL,  
   PLAYERNO SMALLINT NOT NULL,  
   DIVISION CHAR(6) NOT NULL,  
   PRIMARY KEY (TEAMNO),  
   UNIQUE (PLAYERNO),  
   FOREIGN KEY (PLAYERNO)  
     REFERENCES PLAYERS (PLAYERNO),  
   CHECK (TEAMNO >= 1);
```

Uji Coba - CREATE Tabel (contd-5)

Contoh: Buat tabel Committee_Members!

Statemen SQL:

```
CREATE TABLE COMMITTEE_MEMBERS  
  (PLAYERNO SMALLINT NOT NULL,  
   BEGIN_DATE DATE NOT NULL,  
   END_DATE DATE ,  
   POSITION CHAR(20) ,  
   PRIMARY KEY (PLAYERNO, BEGIN_DATE) ,  
   FOREIGN KEY (PLAYERNO)  
     REFERENCES PLAYERS (PLAYERNO) ,  
   CHECK (POSITION IN ('Chairman', 'Secretary',  
'Treasurer', 'General member')),  
   CHECK (BEGIN_DATE >= '1990-01-01'),  
   CHECK (END_DATE >= BEGIN_DATE) );
```

Uji Coba - CREATE Tabel (contd-6)

Contoh: Buat tabel TEAMS!

Statemen SQL:

```
CREATE TABLE TEAMS  
  (TEAMNO SMALLINT NOT NULL PRIMARY KEY  
    CHECK (TEAMNO >= 1),  
    PLAYERNO SMALLINT NOT NULL UNIQUE,  
    FOREIGN KEY (PLAYERNO)  
    REFERENCES PLAYERS (PLAYERNO),  
    DIVISION CHAR(6) NOT NULL);
```

Uji Coba - Drop Table

DROP TABLE tbl_name [,tbl_name]

Contoh: Drop seluruh tabel dalam basisdata tennis!

Statemen SQL:

```
DROP TABLE COMMITTEE_MEMBERS;  
DROP TABLE PENALTIES;  
DROP TABLE MATCHES;  
DROP TABLE TEAMS;  
DROP TABLE PLAYERS;
```

Uji Coba - RENAME Tabel

- **RENAME TABLE tbl_name TO new_tbl_name**
[**,tbl_name2 TO new_tbl_name2**] ...

Contoh: Ubah nama tabel Teams menjadi Groups!

Statemen SQL:

RENAME TABLE Teams TO Groups

RENAME tabel

Uji Coba - ALTER Tabel (contd-1)

- **ALTER TABLE *tbl_name***
alter_specification [, alter_specification] ...

alter_specification:

- ADD**
- CHANGE & MODIFY**
- DROP**
- RENAME**

ALTER tabel

Uji Coba - ALTER Tabel ADD

**ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name]
ADD [COLUMN] (col_name column_definition,...)**

Contoh: Tambahkan satu kolom bernama TYPE di dalam tabel TEAMS.
Kolom TYPE ini untuk memberikan identifikasi tim Pria dan Wanita.

Statemen SQL:

```
ALTER TABLE TEAMS  
ADD COLUMN TYPE CHAR (1);
```

Contoh: Dengan adanya kolom TYPE, maka (misalkan) tim 2 sebagai tim Pria harus di-update.

Statemen:

```
UPDATE TEAMS  
SET TYPE = 'M'  
WHERE TEAMNO = 2;
```

ALTER tabel : ADD

Uji Coba - ALTER Tabel CHANGE & MODIFY

**CHANGE [COLUMN] old_col_name new_col_name
column_definition [FIRST|AFTER col_name]**

**MODIFY [COLUMN] col_name column_definition
[FIRST | AFTER col_name]**

Contoh: Tambahkan panjang karakter kolom TOWN
dari 10 menjadi 20!

**ALTER TABLE PLAYERS
MODIFY COLUMN TOWN CHAR (20);**

ALTER tabel : CHANGE & MODIFY

Latihan - DROP

DROP [COLUMN] col_name

DROP PRIMARY KEY

DROP FOREIGN KEY fk_symbol

Contoh: Hapuskan kolom TYPE dari tabel TEAMS!

Statemen:

ALTER TABLE TEAMS

DROP COLUMN TYPE

Uji Coba - ALTER Tabel RENAME

RENAME [TO] new_tbl_name

Contoh: Ubah nama tabel Teams menjadi Groups!

Statemen SQL:

ALTER TABLE Teams RENAME Groups

ALTERtabel : RENAME

DML (DATA MANIPULATION LANGUAGE)

- 1) *INSERT* data ke dalam tabel
- 2) *SELECT*(dari 1 tabel)
- 3) *SELECT*(lebih dari 1 tabel)
- 4) *UPDATE* data dalam tabel
- 5) *DELETE* data dalam tabel

SQL

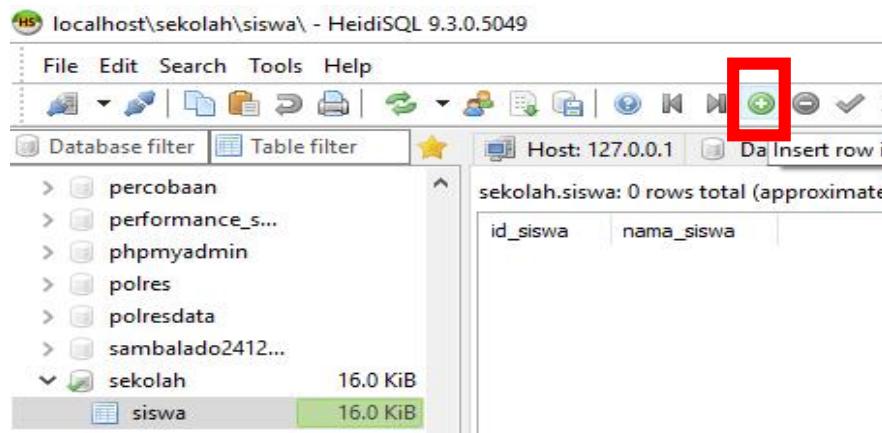
c) DML

- ❑ Pengelolaan **data** dalam tabel.
- ❑ Bentuk **CRUD**:
 1. **Create**
 2. **Read** (Max, Min, Sum, dll)
 3. **Update**
 4. **Delete**

SQL

c) DML Script - Create

GUI



This screenshot shows the result of the insertion. The HeidiSQL interface is identical to the previous one, but now it displays '1 rows total (approximately)'. The table view for 'siswa' shows the newly inserted row: 'id_siswa' is '2016111234' and 'nama_siswa' is 'Furi hikmawati'. The status bar at the bottom indicates 'Post (Ctrl+Enter)'.

id_siswa	nama_siswa
2016111234	Furi hikmawati

CLI

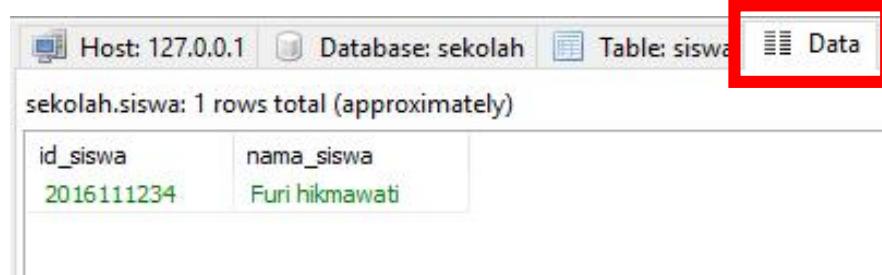
INSERT INTO

```
 `sekolah`.`siswa`  
(`id_siswa`,  
 `nama_siswa`) VALUES  
('2016111234', 'Furi  
Hikmawati');
```

SQL

c) DML Script - Read

GUI



A screenshot of the MySQL Workbench interface. At the top, there are tabs for Host: 127.0.0.1, Database: sekolah, Table: siswa, and Data. The Data tab is highlighted with a red box. Below the tabs, a message says "sekolah.siswa: 1 rows total (approximately)". A table below shows one row of data:

id_siswa	nama_siswa
2016111234	Furi hikmawati

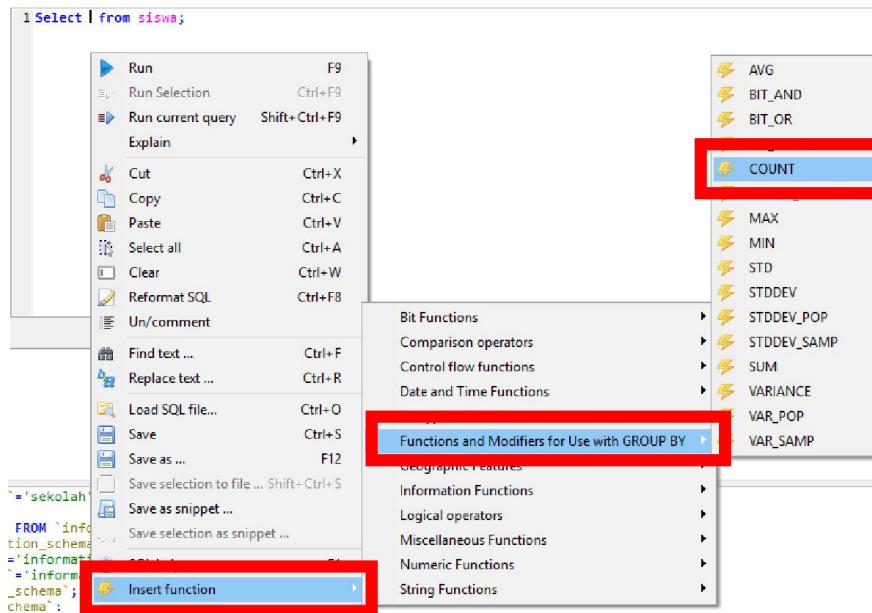
CLI

```
SELECT * FROM  
`sekolah`.`siswa`;
```

SQL

c) DML Script - Read (count)

GUI



CLI

```
Select COUNT(*) from  
siswa;
```

SQL

c) DML Script – Read (count)

Statement SELECT

Cari nomor pemain yang telah melakukan setidaknya dua kali penalti yang jumlahnya lebih dari \$25! Urutkan hasil berdasarkan nomor pemainnya!

Query:

```
SELECT PLAYERNO FROM PENALTIES WHERE AMOUNT >  
25 GROUP BY PLAYERNO HAVING COUNT (*) > 1 ORDER  
BY PLAYERNO;
```

SQL

c) DML Script – Read (count)

Query:

```
SELECT          PLAYERO NO  
FROM           PENALTIES  
WHERE          AMOUNT > 25  
GROUP BY       PLAYERO NO  
HAVING         COUNT (*) > 1  
ORDER BY       PLAYERO NO;
```

FROM

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
3	27	1983-09-10	100.00
4	104	1984-12-08	50.00
5	44	1980-12-08	25.00
6	8	1980-12-08	25.00
7	44	1982-12-30	30.00

WHERE

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
3	27	1983-09-10	100.00
4	104	1984-12-08	50.00
7	44	1982-12-30	30.00
8	27	1984-11-12	75.00

GROUP BY

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
7	44	1982-12-30	30.00
3	27	1983-09-10	100.00
8	27	1984-11-12	75.00
4	104	1984-12-08	50.00

HAVING

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
2	44	1981-05-05	75.00
7	44	1982-12-30	30.00
3	27	1983-09-10	100.00
8	27	1984-11-12	75.00

SELECT

PLAYERO NO
44
27

ORDER BY

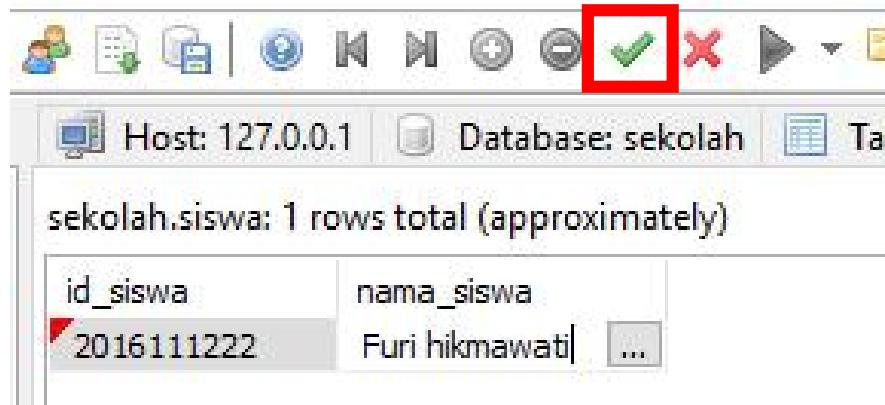
PLAYERO NO
27
44

A
G

SQL

c) DML Script - Update

GUI



CLI

```
UPDATE `sekolah`.`siswa`
SET
`id_siswa`='2016111222'
WHERE
`id_siswa`='2016111234'
AND `nama_siswa`='Furi
hikmawati' LIMIT 1;
```

SQL

c) DML Script - Delete

GUI



CLI

DELETE FROM

```
`sekolah`.`siswa` WHERE
`id_siswa`='2016111222'
AND `nama_siswa`='Furi
hikmawati' LIMIT 1;
```

Start MySQL

```
C:\Users\Smart>
```

```
cd c:\xampp\mysql\bin\
```

```
c:\xampp\mysql\bin>
```

```
mysql -u root -p
```

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Microsoft Windows
[Version 6.1.7600]

```
MariaDB [(none)]> show databases;
```

Database
fifin
information_schema
mysql
performance_schema
phpmyadmin
test

6 rows in set (0.11 sec)

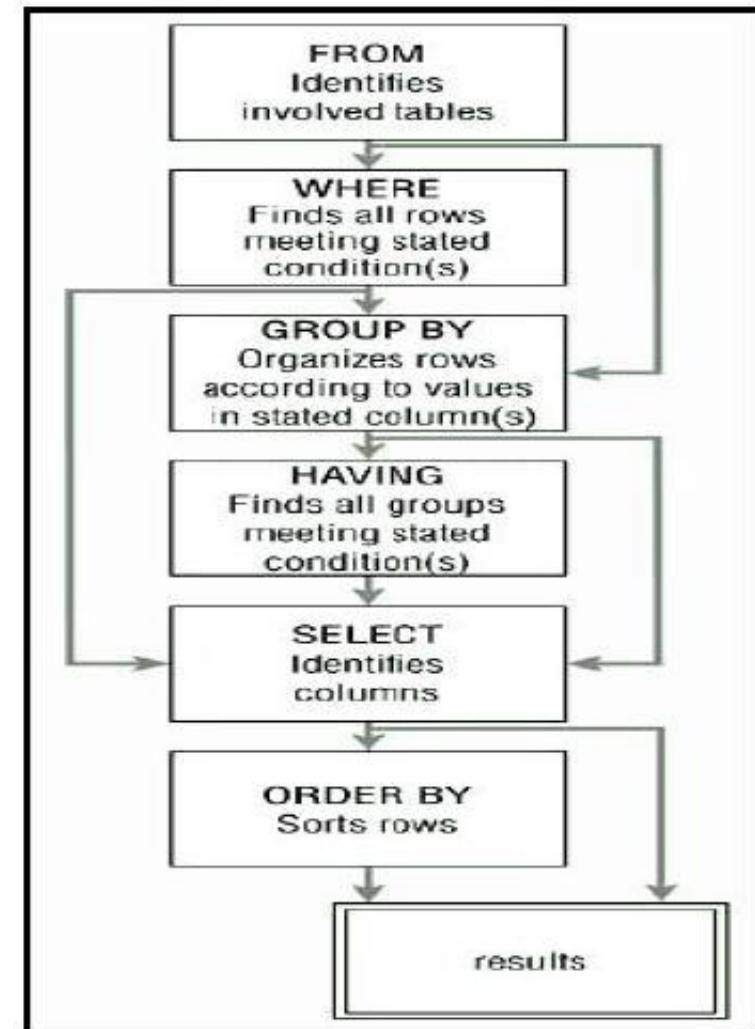
```
MariaDB [(none)]> use fifin;
```

Database changed

```
MariaDB [fifin]>
```

Data Manipulation Language (DML)

- *INSERT / UPDATE / DELETE, SELECT*
- Klausula-klausula *SELECT*:
 - **FROM** --> menentukan tabel(-tabel) sumber
 - **WHERE** --> memilih baris(-baris) yang memenuhi kondisi (-kondisi)
 - **GROUP BY** --> menggabungkan baris(-baris) yang kolomnya memiliki nilai yang sama
 - **HAVING** --> memilih grup yang memenuhi kondisi tertentu
 - **SELECT** --> memilih kolom(-kolom)
 - **ORDER BY** --> mengurutkan baris-baris berdasarkan nilai-nilai yang ada dalam kolom(-kolom)



DML: Insert data ke dalam Tabel

1. INSERT satu baris

INSERT INTO <table specification>

[<column list>]

VALUES (<expression> [{,<expression>}...])

2. INSERT dari tabel lain (Mengisi suatu tabel dari isi tabel lain)

INSERT INTO<table specification>

[<column list>]

<select clause>

<from clause>

[<where clause>]

[<group by clause>]

[<having clause>]

DML: Insert data ke dalam Tabel (contd -2)

Contoh 1: Sebuah tim baru dengan pemain nomor 100 sebagai kaptennya telah bergabung di dalam liga. Tim ketiga ini akan bermain di dalam divisi ketiga.

Query:

```
INSERT INTO TEAMS (TEAMNO,  
PLAYERNO, DIVISION) VALUES  
(3,100,'third');
```

atau:

```
INSERT INTO TEAMS VALUES  
(81,100,'third');
```

DML: Insert data dari Tabel lain (contd -3)

- **Contoh 1.1:** Buatlah suatu tabel baru yang mencatat nomor pemain, nama, kota dan nomor telepon dari masing-masing pemain yang tidak memiliki nomor liga (tidak pernah bermain di dalam liga)!

Query:

- Membuat tabel baru:

```
CREATE TABLE RECR_PLAYERS (PLAYERNO  SMALLINT NOT NULL,  
NAME CHAR(15) NOT NULL, TOWN CHAR(10) NOT NULL, PHONENO  
CHAR(10), PRIMARY KEY (PLAYERNO) )
```

- Insert data ke tabel RECR_PLAYERS dari tabel PLAYERS:

```
INSERT INTO RECR_PLAYERS  
SELECT PLAYERNO, NAME, TOWN, PHONENO  
FROM PLAYERS WHERE LEAGUENO IS NULL
```

KLAUSA : SELECT (dari 1 tabel)

Contoh 2: Tampilkan seluruh data penalti.

Query:

```
SELECT * FROM Penalties;
```

Contoh 3: Tampilkan nomor pembayaran, nomor pemain dan jumlah dari masing-masing penalti

Query:

```
SELECT paymentno, playerno,  
amount FROM Penalties;
```

Contoh 3.1 : Tampilkan nomor pemain, nama dan umur ketika bergabung di klub tenis dari masing-masing pemain.

Query:

```
SELECT playerno, name,  
joined - Year(birth_date) AS join_age  
FROM Players
```

KLAUSA : SELECT - WHERE (dari 1 tabel)

Contoh 4: Dapatkan nomor pemain dan nomor liga dari masing-masing pemain yang memang benar-benar memiliki nomor liga.

Query:

```
SELECT playerno, leagueno
FROM Players
WHERE leagueno IS NOT NULL;
```

Contoh 4.1 : Dapatkan nomor, nama, jenis kelamin dan tanggal lahir dari masing-masing pemain pria yang lahir setelah tahun 1970.

```
SELECT playerno, name, sex, birth_date
FROM Players
WHERE sex = 'M'
AND Year(birth_date) > 1970
```

KLAUSA : SELECT - BETWEEN (dari 1 tabel)

Contoh 5: Cari nomor dan tanggal lahir dari masing-masing pemain yang lahir antara tahun 1962 sampai 1964.

Query:

```
1 SELECT playerno, birth_date
2 FROM Players
3 WHERE Year(birth_date) BETWEEN 1962 AND 1964
```

playerno	birth_date
6	1964-06-25
7	1963-05-11
8	1962-07-08
27	1964-12-28
28	1963-06-22
44	1963-01-09
95	1963-05-14
100	1963-02-28
112	1963-10-01

KLAUSA : SELECT - IN (dari 1 tabel)

Contoh 6: Cari nomor, nama dan kota dari masing-masing pemain yang tinggal di *Inglewood*, *Plymouth*, *Midhurst* atau

Douglas.

Query dengan OR:

```
SELECT playerno, name, town
FROM Players
WHERE town = 'Inglewood' OR town =
'Plymouth' OR
town = 'Midhurst' OR town = 'Douglas';
```

Query dengan IN:

```
SELECT playerno, name, town
FROM Players
WHERE town IN
('Inglewood', 'Plymouth', 'Midhurst',
'Douglas');
```

KLAUSA : SELECT – LIKE, ORDER BY (dari 1 tabel)

Contoh 7: Dapatkan nama dan nomor dari masing-masing pemain yang memiliki huruf “e” pada posisi huruf sebelum huruf terakhir dari namanya.

Query:

```
SELECT name, playerno  
FROM Players  
WHERE name LIKE '%e_';
```

Contoh 8: Buat daftar nomor pembayaran dan nomor pemain untuk masing-masing penalti; urutkan hasil berdasarkan nomor pemain dan nomor pembayaran untuk masing-masing pemain.

Query:

```
SELECT paymentno, playerno  
FROM Penalties  
ORDER BY playerno, paymentno;
```

Seluruh Klausa dalam 1 Statement

- **Contoh 8.1:** Cari nomor dari masing-masing pemain yang telah melakukan lebih dari 1 penalti yang besarnya lebih dari 25.
- Query:

```
SELECT playerno  
FROM Penalties  
WHERE amount > 25  
GROUP BY playerno  
HAVING COUNT(*) > 1  
ORDER BY playerno
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Count

Contoh 9: Hitung jumlah pemain yang tercatat di dalam tabel PLAYERS!

SQL:

```
SELECT COUNT(*) FROM PLAYERS;
```

Contoh 10: Ada berapa nama kota yang tercatat di dalam kolom TOWN dalam tabel PLAYERS?

SQL:

```
SELECT COUNT(DISTINCT(TOWN)) FROM PLAYERS;
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Max & Min

Contoh 11: Berapakah jumlah penalti yang tertinggi?

SQL:

```
SELECT MAX(AMOUNT) FROM PENALTIES;
```

Contoh 12: Berapakah jumlah penalti yang terendah?

SQL:

```
SELECT MIN(AMOUNT) FROM PENALTIES;
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Sum

Contoh 13: Berapa banyak total SET yang telah dimenangkan, total SET yang telah kalah, dan berapa perbedaan di antara keduanya?

SQL:

```
SELECT SUM(WON),SUM(LOST),SUM(WON)-  
SUM(LOST) AS Difference
```

```
FROM MATCHES;
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Avg

Contoh 14: Berapakah jumlah rata-rata penalti yang dilakukan oleh pemain nomor 44?

SQL:

```
SELECT AVG(AMOUNT)  
FROM PENALTIES  
WHERE PLAYERNO = 44;
```

KLAUSA : SELECT (lebih dari 1 tabel)

Contoh 15: Cari nama dan inisial pemain yang telah bermain di dalam pertandingan minimal sebanyak 1 kali!

Query:

```
SELECT DISTINCT P.NAME, P.INITIALS  
FROM PLAYERS AS P, MATCHES AS M  
WHERE P.PLAYERNO = M.PLAYERNO;
```

KLAUSA : SELECT (lebih dari 1 tabel)

Contoh 16: Cari nomor dan nama pemain yang tinggal di kota yang sama dengan pemain nomor 27!

Query:

SELECT P.PLAYERNO, P.NAME

FROM PLAYERS AS P, PLAYERS AS P27

WHERE P.TOWN = P27.TOWN

AND P27.PLAYERNO = 27

AND P.PLAYERNO <> 27

Operator Union

- UNION = $(A \cup B)$
- Aturan Penggunaan:
 - Klausa SELECT dari seluruh blok select harus memiliki jumlah ekspresi yang sama, dan hasil ekspresi yang akan ditempatkan dalam kolom yang sama dengan hasil ekspresi yang lainnya harus memiliki tipe data yang sama atau masih dapat ditransformasi menjadi tipe data yang sama
 - Klausa ORDER BY hanya dapat diletakkan setelah blok select yang terakhir
 - Di klausa SELECT tidak boleh menggunakan DISTINCT; SQL secara otomatis akan menghapus duplikasi baris yang memiliki nilai yang sama

Operator Union (contd-2)

- Sintaknya :

Union-Distinct

Table1	
column1	column2
a	b
a	c
a	d

Table 2	
column1	column2
b	c
a	d

Table1 Union Table2	
column1	column2
a	b
a	c
a	d
b	c

Duplicate row
not repeated in
results

Union-ALL

Table1	
column1	column2
a	b
a	c
a	d

Table 2	
column1	column2
b	c
a	d

Table1 Union Table2	
column1	column2
a	b
a	c
b	c
a	d
a	d

Duplicate Rows
are Repeated in
Results

SELECT `column1`, `column1` FROM
'table1'
UNION DISTINCT
SELECT `column1`, `column1` FROM
'table2';

Sintaknya :

SELECT `column1`, `column1` FROM
'table1'
UNION ALL
SELECT `column1`, `column1` FROM
'table2';

Operator Union (contd-3)

- Contoh 1: Cari nomor dan tempat tinggal masing-masing pemain yang berasal dari Inglewood and Plymouth! Urutkan berdasarkan nomor pemain!
- Catatan: “berasal dari Inglewood dan Plymouth” berarti “berasal dari Inglewood atau dari Plymouth”.
- Dengan menggunakan operator UNION:

```
SELECT PLAYERO NO , TOWN  
FROM PLAYERS  
WHERE TOWN = 'Inglewood'  
UNION atau tanpa DISTINCT  
SELECT PLAYERO NO , TOWN  
FROM PLAYERS  
WHERE TOWN = 'Plymouth'  
ORDER BY PLAYERO NO ;
```

Dengan menggunakan OR :

```
SELECT PLAYERO NO , TOWN  
FROM PLAYERS  
WHERE TOWN = 'Inglewood'  
OR TOWN = 'Plymouth'  
ORDER BY PLAYERO NO ;
```

PLAYERO NO	TOWN
8	Inglewood
44	Inglewood
112	Plymouth

Operator Union (contd-4)

- Contoh 2: Buatlah daftar dari nomor pemain yang telah melakukan penalti setidaknya sebanyak 1 kali, atau pemain yang menjadi kapten tim atau pemain yang memenuhi kedua kondisi tersebut.

penalties (4x8)				teams (3x4)		
PAYMENTNO	PLAYERNO	PAYMENT_DATE	AMOUNT	TEAMNO	PLAYERNO	DIVISION
1	6	1980-12-08	100,00	1	6	first
2	44	1981-05-05	75,00	2	27	second
3	27	1983-09-10	100,00	3	100	third
4	104	1984-12-08	50,00	81	100	third
5	44	1980-12-08	25,00			
6	8	1980-12-08	25,00			
7	44	1982-12-30	30,00			
8	27	1984-11-12	75,00			

Dengan operator UNION:

```
SELECT PLAYERNO FROM PENALTIES  
UNION  
SELECT PLAYERNO FROM TEAMS;
```

Hasil #1 (1x6)	
PLAYERNO	
6	
8	
27	
44	
104	
100	

Atau dengan operator OR:

```
SELECT PLAYERNO FROM PLAYERS  
WHERE PLAYERNO IN  
    (SELECT PLAYERNO FROM PENALTIES)  
OR PLAYERNO IN  
    (SELECT PLAYERNO FROM TEAMS);
```

Operator Union (contd-5)

- Contoh 3: Buatlah daftarn omor pemain dan nomor tim dari para pemain yang menjadi kapten tim atau pemain yang pernah bermain minimal 1 kali dalam suatu pertandingan atau pemain yang memenuhi kedua kondisi tersebut! Urutkan berdasarkan nomor pemain dan nomor tim.

Dengan operator UNION:

```
SELECT PLAYERO NO, TEAMNO  
FROM TEAMS  
UNION  
SELECT PLAYERO NO, TEAMNO  
FROM MATCHES  
ORDER BY PLAYERO NO, TEAMNO;
```

```
1 SELECT PLAYERO NO, TEAMNO  
2 FROM MATCHES  
3 ORDER BY PLAYERO NO, TEAMNO;
```

matches (2x13)	
PLAYERO NO	TEAMNO
2	1
6	1
6	1
6	1
8	1
8	2
27	2
44	1
57	1
83	1
104	2
112	2
112	2

```
1 SELECT PLAYERO NO, TEAMNO  
2 FROM TEAMS  
3 ORDER BY PLAYERO NO, TEAMNO;
```

teams (2x4)	
PLAYERO NO	TEAMNO
6	1
27	2
100	3
100	81

Hasil #1 (2x12)	
PLAYERO NO	TEAMNO
2	1
6	1
8	1
8	2
27	2
44	1
57	1
83	1
100	3
100	81
104	2
112	2

Bagaimana Kalau dengan Operator OR?

OPERATOR UNION ALL

- UNION ALL = $(A \cup B)$
- Perbedaan dengan OPERATOR UNION :
 - Duplikasi baris yang memiliki nilai yang sama tidak dihapuskan
- Contoh 4: Buatlah daftar dari nomor pemain yang telah melakukan penalti setidaknya sebanyak 1 kali, atau pemain yang menjadi kapten tim atau pemain yang memenuhi kedua kondisi tersebut.

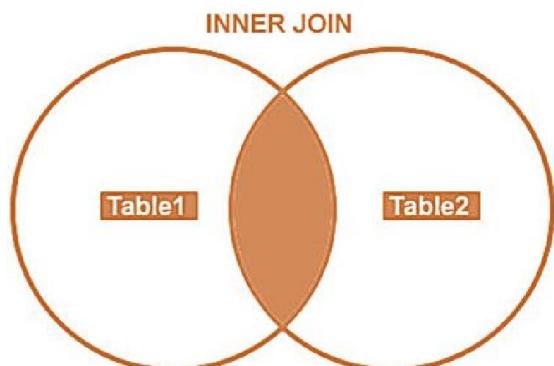
```
SELECT PLAYERO NO  
FROM PENALTIES  
UNION ALL  
SELECT PLAYERO NO  
FROM TEAMS;
```

penalties (4x8)				PLAYERO NO
PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT	
1	6	1980-12-08	100,00	6
2	44	1981-05-05	75,00	8
3	27	1983-09-10	100,00	27
4	104	1984-12-08	50,00	44
5	44	1980-12-08	25,00	44
6	8	1980-12-08	25,00	27
7	44	1982-12-30	30,00	44
8	27	1984-11-12	75,00	104

teams (3x4)				PLAYERO NO
TEAMNO	PLAYERO NO	DIVISION		
1	6	first		6
2	27	second		27
3	100	third		27
81	100	third		100
				100

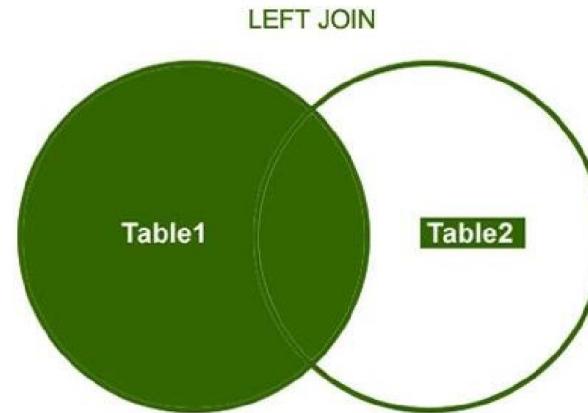
JOIN

Menampilkan semua data sebelah kiri dari table yang di joinkan dan menampilkan data sebelah kanan yang cocok dengan kondisi join.

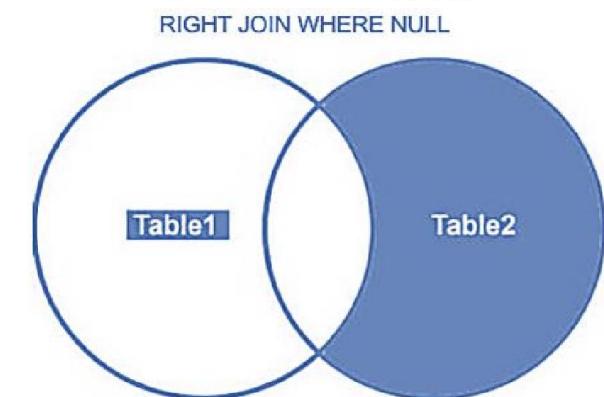
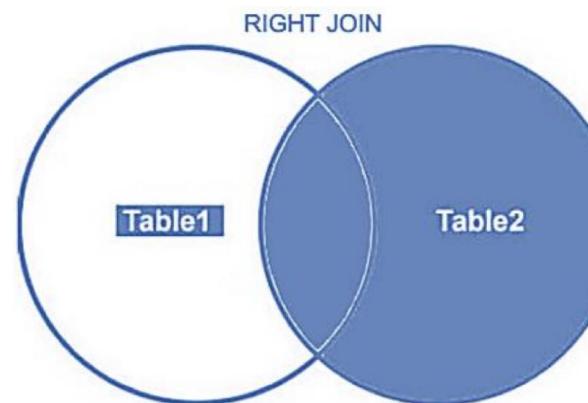
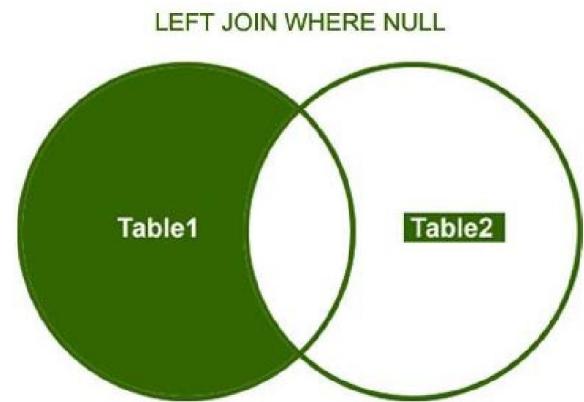


Menampilkan data pada kolom yang sesuai dengan yang dibandingkan

```
SELECT columns FROM TableA  
INNER JOIN TableB  
ON A.columnName =  
B.columnName;
```



```
SELECT columns  
FROM TableA  
LEFT OUTER JOIN TableB  
ON A.columnName = B.columnName  
WHERE B.columnName IS NULL
```



Menampilkan semua data yang ada di table sebelah kanan dan mencari kecocokan key pada table sebelah kiri.

```
SELECT columns  
FROM TableA  
RIGHT OUTER JOIN TableB  
ON A.columnName =  
B.columnName  
WHERE A.columnName IS NULL
```

KLAUSA : SELECT (lebih dari 1 tabel)

Join: Inner

Contoh 17: Tampilkan nomor tim dan nama kapten dari masing-masing tim *Solusi menggunakan “Equi JOIN”*:

Query:

```
SELECT TEAMNO, NAME  
FROM TEAMS, PLAYERS  
WHERE TEAMS.PLAYERNO = PLAYERS.PLAYERNO;
```

Solusi menggunakan INNER JOIN:

Query:

```
SELECT TEAMNO, NAME  
FROM PLAYERS P INNER JOIN TEAMS T  
ON (P.PLAYERNO = T.PLAYERNO);
```

KLAUSA : SELECT (lebih dari 1 tabel)

Join: Outer (Left/Right)

Contoh 18: Untuk masing-masing pemain, buatlah daftar nama dan nomor teleponnya (jika memang terdaftar)! Khusus untuk pemain yang menjadi kapten suatu tim, cantumkan juga nomor timnya dan divisinya.

Solusi menggunakan **LEFT JOIN**:

```
SELECT NAME, PHONENO, TEAMNO, DIVISION  
FROM PLAYERS AS P LEFT JOIN TEAMS AS T  
ON P.PLAYERNO = T.PLAYERNO;
```

Solusi menggunakan **RIGHT JOIN**:

```
SELECT NAME, PHONENO, TEAMNO, DIVISION  
FROM TEAMS AS T RIGHT JOIN PLAYERS AS P  
ON T.PLAYERNO = P.PLAYERNO;
```

KLAUSA : SELECT (lebih dari 1 tabel) (Contd-2)

Join: Outer (Left/Right)

Contoh 18.1: Untuk seluruh pemain, buatlah daftar nomor pemain, total banyaknya penalti yang telah mereka lakukan dan total jumlah penalti yang harus mereka bayar!

Solusi menggunakan **UNION**:

```
SELECT PLAYERNO, COUNT(*) AS NUMBER_of_PENALTIES, SUM(AMOUNT) AS  
TOTAL_PENALTIES FROM PENALTIES GROUP BY PLAYERNO  
UNION  
SELECT PLAYERNO, 0, 0.00 FROM PLAYERS WHERE PLAYERNO NOT IN  
(SELECT PLAYERNO FROM PENALTIES) ORDER BY 2 DESC;
```

Solusi menggunakan **JOIN**:

```
SELECT P.PLAYERNO, COUNT(PAYMENTNO) AS NUMBER_of_PENALTIES,  
IFNULL(SUM(AMOUNT), 0.00) AS TOTAL_PENALTIES  
FROM PLAYERS AS P LEFT OUTER JOIN PENALTIES AS PEN  
ON P.PLAYERNO = PEN.PLAYERNO  
GROUP BY P.PLAYERNO  
ORDER BY 2 DESC;
```

KLAUSA : SELECT (lebih dari 1 tabel) (Contd-3)

Join: Outer (Left/Right)

Solusi menggunakan **UNION**:

Hasil #1 (3x18)		
PLAYERNO	NUMBER_of_PENALTIES	TOTAL_PENALTIES
44	3	130,00
27	2	175,00
6	1	100,00
8	1	25,00
104	1	50,00
7	0	0,00
28	0	0,00
39	0	0,00
57	0	0,00
83	0	0,00
95	0	0,00
2	0	0,00
100	0	0,00
3	0	0,00
112	0	0,00
4	0	0,00
200	0	0,00
5	0	0,00

Solusi menggunakan **JOIN**:

players (3x18)		
PLAYERNO	NUMBER_of_PENALTIES	TOTAL_PENALTIES
44	3	130,00
27	2	175,00
6	1	100,00
104	1	50,00
8	1	25,00
39	0	0,00
2	0	0,00
3	0	0,00
57	0	0,00
4	0	0,00
83	0	0,00
5	0	0,00
95	0	0,00
100	0	0,00
7	0	0,00
112	0	0,00
200	0	0,00
28	0	0,00

Join

```
1 SELECT * FROM TEAMS
```

teams (3x4)

TEAMNO	PLAYERO NO	DIVISION
1	6	first
2	27	second
3	100	third
81	100	third

```
1 SELECT * FROM PLAYERS
```

players (12x18)

PLAYERO NO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13
5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Semarang	0811-1111-	14
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319
200	Sisi	Ss	(NULL)	F	0	(NULL)	(NULL)	(NULL)	Madura	(NULL)	(NULL)

Gabungkan ke Dua Tabel tersebut,
kemudian tampilkan Teamno dan name

Join

```
1 SELECT * FROM TEAMS INNER JOIN PLAYERS;
```

Hasil #1 (15x72)

TEAMNO	PLAYERO NO	DIVISION	TEAMNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOW
1	6	first	2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	St
2	27	second	2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	St
3	100	third	2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	St
81	100	third	2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	St
1	6	first	3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Su
2	27	second	3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Su
3	100	third	3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Su
81	100	third	3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Su
1	6	first	4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bc
2	27	second	4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bc
3	100	third	4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bc
81	100	third	4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bc
1	6	first	5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Se
2	27	second	5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Se
3	100	third	5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Se
81	100	third	5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Se
1	6	first	6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	St
2	27	second	6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	St
3	100	third	6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	St
81	100	third	6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	St

```
SELECT TEAMNO, NAME
FROM TEAMS INNER JOIN PLAYERS
ON (TEAMS.PLAYERNO =
PLAYERS.PLAYERNO);
```

```
1 SELECT TEAMNO, NAME
2 FROM TEAMS INNER JOIN PLAYERS
3 ON (TEAMS.PLAYERNO = PLAYERS.PLAYERNO);
```

Hasil #1 (2x4)

TEAMNO	NAME
1	Parmenter
2	Collins
3	Parmenter
81	Parmenter

Pertanyaan :

- Jelaskan KAPAN MENGGUNAKAN: UNION dan kapan JOIN

T1 C	T2 C	T3 C	T4 C
1	2	?	?
2	3	2	2
3	4		3

Cobalah tentukan hasil dari query berikut:

1.

```
SELECT T1.C, T2.C
      FROM T1 INNER JOIN T2 ON (T1.C = T2.C)
```
2.

```
SELECT T1.C, T2.C
      FROM T1 LEFT JOIN T2 ON (T1.C = T2.C)
```
3.

```
SELECT T1.C, T2.C
      FROM T1 RIGHT JOIN T2 ON (T1.C = T2.C)
```
4.

```
SELECT T1.C, T3.C
      FROM T1 RIGHT JOIN T3 ON (T1.C = T3.C)
```
5.

```
SELECT T1.C, T3.C
      FROM T1 LEFT JOIN T3 ON (T1.C = T3.C)
```
6.

```
SELECT T3.C, T4.C
      FROM T3 LEFT JOIN T4 ON (T3.C = T4.C)
```
7.

```
SELECT T3.C, T4.C
      FROM T3 RIGHT JOIN T4 ON (T3.C = T4.C)
```

UPDATE data dalam Tabel

```
UPDATE <table specification>
SET <column name = expression>
[ WHERE <condition> ]
```

Contoh 19: Keluarga Parmenter telah pindah rumah. Sekarang mereka tinggal di Palmer Street nomor 83, kota Inglewood, kode pos 1234UU. Nomor telepon mereka yang baru belum diketahui.

Query:

```
UPDATE PLAYERS
SET STREET = 'Palmer Street', HOUSENO = '83',
    TOWN = 'Inglewood', POSTCODE = '1234UU',
    PHONENO = NULL
WHERE NAME = 'Parmenter';
```

UPDATE data dalam Tabel

Contoh 20: Naikkan jumlah penalti sebanyak 5%!

Query:

```
UPDATE PENALTIES  
SET AMOUNT = AMOUNT * 1.05;
```

Truncate Table

- Salah satu perintah atau statement di database MySQL untuk mengosongkan atau menghapus semua data yang ada di table.
- Perintah Truncate Table mirip dengan perintah DELETE table, hanya lebih singkat dan sederhana tanpa menggunakan WHERE clause.
- Perintah Truncate Table akan mereset ***sequence*** yang ada di table tersebut apabila salah satu kolomnya menggunakan constraint AUTO INCREMENT.
- Sintaknya : **TRUNCATE TABLE** *nama_table*;

Truncate Vs Delete

TRUNCATE	DELETE
Merupakan kategori DDL (Data Definition Language)	Merupakan kategori DML (Data Manipulation Language)
Akan menghapus seluruh record di table tanpa WHERE clause	Akan menghapus record berdasarkan WHERE clause
Proses lebih cepat dan menggunakan lebih sedikit resource termasuk log transaksi	Proses lebih lambat dan menggunakan lebih banyak resource termasuk log transaksi
Tidak dapat di rollback	Dapat di rollback
Tidak dapat mengaktifkan trigger	Dapat mengaktifkan trigger
Dapat mereset sequence	Tidak dapat mereset sequence

Truncate Table (contd-2)

- Pada sebelumnya kita sudah mempunyai sebuah tabel RECR_PLAYERS

```
1 SELECT * from RECR_PLAYERS
```

PLAYERNO	NAME	TOWN	PHONENO
7	Wise	Stratford	070-347689
28	Collins	Midhurst	010-659599
39	Bishop	Stratford	070-393435
95	Miller	Douglas	070-867564
200	Sisi	Madura	(NULL)

```
1 desc RECR_PLAYERS
```

Field	Type	Null	Key	Default	Extra
PLAYERNO	smallint(6)	NO	PRI	(NULL)	
NAME	char(15)	NO		(NULL)	
TOWN	char(10)	NO		(NULL)	
PHONENO	char(10)	YES		(NULL)	

Truncate Table (contd-3)

- Contoh 16 : Hapus semua isi tabel RECR_PLAYERS

```
TRUNCATE TABLE RECR_PLAYERS;
```

```
1 TRUNCATE TABLE RECR_PLAYERS;
2 SELECT * FROM RECR_PLAYERS;
```

recr_players (4x0)			
PLAYERNO	NAME	TOWN	PHONENO

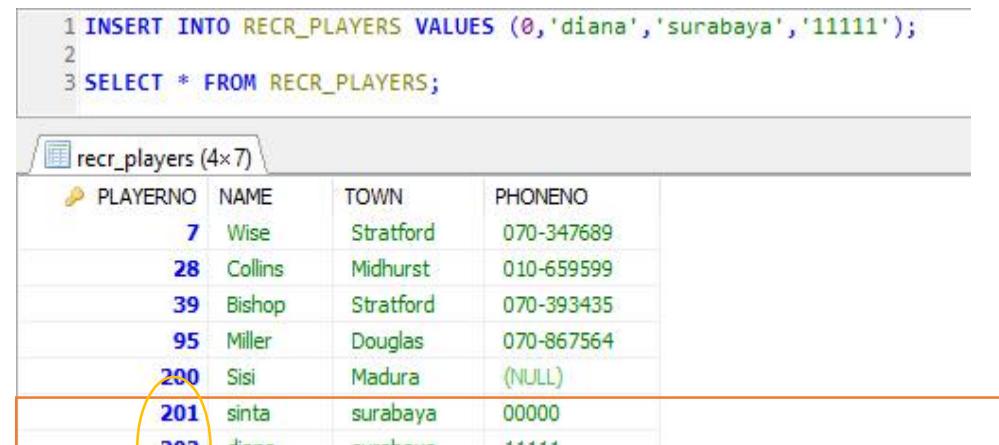
Intruksi Lanjutan : Lakukan Perubahan Struktur pada tabel RECR_PLAYERS dengan membuat Playerno menjadi Auto Increment, sehingga data ketika diberi nilai 0 atau null akan terisi sendiri

```
ALTER table RECR_PLAYERS
MODIFY COLUMN PLAYERNO SMALLINT(6)
AUTO_INCREMENT ;
```

Truncate Table (contd-4)

- **Intruksi Lanjutan** : setelah itu coba masukan data untuk mencoba hasil perubahan tabel

```
INSERT INTO RECR_PLAYERS VALUES  
(0,'sinta','surabaya','00000');  
INSERT INTO RECR_PLAYERS VALUES  
(0,'diana','surabaya','11111');  
  
SELECT * FROM RECR_PLAYERS;
```



```
1 INSERT INTO RECR_PLAYERS VALUES (0,'diana','surabaya','11111');  
2  
3 SELECT * FROM RECR_PLAYERS;
```

recr_players (4x7)			
PLAYERNO	NAME	TOWN	PHONENO
7	Wise	Stratford	070-347689
28	Collins	Midhurst	010-659599
39	Bishop	Stratford	070-393435
95	Miller	Douglas	070-867564
200	Sisi	Madura	(NULL)
201	sinta	surabaya	00000
202	diana	surabaya	11111

Intruksi Lanjutan : Setelah itu lakukan perintah Truncate Tabel

```
TRUNCATE TABLE RECR_PLAYERS;
```

Truncate Table (contd-5)

- Intruksi Lanjutan : Masukan data lagi untuk menguji hasilnya

```
INSERT INTO RECR_PLAYERS VALUES  
(0, 'diana', 'surabaya', '11111');
```

```
INSERT INTO RECR_PLAYERS VALUES  
(0, 'Toni', 'surabaya', '00000');
```

1	INSERT INTO RECR_PLAYERS VALUES (0,'diana','surabaya','11111');
2	
3	SELECT * FROM RECR_PLAYERS;

recr_players (4x7)						
PLAYERNO	NAME	TOWN	PHONENO			
7	Wise	Stratford	070-347689			
28	Collins	Midhurst	010-659599			
39	Bishop	Stratford	070-393435			
95	Miller	Douglas	070-867564			
200	Sisi	Medura	(NULL)			
201	sinta	surabaya	00000			
202	diana	surabaya	11111			

Auto Incremennya di mulai dari awal lagi

```
SELECT * FROM RECR_PLAYERS;
```

1	INSERT INTO RECR_PLAYERS VALUES (0,'Toni','surabaya','00000');
2	SELECT * FROM RECR_PLAYERS;
3	

recr_players (4x2)			
PLAYERNO	NAME	TOWN	PHONENO
1	diana	surabaya	11111
2	Toni	surabaya	00000

DELETE data dalam Tabel

```
DELETE  
FROM <table specification>  
[ WHERE <condition> ]
```

Contoh 21: Hapus seluruh data penalti yang dilakukan oleh pemain nomor 44 pada tahun 1980!

Query:

```
DELETE  
FROM PENALTIES  
WHERE PLAYERNO = 44  
AND YEAR(PAYMENT_DATE) = 1980;
```

Function

Operator IS NULL

Operator IN dalam subquery

Operator EXISTS

Operator ALL & ANY

DISTINCT

Fungsi COUNT

Fungsi MAX dan MIN

Fungsi SUM

Fungsi AVG

Ekspresi CASE

Operator IS NULL

- ❑ Digunakan untuk memilih baris (record) dalam kolom tertentu yang tidak memiliki nilai
- ❑ Contoh 1: Buat daftar seluruh pemain yang memiliki nomor liga!

Query:

```
SELECT PLAYERNO, LEAGUENO  
FROM PLAYERS  
WHERE LEAGUENO IS NOT NULL
```

Contoh 2: Cari nomor pemain yang tidak memiliki nomor liga!

Query:

```
SELECT PLAYERNO  
FROM PLAYERS  
WHERE LEAGUENO IS NULL
```

Operator IN dalam subquery (contd-1)

- Contoh 3: Cari nomor, nama dan inisial dari masing-masing pemain yang telah bermain minimal satu kali pertandingan

Query:

```
SELECT PLAYERNO  
FROM MATCHES
```

Query:

```
SELECT PLAYERNO, NAME, INITIALS  
FROM PLAYERS  
WHERE PLAYERNO IN  
(2,6,8,27,44,57,83,104,112)
```

Operator IN dalam subquery (contd-2)

- Contoh 3: Cari nomor, nama dan inisial dari masing-masing pemain yang telah bermain minimal satu kali pertandingan
- Query:

```
SELECT PLAYERNO, NAME, INITIALS  
FROM PLAYERS  
WHERE PLAYERNO IN  
(SELECT PLAYERNO  
FROM MATCHES)
```

Intermediate Result:
(2,6,8,27,44,57,83,104, 112)

Operator IN dalam subquery (contd-3)

- Contoh 4: Cari nomor dan nama dari masing-masing pemain dari tim 1 yang telah bermain setidaknya satu kali pertandingan!

Query:

```
SELECT PLAYERNO, NAME,  
FROM PLAYERS  
WHERE PLAYERNO IN  
(SELECT PLAYERNO  
FROM MATCHES)  
WHERE TEAMNO = 1)
```



Intermediate Result:
(2,6,8,44,57,83)

Operator IN dalam subquery (contd-4)

- Contoh 5: Cari nomor dan nama dari masing-masing pemain yang telah bermain setidaknya satu kali pertandingan dari suatu tim yang kaptennya bukan pemain no 6!

Query:

```
SELECT PLAYERNO, NAME,  
      FROM PLAYERS  
      WHERE PLAYERNO IN  
            (SELECT PLAYERNO  
                  FROM MATCHES  
                  WHERE TEAMNO NOT IN  
                        (SELECT TEAMNO  
                              FROM TEAMS  
                              WHERE PLAYERNO = 6))
```

Intermediate Result:
(8,27,104,112)

Intermediate Result:
(1)

Operator EXISTS

- Contoh 6: Cari nama dan inisial dari masing-masing pemain yang telah melakukan minimal satu kali penalti!

Query 1:

```
SELECT NAME, INITIALS  
FROM PLAYERS  
WHERE PLAYERNO IN  
(SELECT PLAYERNO  
FROM PENALTIES)
```

Query 2:

```
SELECT NAME, INITIALS  
FROM PLAYERS  
WHERE EXISTS  
(SELECT *  
FROM PENALTIES  
WHERE PLAYERNO = PLAYERS.PLAYERNO)
```

Operator EXISTS (contd-2)

- Contoh 7: Cari nama dan inisial dari para pemain yang tidak menjadi kapten dalam suatu tim!
- Query:

```
SELECT NAME, INITIALS  
FROM PLAYERS  
WHERE NOT EXISTS
```

```
(SELECT * FROM TEAMS  
WHERE PLAYERNO = PLAYERS.PLAYERNO)
```

Operator ALL & ANY

- ❑ Fungsinya seperti Operator IN dalam subquery
- ❑ Contoh 8: Cari nomor, nama dan tanggal lahir dari pemain yang usianya paling tua! Pemain yang paling tua adalah pemain yang tanggal lahirnya adalah lebih kecil atau sama dengan pemain-pemain yang lain.

- ❑ Query:

```
SELECT PLAYERNO, NAME, BIRTH_DATE  
FROM PLAYERS  
WHERE BIRTH_DATE <= ALL  
      (SELECT BIRTH_DATE  
       FROM PLAYERS)
```

Operator ALL & ANY (contd 2)

- ❑ Contoh 9: Cari nomor, nama dan tanggal lahir dari masingmasing pemain terkecuali pemain yang usianya paling tua!
- ❑ Query:

```
SELECT PLAYERNO, NAME, BIRTH_DATE  
FROM PLAYERS  
WHERE BIRTH_DATE > ANY  
      (SELECT BIRTH_DATE  
       FROM PLAYERS)
```

DISTINCT

- Menghilangkan baris yang isinya sama
- Contoh 10: Cari nama-nama kota yang berbeda dalam tabel PLAYERS!
- Query:
SELECT DISTINCT TOWN FROM PLAYERS

Function dalam SELECT

- COUNT ([DISTINCT | ALL] { * | <expression> }) |
- MIN ([DISTINCT | ALL] <expression>) |
- MAX ([DISTINCT | ALL] <expression>) |
- SUM ([DISTINCT | ALL] <expression>) |
- AVG ([DISTINCT | ALL] <expression>)

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Count

Contoh 9: Hitung jumlah pemain yang tercatat di dalam tabel PLAYERS!

SQL:

```
SELECT COUNT(*) FROM PLAYERS;
```

Contoh 10: Ada berapa nama kota yang tercatat di dalam kolom TOWN dalam tabel PLAYERS?

SQL:

```
SELECT COUNT(DISTINCT(TOWN)) FROM PLAYERS;
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Max

Contoh 11: Berapakah jumlah penalti yang tertinggi?

SQL:

```
SELECT MAX(AMOUNT) FROM PENALTIES;
```

Contoh 11.1: Buatlah daftar nomor dan tanggal lahir para pemain yang lahir di tahun yang sama dengan pemain termuda yang bermain untuk Tim 1!

Query: `SELECT PLAYERNO, BIRTH_DATE`

`FROM PLAYERS`

`WHERE YEAR(BIRTH_DATE) =`

`(SELECT MAX(YEAR(BIRTH_DATE)))`

`FROM PLAYERS`

`WHERE PLAYERNO IN`

`(SELECT PLAYERNO`

`FROM MATCHES`

`WHERE TEAMNO = 1))`

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Min

Contoh 12: Berapakah jumlah penalti yang terendah?

SQL:

```
SELECT MIN(AMOUNT) FROM PENALTIES;
```

Contoh 12.1 : Berapakah nilai/jumlah penalti terendah yang dilakukan oleh pemain yang bertempat tinggal di Stratford?

Query:

```
SELECT MIN(AMOUNT)  
FROM PENALTIES  
WHERE PLAYERNO IN  
(SELECT PLAYERNO  
FROM PLAYERS  
WHERE TOWN = 'Stratford')
```

Contoh 12.2 : Berapa banyak penalti yang jumlahnya adalah sama dengan nilai/jumlah penalti yang terendah?

Query:

```
SELECT COUNT(*)  
FROM PENALTIES  
WHERE AMOUNT =  
(SELECT MIN(AMOUNT)  
FROM PENALTIES)
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Sum

Contoh 13: Berapa banyak total SET yang telah dimenangkan, total SET yang telah kalah, dan berapa perbedaan di antara keduanya?

SQL:

```
SELECT SUM(WON),SUM(LOST),SUM(WON)-SUM(LOST) AS Difference
FROM MATCHES;
```

Contoh 13.1 : Berapakah jumlah total penalti yang dilakukan oleh pemain yang berasal dari Inglewood?

Query:

```
SELECT SUM(AMOUNT)
FROM PENALTIES
WHERE PLAYERO NO IN
  (SELECT PLAYERO NO
    FROM PLAYERS
    WHERE TOWN = 'Inglewood')
```

KLAUSA : SELECT (dari 1 tabel)

Fungsi dalam Select: Avg

Contoh 14: Berapakah jumlah rata-rata penalti yang dilakukan oleh pemain nomor 44?

SQL:

```
SELECT AVG(AMOUNT)
FROM PENALTIES
WHERE PLAYERNO = 44;
```

Contoh 14.1 : Berapakah rata-rata penalti yang dilakukan oleh para pemain yang bermain untuk Tim 1?

Query:

```
SELECT AVG(AMOUNT)
FROM PENALTIES
WHERE PLAYERNO IN
  (SELECT PLAYERNO
   FROM MATCHES
   WHERE TEAMNO = 1)
```

Ekspresi CASE

- Merupakan suatu bentuk ekspresi CASE di dalam SQL (serupa dengan pernyataan IF-THEN-ELSE)

<case expression>

CASE <expression>

WHEN <expression> THEN <expression>

ELSE <expression>

END

Ekspresi CASE (contd-2)

- Contoh 15 : Cari nomor, jenis kelamin dan nama dari masingmasing pemain yang telah bergabung di dalam klub setelah tahun 1980! Jenis kelamin harus tercetak sebagai 'Female' atau 'Male' dan bukannya 'F' atau 'M' saja.
- Query:

```
SELECT PLAYERO,
      CASE SEX
          WHEN 'F' THEN 'Female'
          ELSE 'Male' END AS GENDER,
      NAME
FROM PLAYERS
WHERE JOINED > 1980
```

View dan User Authorisation

- 1) View
- 2) Grant
- 3) Revoke
- 4) Privelege
- 5) Adding User
- 6) Limiting User Resource

SuperUser Vs Privileges

- User '**root**' dalam istilah keamanan komputer sering disebut sebagai '**superuser**'.
- **Superuser** user tertinggi dimana user ini dapat melihat, mengubah, bahkan menghapus seluruh database dan menjalankan perintah apapun yang terdapat dalam SQL (**CRUD**)
- **Privileges User** yang dibatasi hak aksesnya.
- Apakah user tersebut dapat membuat, mengubah dan menghapus sebuah tabel, atau user tersebut kita batasi hanya untuk melihat tabel saja (**SELECT**).

Hak Akses

- **Hak akses** didalam Database adalah **hak** yang diberikan **kepada User** untuk dapat **mengakses data** / record tertentu.
- **Hak akses** ini jenisnya bermacam-macam bisa saja untuk memberikan **hak akses Tabel**, **hak akses Kolom** untuk dapat diakses oleh User tertentu.
- Setiap **user** dapat **dibatasi** untuk dapat **mengakses** baik itu sebuah **database** tertentu saja, **tabel** tertentu, atau bahkan hanya **kolom** tertentu.
- Kita dapat membuat user baru yang hanya bisa menjalankan perintah **SELECT** saja, dan user tersebut dibatasi untuk tidak dapat menjalankan query **DROP**.

Users Authorisation

- Untuk proteksi data: USER ,PASSWORD, HAK AKSES
- Hak Akses meliputi:
 - Hak akses terhadap suatu **Kolom** tertentu dari suatu tabel
 - Hak akses terhadap suatu **Tabel**
 - Hak akses terhadap tabel-tabel yang ada pada suatu **Basisdata** tertentu
 - Hak akses suatu **User** terhadap seluruh basisdata yang ada di dalam server

Bentuk otorisasi dalam basis data

- **Read Authorization** – pengguna diperbolehkan membaca data, tetapi tidak dapat memodifikasi.
- **Insert Authorization** – pengguna diperbolehkan menambah data baru, tetapi tidak dapat memodifikasi data yang sudah ada.
- **Update Authorization** – pengguna diperbolehkan memodifikasi data, tetapi tidak dapat menghapus data.
- **Delete Authorization** – pengguna diperbolehkan menghapus data.

Bentuk otorisasi untuk modifikasi skema basis data

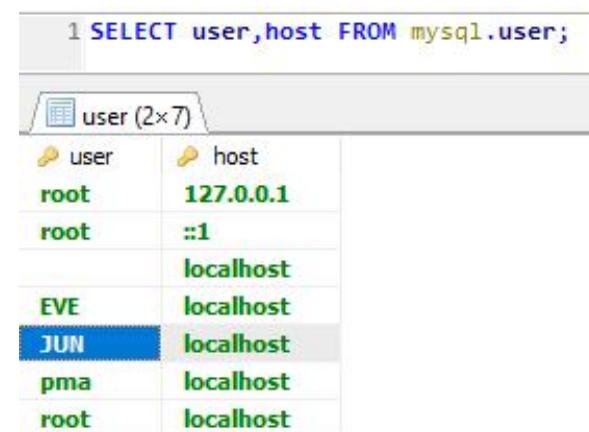
- **Index Authorization** = pengguna diperbolehkan membuat dan menghapus **index** data.
- **Authorization** = pengguna diperbolehkan **membuat relasi-relasi baru**.
- **Alteration Authorization** = pengguna diperbolehkan **menambah/menghapus atribut suatu relasi**.
- **Drop Authorization** = pengguna diperbolehkan **menghapus relasi** yang sudah ada.

Menambah, Melihat & Menghapus USERS

- **CREATE USER 'user_name'@'host_name' IDENTIFIED BY password_user**
- Contoh 1: Buatlah dua user baru, yaitu **EVE** dengan password **EVE_PASS**, dan **JUN** dengan password **JUN_PASS**!
- **CREATE USER**
`'EVE'@'localhost' IDENTIFIED BY 'EVE_PASS',
'JUN'@'localhost' IDENTIFIED BY 'JUN_PASS'`

Melihat user :

```
SELECT user,host FROM mysql.user;
```



The screenshot shows the results of a SQL query in MySQL Workbench. The query is:

```
1 SELECT user,host FROM mysql.user;
```

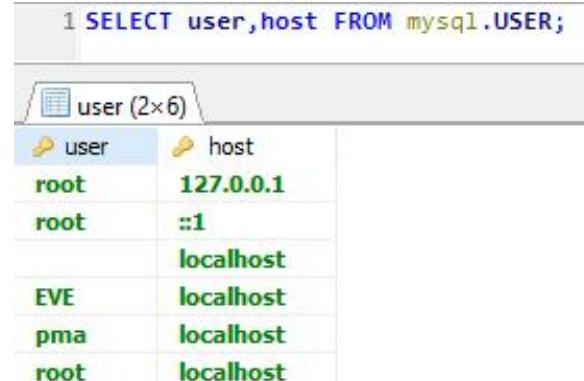
The results are displayed in a table titled "user (2x7)". The table has two columns: "user" and "host". The data is as follows:

user	host
root	127.0.0.1
root	::1
	localhost
EVE	localhost
JUN	localhost
pma	localhost
root	localhost

Menambah, Melihat & Menghapus USERS

- DROP USER 'user_name'@'host_name'
- **Contoh 2:** Hapuslah user JUN!

```
DROP USER 'JUN'@'localhost'
```



The screenshot shows the results of a SQL query in MySQL Workbench. The query is:

```
1 SELECT user,host FROM mysql.USER;
```

The results are displayed in a table titled "user (2x6)".

user	host
root	127.0.0.1
root	::1
	localhost
EVE	localhost
pma	localhost
root	localhost

```
DROP USER  
'JUN'@'localhost', 'EVE'@'localho  
st'
```

Mengubah Nama & Password USERS

- **RENAME USER** 'user_name'@'host_name' TO 'user_name'@'host_name'
- Contoh 3: **Ubahlah** nama user **EVE** dan **JUN** menjadi **EVE1** dan **JUN1**

```
RENAME USER
'EVE'@'localhost' TO 'EVE1'@'localhost',
'JUN'@'localhost' TO 'JUN1'@'localhost'
```

- **SET PASSWORD FOR** 'user_name'@'host_name' = **PASSWORD** ('new_password')
- Contoh 4: **Ubahlah** password user **JUN1** menjadi **JUN1_PASS!**

```
SET PASSWORD FOR 'JUN1'@'localhost' = PASSWORD ('JUN1_PASS')
```

Level Hak Akses - GRANT

- Hak Akses Global (*.*)
- Hak Akses Level Database (nama_database.*)
- Hak Akses Level Tabel (nama_database.nama_tabel)
- Hak Akses Level Kolom (nama_kolom)

Hak Akses Global (*.*)

- Hak akses yang dapat mengakses seluruh bagian didalam suatu database
- Misalkan Tabel, Kolom, dan Data.
- Hak akses jenis ini dapat melakukan apapun CRUD didalam Database.
- Penulisan querynya biasanya (*.*).
- Untuk memberi akses terhadap user dilakukan di User Root.
- **Sql :**
GRANT hak_akses ON *.* TO "nama_user"@"lokasi_user";
GRANT SELECT ON *.* TO 'JUN1'@'localhost', 'EVE1'@'localhost';

Hak Akses : Basisdata

- **SELECT** — user berhak untuk **mengakses seluruh tabel dan view** dari suatu basisdata dengan menggunakan pernyataan **SELECT**
- **INSERT** — user berhak untuk **menambah baris pada seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **INSERT**
- **DELETE** — user berhak untuk **menghapus baris dari seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **DELETE**
- **UPDATE** — user berhak untuk **mengubah nilai dari seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **UPDATE**
- **REFERENCES** — user berhak untuk membuat **Foreign Key (FK)** yang mengacu pada tabel-tabel yang ada dalam suatu basisdata
- **CREATE** — **CREATE** — user berhak untuk **membuat tabel-tabel baru** dalam suatu basisdata dengan menggunakan pernyataan **CREATE TABLE**
- **ALTER** — user berhak untuk **mengubah tabel-tabel** pada suatu basisdata dengan menggunakan pernyataan **ALTER TABLE**

Hak Akses : Basisdata (contd-2)

- **DROP** —user berhak untuk **menghapus seluruh tabel dan view** yang ada dalam suatu basisdata
- **INDEX** —user berhak untuk **menentukan dan menghapus indeks** dari tabel-tabel yang ada dalam suatu basisdata
- **CREATE TEMPORARY TABLES** —user berhak untuk **membuat tabel-tabel sementara** di dalam suatu basisdata
- **CREATE VIEW** — user berhak untuk **membuat view baru** dalam suatu basisdata dengan menggunakan pernyataan **CREATE VIEW**
- **SHOW VIEW** — user berhak untuk melihat **definisi view** dari **seluruh view** yang ada dalam suatu basisdata dengan menggunakan pernyataan **SHOW VIEW**
- **CREATE ROUTINE** —user berhak untuk **membuat prosedur (stored procedure) dan fungsi (stored function)** baru untuk suatu basisdata
- **ALTER ROUTINE** —user berhak untuk **mengubah dan menghapus prosedur (stored procedure) dan fungsi (stored function)** dari suatu basisdata

Hak Akses : Basisdata (contd-3)

- ❑ **EXECUTE ROUTINE** — user berhak untuk membatalkan prosedur (**stored procedure**) dan fungsi (**stored function**) yang ada pada suatu basisdata
- ❑ **LOCK TABLES** — user berhak untuk **memblok tabel-tabel** yang ada dalam suatu basisdata
- ❑ **ALL** atau **ALL PRIVILEGES** —merupakan 'singkatan' atas **seluruh hak akses** yang telah disebutkan sebelumnya

Hak Akses Database - nama_database.*

- Hanya dapat mengakses Database tertentu saja serta dapat mengakses seluruh Tabel dan Kolom didalam Database tersebut.
- Sql :

```
GRANT hak_akses ON nama_database.* TO "nama_user"@"lokasi_user";
```

```
GRANT SELECT ON tennis.* TO 'JUN1'@'localhost';
```

Hak Akses: Basisdata (contd-4)

- **Contoh 8:** Berikanlah hak SELECT untuk JUN1 pada seluruh tabel Yang ada di dalam basisdataTENNIS!

GRANT SELECT ON TENNIS.* TO JUN1

`GRANT SELECT ON TENNIS.* TO 'JUN1'@'localhost';`

- **Contoh 9:** Berikanlah hak CREATE, UPDATE dan REMOVE tabel-tabel dan view baru untuk JUN1 didalam basisdata TENNIS!

GRANT CREATE, ALTER, DROP, CREATE VIEW ON TENNIS.* TO JUN1

`GRANT CREATE, ALTER, DROP, CREATE VIEW ON TENNIS.* TO 'JUN1'@'localhost'`

Hak Akses : Tabel & Kolom

- **Select** : User berhak untuk mengakses tabel tertentu dengan menggunakan pernyataan SELECT
- **INSERT** : user berhak untuk menambah baris pada tabel tertentu dengan menggunakan pernyataan INSERT
- **DELETE** : user berhak untuk menghapus baris dari tabel tertentu dengan menggunakan pernyataan DELETE
- **UPDATE** : user berhak untuk mengubah nilai dari suatu tabel tertentu dengan menggunakan pernyataan UPDATE
- **REFERENCES** : user berhak untuk membuat Foreign Key (FK) yang mengacu pada suatu tabel
- **CREATE** : user berhak untuk membuat tabel dengan suatu nama tertentu
- **ALTER** : user berhak untuk mengubah tabel dengan menggunakan pernyataan ALTER TABLE
- **INDEX** : user berhak untuk menentukan indeks pada satu tabel
- **DROP** : user berhak untuk menghapus tabel
- **ALL** atau **ALL PRIVILEGES** : merupakan 'singkatan' atas seluruh hak akses yang telah disebutkan di atas

Hak Akses :Tabel & Kolom (contd-2)

- **Contoh 5:** Berikanlah hak SELECT untuk JUN1 pada tabel PLAYERS!

GRANT SELECT ON PLAYERS TO JUN1

```
GRANT SELECT ON tennis.players TO 'JUN1'@'localhost';
```

- **Contoh 6:** Berikanlah hak SELECT untuk user baru BOB pada tabel PLAYERS!

GRANT SELECT ON PLAYERS TO 'BOB'@'localhost' IDENTIFIED BY 'BOB_PASS'

```
GRANT SELECT ON PLAYERS TO 'BOB'@'localhost' IDENTIFIED BY 'BOB_PASS'
```

- **Contoh 7:** Berikanlah hak INSERT dan UPDATE untuk EVE1 dan JUN1 pada seluruh kolom dari tabel TEAMS!

GRANT INSERT, UPDATE ON TEAMS TO EVE1, JUN1

```
GRANT insert,update,ALTER ON tennis.teams TO  
'JUN1'@'localhost';
```

Hak Akses Tabel - nama_database.nama_tabel

- User memiliki hak akses pada sebuah tabel beserta Kolomnya yang berada pada sebuah database.
- Hak akses yang dimiliki user hanya terbatas pada level sebuah tabel saja.
- Sql :

```
GRANT hak_akses ON nama_database . nama_tabel TO  
"nama_user"@"lokasi_user";
```

```
GRANT SELECT ON tennis.players TO 'JUN1'@'localhost';
```

Hak Akses Level Kolom - nama_kolom

- ❑ Hak akses ini adalah hak akses paling kecil yang dapat diberikan kepada sebuah user.
- ❑ Dengan hak akses level kolom, user hanya memiliki hak akses untuk beberapa kolom pada sebuah tabel.
- ❑ Level paling akhir ini kita membatasi hak akses user hanya untuk kolom tertentu saja.
- ❑ Penulisan kolom yang diperbolehkan diletakkan di dalam tanda kurung.
- ❑ Sql :

```
GRANT hak_akses (nama_kolom) ON nama_database.nama_tabel  
TO "nama_user"@"lokasi_user";
```

```
GRANT SELECT (nama,umur) ON tennis.players TO 'JUN1'@'localhost';
```

Pemberian Akses DBA ke User account terhadap database:

- Membatasi User akses data **table** baik untuk **melihat struktur table, melihat data** maupun melakukan **operasi manipulasi data** seperti **Insert, Update, Delete** data
- Membatasi user akses **view** database baik melihat, maupun merubah struktur **view**
- Membatasi user akses **strored procedure** baik **execute** dan merubah struktur **SQL** didalam **stored procedure** tersebut.
- Membatasi Host akses yang digunakan user baik local host(**127.0.0.1**), user akses dalam **jaringan LAN** dan Remot IP Public.
- Memberikan **timer user akses database** pada saat **jam kerja** saja misalnya diluar jam kerja user tidak dapat melakukan akses database.
- Memberikan **otoritas user** untuk **create tabel, view function, stored procedure , trigger**

Contoh Perintah SQL

- **GRANT** : memberikan wewenang kepada pemakai
- Syntax : GRANT ON TO
- Contoh :
 - GRANT SELECT ON S TO BUDI
 - GRANT SELECT,UPDATE (STATUS,KOTA) ON S TO SULIS,WENDI
- **REVOKE** : mencabut wewenang yang dimiliki oleh pemakai
- Syntax : REVOKE ON FROM
- Contoh :
 - REVOKE SELECT ON S TO BUDI
 - REVOKE SELECT,UPDATE (STATUS,KOTA) ON S TO SULIS,WENDI
- **Priviledge list** : READ, INSERT, DROP, DELETE, INEX, ALTERATION, RESOURCE

Hak Akses:User

- Contoh 10: Berikanlah hak CREATE,ALTER,dan DROP untuk EVE1 pada seluruh tabel dari seluruh basisdata!

GRANT CREATE, ALTER, DROP ON *.* TO EVE1

GRANT CREATE, ALTER, DROP ON *.* TO EVE1 @'localhost'

- Contoh 11: Berikanlah hak kepada JUN1 untuk membuat user baru!

GRANT CREATE USER ON *.* TO JUN1

GRANT CREATE USER ON *.* TO JUN1@localhost

Meneruskan Hak Akses: WITH GRANTOPTION

- Contoh 12: Berikanlah hak REFERENCES untuk JUN1 pada tabel TEAMS dan berikanlah ijin padanya untuk meneruskan hak tersebut kepada user lain!

GRANT REFERENCES ON TEAMS TO JUN1 WITH GRANT OPTION

GRANT REFERENCES ON TEAMS TO JUN1@localhost WITH GRANT OPTION

- Dengan adanya klausa WITH GRANT OPTION, maka JUN1 dapat meneruskan hak akses REFERENCES tersebut kepada EVE1

GRANT REFERENCES ON TEAMS TO EVE1

GRANT REFERENCES ON TEAMS TO EVE1@localhost

Membatalkan Hak Akses

- Contoh 13: Batalkan hak SELECT untuk JUN1 pada tabel PLAYERS!

REVOKE SELECT ON PLAYERS FROM JUN1

`REVOKE SELECT ON PLAYERS FROM JUN1@localhost`

- Contoh 14: Batalkan hak INSERT dan SELECT untuk EVE1 pada tabel TEAMS!

REVOKE INSERT, SELECT ON TEAMS FROM EVE1

`REVOKE INSERT, SELECT ON teams FROM EVE1@localhost`

Otorisasi dan View

- **View** adalah **objek basis data** yang berisi perintah **query** ke basis data
- Setiap kali sebuah **view diaktifkan**, pemakai akan selalu **melihat hasil querynya**.
- Berbeda dengan tabel, **data** yang ditampilkan **didalam view tidak bisa di ubah**.
- **View** menyediakan mekanisme **pengamanan yang fleksibel** namun **kuat** dengan cara **menyembunyikan sebagian basis data** dari user lain
- **User** dapat **diberikan otorisasi pada View**, tanpa harus diberikan otorisasi terhadap relasi yang digunakan di dalam **definisi view**.
- **View** dapat **meningkatkan keamanan data** dengan **mengizinkan user** untuk **hanya dapat mengakses data** sesuai dengan **pekerjaannya masing-masing**.
- **Kombinasi** antara **level keamanan** ditingkat **relasional** dengan **level keamanan** **dingkat view** dapat digunakan untuk **membatasi hak akses user**, sehingga mereka **hanya mengakses data** sesuai dengan **kebutuhan** saja.

VIEWS

```
CREATE VIEW view_name (column_name) AS  
[SELECT BLOCK]  
[WITH CHECK OPTION]
```

- Merupakan '*derivedtables*' ⇒ harus didefinisikan dalam bentuk query pada tabel atau view yang lain
- Merupakan '*virtual tables*' ⇒ tidak akan dievaluasi, kecuali jika mereka digunakan dalam query lain
- reusable
- Dapat digunakan untuk membuat query yang sulit (atau bahkan tidak mungkin) untuk dilakukan dalam suatu kondisi tertentu perintah INSERT,
- UPDATE, atau DELETE dapat dilakukan terhadap data yang ada di dalam tabel basis melalui view tabel basis tersebut

VIEWS

```
CREATE VIEW view_name (column_name) AS  
[SELECT BLOCK]  
[WITH CHECK OPTION]
```

- ❑ Tabel View bisa berasal dari tabel lain, atau gabungan dari beberapa tabel.
- ❑ Tujuan :
 - ❑ kenyamanan (mempermudah penulisan query),
 - ❑ Keamanan (menyembunyikan beberapa kolom yang bersifat rahasia),
 - ❑ beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan query tersebut secara berulang)

Create VIEWS

- **Contoh 1:** Buatlah **view** untuk membuat daftar seluruh nama kota yang ada dalam tabel **PLAYERS**!
- **CREATE VIEW TOWNS (TOWN)
AS SELECT DISTINCT TOWN
FROM PLAYERS**
- Sintaks Alternatif:
- **CREATE VIEW TOWNS AS
SELECT DISTINCT TOWN FROM
PLAYERS**
- Query SELECT:
- **SELECT * FROM TOWNS**

1	CREATE VIEW TOWNS AS SELECT DISTINCT TOWN FROM PLAYERS
/	players (1x6)
	TOWN
	Stratford
	Inglewood
	Eltham
	Midhurst
	Douglas
	Plymouth

Create VIEWS (contd 2)

- Contoh 2: Buatlah view untuk membuat daftar nomor pemain dan nomor liga dari seluruh pemain yang memiliki nomor liga!
- CREATE VIEW CPLAYERS AS SELECT PLAYERN0, LEAGUENO FROM PLAYERS WHERE LEAGUENO IS NOT NULL**
- Query SELECT:
- SELECT * FROM CPLAYERS**
- Contoh 3: Dapatkan seluruh informasi mengenai pemain yang memiliki nomor liga yang nomor pemainnya adalah antara 6 dan 44!
- SELECT * FROM CPLAYERS WHERE PLAYERN0 BETWEEN 6 AND 44**
- Atau sama dengan :
SELECT * FROM CPLAYERS WHERE PLAYERN0 >= 6 AND PLAYERN0 <=44;

```
1 CREATE VIEW CPLAYERS AS SELECT PLAYERN0, LEAGUENO FROM PLAYERS WHERE LEAGUENO IS NOT NULL
2 SELECT * FROM CPLAYERS
3
```

CPLAYERS (2x10)	
PLAYERN0	LEAGUENO
2	2411
6	8467
8	2983
27	2513
44	1124
57	6409
83	1608
100	6524
104	7060
112	1319

```
1 SELECT * FROM CPLAYERS WHERE PLAYERN0 BETWEEN 6 AND 44
```

CPLAYERS (2x4)	
PLAYERN0	LEAGUENO
6	8467
8	2983
27	2513
44	1124

Create VIEWS (contd 3)

- Contoh 4: Buatlah **view** untuk membuat **daftar seluruh pemain** yang **memiliki nomor liga** dengan **nomor pemain antara 6 dan 27!**
- **CREATE VIEW SEVERAL AS
SELECT * FROM CPLAYERS
WHERE PLAYERNO BETWEEN
6 AND 27**
- Untuk kebalikannya :
- **CREATE VIEW SEVERAL2 AS
SELECT * FROM CPLAYERS
WHERE PLAYERNO not
BETWEEN 6 AND 27**

1 SELECT * FROM CPLAYERS

CPLAYERS (2x10)	
PLAYERNO	LEAGUENO
2	2411
6	8467
8	2983
27	2513
44	1124
57	6409
83	1608
100	6524
104	7060
112	1319

1 SELECT * FROM SEVERAL

SEVERAL (2x3)	
PLAYERNO	LEAGUENO
6	8467
8	2983
27	2513

Create VIEWS (contd 4)

- Contoh 5: Tentukan **(nilai) rata-rata** dari **keseluruhan penalti** yang pernah dilakukan oleh **pemain!**
- Solusi berikut **tidak mungkin** dapat dilakukan:
- **SELECT PLAYERNO, AVG(SUM(AMOUNT)) FROM PENALTIES GROUP BY PLAYERNO;**
- Solusiyang dapat dilakukan dengan menggunakan VIEW:
- **CREATE VIEW SUM_PENALTIES (PLAYERNO, SUM_AMOUNT) AS SELECT PLAYERNO, SUM(AMOUNT) FROM PENALTIES GROUP BY PLAYERNO;**
- **SELECT AVG(SUM_AMOUNT) FROM SUM_PENALTIES;**

```
1 CREATE VIEW SUM_PENALTIES (PLAYERNO, SUM_AMOUNT) AS  
2 SELECT PLAYERNO, SUM(AMOUNT) FROM PENALTIES GROUP BY PLAYERNO;
```

Create VIEWS (contd 5)

- Sintaks Alternatif untuk membuat VIEW:
- **CREATE VIEW**
SUM_PENALTIES AS
SELECT PLAYERNO,
SUM(AMOUNT) AS
SUM_AMOUNT FROM
PENALTIES GROUP BY
PLAYERNO;

```
1 SELECT AVG(SUM_AMOUNT)
2 FROM SUM_PENALTIES;|
```

Hasil #1 (1x1)

Avg(SUM_AMOUNT)
96,000000

Create VIEWS (contd 5)

- Contoh 6: Buatlah **view** untuk **membuat daftar** urutan digit dari 0 sampai 9!

- **CREATE VIEW DIGITS AS**

```
SELECT 0 DIGIT UNION
```

```
SELECT 1 UNION
```

```
SELECT 2 UNION
```

```
SELECT 3 UNION
```

```
SELECT 4 UNION
```

```
SELECT 5 UNION
```

```
SELECT 6 UNION
```

```
SELECT 7 UNION
```

```
SELECT 8 UNION
```

```
SELECT 9
```

- **SELECT * FROM DIGITS**

```
1 SELECT * FROM DIGITS
```

Hasil #1 (1×10)

DIGIT
0
1
2
3
4
5
6
7
8
9

Create VIEWS (contd 6)

- Contoh 7 : Misalkan kita ingin menampilkan nama pemain yang berjenis kelamin female.

```
SELECT PLAYERO NO, NAME, INITIALS , sex, TOWN FROM  
players WHERE sex='F'
```

1 SELECT PLAYERO NO, NAME, INITIALS , sex, TOWN FROM players WHERE sex='F'
2 |

players (5x5)

PLAYERO NO	NAME	INITIALS	sex	TOWN
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

Bagaimana jika query tersebut akan dijalankan setiap beberapa detik (diakses dari website yang sibuk)???

Solusinya? view

Create VIEWS (contd 7)

- Contoh 8 : Solusinya dengan membuat View untuk menampilkan nama pemain yang berjenis kelamin female. Sehingga beban server berkurang dan VIEW dapat menyembunyikan beberapa kolom dari tabel players.

```
CREATE VIEW sex_female AS SELECT  
PLAYERNO, NAME, INITIALS, sex,  
TOWN FROM players WHERE sex='F'
```

```
SELECT * FROM sex_female
```

```
SELECT name FROM sex_female  
WHERE playerno='8'
```

SELECT * from sex_female

sex_female (5x5)				
PLAYERNO	NAME	INITIALS	sex	TOWN
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

```
1 SELECT name FROM sex_female WHERE playerno='8'
```

sex_female (1x1)

NAME
Newcastle

Create VIEWS (contd 8)

- Bagaimana jika tabel utama di update?

```
1 SELECT * from players
```

players (12x14)											
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319

Create VIEWS (contd 9)

INSERT INTO PLAYERS VALUES

```
(3, 'Dian', 'Di', '1999-05-20', 'F', 2000, 'Surabaya', '11', '60208', 'Surabaya', '0856-4868-8777', '12'),
(4, 'Diana', 'Da', '1999-06-21', 'F', 2000,
'Bojonegoro', '12', '62115', 'Bojonegoro',
'0800-0000-1111', '13'),
(5, 'Dinda', 'Dn', '1999-07-22', 'F', 2000,
'Semarang', '13', '63115',
'Semarang', '0811-1111-0000', '14');
```

1 SELECT * FROM players												
players (12x17)												
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO	
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411	
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-12		
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-13		
5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Semarang	0811-1111-14		
6	Parmenter	R	1964-06-25	M	1.977	Haselton Lane	80	1234K	Stratford	070-476537	8467	
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)	
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WD	Inglewood	070-458458	2983	
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457OK	Eltham	079-234857	2513	
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294K	Midhurst	010-658599	(NULL)	
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	76	9629CD	Stratford	070-393495	(NULL)	
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4441J	Inglewood	070-368753	1124	
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377GB	Stratford	070-473458	6409	
83	Hope	PK	1956-11-11	M	1.982	Magnolene Road	16A	1812JP	Stratford	070-353548	1608	
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746GP	Douglas	070-867564	(NULL)	
100	Parmenter	P	1963-02-28	M	1.979	Haselton Lane	80	6494SG	Stratford	070-494593	6524	
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AQ	Eltham	079-987571	7060	
112	Bailey	IP	1963-10-01	F	1.984	Viven Road	8	6392LK	Plymouth	010-548745	1319	

1 SELECT * FROM sex_female				
sex_female (5x8)				
PLAYERNO	NAME	INITIALS	sex	TOWN
3	Dian	Di	F	Surabaya
4	Diana	Da	F	Bojonegoro
5	Dinda	Dn	F	Semarang
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

Terlihat bahwa VIEW juga otomatis diupdate.

Create VIEWS (contd 10)

- Apakah kita bisa menambahkan data ke dalam VIEW? Nama Viewnya : sex_female

```
INSERT INTO sex_female VALUES  
(200, 'Sisi',  
'Ss', 'F', 'Madura');
```

Ternyata ketika kita update pada VIEW. Pada tabel asal juga mengalami Update meskipun akan terdapat nilai **NULL** di dalam tabel asli

```
1 SELECT * FROM sex_female
```

sex_female (5x9)

PLAYERNO	NAME	INITIALS	sex	TOWN
3	Dian	Di	F	Surabaya
4	Diana	Da	F	Bojonegoro
5	Dinda	Dn	F	Semarang
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth
200	Sisi	Ss	F	Madura

```
1 SELECT * FROM players
```

players (12x18)

PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13
5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Semarang	0811-1111-	14
6	Parmenter	R	1964-06-25	M	1.977	Haseline Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	15	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1.979	Haseline Lane	80	6494SG	Stratford	070-494593	6524
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319
200	Sisi	Ss	(NULL)	F	0	(NULL)	(NULL)	(NULL)	Madura	(NULL)	(NULL)

Update VIEWS -- WITH CHECK OPTION

- Contoh 9 : Buatlah view untuk membuat daftar pemain yang lahir sebelum tahun 1960!
- **CREATE VIEW VETERANS AS SELECT * FROM PLAYERS WHERE BIRTH_DATE < '1960-01-01';**
- **SELECT * FROM VETERANS;**

The screenshot shows a MySQL Workbench environment. On the left, a code editor displays the SQL command: `1 SELECT * FROM VETERANS;`. To the right is a results grid titled "VETERANS (12x3)" containing player data. A toolbar on the far right includes icons for Delay, Delete, Description, Delete, and Direct.

PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608

Update VIEWS -- WITH CHECK OPTION (Contd-2)

- **SELECT * FROM VETERANS;**

1 `SELECT * FROM VETERANS;`

PLAYERO NO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608

- Kemudian lakukan perubahan tanggal lahir dari pemain nomor 2 (dari 1 September 1948 menjadi 1 September 1970)

`UPDATE VETERANS`

`SET BIRTH_DATE = '1970-09-01'`

`WHERE PLAYERO NO = 2;`

`SELECT * FROM VETERANS;`

PLAYERO NO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608

Update VIEWS -- WITH CHECK OPTION (contd-3)

- Penggunaan WITH CHECK OPTION pada VIEW adalah melakukan pengecekan validitas perintah UPDATE, INSERT dan DELETE:
 - Perintah UPDATE adalah benar jika seluruh baris yang di-UPDATE tetap menjadi isi ('virtual') tabel VIEW
 - Perintah INSERT adalah benar jika baris baru yang di INSERT merupakan isi ('virtual') tabel VIEW
 - Perintah DELETE adalah benar jika baris baru yang di DELETE merupakan isi ('virtual') tabel VIEW
- Maka, view pada **Contoh 9** diubah menjadi seperti berikut: `CREATE OR REPLACE VIEW VETERANS AS
SELECT * FROM PLAYERS
WHERE BIRTH_DATE < '1960-01-01'
WITH CHECK OPTION`

Delete VIEWS

- Contoh 9: Hapuslah view CPLAYERS!

DROP VIEW CPLAYERS;

- Contoh 10: Hapuslahview VETERANS!

DROP VIEW VETERANS

```
1 SELECT * from CPLAYERS;
2 SELECT * from VETERANS
3
4 DROP VIEW VETERANS
5 DROP VIEW CPLAYERS;
```

CPLAYERS (2×13)		VETERANS (12×2)	
PLAYERNO	LEAGUENO	PLAYERNO	LEAGUENO
2	2411		
3	12		
4	13		
5	14		
6	8467		
8	2983		
27	2513		
44	1124		
57	6409		
83	1608		
100	6524		
104	7060		
112	1319		

Stored Procedure

- 1) Function Lanjutan
- 2) Stored Procedure
- 3) Trigger

Function Lanjutan

- ❑ Stored function merupakan sekumpulan perintah SQL yang disusun dalam sebuah fungsi yang memiliki nama, kegunaan, dan pembalian nilai (return value)
- ❑ Keuntungan dengan stored function yakni kita dapat membuat fungsi yang tidak tersedia.
- ❑ Aturan pembuatan stored function mirip dengan stored procedure. Perbedaanya adalah dalam function adanya nilai RETURN.
- ❑ Sama seperti halnya prosedur, dalam tubuh fungsi dapat berisi statement seperti deklarasi variabel, perintah pemilihan, dan perulangan

Function Lanjutan

- Stored Function hampir sama seperti stored procedure, yaitu mempunyai sekumpulan statemen sql dan statemen procedural yang tersimpan pada database MySQL dan dapat dipanggil dari aplikasi maupun database MySQL.
- Poin :
 - Terdapat statemen sql dan procedural language
 - Tersimpan pada database server (MySQL)
 - Dapat dipanggil dari program aplikasi dan Database server (MySQL)

Sintaks :

```
CREATE FUNCTION <namafungsi>(IN/OUT/INOUT
parameter TYPE)
RETURNS <tipedata>
BEGIN
    <statement>
END
```

Function Lanjutan - Sintak

```
CREATE FUNCTION sp_name ([func_parameter[, ...]])  
    RETURNS type  
    [characteristic ...] routine_body  
proc_parameter:  
    [ IN | OUT | INOUT ] param_name type  
func_parameter:  
    param_name type  
type:  
    Any valid MySQL data type  
characteristic:  
    LANGUAGE SQL  
    | [NOT] DETERMINISTIC  
    | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIESTEXT DATA }  
    | SQL SECURITY { DEFINER | INVOKER }  
    | COMMENT 'string'  
routine_body:  
    Valid SQL procedure statement or statements
```

Keterangan Sintak Function

- ❑ **sp_name:** Nama *routine* yang akan dibuat
- ❑ **proc_parameter:** Spesifikasi parameter sebagai IN, OUT, atau INOUT valid hanya untuk PROCEDURE. (parameter FUNCTION selalu sebagai parameter IN)
- ❑ **returns:** Klausa RETURNS dispesifikan hanya untuk suatu FUNCTION. Klausa ini digunakan untuk mengembalikan tipe fungsi, dan *routine_body* harus berisi suatu statemen nilai RETURN.
- ❑ **comment:** Klausa COMMENT adalah suatu ekstensi MySQL, dan mungkin digunakan untuk mendeskripsikan *stored procedure*. Informasi ini ditampilkan dengan statemen SHOW CREATE PROCEDURE dan SHOW CREATE FUNCTION.

Keterangan Sintak Function MYSQL

- **DELIMITER** adalah untuk memberi tahu kepada mysql soal delimiter yang digunakan, secara default menggunakan ; jadi bila ada tanda ; mysql akan mengartikan akhir dari statement, pada contoh di atas delimiter yang digunakan // jadi akhir statementnya adalah //
- **CREATE FUNCTION** adalah header untuk membuat sebuah function.
- **RETURNS** adalah untuk menentukan tipe data yang di return-kan oleh sebuah function.

Keterangan Sintak Function MYSQL

- **DETERMINISTIC/ NOT DETERMINISTIC** adalah untuk menentukan yang bisa menggunakan function ini adalah user pembuatnya saja (deterministic) atau user siapa saja (not deterministic).
- Untuk penulisan **DETERMINISTIC** bisa ditulis secara implisit dengan memberikan setting global pada mysql dan secara default benilai **NOT DETERMINISTIC**.
- **BEGIN END** adalah body dari function jadi semua SQL nya di tulis disini.

Function Lanjutan - Contoh

Contoh 1 : Buatlah Function dengan Case untuk menentukan jumlah diskon jika jumlah nilainya lebih dari 100 maka diskonnya 10, jika jumlah nilainya dibawah 100 s.d 50 maka diskonya 5, dan jika jumlah nilanya dibawah 50 hingga 20 maka diskonnya 3 dan jika tidak maka tidak ada diskon.

```
DELIMITER //
CREATE FUNCTION getDiskon(jumlah INT)
RETURNS int(11)
BEGIN
    DECLARE diskon INT; CASE
        WHEN (jumlah >= 100) THEN SET diskon = 10;
        WHEN (jumlah >= 50 AND jumlah < 100) THEN SET diskon = 5;
        WHEN (jumlah >= 20 AND jumlah < 50) THEN SET diskon = 3;
        ELSE SET diskon = 0; END CASE;
    RETURN diskon;
END//
```

Perintah Untuk memanggilnya :

```
SELECT getdiskon('20');
```



Function Lanjutan - Contoh

Contoh 2 : Buatlah Function untuk menentukan volume segitiga

Delimiter //

```
create function volume (panjang int, lebar int,  
tinggi int) returns int  
deterministic  
begin  
declare volum INT;  
set volum = panjang * lebar * tinggi;  
return volum;  
END//
```

Contoh Perintah Untuk memanggilnya :

```
Select volume (5,3,7);
```

1 Select volume (5,3,7);
Hasil #1 (1x1)
volume (5,3,7)
105

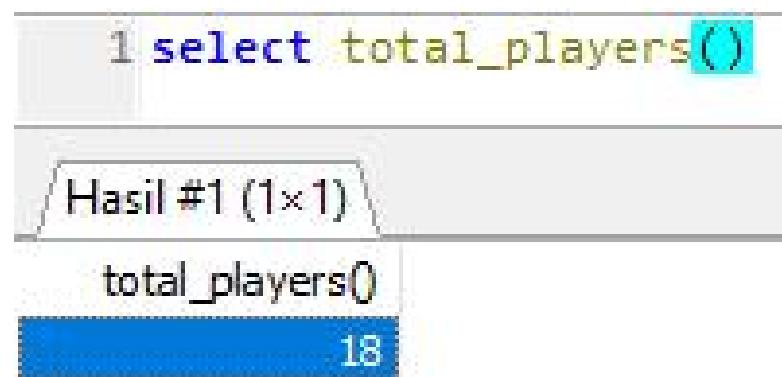
Function Lanjutan - Contoh

Contoh 3: Buatlah Function untuk menghitung jumlah pemain yang tercatat di dalam tabel PLAYERS!

```
Delimiter //
CREATE FUNCTION total_players()
RETURNS Int
BEGIN
RETURN (SELECT COUNT(*) FROM PLAYERS);
end
```

Perintah Untuk memanggilnya :

```
select total_players()
```



The screenshot shows a MySQL command-line interface. A query is being run: "1 select total_players()". The result set is labeled "Hasil #1 (1x1)" and contains a single row with the value "total_players0". The number "18" is visible at the bottom right of the interface.

total_players()
total_players0

18

Function Lanjutan - Contoh

- **Contoh 4:** Buatlah Function untuk menghitung Ada berapa nama kota yang tercatat di dalam kolom TOWN dalam tabel PLAYERS, dengan menghilangkan data yang duplikat?

Delimiter //

```
CREATE FUNCTION number_players()
RETURNS Int
BEGIN
RETURN (SELECT COUNT(DISTINCT (TOWN)) FROM PLAYERS);
end
```

Perintah Untuk memanggilnya :

```
select number_players()
```

number_players()
10

Stored Procedures

- Merupakan sekumpulan sintaks SQL yang tersimpan pada server
- Memiliki beberapa keunggulan
 - Karena sintaks sql pada stored procedure tersimpan pada server maka pemanggilan lebih cepat.
 - Reuseable artinya cukup ditulis sekali dapat digunakan berkali-kali
 - Meningkatkan keamanan

Stored Procedure vs Trigger

□ Persamaan:

- Obyeknya dapat berupa TABLE, VIEW, dll
- Tersimpan di dalam System Catalog basisdata
- Terdiri dari pernyataan SQL yang deklaratif (seperti CREATE, UPDATE, dan SELECT) dan prosedural (seperti IF-THEN-ELSE dan WHILE-DO)

□ Perbedaan:

- Stored procedure diaktifkan sebagai suatu pernyataan oleh SQL editor, program, atau oleh stored procedure atau trigger lain
- Trigger diaktifkan hanya oleh RDBMS dalam suatu kondisi tertentu (ketika pernyataan INSERT, UPDATE, DELETE dieksekusi)

Komponen Stored Procedure

- Nama Stored Procedure
- Parameter
- Body
- Sintaknya :

```
Create procedure nama_prosedur(param1,  
                                param2,...,paramn)  
Begin  
    <sintaks SQL>  
end
```

Contd. Stored Procedure

```
CREATE PROCEDURE DELETE_MHS  
    (IN P_MHSID INTEGER)  
BEGIN  
    DELETE FROM MAHASISWA  
    WHERE ID_MHS = P_MHSID;  
END
```

Nama Procedure

Parameter

Body

Stored Procedure - Body

- Berisi semua statement yang akan dieksekusi.
- Diawali keyword BEGIN dan diakhiri END
- Statement SQL dapat berupa : DDL, DML, DCL
- Procedural Language : IF-THEN-ELSE, WHILE-DO
- Dapat mendeklarasikan variabel (local variabel)

Stored Procedure

□ Create Stored Procedure:

```
<create procedure statement> ::=  
CREATE PROCEDURE <procedure name> ( [ <parameter list> ] )  
<routine body>  
<parameter list> ::=  
<parameter specification> [ , <parameter specification> ]...  
<parameter specification> ::=  
[ IN | OUT | INOUT ] <parameter> <data type>  
<routine body> ::= <begin-end block>  
<begin-end block> ::=  
[ <label> : ] BEGIN <statement list> END [ <label> ]  
<statement list> ::= { <body statement> ; }...  
<statement in body> ::=  
<declarative statement> | <procedural statement>
```

Stored Procedure

□ Aktivasi /pemanggilan Stored Procedure :

```
<call statement> ::=  
CALL [ <database name> . ] <stored procedure  
name>  
( [ <scalar expression> [ , <scalar expression>  
]... ] )
```

□ Menghapus Stored Procedure:

```
<drop procedure statement> ::=  
DROP PROCEDURE [ IF EXISTS ]  
[ <database name> . ] <procedure NAME>
```

Parameter Stored Procedure

□ IN

- Parameter yang merupakan mode default ini mengindikasikan bahwa sebuah parameter dapat di-pass ke dalam stored procedure tetapi nilainya tidak dapat diubah (dari dalam stored procedure)/
- untuk inputan parameter yang akan disimpan dalam stored procedure

□ OUT

- stored procedure dapat mengubah parameter dan mengirimkan kembali ke program pemanggil./
- sebagai output parameter yang diperoleh dalam stored procedure

□ INOUT

- Kombinasi dari mode IN dan OUT.
- kita bisa mengirimkan parameter kedalam stored procedure dan mendapatkan nilai kembalian yang baru dari stored procedure yang didefinisikan./
- sebagai parameter yang dapat digunakan sebagai input maupun output

Variabel Stored Procedure

- ❑ Dideklarasikan dengan keyword “DECLARE” kemudian diikuti dengan nama variabel dan tipe data.
- ❑ Sintaks:

DECLARE namavariabel **TYPE** **DEFAULT** nilai

Pemilihan Stored Procedure

- ❑ Perintah pemilihan ini berupa statement-statement yang akan mengerjakan instruksi jika kondisi benar/terpenuhi
- ❑ Sintaks :

```
IF [val] THEN  
[result1]  
END IF ;
```

```
IF [val] THEN  
[result1]  
ELSE  
[result2]  
END IF ;
```

Perulangan Stored Procedure

- Perintah perulangan dengan menggunakan statement LOOP, WHILE, dan REPEAT.
- Penggunaan statement LOOP diawali dengan menentukan nama perulangan : LOOP dan diakhiri dengan END LOOP.

Sintak Loop :

```
Loop_name : LOOP  
[statement1]  
[statement2]  
END LOOP loop_name
```

Sintak While :

```
Loop_Name : WHILE  
[condition] DO  
[statement1]  
[statement2]  
END WHILE  
Loop_Name;
```

Sintak Repeat :

```
[begin_label:] REPEAT  
statement_list  
UNTIL search_condition  
END REPEAT [end_label]
```

Contoh Stored Procedure

```
select * from players;
```

players (12x18)											
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13
5		(NULL)			0		(NULL)	(NULL)		(NULL)	98
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319
200	Sisi	Ss	(NULL)	F	0		(NULL)	(NULL)	Madura	(NULL)	(NULL)

Contoh Stored Procedure

Contoh 1: Buatlah stored procedure untuk memanggil isi pada tabel players

```
delimiter //
CREATE procedure getPlayers()
BEGIN
SELECT * from players;
end //
delimiter;
```

Perintah Untuk memanggilnya :

```
CALL getPlayers();
```

1 CALL getPlayers();													
/players (12x18)/													
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO		
2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411		
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12		
4	Diana	Da	1999-05-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13		
5		(NULL)			0		(NULL)	(NULL)		(NULL)	98		
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467		
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)		
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983		
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513		
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)		
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)		
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124		
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409		
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608		
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)		
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524		
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060		
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319		
200	Sisi	Ss	(NULL)	F	0		(NULL)	(NULL)	Madura	(NULL)	(NULL)		

Contoh Stored Procedure

- **Contoh 2:** Buatlah stored procedure untuk menghapus seluruh pertandingan yang pernah dimainkan oleh suatu pemain tertentu!

```
DELIMITER $$  
CREATE PROCEDURE DELETE_MATCHES (IN P_PLAYERNO INTEGER)  
BEGIN  
DELETE FROM MATCHES  
WHERE PLAYERNO = P_PLAYERNO;  
END$$  
DELIMITER ;
```

- **Contoh 2:** Hapuslah seluruh pertandingan yang pernah dimainkan oleh pemain nomor 8 dengan penggunaan prosedur DELETE_MATCHES!

```
CALL DELETE_MATCHES (8);
```

Contoh Stored Procedure (Contd-2)

- **Contoh 3:** Hapuslah pemain nomor 83 dari basisdata TENNIS. Akan tetapi jika pemain tersebut adalah kapten tim, maka pemain tidak boleh dihapus dari basisdata!

Alur 1 : Mengecek apakah pemain nomor 83 adalah kapten tim:

```
SELECT COUNT(*) AS IS_CAPTAIN  
FROM TEAMS WHERE PLAYERO NO = 83
```

IS_CAPTAIN

0

Karena pemain nomor 83 bukanlah kapten suatu tim,Sehingga dapat diketahui,
Kita dapat menghapus semua data mengenai pemain tersebut:

```
DELETE FROM MATCHES  
WHERE PLAYERO NO = 83;  
DELETE FROM COMMITTEE_MEMBERS  
WHERE PLAYERO NO = 83;  
DELETE FROM PENALTIES  
WHERE PLAYERO NO = 83;  
DELETE FROM PLAYERS  
WHERE PLAYERO NO = 83;
```

Contoh Stored Procedure (Contd-3)

Percabangan

```
delimiter //  
CREATE PROCEDURE DELETE_PLAYER_83()  
BEGIN  
    DECLARE NUMBER_OF_TEAMS INTEGER;  
  
    SELECT COUNT(*)  
    INTO NUMBER_OF_TEAMS  
    FROM TEAMS  
    WHERE PLAYERO NO = 83;  
  
    IF NUMBER_OF_TEAMS = 0 THEN  
        DELETE FROM MATCHES  
        WHERE PLAYERO NO = 83;  
        DELETE FROM COMMITTEE_MEMBERS  
        WHERE PLAYERO NO = 83;  
        DELETE FROM PENALTIES  
        WHERE PLAYERO NO = 83;  
        DELETE FROM PLAYERS  
        WHERE PLAYERO NO = 83;  
    END IF;  
END//  
delimiter
```

Aktivasi/pemanggilan stored procedure:

```
CALL DELETE_PLAYER_83
```

Menghapus stored procedure:

```
DROP PROCEDURE DELETE_PLAYER_83
```

Contoh Stored Procedure Percabangan

```
DELIMITER //
CREATE PROCEDURE cobaIF(
IN bil INT(3)
)
BEGIN
/*Deklarasi Variabel*/
    DECLARE str VARCHAR(50);
    if (bil<0) then
        SET str ='Bilangan Negetif';
    ELSE
        SET str='Bilangan Posistif';
    END if;
    SELECT str;
END///
DELIMITER;
```

```
1 call cobaIF(9);
```

Hasil #1 (1×1)

```
str
```

Bilangan Positif

```
1 call cobaIF(-3);
```

Hasil #1 (1×1)

```
str
```

Bilangan Negetif

Contoh Stored Procedure Pengulangan

```
DELIMITER |
CREATE PROCEDURE ExLooping (
IN bil INT(3)
)
BEGIN
/*Deklarasi Variabel*/
    DECLARE str VARCHAR(50);
    DECLARE i int(3);
    Set i=1;
    Set str='';

    While i<= bil do
        Set str=concat (str," ",i);
        Set i=i+1;
    END WHILE;
    SELECT str;

END;
|
DELIMITER ;
```

```
CALL ExLooping(10);
```

str
1 2 3 4 5 6 7 8 9 10

Contoh Stored Procedure

IN (Definisi dengan 1 parameter)

- **Contoh 4:** Buatlah stored procedure untuk menampilkan jenis kelamin yang inputannya ditentukan oleh user dengan 1 parameter.

DELIMITER \$\$

```
create procedure getNamaByJenisKelamin(IN jeniskelamin
varchar(1))
begin
select * from players where sex = jeniskelamin;
END$$
```

DELIMITER ;

Perintah Untuk memanggilnya : **call** getNamaByJenisKelamin ('m') ;
call getNamaByJenisKelamin ('F') ;

players (12x8)											
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONE NO	LEAGUENO
3	Dian	Di	1999-05-20	F	2,000	Surabaya	11	60208	Surabaya	0856-4668-	12
4	Diana	Da	1999-06-21	F	2,000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13
8	Newcastle	B	1962-07-08	F	1,980	Station Road	4	6584WIO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1,983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1,983	Old Main Road	10	1294QK	Midhurst	010-599599	(NULL)
104	Moorman	D	1970-05-10	F	1,984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1,984	Vixen Road	8	6392LK	Plymouth	010-546745	1319
200	Siel	Ss	(NULL)	F	0	(NULL)	(NULL)	(NULL)	Madura	(NULL)	(NULL)

players (12x8)											
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONE NO	LEAGUENO
2	Everett	R	1970-09-01	M	1,975	Stoney Road	43	3575NH	Stratford	070-237893	2411
6	Parmenter	R	1964-06-25	M	1,977	Haseline Lane	80	123KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1,981	Edgecombe Way	39	9758VB	Stratford	070-317689	(NULL)
39	Bishop	D	1956-10-29	M	1,989	Eaton Square	78	9628CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1,990	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1,995	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
95	Miller	P	1963-05-14	M	1,972	High Street	33A	5746CP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1,979	Haseline Lane	80	6494SG	Stratford	070-494993	6524

Contoh Stored Procedure IN (Definisi dengan > 1 parameter)

- **Contoh 4:** Buatlah stored procedure untuk menampilkan jenis kelamin dan kota yang inputannya ditentukan oleh user dengan lebih parameter IN.

```
DELIMITER $$  
create procedure GETjkt( IN kota VARCHAR(20), IN jk  
VARCHAR(1))  
begin  
    select * from players WHERE SEX = jk AND TOWN = kota;  
END$$  
DELIMITER ;
```

Perintah Untuk memanggilnya :

```
call  
GETjkt ('Stratford', 'm');
```

The screenshot shows the MySQL Workbench interface. At the top, there is a command line with the following text:
1 call GETjkt('Stratford', 'm');
Below the command line is a results grid titled "players (12x6)". The grid displays 12 rows of data from the "players" table, filtered by the stored procedure. The columns are labeled: PLAYENO, NAME, INITIALS, BIRTH_DATE, SEX, JOINED, STREET, HOUSENO, POSTCODE, TOWN, PHONENO, and LEAGUENO. The data includes various names like Everett, Parmenter, Wise, Bishop, Brown, and Parmenter, along with their respective details such as birth dates (e.g., 1970-09-01, 1964-06-25) and addresses (e.g., Stoney Road, Haseltine Lane).

PLAYENO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1970-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524

Contoh Stored Procedure IN (Penambahan Data)

- Contoh 5: Buatlah stored procedure untuk memasukan data pada tabel teams yang inputan operasi penambahan, data – data terkait diisikan melalui argumen

```
DELIMITER $$  
create procedure AddTeam( IN TEAMNO  
SMALLINT(10), IN PLAYERO NO SMALLINT(10), IN  
DIVISION VARCHAR(6))  
BEGIN  
    insert into teams values (TEAMNO, PLAYERO NO,  
DIVISION);  
END $$  
DELIMITER ;
```

Perintah Untuk memanggilnya :

```
call AddTeam(4,57,'first');
```

1 SELECT * from teams;



teams (3x4)		
TEAMNO	PLAYERO NO	DIVISION
1	6	first
2	27	second
3	100	third
81	100	third

1 SELECT * from teams;



teams (3x5)		
TEAMNO	PLAYERO NO	DIVISION
1	6	first
2	27	second
3	100	third
4	57	first
81	100	third

Contoh Stored Procedure

Parameter Out

- **Contoh 6:** Buatlah stored procedure untuk menghitung jumlah pemain yang tercatat di dalam tabel PLAYERS, dengan parameter OUT procedure dapat mengubah parameter dan mengirimkan kembali ke program pemanggil.

```
DELIMITER $$  
create procedure JumlahPlayers(OUT jumlah_players INT(3))  
begin  
    select count(playerno) into jumlah_players from players;  
END$$  
DELIMITER ;
```

Perintah Untuk memanggilnya :

```
call JumlahPlayers(@jumlah_players);  
select @jumlah_players;
```

@jumlah_players

Contoh Stored Procedure

Parameter Out

- **Contoh 7:** Buatlah stored procedure untuk menghitung nilai tertinggi dalam tabel Penalties, dengan parameter OUT.

```
DELIMITER $$  
create PROCEDURE ambilPlay(OUT  
besar INT(20))  
begin  
select max(amount) into besar FROM  
penalties;  
END$$  
DELIMITER ;
```

1 SELECT * FROM penalties;

penalties (4x8)			
PAYMENTNO	PLAYERNO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100,00
2	44	1981-05-05	75,00
3	27	1983-09-10	100,00
4	104	1984-12-08	50,00
5	44	1980-12-08	25,00
6	8	1980-12-08	25,00
7	44	1982-12-30	30,00
8	27	1984-11-12	75,00

Perintah Untuk memanggilnya :

```
CALL ambilPlay(@besar);  
SELECT @besar;
```

@besar

100

Contoh Stored Procedure

2 Parameter OUT

- **Contoh 8:** Buatlah stored procedure untuk menghitung nilai tertinggi dalam tabel Penalties, dengan 2 parameter OUT.

```
DELIMITER $$  
create PROCEDURE hitungamount(OUT besar INT(20), OUT rata2  
DECIMAL(7,2))  
begin  
select max(amount), avg(amount) into besar, rata2 FROM  
penalties;  
END$$  
DELIMITER ;
```

Perintah Untuk memanggilnya :

```
CALL hitungamount(@besar, @rata2);  
SELECT @besar, @rata2;
```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the stored procedure definition. Below it, a results window titled "Hasil #1 (2x1)" displays the output of the procedure call. The result is a single row with two columns: "@besar" and "@rata2". The value for "@besar" is 100, and the value for "@rata2" is 60,00.

@besar	@rata2
100	60,00

Contoh Stored Procedure

Parameter INOut

- **Contoh 9:** Buatlah stored procedure untuk menghitung jumlah pemain yang tercatat di dalam tabel PLAYERS yang berjenis kelamin laki-laki atau perempuan yang inputannya dari argument, menggunakan parameter INOUT.

```
DELIMITER $$  
create procedure CountByGender(  
IN gender VARCHAR(2), OUT total INT(3))  
begin  
select count(playerno) into total from players where sex = gender;  
END$$  
DELIMITER ;
```

Perintah Untuk memanggilnya :

```
call CountByGender ('M',@total);  
SELECT @total;
```

The screenshot shows the MySQL Workbench interface with two tabs. The top tab is titled 'Hasil #1 (1x1)' and contains the SQL command: '1 call CountByGender('M',@total); 2 SELECT @total;'. The bottom tab is titled '@total' and displays the result: '8'.

Next Trigger

5) Kontrak Perkuliahan

- a). Tujuan Perkuliahan
- b). Metode Pengajaran
- c). Metode Penilaian
- d). Tugas dan Projek

Learning Outcomes

Diharapkan mahasiswa mampu:

- Merancang Dan Memodelkan Basis Data
- Melakukan Desain Database Dengan Benar
- Menggunakan Bahasa Query Dan Menjelaskan Konsep Pemrosesan Query
- Menyusun Stored Procedure Dan Trigger Yang Optimal
- Menerapkan Atau Mengimplementasi SMBD Pada Aplikasi Yang Sesuai.

Metode Pengajaran

□ Tatap muda di kelas & Praktikum

- Memberikan framework atau roadmap untuk mengorganisasi informasi mengenai perkuliahan
- Menjelaskan subjek dan perkuat gagasan besar yang penting
- Mengimplementasikan hasil perkuliahan pada praktikum di laboratorium.

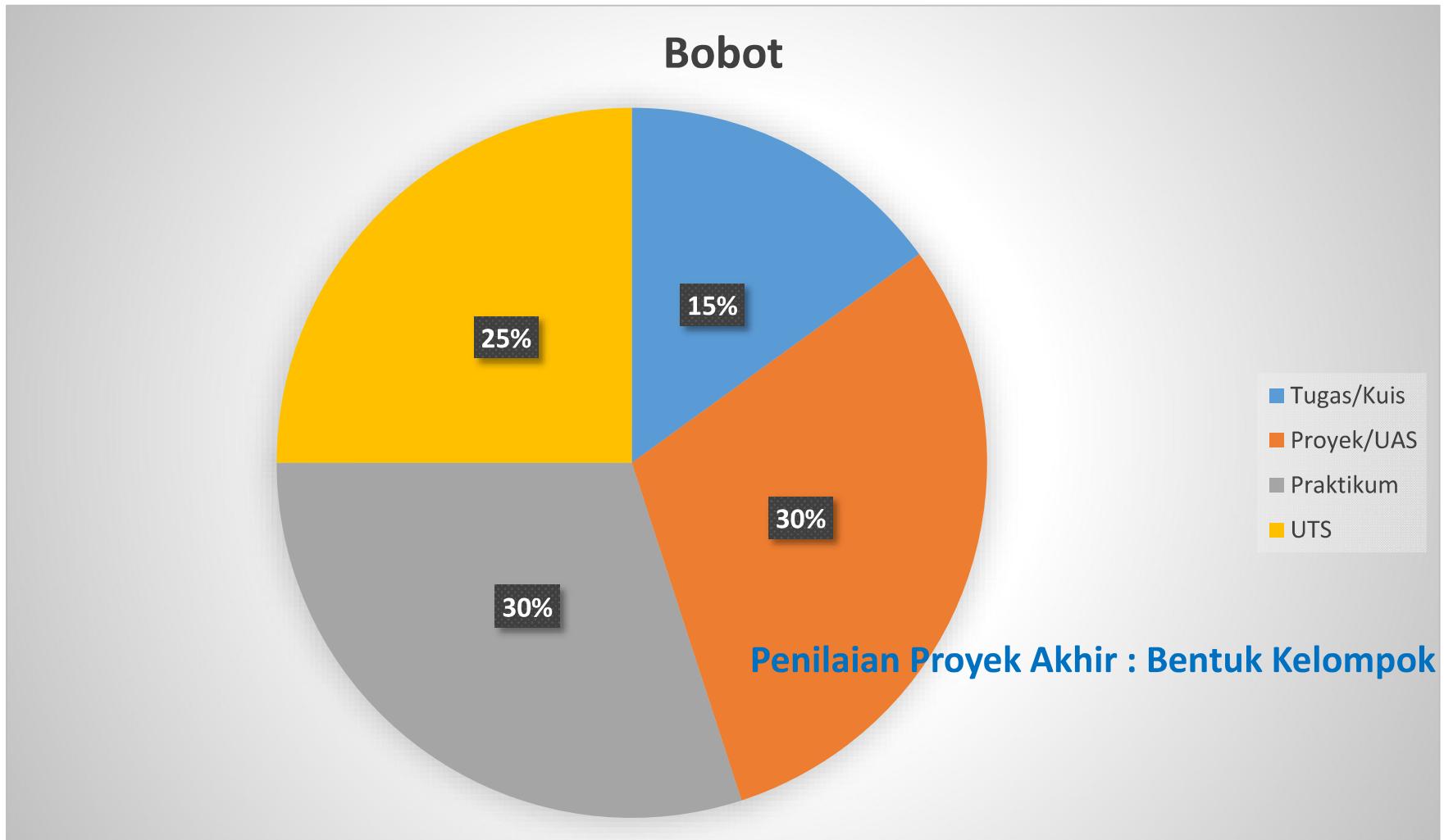
□ Bimbingan dan Arahan

- Meminta mahasiswa mengungkapkan apa yang belum dimengerti, sehingga Dosen dapat membantunya
- Mempersilakan mahasiswa mempraktikkan keterampilan yang diperlukan untuk menguasai penerapannya

Tata Tertib Perkuliahan

- Masuk sesuai jadwal 7.15 WIB, Toleransi keterlambatan adalah 15 menit.
- PAKAIAN** bebas **RAPI, BERKERAH, berSEPATU.**
- Setiap mahasiswa **TIDAK DIPERKENANKAN MENCONTEK, PLAGIAT**, dalam pengerojaan tugas dan ujian, jika terjadi maka pengerojaan tugas dan ujian akan dikurangi 20% atau Gugur.
- Setiap mahasiswa **WAJIB MENGIKUTI UJIAN** dan **TUGAS** baik tugas **MANDIRI, berKELOMPOK** atau **PRAKTIKUM.**
- Wajib untuk **BERTUTUR KATA** yang **SOPAN** dan **SANTUN di DALAM KELAS.**

Metode Penilaian



Tugas

- ❑ Tugas personal akan diberikan pada waktu perkuliahan
- ❑ Untuk pelaksanaan praktikum dilaksanakan berbarengan dengan waktu perkuliahan sesuai dengan jadwal pada lab yang digunakan.

Proyek Akhir

- Membuat aplikasi sederhana dengan fokus **Penerapan Database** ke Aplikasi untuk menyimpan transaksi
- **Tahapannya :**
 - Penentuan Studi Kasus
 - Perancangan Database beserta Relasi Tabelnya
 - Pada database terdapat beberapa SQL Langguage yang dilakukan diantaranya : CRUD, Transactions, Function, Stored Procedure & Trigger, System Catalog hingga hak akses.
 - Untuk Aplikasi boleh Web atau Desktop, fokus pada penerapan Database.
 - Pembuatan Laporan atau Dokumentasi.
- **Poin penilaian:** Aplikasi (Penerapan Database), Dokumentasi, Presentasi.

6) Kebutuhan Software

Kebutuhan Software

Browser

- Adobe flash
- Chrome
- Firefox

Localserver

- Xampp
- Laragon

Desain Tools

- Power Designer
- Sparx Enterprise Architect

Editor

- Notepad++
- Sublime Text

Database GUI

- PostgreSQL
- HeidiSQL
- SQLYog
- FlySpeed SQL

Database

- Mysql
- Oracle

7) Contact

Contact

- ❑ Bahan Kuliah : github.com/doniaft
- ❑ Email : doniaft@gmail.com
- ❑ WA/Telegram :
- ❑ Komting SMBD SI4A Romi : [0857 0681 7980](tel:085706817980)

8) Referensi

Referensi (I)

- Raghu Ramakhrisnan, Johannes Gehrke , “Database Management System” 3rd Edition, Mc Graw Hill,2003.
- Rick van der Lans, Introduction to SQL, Mastering Relational Database Language 2nd Edition, Addison-Wesley, 2000.
- Chris Bates, Web Programming: Building Internet Applications, Third Edition, John Wiley & Sons Ltd, England, 2006.
- Sebesta, R.W., Programming the World Wide Web, Addison Wesley, 2002.
- Elliot White III, Jonathan Eisenhamer, PHP 5 in Practice, Sams, 2006.
- SQL For MySQL Developers, Rick F. van der Lans, Addison Wesley, 2007
- MySQL Reference Manual, MySQL 2003
- Database Systems - A Practical Approach to Design, Implementation, and Management, Thomas Connoly and Carolyn Begg, Addison Wesley 1999