

SISTEM MANAJEMEN BASIS DATA

09. Trigger 11. System Catalog

Doni Abdul Fatah
github.com/doniaft
Universitas Trunojoyo Madura

Pokok Bahasan

01. Overview SMDB– 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) System Catalog
- 2) Trigger
- 3) Kontrak Perkuliahan
- 4) Kebutuhan Software
- 5) Contact
- 6) Referensi

Trigger

Stored Procedure vs Trigger

❑ **Persamaan:**

- Obyeknya dapat berupa TABEL, 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)



Trigger

- ❑ Kumpulan kode yang terdiri dari procedural dan declarative statements
- ❑ Tersimpan dalam katalog dan
- ❑ Diaktifkan oleh server database jika operasi yang spesifik dieksekusi dalam database dan jika ada kondisi yang ditentukan



Trigger

- **BEFORE INSERT** – activated before data is inserted into the table.
- **AFTER INSERT** – activated after data is inserted into the table.
- **BEFORE UPDATE** – activated before data in the table is updated.
- **AFTER UPDATE** – activated after data in the table is updated.
- **BEFORE DELETE** – activated before data is removed from the table.
- **AFTER DELETE** – activated after data is removed from the table.

Mengakses Nilai Baru dan Lama

- ❑ Dalam trigger untuk mengakses data lama dan data baru, data lama dapat direference dengan record OLD dan data baru dapat direference dengan Record NEW.

OPERASI	NEW (READ/WRITE)	OLD (READ)
INSERT	√	
UPDATE	√	√
DELETE		√

- ❑ Untuk mengacu ke sebuah field dapat ditulis dengan NEW.namafiled atau OLD.namafiled.

Trigger (Contd-2)

```
<create trigger statement> ::=  
CREATE TRIGGER <trigger name>  
<trigger moment>  
<trigger event>  
[ <trigger condition> ]  
<trigger action>
```

```
<trigger moment> ::=  
BEFORE | AFTER | INSTEAD OF
```

```
<trigger event> ::=  
{ INSERT | DELETE | UPDATE [ OF <column list> ] }  
{ ON | OF | FROM | INTO } <table specification>  
[ REFERENCING { OLD | NEW | OLD_TABLE | NEW_TABLE }  
AS <variable> ] FOR EACH { ROW | STATEMENT }
```

```
<trigger condition> ::= ( WHEN <condition> )  
<trigger action> ::= <begin-end BLOCK>
```

Contoh Trigger

Contoh 1: Buatlah tabel CHANGES!

```
CREATE TABLE CHANGES
    (USER CHAR(30) NOT NULL,
    CHA_TIME TIMESTAMP NOT NULL,
    CHA_PLAYERNO SMALLINT NOT NULL,
    CHA_TYPE CHAR(1) NOT NULL,
    CHA_PLAYERNO_NEW INTEGER,
    PRIMARY KEY (USER, CHA_TIME,
    CHA_PLAYERNO, CHA_TYPE)) ;
```

Contoh Trigger Lanjutan

Contoh 2: Buatlah trigger yang akan meng-update tabel CHANGES secara otomatis setiap kali ada penambahan baris baru di dalam tabel PLAYERS!

```
DELIMITER $$

CREATE TRIGGER INSERT_PLAYERS
AFTER INSERT ON PLAYERS FOR EACH ROW
BEGIN
INSERT INTO CHANGES
(USER, CHA_TIME, CHA_PLAYERNO, CHA_TYPE, CHA_PLAYERNO_NEW)
VALUES (USER, CURDATE(), NEW.PLAYERNO, "I", NULL);

END$$
DELIMITER ;
```

Contoh Trigger Lanjutan

Contoh 4: Buatlah trigger yang meng-update tabel CHANGES setiap kali ada baris di dalam tabel PLAYERS yang dihapus!

```
DELIMITER $$

CREATE TRIGGER DELETE_PLAYER
AFTER DELETE ON PLAYERS FOR EACH ROW
BEGIN
    CALL INSERT_CHANGE (OLD.PLAYERNO, 'D', NULL);

END$$
DELIMITER ;
```

Contoh Trigger Lanjutan

Contoh 5: Buatlah trigger yang meng-update tabel CHANGES setiap kali ada baris di dalam tabel PLAYERS yang di-update!

```
DELIMITER $$
```

```
CREATE TRIGGER UPDATE_PLAYER
```

```
AFTER UPDATE ON PLAYERS FOR EACH ROW
```

```
BEGIN
```

```
CALL INSERT_CHANGES (NEW.PLAYERNO, 'U', OLD.PLAYERNO);
```

```
END$$
```

```
DELIMITER ;
```

Contoh Trigger Lanjutan

Contoh 6 : Buat trigger untuk menyimpan history division, jika division berubah, maka division lama harus disimpan ke tabel history division.

```
DELIMITER $$
DROP TRIGGER IF EXISTS
coba_update_teams$$
CREATE TRIGGER coba_update_teams
AFTER UPDATE ON teams FOR EACH ROW
BEGIN
INSERT INTO history_div_teams VALUES
(NOW(), old.teamno, old.division,
USER());
END$$
DELIMITER ;
```

Contoh Penggunaan Trigger

```
UPDATE teams SET division =
"thrid" WHERE teamno=1;
```

```
SELECT * FROM teams;
SELECT * FROM history_div_teams;
```

waktu	teamno	playerno	division	oleh
2019-04-11 20:04:49	1	(NULL)	first	root@localhost
2019-04-11 20:05:09	1	(NULL)	first	root@localhost
2019-04-11 20:05:24	1	(NULL)	second	root@localhost
2019-04-11 20:26:20	1	(NULL)	thrid	root@localhost
2019-04-11 20:26:26	1	(NULL)	coba	root@localhost
2019-04-11 20:26:30	1	(NULL)	lagi	root@localhost
2019-04-11 20:31:14	1	(NULL)	sekali	root@localhost
2019-04-11 20:33:12	1	(NULL)	firste	root@localhost

Untuk melihat semua division yang pernah digunakan oleh teams yang bernomor=1

```
(SELECT NOW() waktu, teamno, division
FROM teams WHERE teamno=1)
UNION
(SELECT waktu, teamno, division FROM
history_div_teams WHERE teamno=1)
ORDER BY waktu DESC;
```

2019-04-11 20:26:20	1	thrid
2019-04-11 20:05:24	1	second
2019-04-11 20:05:09	1	first
2019-04-11 20:04:49	1	first

Contoh Trigger Lanjutan

Trigger yang pertama mempunyai kekurangan yaitu ketika ada perubahan di tabel teams walaupun tdk mengubah kolom division, maka statement INSERT di tabel history akan dijalankan.

```
DELIMITER $$
DROP TRIGGER if EXISTS
coba_update_teams$$
CREATE TRIGGER coba_update_teams
AFTER UPDATE ON teams FOR EACH ROW
BEGIN
  if old.division <> new.division
  then
    INSERT INTO history_div_teams
    VALUES (NOW(), old.teamno,
    old.division, USER());
  END if;
END$$
DELIMITER ;
```

Contoh Penggunaan Trigger

```
UPDATE teams SET division =
"sekali" WHERE teamno=1;
```

```
UPDATE teams SET division =
"firste" WHERE teamno=1;
```

```
UPDATE teams SET division =
"first" WHERE teamno=1;
```

```
SELECT * FROM teams;
```

```
SELECT * from history_div_teams;
```

Untuk melihat semua division yang pernah digunakan oleh teams yang bernomor =1

```
(SELECT NOW() waktu, teamno,
division FROM teams WHERE
teamno=1)
```

```
UNION
```

```
(SELECT waktu, teamno, division
FROM history_div_teams WHERE
teamno=1)
```

```
ORDER BY waktu DESC;
```

Contoh Trigger Lanjutan

Contoh 7 : Buat trigger akan dieksekusi ketika ada perubahan teamno di tabel teams yang akan melakukan update ke tabel history_div_teams untuk menyesuaikan Teamno agar relasi tidak terlepas.

```
DELIMITER $$
DROP TRIGGER if EXISTS coba_update_teams$$
CREATE TRIGGER coba_update_teams
AFTER UPDATE ON teams FOR EACH ROW
BEGIN
    if old.division <> new.division then
        INSERT INTO history_div_teams VALUES (NOW(),
        old.teamno, old.division, USER());
    END if;
    if old.teamno <> new.teamno then
        UPDATE history_div_teams SET
        teamno=new.teamno WHERE teamno=old.teamno;
    END if;
END$$
DELIMITER ;
```

Contoh Penggunaan Trigger

```
ALTER table history_div_teams
ADD COLUMN playerno
SMALLINT(6) AFTER teamno;
```

```
SELECT * from history_div_teams;
```


Contoh Trigger Lanjutan

Contoh 7 : Buat trigger akan dieksekusi ketika ada perubahan teamno di tabel teams yang akan melakukan update ke tabel history_div_teams untuk menyekusiakan Teamno agar relasi tidak terlepas.

```
DELIMITER $$
DROP TRIGGER if EXISTS coba_update_teams$$
CREATE TRIGGER coba_update_teams
AFTER UPDATE ON teams FOR EACH ROW
BEGIN
    if old.division <> new.division then
        INSERT INTO history_div_teams VALUES (NOW(),
        old.teamno, old.division, USER());
    END if;
    if old.playerno <> new.playerno then
        UPDATE history_div_teams SET
        playerno=new.playerno WHERE
        playerno=old.playerno;
    END if;
END$$
DELIMITER ;
```

Contoh Penggunaan Trigger

```
UPDATE teams SET teamno
= 111 WHERE teamno=1;
```

```
UPDATE teams SET
playerno =111 WHERE
playerno=6;
```

11. System Catalog



Sub Pokok Bahasan

- ☐ *System Catalog*
- ☐ *Databases & The Catalog*
- ☐ *Tables & The Catalog*
- ☐ *Views & The Catalog*
- ☐ *User Authorisation & The Catalog*
- ☐ *Stored Procedure & The Catalog*
- ☐ *Trigger & The Catalog*

System Catalog

- ❑ MySQL menyimpan informasi mengenai *system catalog* (= catalog; metadata) di dalam tabel-tabel basisdata **INFORMATION_SCHEMA**
- ❑ Query yang dapat dilakukan pada **INFORMATION_SCHEMA** hanya pernyataan **SELECT (no UPDATE and DELETE!)** = **read-only tables = views tables**
- ❑ Query pada tabel-tabel catalog dapat dilakukan dengan **tujuan** sebagai:
 - ❑ **Fungsi Bantuan:** user baru dapat menemukan tabel mana saja yang dimiliki oleh suatu basisdata dan kolom apa saja yang dimiliki oleh suatu tabel
 - ❑ **Fungsi Kontrol:** user dapat melihat, misalnya, daftar views, hak akses, atau index mana saja yang akan terhapus apabila tabel tertentu dihapus
 - ❑ **Fungsi Pemrosesan/Fungsi Bantuan** untuk MySQL ketika mengeksekusi suatu pernyataan

Databases & The Catalog

- ❑ Informasi mengenai BASISDATA disimpan oleh INFORMATION_SCHEMA di dalam tabel SCHEMATA:

```
MariaDB [(none)]> desc information_schema.schemata;
```

Field	Type	Null	Key	Default	Extra
CATALOG_NAME	varchar(512)	NO			
SCHEMA_NAME	varchar(64)	NO			
DEFAULT_CHARACTER_SET_NAME	varchar(32)	NO			
DEFAULT_COLLATION_NAME	varchar(32)	NO			
SQL_PATH	varchar(512)	YES		NULL	

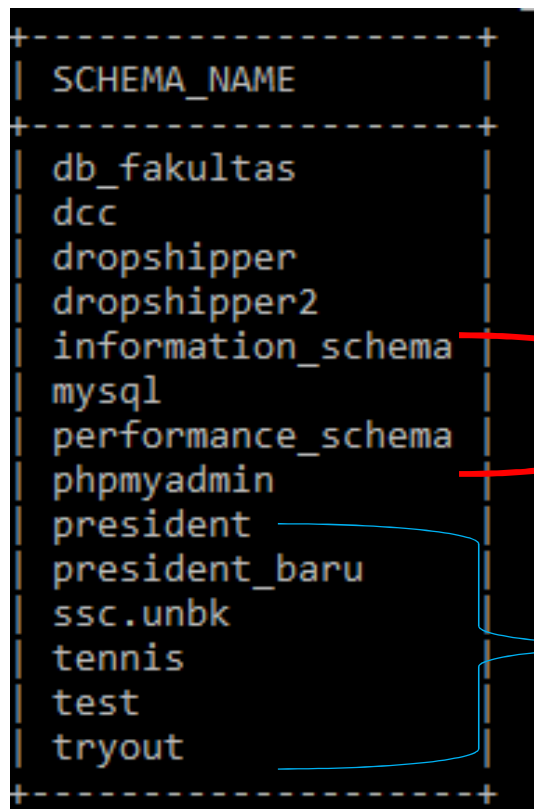
```
5 rows in set (0.03 sec)
```

Perintah di atas sama **show databases** pada mysql

Databases & The Catalog

- ❑ **Contoh 1:** Buatlah daftar seluruh basisdata yang ada di dalam MySQL Server!

```
SELECT schema_name FROM information_schema.schemata;
```



SCHEMA_NAME
db_fakultas
dcc
dropshipper
dropshipper2
information_schema
mysql
performance_schema
phpmyadmin
president
president_baru
ssc.unbk
tennis
test
tryout

default (created during instalation) – DO NOT DELETE THEM!

dibuat oleh user

Databases & The Catalog

- ❑ **Contoh 2:** Untuk masing-masing basisdata yang ada, carilah nama basisdata, default character set dan default collation-nya!

```
MariaDB [(none)]> SELECT SCHEMA_NAME, DEFAULT_CHARACTER_SET_NAME,  
-> DEFAULT_COLLATION_NAME  
-> FROM INFORMATION_SCHEMA.SCHEMATA;
```

SCHEMA_NAME	DEFAULT_CHARACTER_SET_NAME	DEFAULT_COLLATION_NAME
db_fakultas	latin1	latin1_swedish_ci
dcc	latin1	latin1_swedish_ci
dropshipper	latin1	latin1_swedish_ci
dropshipper2	latin1	latin1_swedish_ci
information_schema	utf8	utf8_general_ci
mysql	latin1	latin1_swedish_ci
performance_schema	utf8	utf8_general_ci
phpmyadmin	utf8	utf8_bin
president	latin1	latin1_swedish_ci
president_baru	latin1	latin1_swedish_ci
ssc.unbk	latin1	latin1_swedish_ci
tennis	latin1	latin1_swedish_ci
test	latin1	latin1_swedish_ci
tryout	latin1	latin1_swedish_ci

```
14 rows in set (0.00 sec)
```

Databases & The Catalog

- ❑ **Contoh 3:** Untuk masing-masing basisdata yang ada, carilah nama database, table, dan columnnya!

```
MariaDB [(none)]> select table_schema, table_name, column_name from information_schema.columns;
```

table_schema	table_name	column_name
db_fakultas	contoh_at	no
db_fakultas	contoh_at	nama
db_fakultas	contoh_at	umur
db_fakultas	contoh_at	kodepos
db_fakultas	contoh_bin	bin
db_fakultas	contoh_bin	varbin
db_fakultas	contoh_cha	cha
db_fakultas	contoh_cha	varcha
db_fakultas	contoh_dec	satuan
db_fakultas	contoh_dec	puluhan
db_fakultas	contoh_dec	ribuan
db_fakultas	contoh_dec	normal
db_fakultas	contoh_dec	tambah
db_fakultas	contoh_float	satuan
db_fakultas	contoh_float	puluhan
db_fakultas	contoh_float	ribuan
db_fakultas	contoh_float	positif
db_fakultas	contoh_float	tambah
db_fakultas	contoh_int	mini
db_fakultas	contoh_int	kecil
db_fakultas	contoh_int	sedang
db_fakultas	contoh_int	biasa

Tables & The Catalog

- ❑ Untuk mendapatkan Informasi mengenai TABEL dan KOLOM disimpan oleh INFORMATION_SCHEMA
- ❑ Ada 2 cara untuk mendapatkannya **(1) melalui table information_schema.tables,** atau **(2) information_schema.columns.**
- ❑ Perbedaannya pada **table** information_schema.tables tidak terdapat nama column dari table yang ada. Sehingga penggunaannya disesuaikan dengan kebutuhan.

Tables & The Catalog

❑ Deskripsi mengenai tabel catalog TABLES:

```
MariaDB [(none)]> desc information_schema.tables;
```

Field	Type	Null	Key	Default	Extra
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
TABLE_TYPE	varchar(64)	NO			
ENGINE	varchar(64)	YES		NULL	
VERSION	bigint(21) unsigned	YES		NULL	
ROW_FORMAT	varchar(10)	YES		NULL	
TABLE_ROWS	bigint(21) unsigned	YES		NULL	
AVG_ROW_LENGTH	bigint(21) unsigned	YES		NULL	
DATA_LENGTH	bigint(21) unsigned	YES		NULL	
MAX_DATA_LENGTH	bigint(21) unsigned	YES		NULL	
INDEX_LENGTH	bigint(21) unsigned	YES		NULL	
DATA_FREE	bigint(21) unsigned	YES		NULL	
AUTO_INCREMENT	bigint(21) unsigned	YES		NULL	
CREATE_TIME	datetime	YES		NULL	
UPDATE_TIME	datetime	YES		NULL	
CHECK_TIME	datetime	YES		NULL	
TABLE_COLLATION	varchar(32)	YES		NULL	
CHECKSUM	bigint(21) unsigned	YES		NULL	
CREATE_OPTIONS	varchar(2048)	YES		NULL	
TABLE_COMMENT	varchar(2048)	NO			

```
21 rows in set (0.02 sec)
```

Tables & The Catalog

❑ Deskripsi mengenai tabel catalog COLUMNS:

```
MariaDB [(none)]> desc information_schema.columns;
```

Field	Type	Null	Key	Default	Extra
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
COLUMN_NAME	varchar(64)	NO			
ORDINAL_POSITION	bigint(21) unsigned	NO		0	
COLUMN_DEFAULT	longtext	YES		NULL	
IS_NULLABLE	varchar(3)	NO			
DATA_TYPE	varchar(64)	NO			
CHARACTER_MAXIMUM_LENGTH	bigint(21) unsigned	YES		NULL	
CHARACTER_OCTET_LENGTH	bigint(21) unsigned	YES		NULL	
NUMERIC_PRECISION	bigint(21) unsigned	YES		NULL	
NUMERIC_SCALE	bigint(21) unsigned	YES		NULL	
DATETIME_PRECISION	bigint(21) unsigned	YES		NULL	
CHARACTER_SET_NAME	varchar(32)	YES		NULL	
COLLATION_NAME	varchar(32)	YES		NULL	
COLUMN_TYPE	longtext	NO		NULL	
COLUMN_KEY	varchar(3)	NO			
EXTRA	varchar(27)	NO			
PRIVILEGES	varchar(80)	NO			
COLUMN_COMMENT	varchar(1024)	NO			

```
20 rows in set (0.02 sec)
```

Tables & The Catalog

- ❑ **Contoh 4:** Buatlah daftar nama-nama tabel yang ada di dalam basisdata TENNIS!

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_SCHEMA = 'tennis' ORDER BY TABLE_NAME;
```

```
MariaDB [(none)]> SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES  
-> WHERE TABLE_SCHEMA = 'tennis' ORDER BY TABLE_NAME;
```

TABLE_NAME
changes
changes2
committee_members
cplayers
digits
history_div_teams
hitung
matches
penalties
players
players_x
players_z
recre_players
several
several2
several21
several4
sex_female
sum_penalties
teams
towns
veterans

```
22 rows in set (0.00 sec)
```

Tables & The Catalog

Contoh 5: Buatlah daftar nama, tipe data, is_nullable, dan nomor urutan dari seluruh kolom yang ada di dalam tabel PLAYERS(dalam basisdata TENNIS); urutkan berdasarkan nomor urutannya!

```
SELECT
COLUMN_NAME,
DATA_TYPE,
IS_NULLABLE,
ORDINAL_POSITION
FROM
INFORMATION_SCHEMA
.COLUMNS WHERE
TABLE_NAME
='PLAYERS'
AND TABLE_SCHEMA =
'tennis' ORDER BY
ORDINAL_POSITION;
```

```
MariaDB [(none)]> SELECT COLUMN_NAME, DATA_TYPE, IS_NULLABLE, ORDINAL_POSITION
-> FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'PLAYERS'
-> AND TABLE_SCHEMA = 'tennis' ORDER BY ORDINAL_POSITION;
```

COLUMN_NAME	DATA_TYPE	IS_NULLABLE	ORDINAL_POSITION
PLAYERNO	smallint	NO	1
NAME	char	NO	2
INITIALS	char	NO	3
BIRTH_DATE	date	YES	4
SEX	char	NO	5
JOINED	smallint	NO	6
STREET	char	NO	7
HOUSENO	char	YES	8
POSTCODE	char	YES	9
TOWN	char	NO	10
PHONENO	char	YES	11
LEAGUENO	char	YES	12

```
12 rows in set (0.01 sec)
```

Tables & The Catalog

❑ **Contoh 6:** Carilah jumlah kolom dari masing-masing tabel yang ada di dalam basisdata TENNIS!

```
SELECT 'PLAYERS' AS TABLE_NAME, (SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='PLAYERS' AND TABLE_SCHEMA='tennis') AS
NUMBER_COLUMNS
UNION
SELECT 'TEAMS', (SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='TEAMS' AND TABLE_SCHEMA='tennis')
UNION
SELECT 'PENALTIES', (SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='PENALTIES' AND TABLE_SCHEMA='tennis')
UNION
SELECT 'MATCHES', (SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='MATCHES' AND TABLE_SCHEMA='tennis')
UNION
SELECT 'COMMITTEE_MEMBERS', (SELECT COUNT(*) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='COMMITTEE_MEMBERS' AND
TABLE_SCHEMA='tennis')
ORDER BY 1;
```

```
MariaDB [(none)]> SELECT 'PLAYERS' AS TABLE_NAME, (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME='PLAYERS' AND TABLE_SCHEMA='tennis') AS
-> NUMBER_COLUMNS
-> UNION
-> SELECT 'TEAMS', (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME='TEAMS' AND TABLE_SCHEMA='tennis')
-> UNION
-> SELECT 'PENALTIES', (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME='PENALTIES' AND TABLE_SCHEMA='tennis')
-> UNION
-> SELECT 'MATCHES', (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME='MATCHES' AND TABLE_SCHEMA='tennis')
-> UNION
-> SELECT 'COMMITTEE_MEMBERS', (SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME='COMMITTEE_MEMBERS' AND TABLE_SCHEMA='tennis')
-> ORDER BY 1;
```

TABLE_NAME	NUMBER_COLUMNS
COMMITTEE_MEMBERS	4
MATCHES	5
PENALTIES	4
PLAYERS	12
TEAMS	3

5 rows in set (0.16 sec)

TABLE_NAME	NUMBER_COLUMNS
COMMITTEE_MEMBERS	4
MATCHES	5
PENALTIES	4
PLAYERS	12
TEAMS	3

5 rows in set (0.16 sec)

Tables & The Catalog

- ❑ Pernyataan/perintah SHOW juga dapat digunakan untuk menampilkan data catalog
- ❑ **Contoh 7:** Tampilkan data deskriptif dari kolom-kolom yang ada di dalam tabel PLAYERS (dalam basisdata TENNIS)!

SHOW COLUMNS
FROM
`tennis.PLAYERS;`

```
MariaDB [(none)]> SHOW COLUMNS FROM tennis.PLAYERS;
```

Field	Type	Null	Key	Default	Extra
PLAYERNO	smallint(6)	NO	PRI	NULL	
NAME	char(15)	NO		NULL	
INITIALS	char(3)	NO		NULL	
BIRTH_DATE	date	YES		NULL	
SEX	char(1)	NO		NULL	
JOINED	smallint(6)	NO		NULL	
STREET	char(15)	NO		NULL	
HOUSENO	char(4)	YES		NULL	
POSTCODE	char(6)	YES		NULL	
TOWN	char(10)	NO		NULL	
PHONENO	char(10)	YES		NULL	
LEAGUENO	char(4)	YES		NULL	

```
12 rows in set (0.01 sec)
```

Tables & The Catalog

- ❑ **Contoh 8:** Tampilkan pernyataan CREATE TABLE untuk tabel PLAYERS (dalam basisdata TENNIS)!

```
SHOW CREATE TABLE  
tennis.PLAYERS;
```

```
MariaDB [(none)]> SHOW CREATE TABLE tennis.PLAYERS;
```

```
+-----+-----+  
+-----+-----+  
+-----+-----+  
| Table   | Create Table  
+-----+-----+  
+-----+-----+  
+-----+-----+  
| PLAYERS | CREATE TABLE `players` (  
  `PLAYERNO` smallint(6) NOT NULL,  
  `NAME` char(15) NOT NULL,  
  `INITIALS` char(3) NOT NULL,  
  `BIRTH_DATE` date DEFAULT NULL,  
  `SEX` char(1) NOT NULL,  
  `JOINED` smallint(6) NOT NULL,  
  `STREET` char(15) NOT NULL,  
  `HOUSENO` char(4) DEFAULT NULL,  
  `POSTCODE` char(6) DEFAULT NULL,  
  `TOWN` char(10) NOT NULL,  
  `PHONENO` char(10) DEFAULT NULL,  
  `LEAGUENO` char(4) DEFAULT NULL,  
  PRIMARY KEY (`PLAYERNO`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |  
+-----+-----+  
+-----+-----+  
1 row in set (0.00 sec)
```


Views & The Catalog

□ Informasi mengenai VIEW disimpan oleh INFORMATION_SCHEMA di dalam tabel VIEWS

▪ Deskripsi mengenai tabel catalog VIEWS:

```
DESC INFORMATION_SCHEMA.VIEWS;
```

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.VIEWS;
```

Field	Type	Null	Key	Default	Extra
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
VIEW_DEFINITION	longtext	NO		NULL	
CHECK_OPTION	varchar(8)	NO			
IS_UPDATABLE	varchar(3)	NO			
DEFINER	varchar(189)	NO			
SECURITY_TYPE	varchar(7)	NO			
CHARACTER_SET_CLIENT	varchar(32)	NO			
COLLATION_CONNECTION	varchar(32)	NO			
ALGORITHM	varchar(10)	NO			

```
11 rows in set (0.03 sec)
```

User Authorisations & The Catalog

- ❑ Informasi mengenai User Authorisations disimpan Oleh INFORMATION_SCHEMA di dalam tabel USER_PRIVILEGES, SCHEMA_PRIVILEGES, TABLE_PRIVILEGES, dan COLUMN_PRIVILEGES
 - Deskripsi mengenai tabel catalog USER_PRIVILEGES:

DESC INFORMATION_SCHEMA.USER_PRIVILEGES:

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.USER_PRIVILEGES;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| GRANTEE        | varchar(190)  | NO   |     |         |       |
| TABLE_CATALOG | varchar(512)  | NO   |     |         |       |
| PRIVILEGE_TYPE  | varchar(64)   | NO   |     |         |       |
| IS_GRANTABLE    | varchar(3)    | NO   |     |         |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

User Authorisations & The Catalog

- ❑ Informasi mengenai User Authorisations disimpan Oleh INFORMATION_SCHEMA di dalam tabel USER_PRIVILEGES, SCHEMA_PRIVILEGES, TABLE_PRIVILEGES, dan COLUMN_PRIVILEGES
 - Deskripsi mengenai tabel catalog SCHEMA_PRIVILEGES:

```
DESC INFORMATION_SCHEMA.SCHEMA_PRIVILEGES;
```

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.SCHEMA_PRIVILEGES;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(190)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

```
5 rows in set (0.02 sec)
```

User Authorisations & The Catalog

- ❑ Informasi mengenai User Authorisations disimpan Oleh INFORMATION_SCHEMA di dalam tabel USER_PRIVILEGES, SCHEMA_PRIVILEGES, TABLE_PRIVILEGES, dan COLUMN_PRIVILEGES
 - Deskripsi mengenai tabel catalog TABLE_PRIVILEGES:

```
DESC INFORMATION_SCHEMA.TABLE_PRIVILEGES;
```

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.TABLE_PRIVILEGES;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(190)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

```
6 rows in set (0.03 sec)
```

User Authorisations & The Catalog

- ❑ Informasi mengenai User Authorisations disimpan Oleh INFORMATION_SCHEMA di dalam tabel USER_PRIVILEGES, SCHEMA_PRIVILEGES, TABLE_PRIVILEGES, dan COLUMN_PRIVILEGES
 - Deskripsi mengenai tabel catalog COLUMN_PRIVILEGES:
`DESC INFORMATION_SCHEMA.COLUMN_PRIVILEGES;`

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.COLUMN_PRIVILEGES;
```

Field	Type	Null	Key	Default	Extra
GRANTEE	varchar(190)	NO			
TABLE_CATALOG	varchar(512)	NO			
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
COLUMN_NAME	varchar(64)	NO			
PRIVILEGE_TYPE	varchar(64)	NO			
IS_GRANTABLE	varchar(3)	NO			

```
7 rows in set (0.02 sec)
```

Stored Procedure & Catalog

- ❑ Informasi mengenai Stored Procedure (dan Function) disimpan oleh INFORMATION_SCHEMA di dalam tabel ROUTINES.

```
DESC INFORMATION_SCHEMA.ROUTINES;
```

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.ROUTINES;
```

Field	Type	Null	Key	Default	Extra
SPECIFIC_NAME	varchar(64)	NO			
ROUTINE_CATALOG	varchar(512)	NO			
ROUTINE_SCHEMA	varchar(64)	NO			
ROUTINE_NAME	varchar(64)	NO			
ROUTINE_TYPE	varchar(9)	NO			
DATA_TYPE	varchar(64)	NO			
CHARACTER_MAXIMUM_LENGTH	int(21)	YES		NULL	
CHARACTER_OCTET_LENGTH	int(21)	YES		NULL	
NUMERIC_PRECISION	int(21)	YES		NULL	
NUMERIC_SCALE	int(21)	YES		NULL	
DATETIME_PRECISION	bigint(21) unsigned	YES		NULL	
CHARACTER_SET_NAME	varchar(64)	YES		NULL	
COLLATION_NAME	varchar(64)	YES		NULL	
DTD_IDENTIFIER	longtext	YES		NULL	
ROUTINE_BODY	varchar(8)	NO		EXTERNAL_LANGUAGE	varchar(64) YES NULL
ROUTINE_DEFINITION	longtext	YES		PARAMETER_STYLE	varchar(8) NO
EXTERNAL_NAME	varchar(64)	YES		IS_DETERMINISTIC	varchar(3) NO
EXTERNAL_LANGUAGE	varchar(64)	YES		SQL_DATA_ACCESS	varchar(64) NO
PARAMETER_STYLE	varchar(8)	NO		SQL_PATH	varchar(64) YES NULL
				SECURITY_TYPE	varchar(7) NO
				CREATED	datetime NO 0000-00-00 00:00:00
				LAST_ALTERED	datetime NO 0000-00-00 00:00:00
				SQL_MODE	varchar(8192) NO
				ROUTINE_COMMENT	longtext NO NULL
				DEFINER	varchar(189) NO
				CHARACTER_SET_CLIENT	varchar(32) NO
				COLLATION_CONNECTION	varchar(32) NO
				DATABASE_COLLATION	varchar(32) NO

31 rows in set (0.03 sec)

Stored Procedure & Catalog

❑ **Contoh 9:** Buatlah daftar kolom-kolom yang dimiliki oleh tabel ROUTINES!

```
SELECT COLUMN_NAME FROM  
INFORMATION_SCHEMA.COLUMNS  
WHERE TABLE_SCHEMA =  
'INFORMATION_SCHEMA'  
AND TABLE_NAME = 'ROUTINES'  
ORDER BY ORDINAL_POSITION;
```

```
MariaDB [(none)]> SELECT COLUMN_NAME FROM  
-> INFORMATION_SCHEMA.COLUMNS  
-> WHERE TABLE_SCHEMA = 'INFORMATION_SCHEMA'  
-> AND TABLE_NAME = 'ROUTINES'  
-> ORDER BY ORDINAL_POSITION;
```

COLUMN_NAME
SPECIFIC_NAME
ROUTINE_CATALOG
ROUTINE_SCHEMA
ROUTINE_NAME
ROUTINE_TYPE
DATA_TYPE
CHARACTER_MAXIMUM_LENGTH
CHARACTER_OCTET_LENGTH
NUMERIC_PRECISION
NUMERIC_SCALE
DATETIME_PRECISION
CHARACTER_SET_NAME
COLLATION_NAME
DTD_IDENTIFIER
ROUTINE_BODY
ROUTINE_DEFINITION
EXTERNAL_NAME
EXTERNAL_LANGUAGE
PARAMETER_STYLE
IS_DETERMINISTIC
SQL_DATA_ACCESS
SQL_PATH
SECURITY_TYPE
CREATED
LAST_ALTERED
SQL_MODE
ROUTINE_COMMENT
DEFINER
CHARACTER_SET_CLIENT
COLLATION_CONNECTION
DATABASE_COLLATION

```
31 rows in set (0.03 sec)
```

Stored Procedure & Catalog

- ❑ Pernyataan/perintah SHOW juga dapat digunakan untuk menampilkan data catalog
- ❑ **Contoh 10:** Carilah daftar karakteristik dari stored procedure DELETE_PLAYER!

```
SHOW PROCEDURE STATUS LIKE 'DELETE_PLAYER'
```


Stored Procedure & Catalog

❑ **Contoh 11:** Tampilkan pernyataan CREATE PROCEDURE untuk stored procedure DELETE_MATCHES!

SHOW CREATE PROCEDURE tennis.DELETE_MATCHES;

```
MariaDB [(none)]> SHOW CREATE PROCEDURE tennis.DELETE_MATCHES;
+-----+-----+-----+-----+
| Procedure | sql_mode | Create Procedure | character_set_client | collation_connection | Database Collation |
+-----+-----+-----+-----+
| DELETE_MATCHES | NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | CREATE DEFINER='root'@'localhost' PROCEDURE `DELETE_MATCHES` (IN P_PLAYERNO INTEGER) BEGIN DELETE FROM MATCHES WHERE PLAYERNO = P_PLAYERNO; END | utf8mb4 | utf8mb4_general_ci | latin1_swedish_ci |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Trigger & The Catalog

- ❑ Informasi mengenai TRIGGER disimpan oleh INFORMATION_SCHEMA di dalam tabel TRIGGERS
 - Deskripsi mengenai tabel catalog TRIGGERS:

DESC

INFORMATION_SCHEMA
.TRIGGERS;

```
MariaDB [(none)]> DESC INFORMATION_SCHEMA.TRIGGERS;
```

Field	Type	Null	Key	Default	Extra
TRIGGER_CATALOG	varchar(512)	NO			
TRIGGER_SCHEMA	varchar(64)	NO			
TRIGGER_NAME	varchar(64)	NO			
EVENT_MANIPULATION	varchar(6)	NO			
EVENT_OBJECT_CATALOG	varchar(512)	NO			
EVENT_OBJECT_SCHEMA	varchar(64)	NO			
EVENT_OBJECT_TABLE	varchar(64)	NO			
ACTION_ORDER	bigint(4)	NO		0	
ACTION_CONDITION	longtext	YES		NULL	
ACTION_STATEMENT	longtext	NO		NULL	
ACTION_ORIENTATION	varchar(9)	NO			
ACTION_TIMING	varchar(6)	NO			
ACTION_REFERENCE_OLD_TABLE	varchar(64)	YES		NULL	
ACTION_REFERENCE_NEW_TABLE	varchar(64)	YES		NULL	
ACTION_REFERENCE_OLD_ROW	varchar(3)	NO			
ACTION_REFERENCE_NEW_ROW	varchar(3)	NO			
CREATED	datetime	YES		NULL	
SQL_MODE	varchar(8192)	NO			
DEFINER	varchar(189)	NO			
CHARACTER_SET_CLIENT	varchar(32)	NO			
COLLATION_CONNECTION	varchar(32)	NO			
DATABASE_COLLATION	varchar(32)	NO			

```
22 rows in set (0.03 sec)
```

5) Kontrak Perkuliahan

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



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.

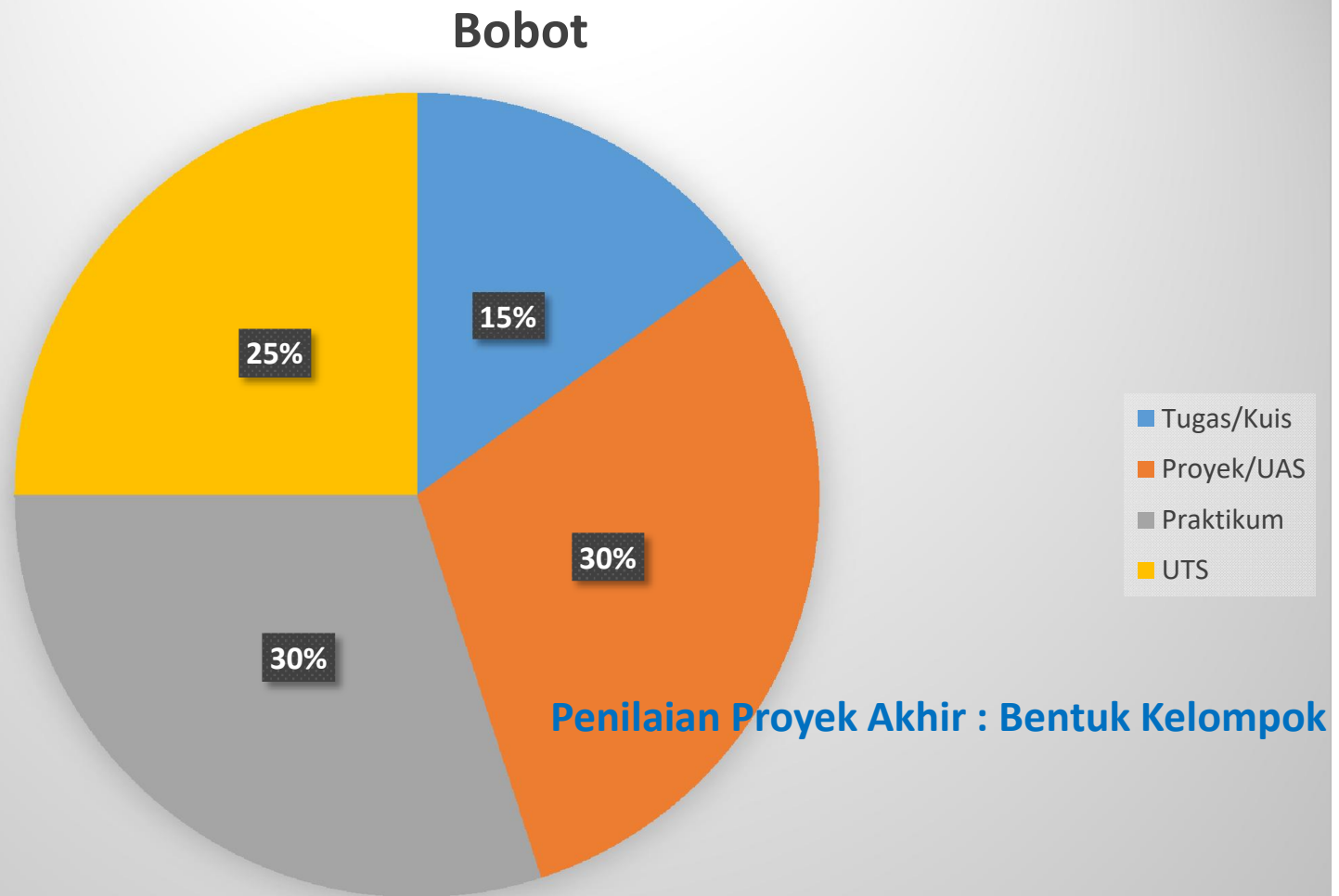
☐ **Bimbingan dan Arah**

- ☐ 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 pengerjaan tugas dan ujian, jika terjadi maka pengerjaan 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 Language 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

8) Referensi



Referensi (1)

- ❑ Raghu Ramakrishnan, 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 Connolly and Carolyn Begg, Addison Wesley 1999