

SISTEM MANAJEMEN BASIS DATA

- 06. Transactional SQL
- 11. System Catalog
- 12. Embedded SQL
- 13. Basis Data NoSQL

Doni Abdul Fatah
github.com/doniaft
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) System Catalog
- 2) Transactional SQL
- 3) Embedded SQL
- 4) Basis Data NoSQL
- 5) Kontrak Perkuliahan
- 6) Kebutuhan Software
- 7) Contact
- 8) Referensi

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

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   |   | NULL    |       |
| TABLE_SCHEMA  | varchar(64)  | NO   |   | NULL    |       |
| TABLE_NAME    | varchar(64)  | NO   |   | NULL    |       |
| TABLE_TYPE    | varchar(64)  | NO   |   | NULL    |       |
| 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   |   | NULL    |       |
+-----+-----+-----+-----+-----+
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  |  
| recr_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;
```

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;

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 |  
+-----+  
| PLAYERS |  
+-----+  
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;
```

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;
```

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;
```

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;
```

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_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;

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)

Transaction Sql

Definisi

- ❑ Transactions adalah sekelompok operasi sekuensial untuk memanipulasi data dalam database.
- ❑ Transactions akan berhasil jika setiap operasi individu dalam grup juga berhasil. Jika salah satu operasi dalam transactions gagal, seluruh transactions akan gagal.
- ❑ Sebuah transactions dimulai dengan pernyataan SQL executable. Sebuah transactions berakhir saat commit atau rolled back, baik secara eksplisit dengan pernyataan COMMIT atau ROLLBACK atau secara implisit ketika DDL (Data Definition Language) digunakan untuk mengelola struktur tabel dan indeks dengan pernyataan CREATE, ALTER, RENAME, DROP dan TRUNCATE

COMMIT and ROLLBACK

❑ COMMIT

to apply the transaction by saving the database changes.

❑ ROLLBACK

to undo all changes of a transaction.

❑ SAVEPOINT

to divide the transaction into smaller sections. It defines breakpoints for a transaction to allow partial rollbacks.

Tipe Transaction

Tipe	Deskripsi
DDL	Terdiri dari satu statement DDL
DML	Terdiri dari beberapa statement DML
DCL	Terdiri dari satu statement DCL

Properti Transaksi Basis Data

ACID

Atomicity

Transaksi dilakukan sekali dan sifatnya *atomic*, artinya merupakan satu kesatuan tunggal yang tidak dapat dipisah, baik itu pekerjaan yang dilaksanakan secara keseluruhan, ataupun tidak satupun.

Consistency

Jika basis data pada awalnya dalam **keadaan konsisten**, maka pelaksanaan transaksi dengan sendirinya juga harus meninggalkan basis data tetap dalam status konsisten.

Isolation

Isolasi memastikan bahwa secara bersamaan (konkuren) eksekusi transaksi terisolasi dari yang lain.

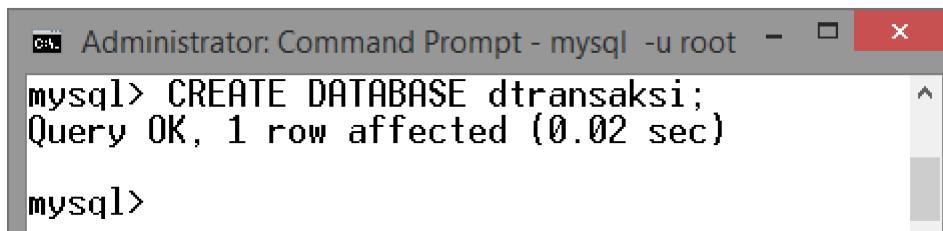
Durability

Begitu transaksi telah dilaksanakan (*di-commit*), maka perubahan yang diakibatkan tidak akan hilang atau tahan lama (*durable*), sekalipun terdapat kegagalan sistem.

Latihan

- Create basis data

CREATE DATABASE dtransaksi;

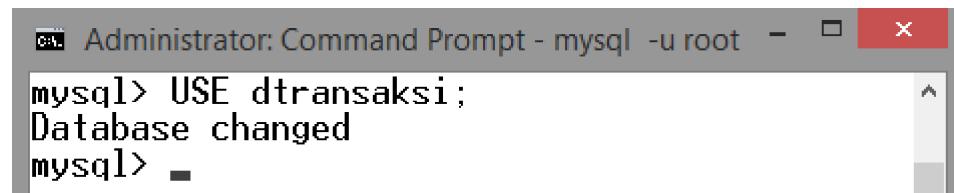


```
Administrator: Command Prompt - mysql -u root
mysql> CREATE DATABASE dtransaksi;
Query OK, 1 row affected (0.02 sec)

mysql>
```

- Gunakan database dtransaksi

USE dtransaksi;



```
Administrator: Command Prompt - mysql -u root
mysql> USE dtransaksi;
Database changed
mysql>
```

Latihan cont..

□ Create tabel

```
CREATE TABLE trans_demo(  
    nama VARCHAR(10) NOT NULL,  
    PRIMARY KEY(nama)  
)ENGINE = InnoDB;
```

```
MariaDB [dtransaksi]> create table trans_demo(  
    -> nama varchar(10) not null,  
    -> primary key(nama)  
    -> )engine = InnoDB;  
Query OK, 0 rows affected (0.31 sec)
```

Latihan Cont ... Implementasi Transaksi

1. Aktifkan transaksi basis data

```
START TRANSACTION;
```

```
MariaDB [dtransaksi]> start transaction;  
Query OK, 0 rows affected (0.00 sec)
```

2. Tambahkan dua baris data ke tabel trans_demo

```
INSERT INTO trans_demo
```

```
VALUES ("mysql"), ("oracle");
```

```
MariaDB [dtransaksi]> insert into trans_demo values ("mysql"),("oracle");  
Query OK, 2 rows affected (0.01 sec)  
Records: 2  Duplicates: 0  Warnings: 0
```

Latihan Cont ... Implementasi Transaksi

3. Periksa hasil penambahan data,

```
SELECT * FROM trans_demo;
```

```
MariaDB [dtransaksi]> select * from trans_demo;
+-----+
| nama |
+-----+
| mysql |
| oracle |
+-----+
2 rows in set (0.00 sec)
```

4. Keluar dari terminal

```
EXIT;
```

```
MariaDB [dtransaksi]> exit;
Bye

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

Latihan Cont ... Implementasi Transaksi

5. Login kembali ke basis data yang sama, kemudian periksa isi tabel “trans_demo”.

```
c:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 83
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use dtransaksi;
Database changed
MariaDB [dtransaksi]> select * from trans_demo;
Empty set (0.00 sec)
```

dengan memanggil **COMMIT**. Adapun penutupan *prompt* mysql mengakibatkan transaksi di-**rollback** secara implisit.

Latihan Cont ... Implementasi Transaksi

6. Ulangi langkah 2 (Tambahkan dua baris data ke tabel trans_demo)

`INSERT INTO trans_demo`

`VALUES ("mysql"), ("oracle");`

```
MariaDB [dtransaksi]> insert into trans_demo values ("mysql"),("oracle");
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

7. Ulangi Langkah 3 (Periksa hasil penambahan data)

`SELECT * FROM trans_demo;`

Latihan Cont ... Implementasi Transaksi

8. ketikkan pernyataan COMMIT.

```
MariaDB [dtransaksi]> select * from trans_demo;
+-----+
| nama   |
+-----+
| mysql  |
| oracle |
+-----+
2 rows in set (0.00 sec)

MariaDB [dtransaksi]> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Latihan Cont ... Implementasi Transaksi

9. Keluar dari terminal

```
MariaDB [dtransaksi]> exit;
Bye

c:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 86
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use dtransaksi;
Database changed

MariaDB [dtransaksi]> select * from trans_demo;
+-----+
| nama |
+-----+
| mysql |
| oracle |
+-----+
2 rows in set (0.00 sec)
```

Latihan 2... ROLLBACK

Pernyataan ROLLBACK digunakan untuk menggugurkan rangkaian perintah. ROLLBACK akan dilakukan manakala ada satu atau lebih perintah yang gagal dilaksanakan.

1. Aktifkan transaksi basis data

START TRANSACTION;

```
MariaDB [dtransaksi]> start transaction;  
Query OK, 0 rows affected (0.00 sec)
```

Latihan Cont ... ROLLBACK

2. Periksa hasil penambahan data,

```
SELECT * FROM trans_demo;
```

3. Tambahkan baris data

```
INSERT INTO trans_demo  
VALUES ("sybase");
```

```
MariaDB [dtransaksi]> insert into trans_demo values ("sybase");  
Query OK, 1 row affected (0.09 sec)
```

```
MariaDB [dtransaksi]>
```

Latihan Cont ... ROLLBACK

4. Tambahkan lagi baris data, namun dengan nilai yang sama

```
INSERT INTO trans_demo  
VALUES ("sybase");
```

```
MariaDB [dtransaksi]> insert into trans_demo values ("sybase");  
ERROR 1062 (23000): Duplicate entry 'sybase' for key 'PRIMARY'  
MariaDB [dtransaksi]> select *from trans_demo;  
+-----+  
| nama |  
+-----+  
| mysql |  
| oracle |  
| sybase |  
+-----+  
3 rows in set (0.00 sec)
```

Latihan Cont ... ROLLBACK

5. Berikan pernyataan untuk membatalkan rangkaian perintah dalam satu transaksi,

ROLLBACK;

SELECT * FROM trans_demo;

```
MariaDB [dtransaksi]> select * from trans_demo;
+-----+
| nama |
+-----+
| mysql |
| oracle |
+-----+
2 rows in set (0.00 sec)
```

Basis Data NoSQL

Pendahuluan Basis Data NoSQL

- Ada beberapa kekurangan dalam database SQL yang pernah ditemukan. Seperti skema database yang kaku (*fixed*), susah membuat query untuk tabel dengan relasi yang kompleks, susah diperbesar sekalanya, dsb.
- Database NoSQL (*Not Only SQL*) hadir untuk menutupi kekurangan-kekurangan tersebut. Selain itu, NoSQL sudah menjadi tuntutan teknologi yang harus dipelajari dalam pengembangan software modern masa kini.

Apa itu database NoSQL?

- Database NoSQL dibuat dengan tujuan khusus untuk model data spesifik dan memiliki skema fleksibel untuk membuat aplikasi modern.
- Database NoSQL dikenal secara luas karena kemudahan pengembangan, fungsionalitas, dan kinerja dalam berbagai skala.
- Database NoSQL menggunakan berbagai model data, termasuk dokumen, grafik, nilai kunci, dalam memori, dan pencarian. Halaman ini termasuk sumber daya untuk membantu Anda memahami lebih baik database NoSQL dan mulai menggunakannya.

Basisdata NoSQL

- ❑ NoSQL merupakan database yang tidak menggunakan realasi antar tabel dan tidak menyimpan data dalam format tabel kaku (kolom yang fix) seperti layaknya Relasional Database.

NoSQL menganut prinsip BASE, Eric Brewer,

- ❑ **Basic Availability**, Setiap request yang diterima mendapat response sukses atau gagal.
- ❑ **Soft State**, State NoSQL dapat berubah secara dinamis tanpa input manual untuk meyakinkan
- ❑ **Eventual Consistency**

❖ Karakteristik NoSQL

- Tidak berdasarkan model relasional.
- Mendukung database relasional terdistribusi.
- Menyediakan skalabilitas tinggi, ketersediaan tinggi dan toleransi kesalahan.
- Mendukung data tersebar yang sangat besar.
- Fokus pada kinerja dibandingkan dengan konsistensi transaksi.

Bagaimana Cara kerja Database (nonrelasional) NoSQL?

- Database NoSQL menggunakan berbagai model data untuk mengakses dan mengelola data, seperti dokumen, grafik, nilai kunci, dalam memori, dan pencarian.
- Jenis database ini dioptimalkan secara khusus untuk aplikasi yang memerlukan volume data besar, latensi rendah, dan model data fleksibel, yang dicapai dengan mengurangi pembatasan konsistensi data dari database lainnya.

Kenapa Anda harus menggunakan database NoSQL?

- Database NoSQL sangat cocok untuk digunakan dengan berbagai aplikasi modern seperti aplikasi seluler, web, dan gaming yang memerlukan database yang fleksibel, dapat diskalakan, berkinerja tinggi, dan memiliki fungsionalitas tinggi untuk memberikan pengalaman pengguna yang baik.
- Fleksibilitas**
- Skalabilitas**
- Kinerja tinggi**
- Fungsionalitas tinggi**

Jenis Database NoSQL

- Nilai-kunci
- Dokumen
- Grafik
- Dalam memori
- Pencarian

Databese SQL (relasional) vs. NoSQL (nonrelasional)

	Database Relasional	Database NoSQL
Beban kerja yang optimal	Database relasional didesain untuk aplikasi transaksional dan aplikasi pemrosesan transaksi online (online transaction processing, OLTP) yang sangat konsisten dan cocok digunakan untuk pemrosesan analisis online (online analytical processing, OLAP).	Database nilai-kunci, dokumen, grafik, dan dalam memori NoSQL didesain untuk OLTP untuk sejumlah pola akses data yang menyertakan aplikasi latensi rendah. Database pencarian NoSQL didesain untuk analisis data yang semi terstruktur.
Model data	Model relasional menormalkan data menjadi tabel yang terdiri dari baris dan kolom. Skema secara ketat mendefinisikan tabel, baris, kolom, indeks, hubungan antara tabel, dan elemen database lain. Database menegakkan integritas referensial dalam hubungan antara tabel.	Database NoSQL menyediakan berbagai model data, antara lain dokumen, grafik, nilai kunci, dalam memori, dan pencarian.

Databese SQL (relasional) vs. NoSQL (nonrelasional)

	Database Relasional	Database NoSQL
Properti ACID	Database relasional menyediakan properti atomicity, consistency, isolation, and durability (ACID):	Database NoSQL sering kali melakukan pertukaran dengan mengurangi beberapa properti ACID database relasional untuk model data yang lebih fleksibel yang dapat dikembangkan secara horizontal.
Performa	Kinerja umumnya tergantung pada subsistem disk. Pengoptimalan kueri, indeks, dan struktur tabel sering kali diperlukan untuk mencapai kinerja puncak.	Kinerja umumnya merupakan fungsi dari ukuran klaster perangkat keras, latensi jaringan, dan aplikasi panggilan.

Databese SQL (relasional) vs. NoSQL (nonrelasional)

	Database Relasional	Database NoSQL
Skala	Database relasional umumnya dapat dikembangkan skalanya dengan meningkatkan kemampuan komputasi perangkat keras atau mengembangkan skala dengan menambahkan replika untuk beban kerja hanya-baca.	Database NoSQL umumnya dapat dipartisi karena pola akses nilai-kunci dapat dikembangkan skalanya dengan menggunakan arsitektur terdistribusi untuk meningkatkan throughput yang menyediakan kinerja yang konsisten pada skala yang tidak terbatas.
API	Permintaan untuk menyimpan dan mengambil data dikomunikasikan menggunakan kueri yang sesuai dengan bahasa kueri terstruktur (SQL). Kueri ini diuraikan dan dijalankan oleh database relasional.	API berbasis objek memungkinkan pengembang aplikasi menyimpan dan mengambil struktur data dalam memori dengan mudah. Kunci partisi mengizinkan aplikasi mencari pasangan nilai-kunci, set kolom, atau dokumen semi terstruktur yang berisi objek dan atribut aplikasi berseri.

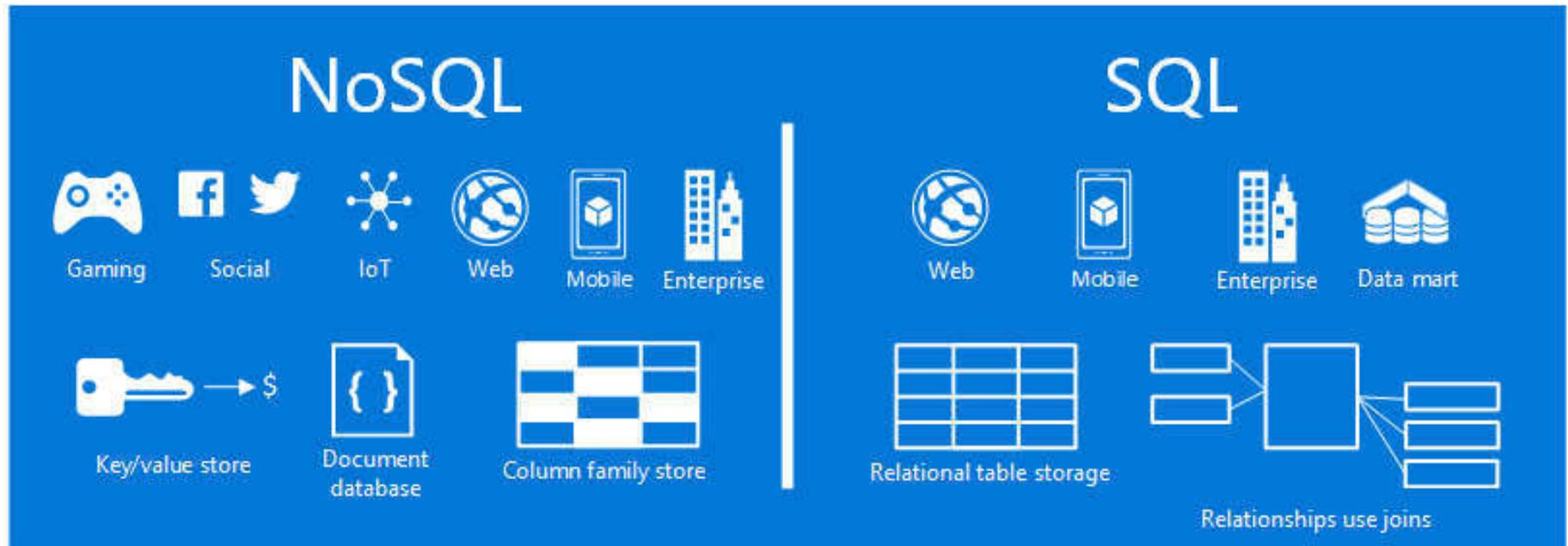
Kelebihan & Kekurangan NoSQL

- ❑ **NoSQL bisa menampung data yang terstruktur, semi terstruktur dan tidak terstruktur secara efisien dalam skala besar (big data/cloud).**
- ❑ **Menggunakan OOP dalam pengaksesan atau manipulasi datanya.**
- ❑ **NoSQL tidak mengenal schema tabel yang kaku dengan format data yang kaku.** NoSQL sangat cocok untuk data yang tidak terstruktur, istilah singkat untuk fitur ini adalah Dynamic Schema.
- ❑ **Autosharding,** istilah sederhananya, jika database noSQL di jalankan di cluster server (multiple server) maka data akan tersebar secara otomatis dan merata keseluruhan server.
- ❑ **Kekurangan NoSQL**
 - Hosting yang mahal
 - Memerlukan waktu untuk belajar format penyimpanan.

Cara penyimpanan basis data NoSQL tersebar dari

- Key-Value based* (disimpan dalam bentuk kunci-isi berpasangan)
- Document based* (disimpan dalam dokumen-dokumen)
- Column based* (disimpan dalam kolom-kolom)
- Graph based* (disimpan dalam bentuk grafik atau diagram)

Ruang lingkup NoSQL & SQL dalam database.



7 Basis Data NoSQL Populer

- ❑ MongoDB
- ❑ CouchDB
- ❑ Cassandra
- ❑ Redis
- ❑ Riak
- ❑ Neo4j
- ❑ OrientDB

I. MongoDB



- ❑ MongoDB merupakan basis data yang paling populer diantara basis data NoSQL lainnya. MongoDB juga merupakan salah satu basis data yang *open source*.
- ❑ MongoDB merupakan basis data NoSQL yang *document based*. Ia menyimpan data-datanya dalam suatu dokumen JSON yang disebut BSON (Binary JSON).
- ❑ Dikembangkan sejak tahun 2009, mongoDB sekarang telah mendukung hampir semua bahasa pemrograman untuk dapat berinteraksi dengan mongoDB.

2. CouchDB



- ❑ Apache CouchDB atau biasa disebut CouchDB saja, adalah basis data noSQL yang dikembangkan oleh Apache.
- ❑ Muncul pada tahun 2005, CouchDB tidak menyimpan datanya dalam tabel melainkan dalam dokumen seperti halnya MongoDB.
- ❑ Basis data ini juga proyek open source, serta dikembangkan dalam Bahasa pemrograman Erlang.

3. Cassandra

- Cassandra merupakan sebuah sistem penyimpanan data terdistribusi untuk menangani jumlah data yang sangat besar dan terstruktur.
 - Cassandra juga merupakan aplikasi open source yang ditulis dalam bahasa Java dengan lisensi Apache License 2.0.
 - Untuk memproses datanya, Cassandra menggunakan bahasa sendiri yang mirip dengan SQL yaitu Cassandra Query Language (CQL).
- Perusahaan-perusahaan besar telah mempercayakan Cassandra :
 - [Facebook](#)
 - [Digg](#)
 - [IBM](#)
 - [Reddit](#)
 - [Rackspace](#)
 - [CERN](#)
 - [Apple](#)
 - [Twitter](#)



4. Redis



- ❑ Redis juga merupakan basis data open source, merupakan basis data yang berbasis *key-value*.
- ❑ Redis merupakan singkatan dari REmot DIrectory Server.
- ❑ Basis data ini dikembangkan oleh Salvatore Sanfilippo, pada tahun 2009 dan ditulis dalam Bahasa C.
- ❑ Redis mendukung banyak bahasa pemrograman seperti ActionScript, C/C++, C#, Clojure, Common Lisp, Dart, Erlang, Go, Haskell, Haxe, Io, Java, JavaScript (Node.js), Lua, Objective-C, Perl, PHP, Pure Data, Python, R, Ruby, Scala, Smalltalk, dan Tcl.

5. Riak



- Riak merupakan basis data NoSQL terdistribusi yang menyimpan datanya dalam bentu key-value. Riak menawarkan fitur high availability, fault tolerance, operational simplicity, dan scalability.
- Riak memiliki dua versi yakni Open source edition dan Enterprise edition.
- Rilis pertama Riak muncul pada tanggal 17 Agustus 2009 (bertepatan dengan 64 tahun Indonesia merdeka).
- [Basho Technologies](#) merupakan perusahaan yang mengembangkan Riak.

6. Neo4j



- Neo4j merupakan basis data NoSQL dengan sistem graf. Neo4j menyimpan relasi antar objek dalam struktur seperti graf, dimana setiap objek merujuk ke objek lainnya secara langsung.
- Dalam menambahkan maupun mengambil data, Neo4j memiliki bahasa sendiri yang disebut Cypher.
- Neo4j ditulis dalam bahasa Java dengan lisensi GPL V3 oleh [Neo Technologies](#) artinya Neo4j termasuk basis data yang open source.

7. OrientDB



- ❑ OrientDB merupakan basis data graf terdistribusi generasi kedua. Basis data ini dibuat dalam bahasa Java oleh [**Orient Technologies LTD**](#) dan dirilis pertama kali tahun 2010. OrientDB diklaim sangat cepat dan mampu menyimpan 220.000 record per detik diperangkat standar.
- ❑ OrientDB menawarkan dua edisi yakni Community Edition yang tersedia secara gratis dan Enterprise Edition yang merupakan aplikasi komersial dikembangkan oleh tim yang sama yang mengembangkan engine OrientDB.
- ❑ OrientDB dapat digunakan bersama JavaScript, .NET, node.js, php, Scala, Ruby, Python, C, Clojure, Java, dan Perl.

Perintah dalam mongoDB

Perintah	Keterangan
Use <nama_db>	Untuk CREATE DATABASE baru atau memilih database yang sudah ada untuk digunakan
Db.mycoll.save(object)	Untuk INSERT data baru kedalam collection, jika sebelumnya namacollection belum ada maka akan dibuatkan secara otomatis
Db.mycoll.update(kondisi)	Untuk UPDATE data yang ada dalam collection
Db.mycoll.remove(kondisi)	Untuk DELETE data yang ada dalam collection
Db.mycoll.find(kondisi)	Untuk SELECT data yang ada dalam collection, perintah ini juga digunakan untuk melakukan pencarian data dalam collection

Perintah Dasar mongoDB

- **use <nama_database>** //CREATE DATABASE baru atau masuk ke dalam database yang sudah ada
- **show dbs** //menampilkan daftar database yang ada pada server MongoDB
- **show collections** //menampilkan daftar collection yang ada pada database
- **show users** //menampilkan nama user pada database
- **db.createCollection()** //membuat sebuah collection baru pada database
- **db.getCollectionNames()** //mengambil seluruh nama collection pada database yang sedang aktif
- **db.getName()** //menampilkan nama database yang sedang aktif
- **db.printCollectionStats()** //menampilkan seluruh collection beserta atributnya
- **db.removeUser(username)**//menghapus user tertentu pada database
- **db.help()** //menampilkan manual untuk manipulasi database
- **db.nama_collection.help()** //menampilkan manual untuk manipulasi collection
- **exit** //keluar dari Mongo Shell
- **db.repairDatabase()**//elakukan repair pada database yang sedang aktif
- **db.addUser(nama_user,password)** //menambah user baru pada database
- **db.auth** //memberikan hak akses pada user
- **db.copyDatabase(database_awal,database_target)** //menyalin isi database ke database lain
- **db.dropDatabase()** //menghapus database yang sedang digunakan sekarang
- **db.printReplicationInfo()** //menampilkan info dari replikasi database

mongoDB dan SQL

SQL	MongoDB
CREATE DATABASE akademik	use rumahsakit;
INSERT INTO mahasiswa(nrp,nama,jurusan) VALUES('09126','DANNIS','Informatika')	db.mahasiswa.save({nrp : "09126", nama : "DANNIS", jurusan : "Informatika"})
SELECT * FROM mahasiswa	db.mahasiswa.find();
UPDATE mahasiswa SET nama = "Budi" WHERE nrp = "090411100029"	db.mahasiswa.update({nrp:"090411100029"}, {\$set:{nilai:"Budi"}}, false,true)
DELETE mahasiswa WHERE Nrp = "090411100029"	db.mahasiswa.remove({nrp:"090411100029"})
SELECT * FROM users WHERE age=33	db.users.find({age:33});
SELECT * FROM users WHERE age>33	db.users.find({age:{\$gt:33}});
SELECT * FROM users WHERE age<33	db.users.find({age:{\$lt:33}});

mongoDB dan SQL

SELECT * FROM users WHERE a=1 OR b=2	db.users.find({\$or:[{a:1},{b:2}]}) ;
SELECT name,age FROM users WHERE age=33	db.users.find({age:33},{name:1,age:1});
SELECT * FROM users WHERE age=33 ORDER BY name	db.users.find({age:33}).sort({name:1});
SELECT * FROM users WHERE age=33 ORDER BY name DESC	db.users.find({age:33}).sort({name:-1});
SELECT * FROM users WHERE name LIKE "%Basith%"	db.users.find({name:/Basith/});
SELECT * FROM users WHERE name LIKE "Kubat%"	db.users.find({name:/^Kubat/});
SELECT DISTINCT last_name FROM users	db.users.distinct('last_name');
SELECT COUNT(*) FROM users	db.users.count();

EMBEDDED SQL

Embedded SQL

- SQL dapat disisipkan dalam bahasa pemrograman procedural
- Bahasa ini meliputi C/C++, Java, Perl, Python, dan PHP.
- Embedded SQL mendukung:
 - Aplikasi yang di custom
 - Background applications yang berjalan tanpa intervensi user.
 - Kombinasi database tools dengan programming tools.
 - Databases pada WWW.

Dua Tipe embedding

Low-level embedding (Contoh C/C++):

- ❑ SQL dan program dicompile ke dalam sebuah executable tunggal
- ❑ Link yang sangat efisien.

ODBC - Open Database Connectivity (Contoh PHP/Java):

- ❑ SQL query dikirim dari program ke database sebagai sebuah string.
- ❑ Hasil dikembalikan sebagai sebuah array atau list.
- ❑ Kebebasan program dan database:
 - Masing-masing bahasa memiliki satu DBI (database interface) untuk semua tipe DBMS. (Cotoh, JDBC untuk Java.)
 - database driver (DBD) yang terpisah untuk masing-masing tipe DBMS .

Low-level embedding (eg. C/C++)

- Query terdiri atas gabungan SQL dan perintah khusus.
- Sebuah cursor melangkah melalui hasil row satu pada satu waktu.
- Sebagai contoh:
 - EXEC SQL SELECT empname INTO :ename
 - FROM employee WHERE eno = :eno;

ODBC database connections

- ❑ Connect to the database.
- ❑ Prepare a query (as a string).
- ❑ Execute the query.
- ❑ Fetch the results (as an array of rows).
- ❑ Finish the query (so that DB can clean up its buffers).
- ❑ Disconnect from the database.

JDBC

- ❑ Java Database Connectivity (JDBC) adalah suatu library yang mirip dengan SQL/CLI, tetapi host language-nya adalah Java.
- ❑ Sebelum dapat mengakses database melalui java, harus install jdbc terlebih dahulu
- ❑ Pada pembahasan ini akan menggunakan database MySql

JDBC

Things to Do

- Import class JDBC
- Buat instance class untuk driver database
- Buat object Connection
- Untuk Query:
 - Buat object statement
 - Buat object ResultSet untuk menampung hasil executeQuery
 - Jalankan method executeQuery()
- Untuk Update & Insert:
 - Buat object statement
 - Jalankan method executeUpdate()

JDBC

Membuat Connection

```
import java.sql.*;           The JDBC classes
public class latJdbc_1 {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver") .newInstance();
            Connection koneksi= DriverManager.getConnection(".....");
        }
    }
}
```

The JDBC classes

The driver
for mySql;
others exist

Creating Instance.

Object Connection

DriverManager.getConnection(...);

URL of the database your
name, and password go here.

JDBC

Membuat Statement Query

```
....  
...  
Object Statement  
try {  
    Statement stmt = koneksi.createStatement();  
    String query = "SELECT * FROM beers;";  
    hasil = stmt.executeQuery(query);  
}  
....  
.....
```

Object ResultSet

Object Statement

Method executeQuery()

```
graph TD; OS[Object Statement] --> S[stmt]; OS --> R[hasil]; MEQ[Method executeQuery()] --> EQ[stmt.executeQuery(query)];
```

JDBC

Membuat Statement Update & Insert

```
.....  
...  
Statement stmtIns = koneksi.createStatement();  
  
try {  
  
    String query =  
  
        "INSERT INTO sells VALUES ('Joe's', 'Heineken', 300);";  
hasil = stmt.executeUpdate(query);  
}  
  
....  
.....
```

Object Statement

Method executeUpdate()

JDBC

Mengakses ResultSet (hasil Query)

- Suatu object type ResultSet mirip cursor.
- Method next() menjalankan “cursor” ke tuple berikutnya.
 - Perintah next() yang pertama akan mengambil tuple pertama.
 - Jika tidak ada tuple lagi, maka method next() akan mengembalikan nilai false.
 - Untuk mengambil isi object ResultSet, gunakan method **getX(i)**,
 - X disesuaikan dengan tipe attribute yang akan diambil.
 - i nama attribute yang akan diambil isinya.

Contoh: `getString('name')`

Artinya, mengambil isi attribute name dengan type String

JDBC

Menampilkan isi ResultSet (hasil Query)

```
String query = "SELECT * FROM beers;";
```

```
hasil = stmt.executeQuery(query);
```

....

.....

```
while( hasil.next() ) {
```

Method next() untuk
pindah ke tuple
berikutnya

```
    String nama = hasil.getString( "name" );
```

Method getString()
untuk mengambil isi
object ResultSet
dengan type String

```
    String pabrik = hasil.getString( "manf" );
```

```
    System.out.println( nama + " " + pabrik );
```

```
}
```

PHP & PEAR/DB

- **PEAR/DB** adalah suatu library yang mirip dengan SQL/CLI, tetapi host language-nya adalah PHP.
- Library ini biasanya sudah ada saat install PHP.
- Pada pembahasan ini akan menggunakan database MySql

PHP the Body

- ❑ Suatu bahasa pemrograman yang digunakan untuk beroperasi dalam HTML.
- ❑ Aturan penulisan, harus berada dalam tag spt berikut:

<?php

....

write your php code here

....

?>

- ❑ Variable dalam php diawali dengan simbol \$
- ❑ Variable tidak harus dideklarasikan *type*-nya

PHP & PEAR/DB

Things to Do

- ❑ Buat Connection

```
$koneksi = mysql_pconnect("localhost", "user", "password");
```

- ❑ Buka database

```
mysql_select_db("namaDB", $koneksi);
```

- ❑ Lakukan query (untuk query, update, insert, delete)

```
$query = "SELECT * FROM beers";  
$hasil = mysql_query($query, $koneksi);
```

- ❑ Mengakses hasil Query

- Menjalankan cursor ke tuple berikutnya
 \$data = mysql_fetch_array(\$hasil)

- Perintah `mysql_fetch_array` akan mengembalikan nilai `false` jika tidak ada tuple lagi.
 - Mengambil data isi attribute pada tuple aktif
 \$data['nama attribute']

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.

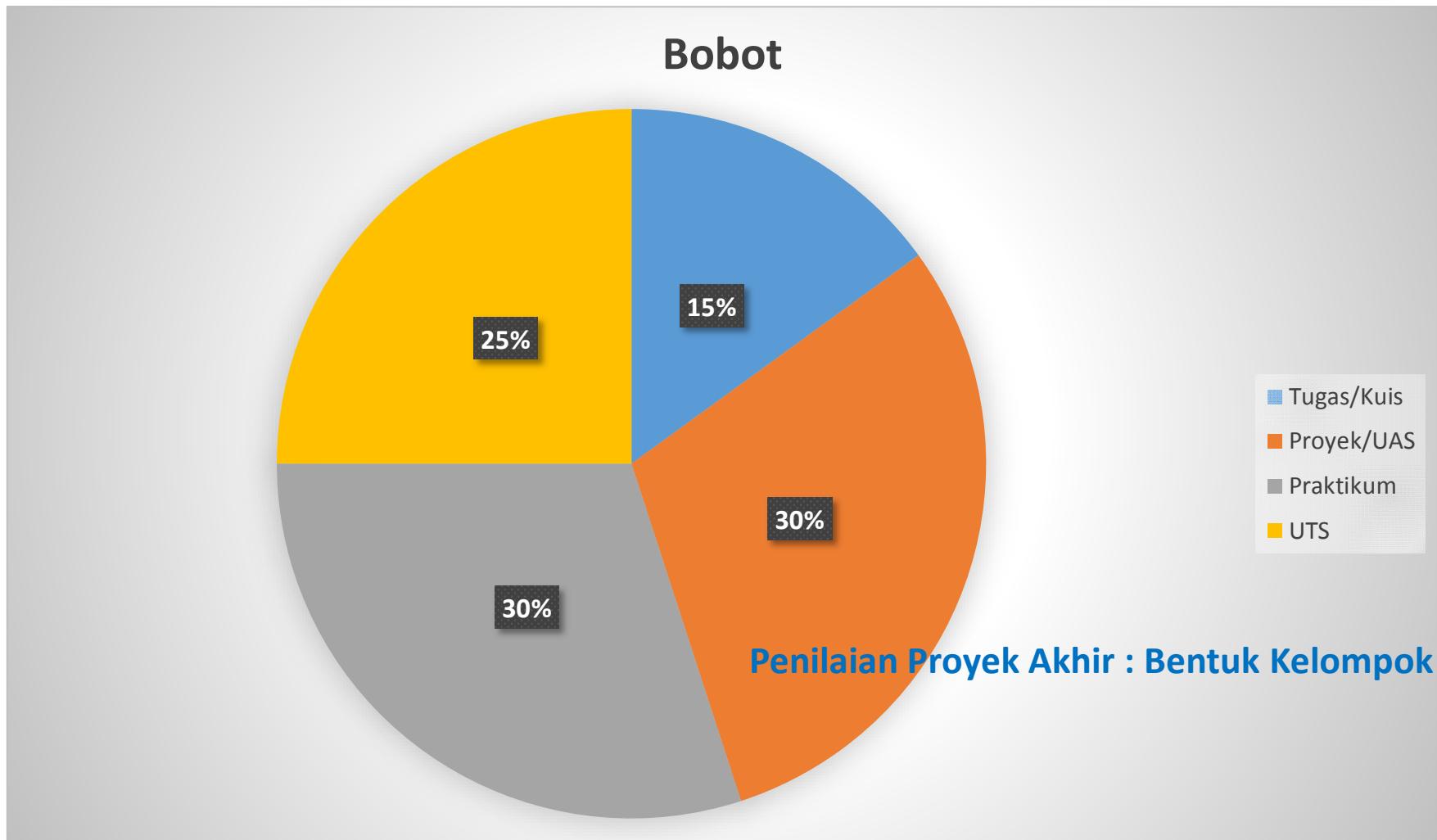
Bimbingan dan Arahan

- 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 penggerjaan tugas dan ujian, jika terjadi maka penggerjaan 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](tel:085706817980)

8) Referensi

Referensi (1)

- ❑ 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