PROGRAM STUDI
TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

MATA KULIAH
Sistem Basis Data

Kekangan Nilai & Query Antar Table

TEKNIK INFORMATIKA S1
UNIVERSITAS DIAN NUSWANTORO

Capaian Pembelajaran

Mahasiswa mampu:

- Menerapkan nilai yang tidak boleh kosong pada atribut suatu table
- Menerapkan nilai yang unik pada atribut suatu table
- Menerapkan nilai bawaan pada atribut suatu table
- Menerapkan auto increment pada atribut suatu table
- Menghubungkan dua table atau lebih menggunakan alias table dengan konsep inner join

Pokok Bahasan

- Membuat nilai tidak kosong (**NOT NULL**)
- Membuat nilai unik (**UNIQUE**)
- Membuat nilai bawaan(**DEFAULT**)
- Membuat kenaikan nilai secara otomatis(**AUTO INCREMENT**)
- Membuat kunci tamu (**FOREIGN KEY**)
- Menggunakan alias table dan query antar table (**INNER JOIN**)

Constraint atau Kekangan Nilai pada Data

- Constraint atau kekangan nilai digunakan untuk menentukan aturan yang mengizinkan atau membatasi nilai yang akan dimasukkan dalam tabel
- Kekangan nilai menyediakan metode yang sesuai untuk memastikan akurasi dan integritas data di dalam table.
  - Mengatur kolom agar terisi atau tidak memiliki nilai NULL
  - Menentukan kunci primer
  - Menentukan nilai yang unik
  - Mengatur nilai default (bawaan) pada suatu kolom
  - Melakukan kontrol nilai dalam kolom tertentu

Membuat Nilai tidak kosong [NOT NULL] (1)

- Constraint NOT NULL mengatur agar data tertentu harus terisi atau tidak kosong.
- Ada kolom tertentu dari suatu table harus terisi dengan nilai valid, sebagai contoh NIM mahasiswa tidak boleh kosong.
- Pengaturan NOT NULL dapat dilakukan pada pendefinisian field yang bersangkutan:

```
CREATE TABLE staf(
nip varchar(5),
nama varchar(50),
posisi varchar(30) not null,
tgl_masuk date,
gaji int,
primary key(nip)
);
```

Membuat Nilai tidak kosong [NOT NULL] (2)

- Ketika NOT NULL sudah diterapkan pada pendefisian field tertentu, maka system akan menolak penyisipan NULL pada kolom.
- Pada contoh berikut, field **posisi** telah diatur dengan NOT NULL
- Insert query memberikan pesan **error** bahwa kolom posisi tidak boleh NULL.

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nip       | varchar(5)  | NO   | PRI | NULL    |       |
| nama      | varchar(50) | YES  |     | NULL    |       |
| posisi    | varchar(30) | NO   |     | NULL    |       |
| tgl_masuk | date        | YES  |     | NULL    |       |
| gaji      | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

```
MariaDB [db_usaha]> insert into staf values
    -> ('a02','devi',null, '2000-05-01', 5000000);
ERROR 1048 (23000): Column 'posisi' cannot be null
```

Membuat Nilai Unik [UNIQUE] (1)

- Constraint ditujukan untuk memastikan bahwa nilai dalam kolom unik, artinya kolom tidak dapat menyimpan nilai duplikat
- MySQL memungkinkan untuk menggunakan lebih 1(satu) kolom dengan constraint unik.
- Penggunaan constraint unik dapat dilakukan dengan pendefinisian field yang bersangkutan

```
CREATE TABLE branch(
id_cabang varchar(5),
alamat varchar(100),
kota varchar(50) not null,
kodepos varchar(10) not null unique);
```

Membuat Nilai Unik [UNIQUE] (2)

- Ketika constraint UNIQUE sudah diterapkan, maka SQL akan menolak penyisipan data yang duplikat (kembar)
- Pada contoh berikut, field kodepos telah diatur UNIQUE
- Insert query memberikan pesan error bahwa kolom **kodepos** memberikan pesan bahwa terdapat *duplicate entry* pada kolom kodepos

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id_cabang | varchar(5)   | NO   | PRI | NULL    |       |
| alamat    | varchar(100) | YES  |     | NULL    |       |
| kota      | varchar(50)  | YES  |     | NULL    |       |
| kodepos   | varchar(10)  | NO   | UNI | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

```
MariaDB [db_usaha]> insert into branch values
    -> ('B003','Solo Baru','Solo','57148');
ERROR 1062 (23000): Duplicate entry '57148' for key 'kodepos'
```

Membuat Nilai Bawaan [DEFAULT] (1)

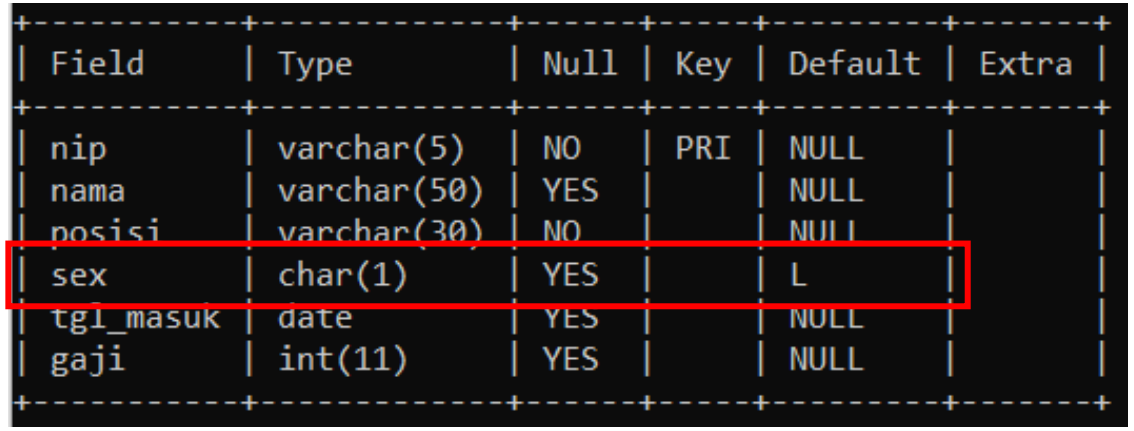
- Apabila field tidak diisi suatu nilai, maka field akan terisi dengan NULL.
- Constraint DEFAULT digunakan untuk memberikan nilai bawaan pada suatu field untuk menghindari field NULL.
- Constraint DEFAULT dapat mengatur jika suatu field tidak diberi nilai eksplisit, maka nilai default atau bawaan dapat diisikan ke field tersebut.
- Penggunaan nilai bawaan dapat dilakukan dengan menambahkan constraint DEFAULT pada pendefinisian field yang bersangkutan

Membuat Nilai Bawaan [DEFAULT] (2)

- Misal pada **table: staf**, ditambahkan kolom sex untuk menampilkan jenis kelamin yaitu "**L**" untuk laki-laki, "**P**" untuk perempuan.

```
ALTER TABLE staf add sex char(1) DEFAULT 'L'
```

- Query diatas mengatur nilai bawaan sex adalah 'L'. Struktur table staf sebagai berikut:



Field	Type	Null	Key	Default	Extra
nip	varchar(5)	NO	PRI	NULL	
nama	varchar(50)	YES		NULL	
posisi	varchar(30)	NO		NULL	
sex	char(1)	YES		L	
tgl_masuk	date	YES		NULL	
gaji	int(11)	YES		NULL	

Membuat Nilai Bawaan [DEFAULT] (3)

- Jika kolom sex tidak diisikan dengan suatu nilai, maka nilai bawaan 'L' yang akan disimpan

```
MariaDB [db_usaha]> insert into staf (nip, nama, posisi, tgl_masuk, gaji) values
    -> ('a02','devi','asisten','2022-09-24',3500000);
Query OK, 1 row affected (0.003 sec)
```

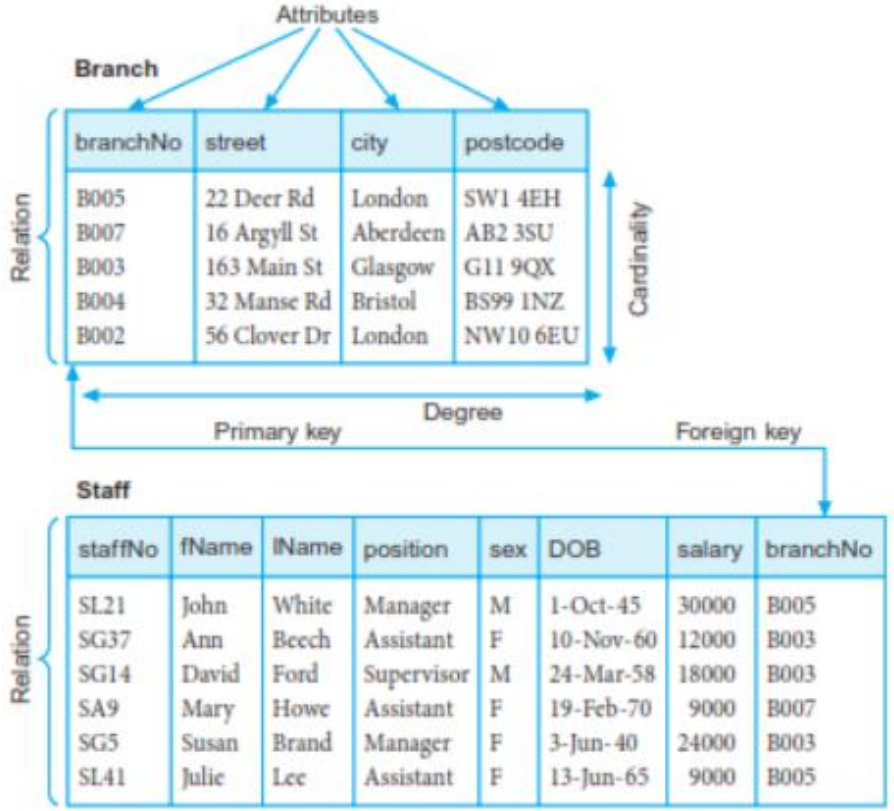
```
+-----+-------+------------+------+------------+----------+
| nip | nama  | posisi     | sex  | tgl_masuk  | gaji     |
+-----+-------+------------+------+------------+----------+
| a01 | wahyu | asisten    | L    | 2014-08-10 |  5000000 |
| a02 | devi  | asisten    | L    | 2022-09-24 |  3500000 |
| m01 | budi  | manajer    | L    | 1996-01-01 | 10000000 |
| s01 | sari  | supervisor | L    | 2000-02-05 |  7500000 |
+-----+-------+------------+------+------------+----------+
```

Membuat Kenaikan Nilai Otomatis [AUTO INCREMENT]

- Auto increment digunakan untuk menaikkan nilai secara otomatis pada field numerik.
- Ada beberapa hal yang harus diperhatikan dalam menggunakan auto increment:
  - Hanya dapat digunakan pada field numerik
  - Field harus bersifat primary key atau unik
  - Field tidak boleh bersifat null
  - Dalam satu table hanya ada satu field yang menggunakan auto increment

Membuat Kunci Tamu [Foreign Key] (1)

- Foreign key digunakan untuk menghubungkan dua table, sering disebut reference key
- Kolom foreign key cocok dengan field primary key pada table lain
- Hal ini menunjukkan bahwa foreign key mengacu pada field primary key table lain
- Sebagai contoh, **`branchNo`** pada table staf merupakan foreign key dari table **Branch**



Membuat Kunci Tamu [Foreign Key] (2)

- Pembuatan foreign key adalah sebagai berikut:

```
CREATE TABLE nama_table(

nama_kolom1 tipe data

nama_kolom2 tipe data

PRIMARY KEY(nama_kolom),

FOREIGN KEY(listForeignKeyKolom)
REFERENCES
table_parent(ListKunciKandidatKolom)

ON UPDATE AksiReferensi

ON DELETE AksiReferensi
```

Contoh:

```
CREATE TABLE staf(
nip varchar(5),
nama varchar(50),
posisi varchar(30) not null,
tgl_masuk date,
gaji int,
id_branch varchar(5),
primary key(nip),
foreign key(id_branch) references
branch(id_branch)
);
```

Membuat Kunci Tamu [Foreign Key] (3)

- Jika table telah dibuat, foreign key dapat ditambahkan dengan ALTER TABLE:

```
ALTER TABLE name_table

ADD FOREIGN KEY(nama_kolom)

REFERENCES
nama_table_parent(kolom_table_parent)
```

- **Note**: sebelum menambahkan foreign key pada **id_cabang**, terlebih dahulu menambahkan kolom id cabang pada table staf:

```
ALTER TABLE staf ADD id_cabang varchar(5);
```

```
MariaDB [db_usaha]> ALTER TABLE staf
    -> ADD FOREIGN KEY(id_cabang)
    -> REFERENCES branch(id_cabang);
Query OK, 4 rows affected (0.026 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [db_usaha]> desc staf;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nip       | varchar(5)  | NO   | PRI | NULL    |       |
| nama      | varchar(50) | YES  |     | NULL    |       |
| posisi    | varchar(30) | NO   |     | NULL    |       |
| sex       | char(1)     | YES  |     | L       |       |
| tgl_masuk | date        | YES  |     | NULL    |       |
| gaji      | int(11)     | YES  |     | NULL    |       |
| id_cabang | varchar(5)  | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.006 sec)
```

Query Antar Table [INNER JOIN]

- Query yang telah dibahas sebelumnya memiliki Batasan, yaitu informasi yang diberikan masih berasal dari satu table.
- Ada kemungkinan bahwa informasi yang dibutuhkan melibatkan tabel lain.
- SQL mempunyai kemampuan untuk menggabungkan dua atau lebih tabel untuk membentuk suatu informasi.
- Umumnya dalam menggabungkan tabel berdasarkan field yang bersesuaian (primary key pada table-1 dengan foreign key pada table-2)
- Hal ini disebut dengan **JOIN**.

Syntax Umum Penulisan SQL Join

- **SELECT** *table1.column, table2.column*
  FROM *table1, table2*
  WHERE *table1.column1 = table2.column2*

```
MariaDB [db_usaha]> SELECT staf.nip, staf.nama, staf.posisi, branch.id_cabang, branch.kota
    -> FROM staf, branch
    -> WHERE staf.id_cabang = branch.id_cabang;
+-----+-------+------------+-----------+----------+
| nip | nama  | posisi     | id_cabang | kota     |
+-----+-------+------------+-----------+----------+
| a01 | wahyu | asisten    | B001      | Semarang |
| a02 | devi  | asisten    | B002      | Solo     |
| m01 | budi  | manajer    | B001      | Semarang |
| s01 | sari  | supervisor | B001      | Semarang |
+-----+-------+------------+-----------+----------+
```

Syntax Umum Penulisan SQL Join dengan ALIAS

- **SELECT** *alias_name_table1.column*, *alias_name_table2.column*
  FROM *table1 alias_name_table1*, *table2 alias_name_table2*
  WHERE *alias_name_table1.column1 = alias_name_table2.column2*

```
MariaDB [db_usaha]> SELECT s.nip, s.nama, s.posisi, b.id_cabang, b.kota
    -> FROM staf s, branch b
    -> WHERE s.id_cabang = b.id_cabang;
+-----+-------+------------+-----------+----------+
| nip | nama  | posisi     | id_cabang | kota     |
+-----+-------+------------+-----------+----------+
| a01 | wahyu | asisten    | B001      | Semarang |
| a02 | devi  | asisten    | B002      | Solo     |
| m01 | budi  | manajer    | B001      | Semarang |
| s01 | sari  | supervisor | B001      | Semarang |
+-----+-------+------------+-----------+----------+
```

pada contoh diatas:
s merupakan alias dari staf
b merupakan alias dari branch

Syntax Umum Penulisan SQL Join – Menggunakan Inner Join

- **SELECT** *table1.column, table2.column*
  FROM *table1*
  INNER JOIN *table2*
  ON *table1.column1 = table2.column2*

```
MariaDB [db_usaha]> SELECT staf.nip, staf.nama, staf.posisi, branch.id_cabang, branch.kota
    -> FROM staf
    -> INNER JOIN branch
    -> ON staf.id_cabang = branch.id_cabang;
+-----+-------+------------+-----------+----------+
| nip | nama  | posisi     | id_cabang | kota     |
+-----+-------+------------+-----------+----------+
| a01 | wahyu | asisten    | B001      | Semarang |
| a02 | devi  | asisten    | B002      | Solo     |
| m01 | budi  | manajer    | B001      | Semarang |
| s01 | sari  | supervisor | B001      | Semarang |
+-----+-------+------------+-----------+----------+
```

Referensi

UTAMA

1. Silberschatz, A., Korth, H. F. & Sudarshan, S., 2022. Database System Concepts. 7th ed. New York: McGraw-Hill Education
2. Connolly, T. & Begg, C., 2015. Database Systems Apracticial Approach to Design, Implementation, and Management. Sixth Edition ed. s.l.:Pearson.
3. Elmasri, R. & Navathe, S. B., 2016. Fundamentalsmof Database Systems. 7th ed. s.l.:Pearson

PENDUKUNG

Aripin., 2005. *Praktikum Basis Data Dengan Database Server MySQL.* Semarang: Fakultas Ilmu Komputer

TERIMA KASIH

ANY QUESTIONS?