

Name: Andi Dwi Saputro

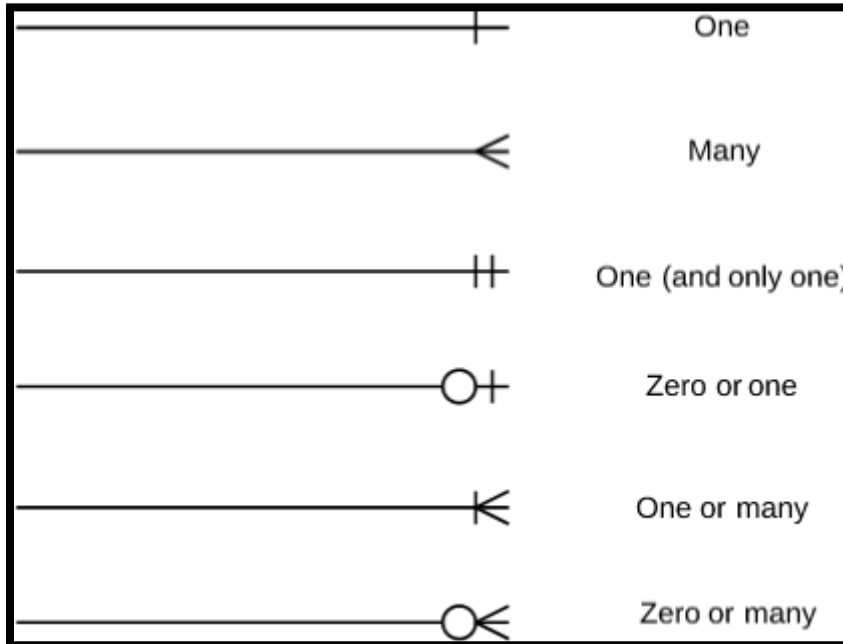
Date: April 17th, 2020

1. Apa yang disebut ordinality dan cardinality di relationship antar table ?

Cardinality: merupakan berapa banyak jumlah maksimum instance dari suatu entity yg saling berhubungan/berelasi dengan instance2 yang ada pada entity yang lain.

Ordinality: merupakan jumlah minimum suatu instance pada suatu entity dapat berasosiasi dengan instance lainnya pada entitas yang saling berelasi.

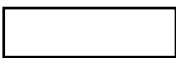
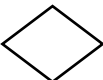
Simbol notasi kardinalitas pada ERD:



2. Dalam ERD yang dirancang dalam tugas library n banking, tuliskan apa saja jenis relationship yang ada antar entitas (table) ?

- Nasabah <-> Admin :: Many to One
- Nasabah <-> Rekening :: Many to Many
- Nasabah <-> Transaksi_bank :: One to Many
- Nasabah <-> Transaksi_PPOB :: One to Many
- Rekening <-> Transaksi_bank :: One to Many
- Rekening <-> Transaksi_PPOB :: One to Many

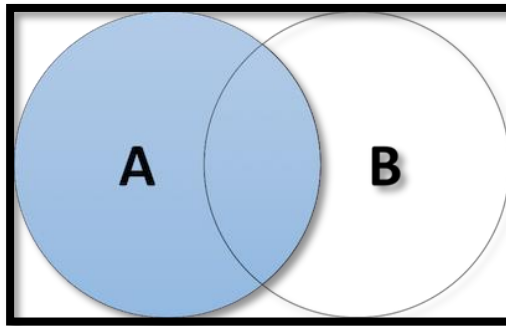
3. Apa bedanya Entitas dan Relationship ?

	Entity	Relationship
Definisi	Menunjukkan objek-objek dasar yang terkait didalam system. E.g: nasabah, rekening, admin,dsb.	Merupakan kejadian atau transaksi yang terjadi antara dua entitas yang keterangannya perlu disimpan dalam database.
Symbol	Persegi Panjang: 	Diamond/Belah Ketupat: 

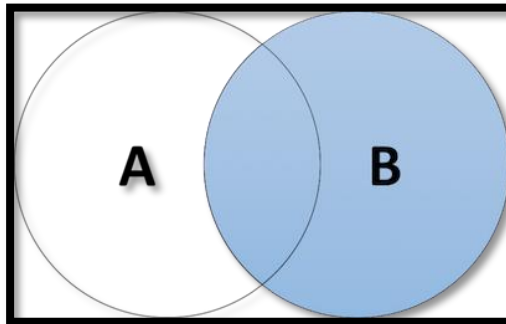
Rules	<ul style="list-style-type: none"> Nama entity berupa kata benda tunggal Nama entity sebisa mungkin menggunakan nama yang mudah dipahami dan menyatakan (deskriptif) maknanya dengan jelas. 	<ul style="list-style-type: none"> Relasi menghubungkan dua entitas Nama relasi menggunakan kata kerja aktif (diawali prefix me) tunggal. Nama relasi sebisa mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.
-------	---	--

4. Apa definisi:

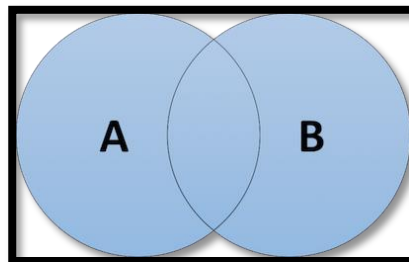
- a. **Left join:** merupakan cara menghubungkan table dan menampilkan seluruh rows data dari table disebelah kiri yang dikenai kondisi **ON** dan hanya baris dari table sebelah kanan yang memenuhi kondisi join.



- b. **Right join:** merupakan kebalikan left join, yakni menghubungkan table dan menampilkan semua rows dari table sebelah kanan yang dikenai kondisi **ON** dengan data dari table sebelah kiri yang memenuhi kondisi join.



- c. **Full join: atau full outer join** merupakan gabungan dari left dan right join, yakni akan menggabungkan dan menampilkan semua rows dari kedua table yang dikenai **ON** termasuk data-data yang bernilai NULL.



5. Contoh query untuk soal no.4 apa saja yang digunakan dalam tugasnya ?

Left Join:

SELECT * FROM rekening LEFT JOIN nasabah ON rekening.id_rekening = nasabah.id_rekening;

✓ Showing rows 0 - 17 (18 total, Query took 0.0193 seconds.)

SELECT * FROM rekening LEFT JOIN nasabah ON rekening.id_rekening = nasabah.id_rekening

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

id_rekening	no_rekening	saldo	jenis_rekening	no_pin	id_nasabah	nama	nik	tgl_lahir	alamat
9	259959170576	1500000	Reguler	32767	4	Ruslan	1231200976	1992-01-12	Jaksel
11	045359170576	5000000	Gold	32767	5	Andi	201543502	1991-01-01	Jakarta
12	050159170576	3347000	Reguler	32767	6	Asep	12345678	1992-01-27	Bandung
13	175359170576	49949000	Platinum	32767	7	Ujang	11223344	1990-01-01	Bogor

Right Join:

SELECT * FROM rekening RIGHT JOIN nasabah ON rekening.id_rekening = nasabah.id_rekening;

✓ Showing rows 0 - 13 (14 total, Query took 0.0309 seconds.)

SELECT * FROM rekening RIGHT JOIN nasabah ON rekening.id_rekening = nasabah.id_rekening

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

id_rekening	no_rekening	saldo	jenis_rekening	no_pin	id_nasabah	nama	nik	tgl_lahir	alamat
NULL	NULL	NULL	NULL	NULL	2	Dudung	12345678	1989-07-11	Jkt
9	259959170576	1500000	Reguler	32767	4	Ruslan	1231200976	1992-01-12	Jaksel
11	045359170576	5000000	Gold	32767	5	Andi	201543502	1991-01-01	Jakarta
12	050159170576	3347000	Reguler	32767	6	Asep	12345678	1992-01-27	Bandung
13	175359170576	49949000	Platinum	32767	7	Ujang	11223344	1990-01-01	Bogor

Full Join:

SELECT * FROM transaksi_bank LEFT JOIN rekening ON transaksi_bank.id_rekening = rekening.id_rekening

UNION ALL

SELECT * FROM transaksi_bank RIGHT JOIN rekening ON transaksi_bank.id_rekening = rekening.id_rekening

WHERE transaksi_bank.id_rekening IS NULL

Showing rows 0 - 17 (18 total, Query took 0.0449 seconds.)

```
SELECT * FROM transaksi_bank LEFT JOIN rekening ON transaksi_bank.id_rekening = rekening.id_rekening UNION ALL SELECT * FROM transaksi_bank RIGHT JOIN rekening ON transaksi_bank.id_rekening = rekening.id_rekening WHERE transaksi_bank.id_rekening IS NULL
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

id_transaksi	jenis_transaksi	nominal	tgl_transaksi	kode_bank	rek_tujuan	biaya_admin	ket_transaksi	id_rekening	id_rekening	no_rekening	saldo
1	Transfer	50000	2020-03-20 20:28:00	21	762512731	1000	Test transfer to my sister	12	12	050159170576	3347000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1	1234567890	100000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	4	058159171881	100000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	673159171881	1000000

Untuk full join pada mysql tidak support hanya dengan query FULL JOIN namun harus menggabungkan LEFT JOIN dan RIGHT JOIN query dengan query UNION:

6. Apakah itu stored procedure, extended procedure, views, materialized view?

Stored Procedure: merupakan sebuah konsep untuk melakukan satu atau lebih statement operasi DML pada database.

Extended Procedure: merupakan sekumpulan routines/statement DML yang terletak pada file DLL (Dynamic Link Library) yang berfungsi sama seperti stored procedure pada umumnya. Mereka dapat menerima parameter-parameter dan mereturn hasil/rows melalui SQL Server Open Data Services API.

Views: pada SQL view merupakan sebuah virtual table berdasarkan result-set dari suatu statement SQL. Views hanya memiliki baris2 dan atau kolom2 sama seperti table pada umumnya namun hanya berdasarkan statemen query yang dioperasikan.

Materialized Views: merupakan view yang “dimaterialisasikan” yakni sebuah object database yang mengandung result set dari sebuah statement query. Perbedaannya dari view biasa adalah, materialized view ditujukan benar2 khusus untuk optimasi performa serta dapat melakukan operasi selain select/join namun tidak mengganggu source table.

7. Apa itu DML dan DDL ?

DML: Data Manipulation Language, merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data dalam table. (SELECT, INSERT, UPDATE, DELETE)

DDL: Data Definition Language merupakan perintah SQL yang berhubungan dengan pendefinisian atau deklarasi suatu struktur database. (CREATE, RENAME, ALTER, DROP)

8. Apakah bedanya views dengan table biasa ?

Kenapa harus pakai views ? apa kelebihanannya ?

	Views	Table
The Differences	Result set of SELECT or JOIN from source table but can't do any update or delete operations, so it can't affect the source table since it's	Result set of any DML or DDL queries. Not a virtual table but the source itself.

	just a virtual table. Except you're using Materialized View instead, to do any query operations besides of SELECT or JOIN queries.	
Why using Views?	Tujuan utama menggunakan view pada SQL adalah untuk mengkombinasikan data dari berbagaimacam sources table tanpa harus membuat table baru lainnya untuk menstore data tersebut	
The Benefits	<ul style="list-style-type: none"> • Restrict akses data • Membuat rangkuman / rekap data dari berbagai macam source table yang biasa digunakan untuk reports 	

9. Apa bedanya views dengan stored procedure ? kapan dipakai ?

View	Stored Procedure
Sebuah alias / table virtual yang tercipta dari sebuah sql statement	Merupakan sekumpulan / set kode atau statement yang dapat melakukan banyak action

Kapan dipakai?

- View akan dipakai ketika kita membutuhkan suatu alias atau virtual table yang merepresentasikan statement query tertentu untuk menghindari kompleksitas dan menyederhanakan query sql.
- Stored procedure akan dipakai ketika membutuhkan suatu set programmatic actions didalam proses manipulasi maupun defining pada database kita.

10. Apa itu trigger ? kapan dipakai ?

Trigger merupakan sebuah SQL procedure yang menginit atau memulai suatu action. Trigger biasa digunakan untuk maintain integritas informasi yang ada pada database. Trigger juga dapat digunakan untuk melakukan logging historical data, sebagai contoh untuk melihat/mentrack log history gaji/salaries pada employees semuanya dilakukan otomatis.

11. Jika saya mau buat history user yang mengupdate transaksi perubahan misalnya di bank/perpustakaan untuk mencatat table apa yang berubah, waktu ubah, value yang diubah, bagaimana saya bisa melakukan sebaiknya ?

Dengan menggunakan trigger kita bisa melakukan maintain dan tracking terhadap log history user yang melakukan update transaksi perubahan misalnya pada table nasabah adalah sebagai berikut step-stepnya menggunakan phpmyadmin:

1. Membuat table log_history_nusabank

Berikut struktur tablenya:

#	Name	Type	Collation	Attributes	Null	Default
<input type="checkbox"/> 1	description	text	latin1_swedish_ci		No	None
<input type="checkbox"/> 2	log_time	timestamp			No	current_timestamp()
<input type="checkbox"/> 3	user	varchar(60)	latin1_swedish_ci		No	None
<input type="checkbox"/> 4	old_data	varchar(45)	latin1_swedish_ci		No	None
<input type="checkbox"/> 5	new_data	varchar(45)	latin1_swedish_ci		No	None

- Description: untuk menampung keterangan detail mengenai perubahan yang terjadi
- Log_time: untuk mencatat keterangan waktu kapan perubahan terjadi
- User: merupakan user database administrator yang melakukan proses data manipulation tersebut
- Old_data: merupakan value data yang sebelumnya
- New_data: merupakan value data terbaru yang diubah atau ditambah.

2. Membuat Trigger After Insert

<title>localhost / 127.0.0.1 / db_nusa | phpMyAdmin 5.0.1</title>

Details

Trigger name: TRIGGER_AFTER_INSERT

Table: nasabah

Time: AFTER

Event: INSERT

Definition:

```

1 INSERT INTO log_history_nusabank
2 (description, user, new_data)
3 VALUES (CONCAT('Insert data ke table nasabah, details:id_nasabah=',
4 NEW.id_nasabah),USER(),NEW.nama);

```

Definer: andi@localhost

Disini trigger yang masuk akan mentrack perubahan setelah operas insert dilakukan, detilnya akan ditambahkan ke table log history, untuk sementara tracking perubahannya hanya pada id dan nama nasabah.

3. Membuat Trigger After Update

<title>localhost / 127.0.0.1 / db_nusa | phpMyAdmin 5.0.1</title>

Details

Trigger name	TRIGGER_AFTER_UPDATE
Table	nasabah
Time	AFTER
Event	UPDATE

Definition

```
1 INSERT INTO log_history_nusabank(description, user,
new_data, old_data)
2 VALUES (concat('Update data ke table nasabah,
details::id_nasabah=',NEW.id_nasabah), USER(), NEW.nama,
OLD.nama);
```

Definer

andi@localhost

4. Membuat Trigger Before Delete

<title>localhost / 127.0.0.1 / db_nusa | phpMyAdmin 5.0.1</title>

Details

Trigger name	TRIGGER_BEFORE_DELETE
Table	nasabah
Time	BEFORE
Event	DELETE

Definition

```
1 INSERT INTO log_history_nusabank(description, USER,
old_data)
2 VALUES (concat('Delete data ke table nasabah,
details::id_nasabah=',OLD.id_nasabah), USER(), OLD.nama);
```

Definer

andi@localhost

5. Uji coba trigger

- a. Test insert query ke table nasabah

```
✓ 1 row inserted.  
Inserted row id: 26  
  
INSERT INTO `nasabah` (`id_nasabah`, `nama`, `nik`, `tgl_lahir`, `alamat`, `photo`, `jenis_kelamin`,  
`pekerjaan`, `alamat_kantor`, `pendapatan`, `email`, `no_hp`, `status`, `nama_ibu`, `username_nasabah`,  
`password_nasabah`, `tgl_pembuatan`, `id_rekening`, `id_admin`) VALUES (NULL, 'Vivaldi Winter',  
'199922233123', '1999-04-01', 'Jl. Tb. Simatupang', '/res/vivaldi_photos.jpg', 'Male', 'Violinist',  
'Studio Vivaldi, Jakarta Selatan', '5000000', 'vivaldi@mail.ru', '085988921232', 'Single', 'Nikky',  
'vivaldi99', 'vivaldi9914', current_timestamp(), '42', '1');
```

- b. Test update query di table nasabah

```
✓ 1 row affected.  
  
UPDATE `nasabah` SET `nama` = 'La Campanella', `username_nasabah` = 'lacampanella', `password_nasabah` =  
'lacamp929' WHERE `nasabah`.`id_nasabah` = 14;
```

- c. Delete duplikat nasabah Jon Bon Jovi
d. Result trigger pada table log_history_nusabank

description	log_time	user	old_data	new_data
Insert data ke table nasabah, details:id_nasabah=...	2020-04-17 17:42:30	root@localhost		Vivaldi Winter
Update data ke table nasabah, details:id_nasabah=...	2020-04-17 17:44:25	root@localhost	Jon Bon Jovi	La Campanella
Delete data ke table nasabah, details:id_nasabah=...	2020-04-17 17:45:27	root@localhost	Jon Bon Jovi	

12. Apa itu PK, FK, Unique, CK ?

Primary Key: merupakan sebuah aturan dimana fungsinya adalah untuk membedakan antara baris satu dengan lainnya yang ada pada table dan bersifat unik, dengan syarat data value pada field yang dijadikan PK tidak boleh terjadi duplikasi serta tidak boleh bernilai null.

Foreign Key: alias kunci asing adalah sebuah atribut atau gabungan atribut yang terdapat dalam suatu table yang digunakan untuk menciptakan relasi antara dua table yang mana field foreign tersebut akan merujuk pada suatu field primary yang ada pada table lain yang direlasikan.

Unique: sama seperti primary key yaitu untuk memastikan bahwa setiap baris data yang terdapt dalam suatu table bersifat unik. Perbedaannya, pada unique key kita diperbolehkan untuk memasukkan value atau nilai NULL.

Candidate Key: adalah suatu atribut atau satu set minimal atribut yang hanya mengidentifikasi secara unik untuk suatu kejadian spesifik dari suatu entitas. Dengan spesifikasi sbb: unique, non-redundan.

13. Mengapa saya harus pakai index ? apa saja yang diindex ?

Index dapat digunakan untuk mempercepat proses pencarian dalam database. Sebagaimana kita saat membaca buku dan ingin mencari pembahasan / topic tertentu pada buku yang kita baca otomatis untuk mencari tau hal tersebut kita bisa menggunakan bantuan index yang ada pada buku tersebut, misal apa yang kita cari ada di halaman sekian bab sekian.

Kriteria field yang harus di index diantaranya adalah sebagai berikut:

- Hanya melakukan index terhadap columns/fields tertentu saja yang bersifat required saat menggunakan perintah WHERE dan ORDER BY atau dalam kondisi join.
- Pastikan field tersebut bersifat unik dan tidak memiliki duplikasi seperti field id, NIK, dan lain sebagainya.
- Columns berisi nilai dengan jangkauan yang luas
- Columns berisi banyak nilai null
- Jangan melakukan indexing pada tables kecil.
- Jangan melakukan indexing pada columns/fields yang sering diubah-ubah/update.

14. Apa yang sebenarnya terjadi pada index ?

Index merupakan objek struktur data tersendiri yang tidak bergantung pada struktur table. Setiap index terdiri dari nilai kolom dan penunjuk/pointer (ROWID) ke baris yang berisi nilai tersebut. Pointer tersebut secara langsung menunjuk ke baris yang tepat pada table, sehingga menghindari terjadinya full table-scan. Akan tetapi lebih banyak index pada table tidak berarti akan mempercepat query. Semakin banyak index pada suatu table menyebabkan kelambatan pemrosesan perintah-perintah yang bersifat DML (Data Manipulation), karena setiap terjadi perubahan data maka index juga harus disesuaikan.