**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:**



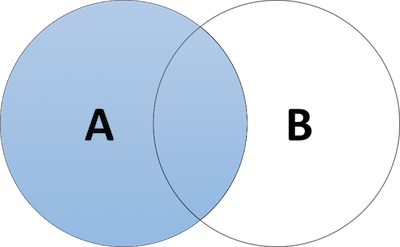
1. **Dalam ERD yang dirancang dalam tugas library n banking, tuliskan apa saja jenis relationship yang ada antar entitas (table) ?**

* 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

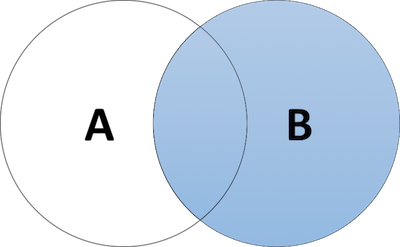
1. **Apa bedanya Entitas dan Relationship ?**

|  |  |  |
| --- | --- | --- |
|  | 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 | * Nama entity berupa kata benda tunggal * Nama entity sebisa mungkin menggunakan nama yang mudah dipahami dan menyatakan (deskriptif) maknanya dengan jelas. | * 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. |

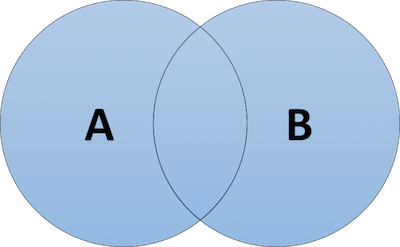
1. **Apa definisi:**
2. **Left join:** merupakan cara menghubungkan table dan menampilkan seluruh rows data dari table disebelah kiri yang dikenai kondisi **ON** dan hanya baris dari table sebelah kanan yang memenuhi kondisi join.



1. **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.



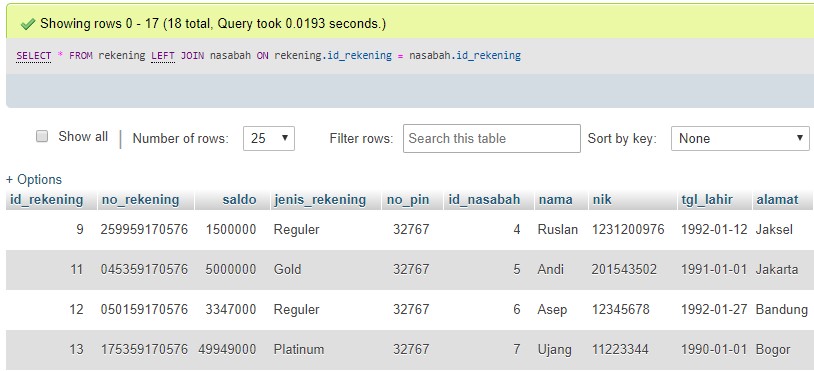
1. **Full join: atau full outer join** merupakan gabungan dari left dan right join, yakni akan menggabungkan dan menampikan semua rows dari kedua table yang dikenai **ON** termasuk data-data yang bernilai NULL.



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

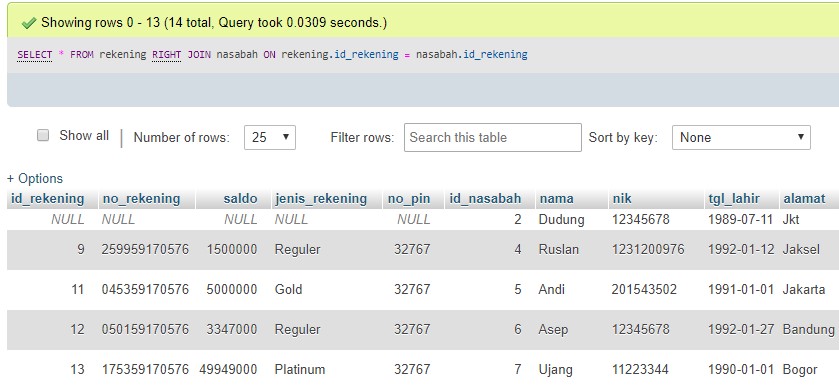
**Left Join:**

*SELECT \* FROM* ***rekening*** *LEFT JOIN* ***nasabah*** *ON rekening.id\_rekening = nasabah.id\_rekening;*



**Right Join:**

*SELECT \* FROM* ***rekening*** *RIGHT JOIN* ***nasabah*** *ON rekening.id\_rekening = nasabah.id\_rekening;*



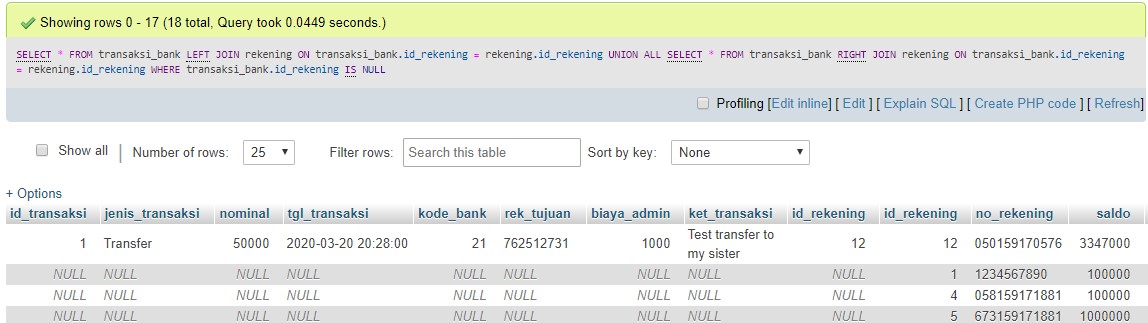
**Full Join***:*

*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*



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

1. **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-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.

1. **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)

1. **Apakah bedanya views dengan table biasa ?**

**Kenapa harus pakai views ? apa kelebihannya ?**

|  |  |  |
| --- | --- | --- |
|  | **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 just a virtual table. Except you’re using Materialized View instead, to do any query operations besides of SELECT or JOIN queries.** | **Result set of any DML or DDL queries.**  **Not a virtual table but the source itself.** |
| **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** | * **Restrict akses data** * **Membuat rangkuman / rekap data dari berbagai macam source table yang biasa digunakan untuk reports** | |

1. **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 manipluasi maupun defining pada database kita.

1. **Apa itu trigger ? kapan dipakai ?**

Trigger merupakan sebuah SQL procedure yang menginit atau memulai suatu action.

Trigger biasa digunakan untuk memaintain integritas informasi yang ada pada database.

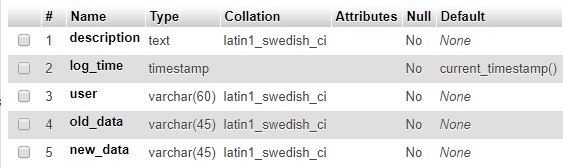
Trigger juga dapat digunakan untuk melakukan logging historical data, sebagai contoh untuk melihat/mentrack log history gaji/salaries pada employees semuanya dilakukan otomatis.

1. **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:

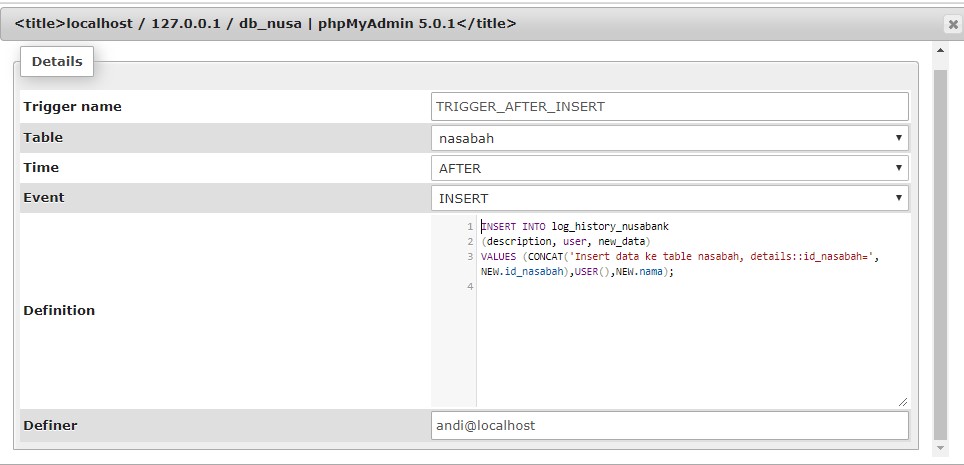
1. **Membuat table log\_history\_nusabank**

Berikut struktur tablenya:



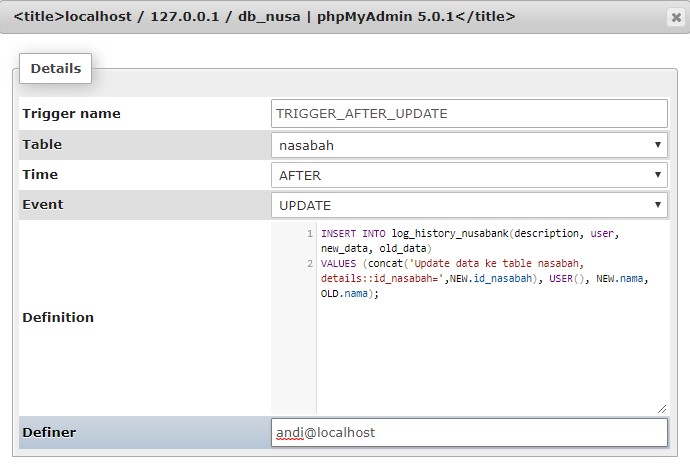
* Description: untuk menampung keterangan detail mengenai perubahan yang terjadi
* Log\_time: untuk mencatat keterangan waktu kapan perubahan terjadi
* User: merupakan user database administrator yang melakukan proses data manipulation tersebut
* Old\_data: merupakan value data yang sebelumnya
* New\_data: merupakan value data terbaru yang diubah atau ditambah.

1. **Membuat Trigger After Insert**

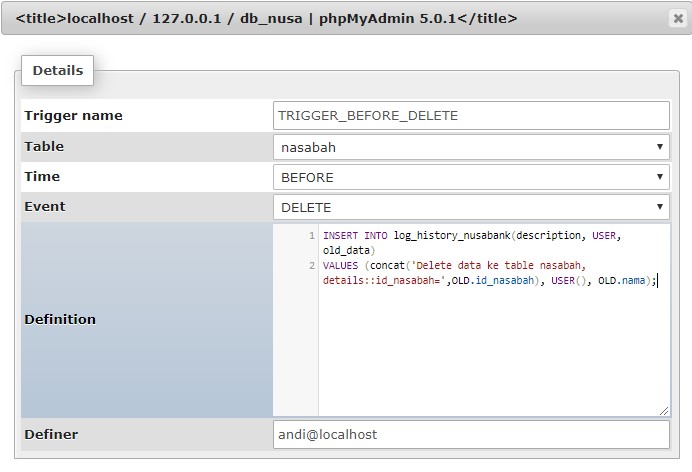


Disini trigger yang masuk akan mentrack perubahan setelah operas insert dilakukan, detilnya akan ditambahkan ke table log history, untuk sementara tracking perubahannya hanya pada id dan nama nasabah.

1. **Membuat Trigger After Update**



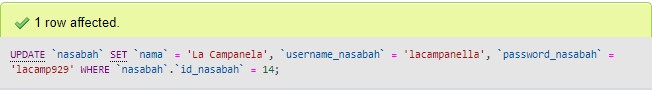
1. **Membuat Trigger Before Delete**



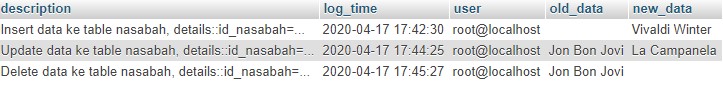
1. **Uji coba trigger**
2. Test insert query ke table nasabah



1. Test update query di table nasabah



1. Delete duplikat nasabah Jon Bon Jovi
2. Result trigger pada table log\_history\_nusabank



1. **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 mengidentifikasikan secara unik untuk suatu kejadian spesifik dari suatu entitas. Dengan spesifikasi sbb: unique, non-redundan.

1. **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 kit abaca otomatis untuk mencari tau hal tersebut kita bisa menggunakan bantuan index yang ada pada buku tersebut, missal 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 sepert 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.

1. **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.