DATABASE INTEGRITY

Integrity Constraints

Constraint : aturan yang diberikan ke tabel agar data yang dimasukkan valid

Integrity Constraints akan melindungi database dari kerusakan dengan memastikan bahwa perubahan yang dilakukan tidak menyebabkan inkonsistensi data

Jenis Integrity Constraint :

1. Inter-relational *Constraint* (constraint pada satu tabel)
   Meliputi : **Entity Integrity Constraint** dan **Domain Constraint**.
2. Intra-relasional *Constraint* (constraint melibatkan > 1 tabe1)
   Meliputi : **Referential integrity**.
3. **Enterprise Integrity (User Defined Integration)**

DATABASE INTEGRITY

Tujuan **database integrity** adalah :

1. Untuk memastikan integritas database berfungsi
2. Untuk melindungi database dari data yang tidak lengkap, tidak akurat, dan tidak konsisten.
3. Untuk memastikan kualitas data agar data bisa menjadi acuan.

upaya lain untuk menjamin kualitas database adalah pengolahan data dengan menggunakan **store procedure** dan **trigger**

DATABASE INTEGRITY

Entity integrity constraints

- Diaplikasikan dengan mendeklarasikan **primary key** untuk memastikan tidak ada **duplicate rows**.
- Option lain yang digunakan untuk memastikan isi entity valid adalah unique, not null, check, enum dan lain-lain

Contoh :

```
CREATE TABLE Buying
        (IDBuy smallint, IDModel smallint,
        descModel VARCHAR(40),);
        PRIMARY KEY (IDBuy),
        UNIQUE (IDModel));
```

implementation of entity integrity constraint

DATABASE INTEGRITY

Domain constraints

- Domain constraint adalah constraint yang paling sederhana. Setiap data baru dimasukkan, akan segera diperiksa oleh sistem.

- Domain constraints mencakup : data type, width, rentang nilai dan lain-lain.

- Penggunaan Domain constraints menyebabkan tidak ada data yang melanggar rentang nilai

DATABASE INTEGRITY

Domain constraints

Jenis constraint pada Domain Constraints :

PRIMARY KEY Constraint

Digunakan untuk spesifikasi atribut pada tabel, seperti NOT NULL, UNIQUE, dan operasi join

FOREIGN KEY Constraint

Digunakan untuk merelasikan ke tabel lain. Atribut foreign key digunakan untuk merelasikan ke attribut primary key dari tabel lain.

UNIQUE Constraint

Digunakan untuk memastikan bahwa data dari suatu atribut tidak diijinkan duplicate.

DATABASE INTEGRITY

Domain constraints

Jenis constraint pada Domain Constraints :

CHECK Constraint

Digunakan untuk membatasi nilai pada atribut dengan batasan nilai tertentu

DEFAULT Constraint

Digunakan untuk memberikan spesifikasi nilai atribut. Jika atribut tidak diisi, maka nilai default yang akan disimpan

NOT NULL

Digunakan untuk memastikan bahwa nilai atribut tidak boleh NULL

DATABASE INTEGRITY

Domain constraints

Contoh :

```
Create table student (regNo char(15) primary key,
        name varchar(25) not null unique, address varchar(30),
        sex char(1) default ‘M’ check (sex = ‘M’ or sex = ‘F’));
```

Catatan :

Atribut **regNo** adalah primary key, not null, dan unique

Atribut **name** adalah not null dan unique

Atribut **Sex** hanya dapat diisi karakter ‘M’ atau ‘F’ ('M' is male and 'F' is female), jika atribut SEX tidak diisi maka atribut akan diisi karakter 'M'

DATABASE INTEGRITY

Referential integrity constraints

- Seperangkat aturan yang mengatur hubungan antara primary key tabel dengan foreign key tabel lain untuk menjaga konsistensi data

- Menjaga integritas nilai atribut dalam tabel yang mengacu pada nilai atribut dari tabel lain

- Referential integrity mencakup operasi :
  - Insert operations
  - Delete operations
  - Update operations

DATABASE INTEGRITY

Referential integrity constraints

Operasi Referential integrity dapat ditulis :

REFERENCES tbl_name (fliedkey) [ON UPDATE {RESTRICT | CASCADE | SET NULL}]

Atau

REFERENCES tbl_name (fliedkey) [ON DELETE {RESTRICT | CASCADE | SET NULL}]

Note :

UPDATE : menyatakan tindakan jika pada tabel induk terjadi operasi update

DELETE : menyatakan tindakan jika pada tabel induk terjadi operasi delete

RESTRICT : operasi delete dan update pada tabel induk ditolak.

CASCADE : Jika nilai primary key dari tabel induk di UPDATE, maka nilai foreign key pada tabel referensi akan diperbarui dengan nilai kunci utama dari tabel induk. Jika di tabel induk di lakukan operasi DELETE, maka nilai foreign key tabel referensi akan dihapus.

SET NULL : Nilai foreign key akan diisi NULL, jika nilai primary key tabel induk di delete atau di update.

DATABASE INTEGRITY

Referential integrity constraints

Contoh :

Tabel Student

CREATE TABLE student (regNo char(8), name varchar(20), PRIMARY KEY (regNo));

Tabel Course

CREATE TABLE course (course_ID char(3), course_name varchar(20), PRIMARY KEY (course_ID));

Tabel Takes

CREATE TABLE takes(regNo char(8), course_ID char(3), grade dec(3,2), PRIMARY KEY (regNo, course_ID), FOREIGN KEY (regNo) REFERENCES student (regNo) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (course_ID) REFERENCES course (course_ID) ON DELETE CASCADE ON UPDATE CASCADE);

DATABASE INTEGRITY

Enterprise Integrity / User Defined Integration / General constraints

Constraint ini mengacu pada aplikasi, aturan bisnis yang didefinisikan oleh user atau DBA (Database Administrator)

Contoh :

Sebuah Kantor cabang hanya boleh memiliki maksimum 20 pegawai, sehingga setiap pegawai baru akan ditempatkan pada kantor cabang yang jumlah pegawainya belum mencapai 20 pegawai

DATABASE INTEGRITY

Store Procedure

Store Procedure adalah fasilitas baru pada MySQL 5.x. Store procedure berisi sekumpulan pernyataan SQL yang disimpan di server MySQL.

Procedure dijalankan dengan cara menuliskan operasi CALL dan dapat mengembalikan nilai melalui variabel output

Manfaat store procedure :

1. Meminimalkan Minimalkan lalu lintas permintaan dari aplikasi ke database, karena semua proses bisnis dilakukan dalam database MySQL dan aplikasi akan menerima hasil proses saja.

2. Meningkatkan keamanan basis data. User tidak dapat mengakses tabel tertentu secara langsung tetapi hanya dapat mengakses melalui store procedure.

DATABASE INTEGRITY

Syntax Umum:

CREATE PROCEDURE <procedure_name> (parameter)
<procedure_characteristic>
<program_body>

Contoh 1:

Store procedure yang digunakan untuk men-display data pegawai berdasarkan job tertentu dari tabel emp :

```
Delimiter  $$  /* change delimiter ; into $$ */
Create procedure display_emp (xjob char(10))
Begin
   Select * from emp where job=xjob;
End;
$$
```

the changes of delimiter is temporary (during the MySQL database is still active)

Store prosedur dapat diakses melalui :

Mysql>call display_emp('salesman') $$

DATABASE INTEGRITY

Contoh 2 :

Store procedure yang digunakan untuk menghitung jumlah record dari tabel emp. Jumlah record disimpan pada variabel param1 :

```
Delimiter $$
Create procedure display_emp 1 (out param1 int)
Begin
    Select count(*) into param1  from emp;
End;
$$
```

Store Procedure dapat diakses melalui :

Mysql>call display_emp1(@hasil) $$
Mysql>select @hasil $$

Jenis parameter yang digunakan pada store procedure terdiri dari :

IN → parameter untuk input (opsional)
OUT → parameter untuk output (menyimpan nilai output)
INOUT → parameter untuk output dan input

DATABASE INTEGRITY

Contoh 3 :

Store procedure yang digunakan untuk mengisi **tabel grade** dengan struktur tabel :

Create table grade (ID char(2) primary key, name varchar(10), exam1 dec(5,2), Exam2 dec(5,2), final dec(5,2), letter_value char(1))

```
Delimiter //
Create procedure grade_fill (x_ID char(2), x_name varchar(10), x_exam1 dec(5,2), x_exam2 dec(5,2))
Begin
   declare x_final dec(5,2);                    /* declaration of temporary variable */
   declare x_letter_value char(1);
   set x_final = (x_exam1 * 0.3 + x_exam2 *. 0.7);
   set x_letter_value = if(x_final>=85,'A',if(x_final>=70,'B', if(x_final>=60,'C', if(x_final>=50,'D','E'))));
   insert into grade values (x_ID, x_name, x_exam1, x_exam2, x_final, x_letter_value);
   Select * from grade;
End;
//
```

Store procedure dapat diakses melalui :

Mysql>call grade_fill('11', 'Ahmad', 80, 90) //

DATABASE INTEGRITY

Trigger

Trigger adalah jenis khusus dari Store Procedure yang melekat pada tabel. Trigger diakses secara otomatis saat terjadi manipulasi data tabel (insert, update, dan delete).

Syntax Umum :

CREATE TRIGGER *trigger_name trigger_time trigger_event* ON *table*
FOR EACH ROW *trigger_statement*

Note :
Trigger_time berisi : BEFORE atau AFTER
Trigger_event berisi :

- INSERT : trigger is active, if insert operations is executed
- UPDATE : trigger is active, if update operations is executed
- DELETE : trigger is active, if delete operations is executed

DATABASE INTEGRITY

Trigger

Contoh 1 :

```
Create trigger i_dept before insert on emp for each row
Begin
    declare xdname char(10);
    declare xloc char(10);
    set  xdname = if(new.deptno='10','ACCOUNTING',  if(new.deptno='20','RESEARCH',
                    if(new.deptno='30','SALES','OPERATIONS')));
     set  xloc = if(new.deptno='10','NEW  YORK', if(new.deptno='20','DALLAS',
                    if(new.deptno='30','CHICAGO','BOSTON')));
     insert into dept values (new.deptno, xdname, xloc);
End;
//
```

Jika pada tabel emp dilakukan operasi insert :

Mysql> **insert into emp values ('8888','JOHN','SALESMAN','7698','1984-10-12',1450,0,'30');**

Maka trigger akan dieksekusi secara otomatis

DATABASE INTEGRITY

Kombinasi store procedure dan trigger

Perhatikan 2 tabel :

```
CREATE TABLE student (regNo  char(8), name varchar(20), PRIMARY KEY (regNo));
CREATE TABLE log_student (event  varchar(15), time datetime);
```

Buat store procedure yang digunakan untuk mengisikan tabel student
Selanjutnya, buat trigger untuk menyimpan aktivitas insert pada tabel log secara otomatis.

Store Procedure

```
Delimiter //
Create procedure i_student (xregNo char(2),xname varchar(20))
Begin
    insert into student values (xregNo, xname);
    select * from student;
End
//
```

Trigger

```
Create trigger  i_log  after  insert  on student for each row
Begin
    insert into log_student  values ('add data', now());
End
//
```

Execution :
Mysql>call i_student ('11','Joni')//
Mysql>call i_student ('22','Smith')//

student

regNo	name
11	Joni
22	Smith

log_student

Event	Time
Add data	2017-10-19 11:30:00
Add data	2017-10-19 11:31:10

DATABASE INTEGRITY

Tugas

1. Buat trigger yang digunakan untuk menyimpan data di tabel log_student, jika operasi update atau delete di tabel student dilakukan:
   a. Jika operasi update dilakukan, maka atribut event pada tabel log_student diisi 'update data'
   b. Jika operasi delete dilakukan, maka atribut event pada tabel log_student diisi 'delete data'

log_student

Event	Time
Add data	2017-10-19 11:30:00
Add data	2017-10-19 11:31:10
....	
Update data	2017-10-19 11:36:10
Delete data	2017-10-19 11:38:20

records are stored from trigger operations result

DATABASE INTEGRITY

2. Perhatikan tabel –tabel berikut :

```
CREATE TABLE stock (ID char(5) primary key, name varchar(20),
                    unit varchar(10), stock_quantity int(5))

CREATE TABLE  sale (sale_no char(5) Primary key, cons_name varchar(10),
                    address varchar(20), city varchar(20), sale_date date,
                    ID char(5), sale_quantity int(4))

CREATE TABLE purchase (purchase_no char(5) Primary key,
                     supplier_name varchar(10), purchase_date date,
                     ID char(5), purchase_quantity int(4))
```

Pada statemen trigger, record tabel dapat diakses dengan menggunakan option **NEW** dan **OLD.**

NEW digunakan untuk mengambil record yang akan diproses (insert or update), sementara **OLD** digunakan untuk mengakses record yang sudah tersimpan (update or delete)

Example :

```
Update stock set stock_quantity = stock_quantity - new.sale_quantity
where ID = new.ID;
```

DATABASE INTEGRITY

a. Buat store procedure yang digunakan untuk memasukan (insert) data penjualan
b. Buat store procedure yang digunakan untuk memasukan (insert) data pembelian
c. Buat trigger yang digunakan untuk mengurangkan atribut stock_quantity dari tabel stock dengan atribut sale_quantity dari tabel sale berdasarkan transaksi pada point a.
d. Buat trigger yang digunakan untuk menambahkan atribut stock_quantity dari tabel stock dengan atribut purchase_quantity dari tabel purchase berdasarkan transaksi pada point b.
e. Buat trigger yang digunakan untuk mengurangkan atribut stock_quantity dari stock dengan atribut purchase_quantity dari tabel purchase, jika pada tabel purchase dilakukan penghapusan data (transaksi purchase dibatalkan). Proses pengurangan dilakukan dengan menyesuaikan ID yang dihapus pada tabel purchase.
f. Buat trigger yang digunakan untuk menambahkan atribut stock_quantity dari tabel stock dengan atribut sale_quantity dari tabel sale, jika pada tabel sale dilakukan penghapusan data (transaksi penjualan dibatalkan). Proses pengurangan dilakukan dengan menyesuaikan ID yang dihapus pada tabel sale.
g. Buat trigger yang digunakan untuk mengupdate atribut stock_quantity dari tabel stock dengan atribut purchase_quantity dari tabel purchase, jika pada tabel purchase dilakukan update data. Proses update dilakukan dengan menyesuaikan ID update pada tabel purchase.
h. Buat trigger yang digunakan untuk mengupdate atribut stock_quantity dari tabel stock dengan atribut sale_quantity dari tabel sale, Jika pada tabel sale dilakukan update data. Proses update dilakukan dengan menyesuaikan ID update pada tabel sale.

DATABASE INTEGRITY

sale

sale_no	cons_name	address	city	sale_date	ID	sale_quantity
S0001	Smith	Jl. A. Yani 20	Semarang	2017-10-11	22222	5
……						

stock

ID	name	unit	Stock_quantity
11111	shirt	piece	10
22222	jacket	piece	15
……			

Stock_quantity + sale_quantity

Stock_quantity - sale_quantity

Example :
Update stock set stock_quantity = stock_quantity - new.sale_quantity where ID = new.ID;

How to update stock_quantity, if the update operation is done ???

if the delete operation is done (transaction is canceled)

if the insert operation is done

Stock_quantity + purchase_quantity

Stock_quantity - purchase_quantity

purchase

purchase_no	supplier_name	ID	purchase_quantity
P0001	CV. Prima	22222	20
……			