1. Set Operators and Subqueries

There are several DBMSs (e.g., MySQL) that do not support “intersect” or “except”. This is because intersect and except operations can be expressed using row subqueries. This section explains how to use row subqueries to express “intersect” and “except” operations.

\[
\text{r(A, B, C)}
\]
\[
\text{s(B, C)}
\]

**INTERSECT**

\[(\text{select B, C from r}) \text{ intersect } (\text{select * from s})\];

can be rewritten as follows:

\[
\text{select distinct B, C from r}
\]
\[
\text{where } (B, C) \text{ in (select B, C from r) AND }
\]
\[
(B, C) \text{ in (select * from s)};
\]

**EXCEPT**

\[(\text{select B, C from r}) \text{ except } (\text{select * from s})\];

can be rewritten as follows:

\[
\text{select distinct B, C from r}
\]
\[
\text{where } (B, C) \text{ not in (select * from s)};
\]
2. Constraints (SQL:1992)
A constraint is a predicate expressing a condition that the database must always satisfy. Many DBMSs support the following types of constraints defined in SQL:1992 SQL standard:

**Table Constraints in a CREATE TABLE statement:**

```sql
[column name] [column type] <DEFAULT NULL (or a value)> <CONSTRAINT [name]>
NOT NULL
(or UNIQUE
or PRIMARY KEY
or FOREIGN KEY REFERENCES [target relation <cand. key>]
  <ON UPDATE (or DELETE) CASCADE
  (or SET NULL
  or SET DEFAULT
  or NO ACTION)>
or check ([predicate])
<deferability>

<CONSTRAINT [name]>
UNIQUE
(or PRIMARY KEY
or FOREIGN KEY REFERENCES [target relation <cand. key>]
  <ON UPDATE (or DELETE) CASCADE
  (or SET NULL
  or SET DEFAULT
  or NO ACTION)>
or check ([predicate])
<deferability>

deferability:
INITIALLY DEFERRED (or INITIALLY IMMEDIATE) <<NOT> DEFERRABLE>

NOTE: When a constraint is INITIALLY DEFERRED the condition is checked at the end of a transaction, and not at intermediate steps. For constraints declared as DEFERRABLE, executing “SET CONSTRAINTS [constraint list] DEFERRED;” as part of a transaction causes the checking to be deferred to the end of the transaction.

**ADDING and DROPPING constraints using ALTER TABLE:**

ALTER TABLE [table name]
ADD [a table constraint];

ALTER TABLE [table name] (in the standard???)
DROP [a table constraint name]
or PRIMARY KEY
or FOREIGN KEY (column list)
or UNIQUE (column list));

**Database Constraints (ASSERTIONs):**

CREATE ASSERTION [assertion name] CHECK ([predicate]) `<deferability>`;
DROP ASSERTION [assertion name];

---

**3. Triggers (SQL:1999)**

Each individual constraint has only “condition”. Triggers differ from constraints in that each trigger consists of not only a condition but also one or more actions that are sequentially taken when the condition is satisfied.

**The standard:**

```sql
CREATE TRIGGER [trigger name]
BEFORE (or AFTER)
INSERT (or DELETE or UPDATE <OF [column names separated by comma])
ON [table name] <REFERENCING alias>
<FOR EACH ROW (or STATEMENT)>
<WHEN (predicate)>
BEGIN ATOMIC [two or more SQL statements*] END (or a single SQL statement*);
```

*alias: OLD (or NEW) <ROW> <AS> alias
  or
  OLD (or NEW) TABLE <AS> alias

*: in the SQL statements, “NEW” and “OLD” (and the corresponding aliases) refer to the old record (or table) and the new record (or table), respectively.

**DROP TRIGGER [trigger name]**

NOTE: <> represents optional parameter

**MySQL Ver. 5.x (not stable, only a beta version exists as of 10/12/2005):**

```sql
CREATE TRIGGER [trigger name]
BEFORE (or AFTER)
INSERT (or DELETE or UPDATE)
ON [table name]
FOR EACH ROW
BEGIN [two or more SQL statements*] END (or a single SQL statement*);
```
*: in the SQL statements, “NEW” and “OLD” refer to the old record and the new record, respectively.

**DROP TRIGGER** [schema_name.]trigger_name

The schema name is optional. Prior to MySQL 5.0.10, the table name was required instead of the schema name (table_name.trigger_name).

**Examples from the official MySQL Ver 5.x manual:**

CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE test4(a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
b4 INT DEFAULT 0);

CREATE TRIGGER testref
BEFORE
INSERT ON test1
FOR EACH ROW
BEGIN
INSERT INTO test2 SET a2 = NEW.a1;
DELETE FROM test3 WHERE a3 = NEW.a1;
UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
END

INSERT INTO test3 (a3) VALUES
(NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL), (NULL);

INSERT INTO test4 (a4) VALUES
(0), (0), (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);

mysql> **INSERT INTO** test1 **VALUES**
-> (1), (3), (1), (7), (1), (8), (4), (4);
Query OK, 8 rows affected (0.01 sec)
Records: 8 Duplicates: 0 Warnings: 0

mysql> **SELECT** * **FROM** test1;

<table>
<thead>
<tr>
<th>a1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Dr. Byunggu Yu
mysql> SELECT * FROM test2;
+------+
| a2 |
| 1 |
| 3 |
| 1 |
| 7 |
| 1 |
| 8 |
| 4 |
| 4 |
| 4 |
+------+

mysql> SELECT * FROM test3;
+----+
| a3 |
| 2 |
| 5 |
| 6 |
| 9 |
| 10 |
+----+

SELECT * FROM test4;
+----+------+
| a4 | b4 |
| 1  | 3   |
| 2  | 0   |
| 3  | 1   |
| 4  | 2   |
| 5  | 0   |
| 6  | 0   |
| 7  | 1   |
| 8  | 1   |
| 9  | 0   |
| 10 | 0   |
+----+------+
You can refer to columns in the table associated with the trigger by using the aliases OLD and NEW. OLD.col_name refers to a column of an existing row before it is updated or deleted. NEW.col_name refers to the column of a new row to be inserted or an existing row after it is updated.

NOTE: Currently, triggers are not activated by cascaded foreign key actions. This limitation will be lifted as soon as possible.

**Oracle**: The actions must be written in Oracle PL/SQL (Examples can be found in the Oracle manual posted on the “References and Resources” page of the class web site).

**MS SQL**: An example can be found in the textbook.

### 4. Transaction Commands (SQL:1992)

```sql
COMMIT;

and

ROLLBACK;
```