How to maintain integrity?

- `PRIMARY KEY, UNIQUE, NOT NULL`
- `FOREIGN KEY` — *referential integrity*
- `CHECK`
  - constraints on *single* attribute / table
- `CREATE ASSERTION`
  - constraints on interaction among *multiple* tables
  - hard to implement, not supported in MySQL
- `CREATE TRIGGER`
  - specify *event-condition-action* rules

MySQL `CHECK` Constraints

```
CREATE TABLE t1 (
    CHECK (c1 != c2),
    c1 INT CHECK (c1 > 10),
    c2 INT CONSTRAINT c2_positive CHECK (c2 > 0),
    c3 INT CHECK (c3 < 100),
    CONSTRAINT c1_nonzero CHECK (c1 <> 0),
    CHECK (c1 > c3)
);
ALTER TABLE t1 DROP CONSTRAINT c2_positive;
ALTER TABLE t1 ADD CONSTRAINT c2_negative
    CHECK (c2 < 0);
```

[CONSTRAINT [*symbol*]] CHECK (*expr*)

forward reference allowed

column constraint

table constraint

This is an ALTER TABLE statement

becomes a table constraint

MySQL `CHECK` Constraints

```
[CONSTRAINT [symbol]] CHECK (expr)
```

- Column constraint `c1 INT CHECK (c1 > 10)`
  - appears within a column definition
  - can refer only to that column
  - checked whenever a row is inserted or that column is updated in some row
- Table constraint `CHECK (c1 > c3)`
  - appears outside any column definition
  - can refer to all columns in the table
  - checked whenever a row is inserted / updated

If *expr* evaluates to `FALSE`, then insertion / update is rejected

If you insert `NULL`, then *expr* evaluates to `UNKNOWN`, so the insertion is allowed

Example

Suppose we update `c2` in some row, then the column constraint on `c1` won't be checked, but all table constraints will be checked, even if they don't involve `c2`

General SQL `CHECK` Constraints

`[CONSTRAINT [`*`symbol`*`]] CHECK (`*`expr`*`)`

- *`expr`* can refer to any attribute / relation via a subquery
- not supported in MySQL
- constraint can sometimes be violated

```
CREATE TABLE Sells (
    bar   CHAR(20),
    beer  CHAR(20) CHECK (beer IN
          (SELECT name FROM Beers)),
    price FLOAT
);
```

Deletion in `Beers` is invisible to this constraint!

Assertions

Type	Where Declared	When Activated	Guaranteed to Hold?
Column CHECK	With attribute	On insertion to relation or attribute update	Not if subqueries
Table CHECK	Element of relation schema	On insertion to relation or tuple update	Not if subqueries
Assertion	Element of database schema	On any change to any mentioned relation	Yes

```
CREATE ASSERTION <name> CHECK (<condition>);
DROP ASSERTION <name>;
```

<condition> may refer to any relation or attribute in the database schema

Assertions

- There cannot be more bars than drinkers

```
CREATE ASSERTION FewBar CHECK (
      (SELECT COUNT(*) FROM Bars) <=
      (SELECT COUNT(*) FROM Drinkers)
);
```

Assertions

- No bar may charge an average of more than $5

```
CREATE ASSERTION NoRipoffBars CHECK (
     NOT EXISTS (
          SELECT bar FROM Sells
          GROUP BY bar
          HAVING 5.00 < AVG(price)
     )
);
```

Triggers: Motivation

- Assertions are powerful, but the DBMS often can’t tell when they need to be checked
- CHECK constraints are checked at known times, but are less powerful
- Triggers let the user decide when to check for any condition, and specify the action to take

MySQL Triggers

```
CREATE TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    trigger_body

trigger_time:  { BEFORE | AFTER  }
trigger_event: { INSERT | UPDATE | DELETE }

DROP TRIGGER schema_name.trigger_name
```

MySQL Triggers

```
CREATE TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    trigger_body
```

- `INSERT`: activates whenever a new row is inserted into the table *`tbl_name`*
- `UPDATE`: activates whenever a row in the table *`tbl_name`* is modified
- `DELETE`: activates whenever a row is deleted from the table *`tbl_name`*
  `DROP TABLE` statements do not activate this trigger
  Instead, the trigger will be dropped

MySQL Triggers

```
CREATE TRIGGER trigger_name
      trigger_time trigger_event
      ON tbl_name FOR EACH ROW
      trigger_body
```

- *trigger_body* is executed once for each row affected by the triggering event
- OLD.*col_name* refers to a column of an existing row
  before it is updated or deleted
  read only
- NEW.*col_name* refers to the column of a new row *to be inserted*
  or the new column value *to be used for update*

If you want to modify NEW.*col_name*, specify *trigger_time* to be BEFORE

MySQL Triggers

```
CREATE TABLE account (acct_num INT, amount FLOAT);

CREATE TRIGGER upd_check
    BEFORE UPDATE ON account
    FOR EACH ROW
    BEGIN
        IF NEW.amount < 0 THEN
            SET NEW.amount = 0;
        ELSEIF NEW.amount > 100 THEN
            SET NEW.amount = 100;
        END IF;
    END;
```

the value must be checked and modified before it is used to update the row

MySQL Triggers

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);

CREATE TRIGGER testref
    BEFORE INSERT ON test1
    FOR EACH ROW
    DELETE FROM test2
        WHERE a2 = NEW.a1;
```

- Emp (emp_id, salary)
- Dept (dept_id, manager_id)
- Works (emp_id, dept_id)

- Employees must have a minimum salary of $1000
- Every manager must be also be an employee
- A manager must always have a higher salary than any employee that he or she manages

- Emp (emp_id, salary)
- Dept (dept_id, manager_id)
- Works (emp_id, dept_id)

- Employees must have a minimum salary of $1000

```
CREATE TABLE Emp (
    emp_id INT PRIMARY KEY,
    salary FLOAT CHECK (salary >= 1000)
);
```

- Emp (emp_id, salary)
- Dept (dept_id, manager_id)
- Works (emp_id, dept_id)

- Every manager must be also be an employee

```
CREATE TABLE Dept (
    dept_id INT PRIMARY KEY,
    manager_id INT,
    FOREIGN KEY (manager_id)
        REFERENCES Emp(emp_id)
);
```

- Emp (emp_id, salary)
- Dept (dept_id, manager_id)
- Works (emp_id, dept_id)

- A manager must always have a higher salary than any employee that he or she manages

```
CREATE ASSERTION ManagerHigherSalary CHECK (
    NOT EXISTS (
        SELECT E.emp_id FROM Emp E, Emp M, Dept D, Works W
        WHERE E.emp_id = W.emp_id AND W.dept_id = D.dept_id
            AND D.manager_id = M.emp_id
            AND M.salary <= E.salary
    )
);
```