>>

Constraints

- Constraints
- Assertions

^ >>

❖ Constraints

So far, we have considered several kinds of constraints:

- attribute (column) constraints — constraint on a single attribute
- relation (table) constraints — general constraint that can involve multiple attributes
- referential integrity constraints — fkey -> pkey integrity

Examples:

```
create table Employee (
    id       integer primary key,
    name     varchar(40),
    salary   real,
    age      integer check (age > 15),
    worksIn integer
                references Department(id),
    constraint PayOk check (salary > age*1000)
);
```

<< ^ >>

❖ Constraints (cont)

Column and table constraints ensure validity of one table.

Ref. integrity constraints ensure connections between tables are valid.

However, specifying validity of entire database often requires constraints involving multiple tables.

Simple example (from banking domain):

```
for all Branches b
   b.assets == (select sum(acct.balance)
                from   Accounts acct
                where  acct.branch = b.location)
```

i.e. assets of a branch is sum of balances of accounts held at that branch

<< ^ >>

❖ Assertions

Assertions are schema-level constraints

- typically involving multiple tables
- expressing a condition that must hold at all times
- need to be checked on each change to relevant tables
- if change would cause check to fail, reject change

SQL syntax for assertions:

```
CREATE ASSERTION name CHECK (condition)
```

The *condition* is expressed as "there are no violations in the database"

Implementation: ask a query to find all the violations; check for empty result

<< ^ >>

❖ Assertions (cont)

Example: #students in any UNSW course must be < 10000

```
create assertion ClassSizeConstraint check (
   not exists (
      select c.id
      from   Courses c
             join Enrolments e on (c.id = e.course)
      group  by c.id
      having count(e.student) > 9999
   )
);
```

Needs to be checked after *every* change to either **Courses** or **Enrolments**

❖ Assertions (cont)

Example: assets of branch = sum of its account balances

```
create assertion AssetsCheck check (
   not exists (
       select branchName from Branches b
       where  b.assets <>
              (select sum(a.balance) from Accounts a
               where a.branch = b.location)
   )
);
```

Needs to be checked after *every* change to either **`Branches`** or **`Accounts`**

<< ^

❖ Assertions (cont)

On each update, it is expensive

- to determine which assertions need to be checked
- to run the queries which check the assertions

A database with many assertions would be way too slow.

So, most RDBMSs do *not* implement general assertions.

Typically, triggers are provided as

- a lightweight mechanism for dealing with assertions
- a general event-based programming tool for databases

Triggers typically enforce assertions rather than *checking* them

Produced: 12 Oct 2020