Integrity Constraints

- Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.
  - A checking account must have a balance greater than $10,000.00
  - A salary of a bank employee must be at least $4.00 an hour
  - A customer must have a (non-null) phone number

Constraints on a Single Relation

- **not null**
- **primary key**
- **unique**
- **check** (*P*), where *P* is a predicate

Not Null Constraint

- Declare *branch_name* for *branch* is **not null**

  branch_name **char**(15) **not null**

- Declare the domain *Dollars* to be **not null**

  create domain *Dollars* **numeric**(12,2) **not null**

The Unique Constraint

- **unique** ($A_1, A_2, \ldots, A_m$)

The unique specification states that the attributes

$A_1, A_2, \ldots A_m$

Form a candidate key.

Candidate keys are permitted to be null
(in contrast to primary keys).

The check clause

- **check** (P), where P is a predicate

Declare *branch_name* as the primary key for *branch* and ensure that the values of *assets* are non-negative.

```
create table branch
        (branch_name    char(15),
         branch_city     char(30),
         assets          integer,
         primary key     (branch_name),
         CHECK           (assets >= 0))
```


The check clause (Cont.)

- The **check** clause permits domains to be restricted:
  - Use **check** clause to ensure that an hourly_wage domain allows only values greater than a specified value.

```
create domain hourly_wage numeric (5,2)
        constraint value_test check(value > = 4.00)
```

  - The domain has a constraint that ensures that the hourly_wage is greater than 4.00
  - The clause **constraint** *value_test* is optional; useful to indicate which constraint an update violated.

Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes also appears for a set of attributes in another relation.
  - Example: If "Perryridge" is a branch name appearing in one of the tuples in the *account* relation, then there exists a tuple in the *branch* relation for branch "Perryridge".
- Primary and candidate keys and foreign keys can be specified as part of the SQL **create table** statement:
  - The primary key clause lists primary key (PK) attributes.
  - The unique key clause lists candidate key attributes
  - The foreign key clause lists foreign key (FK) attributes and the name of the relation referenced by the FK. By default, a FK references PK attributes of the referenced table.

Referential Integrity in SQL – Example

create table *customer*
(*customer_name* **char**(20)**,**
customer_street **char**(30),
customer_city **char**(30),
primary key (*customer_name*))

create table *branch*
(branch_name **char**(15)**,**
branch_city **char**(30),
assets **numeric**(12,2),
primary key (*branch_name*))

Referential Integrity in SQL – Example (Cont.)

create table *account*
(*account_number* **char**(10)**,**
branch_name **char**(15),
balance **integer**,
primary key (*account_number),*
foreign key (*branch_name*) **references** *branch*)

create table *depositor*
(*customer_name* **char**(20)**,**
account_number **char**(10)**,**
primary key (*customer_name, account_number),*
foreign key (*account_number*) **references** *account,*
foreign key (*customer_name*) **references** *customer*)

Assertions

- An **assertion** is a predicate expressing a condition that we wish the database always to satisfy.
- An assertion in SQL takes the form

  create assertion <assertion-name> **check** <predicate>
- When an assertion is made, the system tests it for validity, and tests it again on every update that may violate the assertion
  - This testing may introduce a significant amount of overhead; hence assertions should be used with great care.
- Asserting
  for all X, $P(X)$
  is achieved in a round-about fashion using
  not exists X such that not $P(X)$

Using General Assertions

- Specify a query that violates the condition; include inside a `NOT EXISTS` clause
- Query result must be empty
  - if the query result is not empty, the assertion has been violated

Assertion Example

- Every loan has at least one borrower who maintains an account with a minimum balance or $1000.00

```
create assertion balance_constraint check (not exists
   (select * from loan
    where not exists
        (select *
         from borrower, depositor, account
         where loan.loan_number = borrower.loan_number
         and borrower.customer_name = depositor.customer_name
         and depositor.account_number = account.account_number
         and account.balance >= 1000)))
```

Assertion Example

- The sum of all loan amounts for each branch must be less than the sum of all account balances at the branch.

```
create assertion sum_constraint check
    (not exists (select *
                 from branch
                 where (select sum(amount )
                        from loan
                        where loan.branch_name =
                              branch.branch_name )
                       >= (select sum (amount )
                           from account
                           where loan.branch_name =
                                 branch.branch_name )))
```


Assertions: Another Example

- "The salary of an employee must not be greater than the salary of the manager of the department that the employee works for''

```
CREAT ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS
   (SELECT *
    FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D
    WHERE   E.SALARY > M.SALARY
    AND     E.DNO=D.NUMBER
    AND     D.MGRSSN=M.SSN))
```


SQL Triggers

- Objective: to monitor a database and take action when a condition occurs
- Triggers are expressed in a syntax similar to assertions and include the following:
  - event (e.g., an update operation)
  - condition
  - action (to be taken when the condition is satisfied)

SQL Triggers: An Example

- A trigger to compare an employee's salary to his/her supervisor during insert or update operations:

```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
        SALARY, SUPERVISOR_SSN ON EMPLOYEE
FOR EACH ROW
   WHEN (NEW.SALARY >
                (SELECT SALARY FROM EMPLOYEE
                 WHERE SSN=NEW.SUPERVISOR_SSN))
   INFORM_SUPERVISOR
   (NEW.SUPERVISOR_SSN,NEW.SSN;
```


Summary

- Assertions provide a means to specify additional constraints
- Triggers are a special kind of assertions; they define actions to be taken when certain conditions occur
- Views are a convenient means for creating temporary (virtual) tables