Data Definition Language

Allows the specification of not only a set of relations but also information about each relation, including:

- The schema for each relation.
- The domain of values associated with each attribute.
- Integrity constraints
- The set of indices to be maintained for each relations.
- Security and authorization information for each relation.
- The physical storage structure of each relation on disk.

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

Create Table Construct

- An SQL relation is defined using the **create table** command:

  create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - r is the name of the relation
  - each A_i is an attribute name in the schema of relation r
  - D_i is the data type of values in the domain of attribute A_i

- Example:

  create table *branch*
  (*branch_name* char(15) **not null,**
  branch_city char(30),
  assets integer)

CREATE TABLE

- In SQL2, can use the CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys).
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE   DEPT
(    DNAME                  VARCHAR(10)  NOT NULL,
     DNUMBER                INTEGER      NOT NULL,
     MGRSSN                 CHAR(9),
     MGRSTARTDATE           CHAR(9),
     PRIMARY KEY            (DNUMBER),
     UNIQUE                 (DNAME),
     FOREIGN KEY            (MGRSSN) REFERENCES EMP  );
```

Integrity Constraints in Create Table

- **not null**
- **primary key** $(A_1, ..., A_n)$

Example: 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))
```

primary key declaration on an attribute automatically ensures **not null** in SQL-92 onwards, needs to be explicitly stated in SQL-89

DROP TABLE

- Used to remove a relation (base table) *and its definition*
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

DROP TABLE DEPENDENT;

Drop and Alter Table Constructs

- The **drop table** command deletes all information about the dropped relation from the database.
- The **alter table** command is used to add attributes to an existing relation: **alter table** *r* **add** *A D*

  where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A*.
  - All tuples in the relation are assigned *null* as the value for the new attribute.
- The **alter table** command can also be used to drop attributes of a relation: **alter table** *r* **drop** *A*

  where *A* is the name of an attribute of relation *r*
  - Dropping of attributes not supported by many databases

ALTER TABLE

- Used to add an attribute to one of the base relations
- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is *not allowed* for such an attribute
- Example:
  ALTER TABLE EMPLOYEE
  ADD JOB VARCHAR(12);
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

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