Week 2: Part I
SQL I – Data Definition Language

Domains, Schema Definitions, and Constraints

SQL
- The name is an acronym for **Structured Query Language**. It is actually far richer than a query language: supports both a DML and a DDL.
- First proposal: SEQUEL (IBM Research, 1974); First implementation in SQL/DS (IBM, 1981)
- Standardization crucial for its diffusion
  - Since 1983, *de facto* standard;
  - First official standard, 1986; revised in 1989;
  - Second standard, 1992 (SQL-2 or SQL-92);
  - Third standard, 1999 (SQL-3 or SQL-99)
- Most relational DBMS support the base functionality of the standard and offer proprietary extensions.

Domains

- Domains specify allowable values for attributes.
- Two categories:
  - Elementary (predefined by the standard);
  - User-defined.

Elementary Domains — Character

- **Character**
  - Single characters or strings;
  - Strings may be of variable length;
  - A Character set different from the default one can be used (e.g., Latin, Greek, Cyrillic, etc.)
  - Syntax:
    - `character [ varying ] [ (Length) ]` *character set CharSetName*
    - It is possible to use `char` and `varchar`, for `character` and `character varying` respectively
More Elementary Domains

- **Bit**
  - Single Boolean values or strings of Boolean values (may be variable in length);
  - Syntax:
    - `bit [ varying] [(Length)]`

- Exact numeric domains
  - Exact values, integer or with a fractional part
  - Four alternatives:
    - `numeric(Precision [, Scale])]`
    - `decimal(Precision [, Scale])]`
    - `integer`
    - `smallint` # of significant digits decimal digits

Approximate Numeric Domains

- Approximate numeric domains
- Approximate real values
- Based on a floating point representation
  - `float(Precision)]` e.g., `0.17E16, 0.41E-6`
  - `double precision`
  - `real` — behaves like `float`, but has variable precision

Temporal Instant Domains

- Temporal instants
  - `date` has fields `year, month, day`
  - `time(Precision)[ with time zone]` has fields `hour, minute, second`
  - `timestamp(Precision)[ with time zone]`

- Temporal intervals
  - `interval FirstUnitOfTime [ to LastUnitOfTime ]`
  - Units of time are divided into two groups: (i) year, month, (ii) day, hour, minute, second
  - For example, `year(5) to month` allows intervals up to `99999yrs + 11mo`

User-Defined Domains

- Comparable to definitions of variable types in programming languages.
- A domain is characterized by name, elementary domain, default value, set of constraints
- Syntax:
  - `create domain DomainName as ElementaryDomain [ DefaultValue ] [ Constraints ]`
- Example:
  - `create domain Mark as smallint default null`
### Default Domain Values
- Define the value that the attribute must assume when a value is not specified during row insertion.
- Syntax: `default < GenericValue | user | null >`
- `GenericValue` represents a value compatible with the domain, in the form of a constant or an expression.
- `user` is the login name of the user who assigns a value to this attribute.

### Table Definition
- An SQL table consists of an ordered set of attributes, and a (possibly empty) set of constraints.
- Statement `create table` defines a relation schema, creating an empty instance.
- Syntax:
  ```sql
create table TableName
    ( AttributeName Domain [ DefaultValue ] [ Constraints ]
    {, AttributeName Domain [ DefaultValue ] [ Constraints ]}
    [ OtherConstraints ]
  )
  ```

### Example of `create table`
```sql
create table Employee
  ( RegNo character(6) primary key,
    FirstName character(20) not null,
    Surname character(20) not null,
    Dept character (15) references Department(DeptName) on delete set null on update cascade,
    Salary numeric(9) default 0,
    City character(15),
    unique(Surname,FirstName)
  )
```
Intra-Relational Constraints

- Constraints are conditions that must be verified by every database instance
- Intra-relational constraints involve a single relation
  - `not null` (on single attributes)
  - `unique`: permits the definition of keys; syntax:
    - for single attributes: `unique`, after the domain
    - for multiple: `unique (Attribute {, Attribute })`
  - `primary key`: defines the primary key (once for each table; implies `not null`); syntax like `unique`
  - `check`: described later

Example of Intra-Relational Constraints

- Each pair of `FirstName` and `Surname` uniquely identifies each element
  - `FirstName` char(20) not null,
  - `Surname` char(20) not null,
  - `unique (FirstName, Surname)`
- Note the difference with the following (stricter) definition:
  - `FirstName` char(20) not null unique,
  - `Surname` char(20) not null unique,
  - ...

Inter-Relational Constraints

- Constraints may involve several relations:
  - `check`: checks whether an assertion is true;
  - `references` and `foreign key` permit the definition of referential integrity constraints;
    - Syntax for single attributes
      - `references` after the domain
    - Syntax for multiple attributes
      - `foreign key (Attribute {, Attribute })` `references` ...
  - It is possible to associate reaction policies to violations of referential integrity constraints.

Reaction Policies

Violations arise from
- (a) updates on referred attribute or
- (b) row deletions.

Reactions operate on internal table, after changes to an external table.
- Reactions are:
  - `cascade`: propagate the change;
  - `set null`: nullify the referring attribute;
  - `set default`: assign default value to the referring attribute;
  - `no action`: forbid the change on external table.
- Reactions may depend on the event; syntax:
  - `on < delete | update >
  - < cascade | set null | set default | no action >`
Example

create table Employee
(
    RegNo char(6),
    FirstName char(20) not null,
    Surname char(20) not null,
    Dept char(15),
    Salary numeric(9) default 0,
    City char(15),
    primary key(RegNo),
    foreign key(Dept)
        references Department(DeptName)
        on delete set null
        on update cascade,
    unique(FirstName,Surname)
)

Relational Catalogues

- A relational catalogue contains the data dictionary, i.e., a description of the relational schema \( D \) of the database.
- It is based on a relational schema \( MD \) whose relations describe the relations, columns, domains in \( D \) but also \( MD \) (reflectivity).
- The SQL-2 standard describes a Definition_Schema (composed of tables) and an Information_Schema (composed of views).

Schema Updates

Two SQL statements:
- alter (alter domain...,alter table ...)
- drop < schema | domain | table | view | assertion >
  ComponentName [ restrict | cascade ]

Examples:
- alter table Department
  add column NoOfOffices numeric(4)
- drop table TempTable cascade

Column

<table>
<thead>
<tr>
<th>TableNm</th>
<th>ColName</th>
<th>Pos</th>
<th>Default</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>RegNo</td>
<td>1</td>
<td>Null</td>
<td>N</td>
</tr>
<tr>
<td>Employee</td>
<td>Name</td>
<td>2</td>
<td>Null</td>
<td>Y</td>
</tr>
<tr>
<td>Employee</td>
<td>Dept</td>
<td>3</td>
<td>Null</td>
<td>Y</td>
</tr>
<tr>
<td>Employee</td>
<td>Sal</td>
<td>4</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>Dept</td>
<td>Name</td>
<td>1</td>
<td>Null</td>
<td>N</td>
</tr>
<tr>
<td>Dept</td>
<td>Head</td>
<td>2</td>
<td>Null</td>
<td>Y</td>
</tr>
<tr>
<td>Dept</td>
<td>Address</td>
<td>3</td>
<td>Null</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>TableNm</td>
<td>1</td>
<td>Null</td>
<td>N</td>
</tr>
<tr>
<td>Column</td>
<td>ColName</td>
<td>2</td>
<td>Null</td>
<td>N</td>
</tr>
<tr>
<td>Column</td>
<td>Pos</td>
<td>3</td>
<td>Null</td>
<td>N</td>
</tr>
<tr>
<td>Column</td>
<td>Default</td>
<td>4</td>
<td>Null</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>Nullable</td>
<td>5</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

A Relational Catalogue
What is the DDL for the database schema store containing Employee and Dept on the previous slide?

create schema store {
  create table Employee (
    RegNo char(6),
    FirstName char(20) not null,
    Surname char(20) not null,
    Dept char(15),
    Salary numeric(9) default 0,
    City char(15),
    primary key(RegNo),
    foreign key(Dept) references Dept(DeptName) on delete set null on update cascade,
    unique(FirstName, Surname)
  )
  create table Dept (
    Name char(20) primary key,
    Head char(6) references Employee(RegNo) on delete set null on update cascade,
    Address char(20)
  )
}