**SQL**

“Structured Query Language”

- Standard for relational db systems
- History:
  - Developed at IBM in late 70s
  - First standard: SQL-86
  - Second standard: SQL-92
  - Third standard: SQL-99 or SQL3, well over 1000 pages!
  - Many more …..
  - “The nice things about standards is that you have so many to choose from” -- Andres S. Tannenbaum

---

**SQL Data Definition Language**

- Allows the specification of the database schema
- The name and attributes for each relation.
- The domain of values associated with each attribute.
- Integrity constraints
- The set of indices to be maintained for each relation.
- Security and authorization information for each relation.
- The physical storage structure of each relation on disk.
Some 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 \( d \) 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 Command

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

```sql
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 domain of attribute \( A_i \)

- Example:

```sql
create table branch
    (branch_name char(15) not null,
     branch_city char(30),
     assets integer)
```
Create Table (cont.)

- 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 keywords.

```
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);
```

The check clause

- `check (P)`, where $P$ is a predicate on attribute values

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)
);
```
Drop Table Command

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

```sql
DROP TABLE DEPENDENT;
```

Alter Table Command

- 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 default 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`
  Many databases do not support dropping of attributes
Alter Table (cont.)

• Since new attribute will have NULL values right after the ALTER command is executed, 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.