Structured Query Language (SQL)

- Data Definition Language
- Domains
- Integrity Constraints

The Banking Enterprise

- The banking enterprise database: (used throughout this section)

  branch (branch-name, branch-city, assets)

  customer (customer-name, customer-street, customer-city)

  account (account-number, branch-name, balance)

  loan (loan-number, branch-name, amount)

  depositor (customer-name, account-number)

  borrower (customer-name, loan-number)

Data Definition Language (DDL)

- DDL allows the specification of a set of tables.
- For each table a DDL statement specifies:
  - A name for the table
  - A name for each attribute
  - The domain (i.e., a type) of values associated with each attribute
  - Integrity constraints
  - An associated set of indices
  - Security and authorization information
  - The physical storage structure for the relation

Domain Types in SQL

- Basic SQL Types:

varchar(n)	- Variable length character string, maximum length *n*.
char(n)	- Fixed length character string, with length *n*.
int	- Integer (machine-dependent).
smallint	- Small integer (machine-dependent).
bigint	- Big integer (machine-dependent).
real	- Floating point numbers machine-dependent precision.
double precision	- Floating point numbers machine-dependent precision.
float(n)	- Floating point number, precision of at least *n* digits.
numeric(p,d)	- Fixed point number; *p* digits of precision and *d* digits to the right of decimal point.
plus others…	

Date/Time Types in SQL (Cont.)

- More complex types are also supported:
  - *date* - Dates, containing a year, month and date
  - *time* - Time of day, in hours, minutes and seconds
  - *timestamp* - Date plus time of day
  - *interval* - Period of time
  - *text*, *BLOB*, *CLOB*, *image*, *geometry*, etc.
- Operations on complex types: (typical)
  - Interval values can be added/subtracted to or from a date/time/timestamp value
  - Values of individual fields can be extracted from date/time/timestamp:

    extract (**year from** student.birth-date)

Create Table Construct

- An table 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	- name of the table
A_i	- column name
D_i	- column data type

- Example:

```
create table branch
     (branch-name   varchar(16),
     branch-city    varchar(32),
     assets         numeric(12,2))
```

Integrity Constraints in Create Table

- Integrity constraints:
  - **not null**
  - **primary key** (A_1, ..., A_n) - - *Also enforces* **not null**
  - **check** *(P),* where *P* is a predicate

- Example:

```
create table branch
        (branch-name        varchar(16),
        branch-city         varchar(32)   not null,
        assets              numeric(12,2),
        primary key (branch-name),
        check (assets >= 0))
```

Referential Integrity in SQL

- Key types:
  - **primary key** - enforces uniqueness.
  - **unique key** - also enforces uniqueness, a.k.a, *alternate* or *secondary* key.
  - **foreign key** - attributes in a foreign key and the name of the relation referenced by the foreign key.

- A foreign key references the primary key of the referenced table:

  foreign key (*account-number*) **references** *account*

- Reference columns can be explicitly specified:

  foreign key (*account-number*) **references** *account*(*account-number*)

- Foreign key references have several implications for insertions, deletions and modifications…

DDL Files

- A DDL file typically contains a collection of:
  - **create table** statements
  - **create index** statements
  - statements that create and/or specify other things:
    - Security and authority information
    - Physical storage details

- A DDL file can be coded by hand, or generated by a schema design or modeling tool.

Referential Integrity in SQL – Example

```
create database bankdb;
use bankdb;

create table customer
        (customer-name          varchar(32),
        customer-street         varchar(32),
        customer-city           varchar(16),
        primary key (customer-name));

create table branch
        (branch-name            varchar(16),
        branch-city             varchar(16),
        assets                  numeric(12,2),
        primary key (branch-name));

create table account
        (account-number         char(10),
        branch-name             varchar(16),
        balance                 numeric(9,2),
        primary key (account-number),
        foreign key (branch-name) references branch);

create table depositor
        (customer-name          varchar(32),
        account-number          char(10),
        primary key (customer-name, account-number),
        foreign key (account-number) references account,
        foreign key (customer-name) references customer);

-- Similarly for loan and borrower.
```

*Note tables must be created/loaded/deleted in “foreign key” order

Cascading Actions in SQL

- A foreign key reference can be enhanced to prevent insertion, deletion, and update errors.

```
create table account (
    . . .
    foreign key(branch-name) references branch
                                                     on delete cascade
                                                     on update cascade
  . . . )
```

- If a delete of a tuple in *branch* results in a referential-integrity constraint violation, the delete “cascades” to the *account* relation.

- Cascading updates are similar.

Drop and Alter Table Constructs

- **drop table -** deletes all information about a table.

  drop table customer

- **alter table** - used to add or delete attributes to an existing relation.

  alter table *r* **add** *A D*   *// Attribute A and domain D*

  alter table *r* **drop** *A*   *// Attribute A*

- More generally, the alter table command can be used to modify an existing table in many ways, such as adding indexes, changing permissions, storage properties, etc.

- DO NOT USE THE ALTER COMMAND ON THE PROJECT!!!

Drop and Alter Table Constructs

- Oh, and did I mention…
- DO NOT USE THE ALTER COMMAND ON THE PROJECT!!!