Lecture 8
Data Definition in SQL

Limitations of Relational Algebra

- Too technical for most commercial DBMS users
- SQL provides a high-level declarative language interface
- SQL includes some features from relational algebra
- The SQL syntax is more user-friendly

Background of SQL

- In the early 1970s, IBM **Sequel** language (the original version of SQL)
- Renamed to **Structured Query Language** (SQL)
- ANSI and ISO standard SQL:
  - SQL-86 (SQL1)
  - SQL-89
  - SQL-92 (SQL2)
  - SQL:1999 (SQL3)
  - SQL:2003 (2003 SQL3)
  - SQL:2006
  - SQL:2008
  - SQL:2011
  - SQL:2016

Data Definition Language (DDL)

- Domain Types
  - Basic Domain Types
  - Additional Domain Types
- Schema Definition
  - Create Table
    - Insert
    - Delete
  - Drop Table
  - Alter Table
- Integrity Constraints
  - Constraints on a Single Relation
  - Referential Integrity
  - Assertions

Basic Domain Types

- Character-string
  - **char**(*n*): a fixed length character string
  - **varchar**(*n*): a variable length character string
- Numeric
  - **int/integer**: an integer
  - **smallint**: a small integer
  - **numeric**(*p*,*d*): a fixed point number with user-specified precision
  - **real**: single-precision floating point numbers
  - **double precision**: double-precision floating point numbers
  - **float**(*p*): a floating point number

Additional Domain Types

- Date and time
  - **date**: valid dates (‘yyyy-mm-dd’)
  - **time**: valid time (‘hh:mm:ss’)
  - **time**(*i*): additional *i* digits specifying fractions of a second (‘hh:mm:ss:ii...i’)
  - **timestamp**: has both **date** and **time** components (‘2002-09-27 09:12:47’)
  - **interval**: a period of time specifies a relative value rather than an absolute value
    - year/month interval (‘yyyy-mm’)
    - day/time interval (‘dd hh:mm:ss’)

Create Table

- 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) **primary key**,
  branch_city **char**(30),
  assets **integer**);

Insert

- The **insert** command inserts a tuple into a relation
  - Example:

    insert into *branch*

    values (‘Perryridge ’, ‘Dallas’, **null**);
  - Attribute list can be omitted if it is the same as in **CREATE TABLE**
  - **NULL** and **DEFAULT** values can be specified

Delete

- The **delete** command deletes tuples from a relation
  - Simple form: delete all tuples from a relation
    - Example: **delete from** *branch;*
  - Other forms: allow specific tuples to be deleted (covered later)

Drop Table

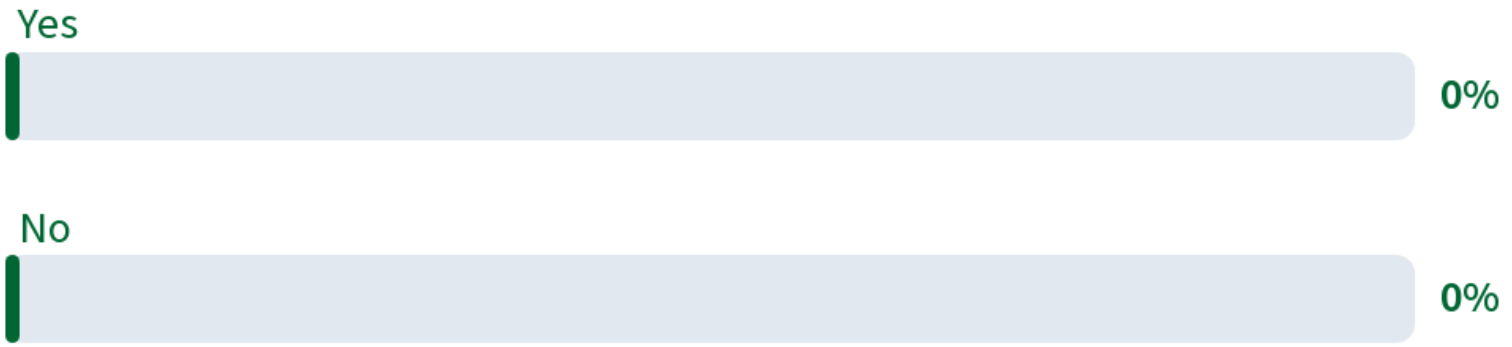
- The **drop table** command deletes all information about the dropped relation from the database
  - Example: **drop table** *branch*;
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

Does DELETE command remove the underlying relation schema?

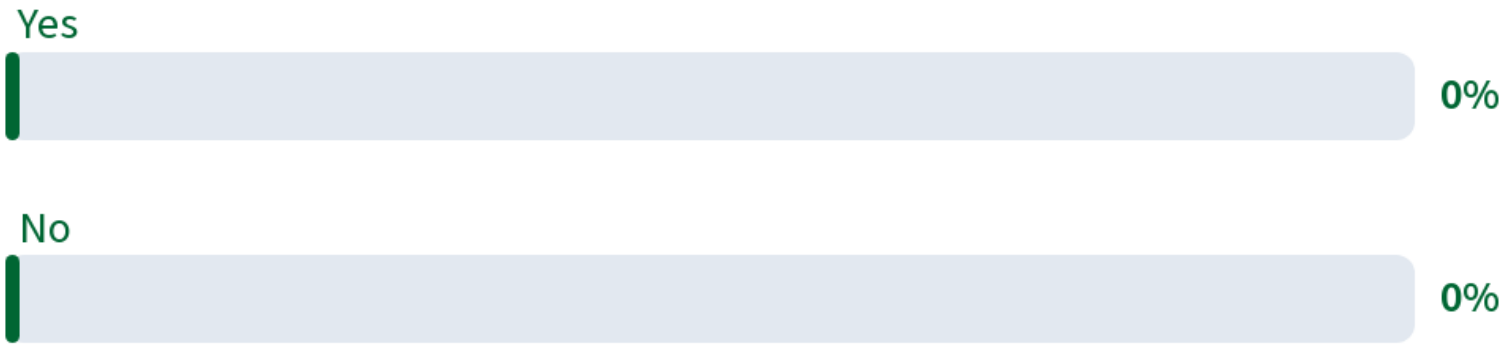
Yes

No

Does DELETE command remove the underlying relation schema?



Does DELETE command remove the underlying relation schema?



Alter Table

- The **alter table** command can be used to add attributes to an existing relation:
  - Example**:**

    alter table *branch* **add** *phone_number* **char**(10);
  - 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:
  - Example:

    alter table *branch* **drop** *branch_city*;

Constraints on a Single Relation

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

Not Null Constraint

- Not null constraint: null value is not permitted
- Example:
  - Declare *branch_name* for *branch* to be **not null**

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

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

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

Primary Key

- **Primary key** (A_1, ..., A_n)
- Example: Declare *branch_name* as the primary key for *branch*

```
create table branch
        (branch_name   char(15),
         branch_city   char(30),
         assets        integer,
         primary key   (branch_name));
```

- **Primary key** declaration implies **not null** and **unique**

Default Constraint

- Specify a default value for an attribute with the **default** clause
- Example:

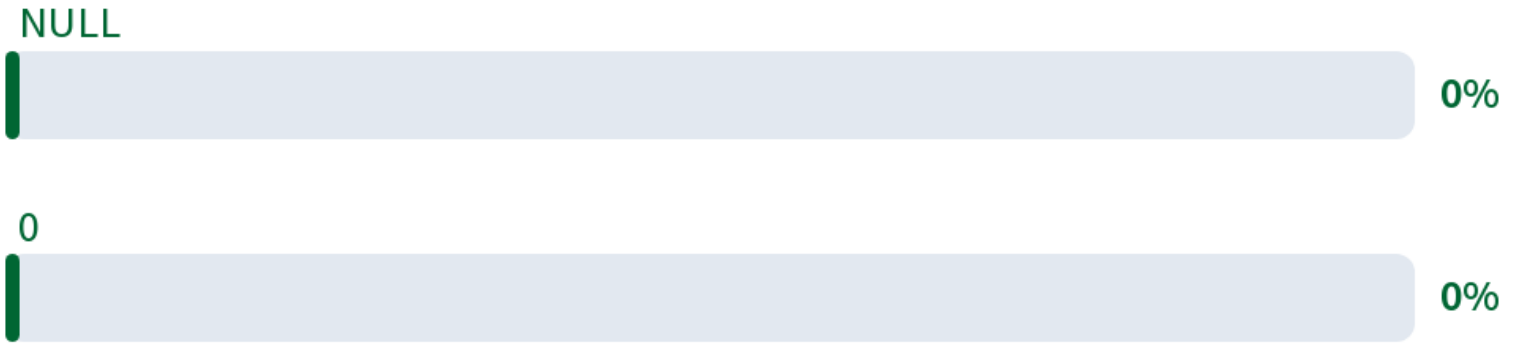
```
create table account
        (account_number    char(10),
        branch_name        char(15),
        balance            integer    default 0,
        primary key (account_number));
```

What's the default value for attributes that don't have the NOT NULL constraint and the DEFAULT clause is not specified?

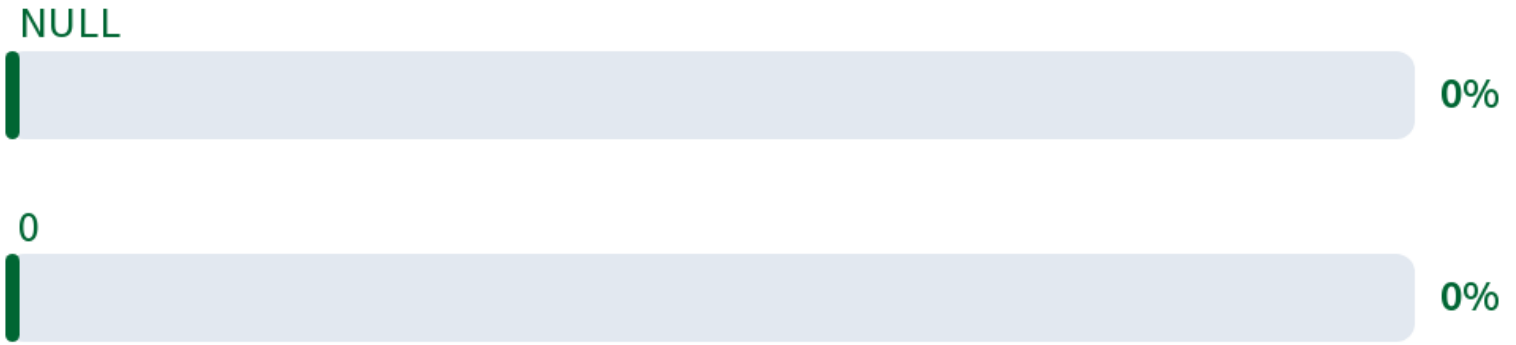
NULL

0

What's the default value for attributes that don't have the NOT NULL constraint and the DEFAULT clause is not specified?



What's the default value for attributes that don't have the NOT NULL constraint and the DEFAULT clause is not specified?



Unique Constraint

- **unique** (A_1, A_2, …, A_m)
- The unique specification states that the attributes (A_1, A_2, …, A_m) form a candidate key

The check Clause

- **check** (*P*), where *P* is a predicate
- The **check** clause can be applied to relation declarations
- Example 1: 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.)

- Example 2:

create table *student*

(*name* **char**(15) **not null**,

student_id **char**(10),

degree-level **char**(15),

primary key (*student_id*),

check (*degree_level* **in** (‘Bachelors’, ‘Masters’, ‘Doctorate’)));

The check Clause (Cont.)

- The **check** clause also permits domains to be restricted
- Example 3: 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** > = 8.00);

The check Clause (Cont.)

- Example 4:

create domain *AccountType* **varchar**(10)

constraint *type_test*

check (**value in** (‘Checking’, ‘Saving’));

Referential Integrity

- Enforce referential integrity by **foreign key** clause
- Example:

create table *branch*
(*branch_name* **char**(15),
…);

create table *account*
(…
branch_name **char**(15),
foreign key (*branch_name*) **references** *branch,*
…);

- A foreign key only references the primary key attributes or the candidate key attributes of the referenced relation

When a Foreign Key Constraint is Violated

- Default procedure: reject the action
- Other procedures: change the tuple in the referencing relation to restore the constraint.
  - Set to null: on delete/update set null
  - Set to default value: on delete/update set default
  - Propagate delete/update: on delete/update cascade
- Example:

```
create table account
        (…
        foreign key (branch_name) references branch
                                  on delete cascade
                                  on update cascade,
        …);
```

Alter Constraint

- Attributes of foreign keys are allowed to be null
- What if any attribute of a foreign key is null?
- The tuple is defined to satisfy the foreign key constraint
- Can we alter integrity constraints to an existing relation?
  - **alter table** *r* **add** *constraint*
  - **alter table** *r* **drop constraint** *constraint name*

Deferred Update

- Transactions consist of several steps
- What if intermediate steps violate referential integrity, but later steps remove the violation?
- **Deferrable**: checked immediately by default, but can be deferred when desired
  - **Initially immediate**: default option
  - **Initially deferred**: the constraint will be checked at the end of a transaction
    - Example: **set constraints** *constraint-list* **deferred**

Assertions

- An **assertion** is a predicate expressing a condition that we wish the database always to satisfy

  create assertion <assertion-name> **check** <predicate>

- Assertion testing may introduce a significant amount of overhead
- Asserting
  for all X, $P(X)$
  by using
  not exists X such that not $P(X)$

Assertions

- 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 account.branch_name = branch.branch_name )));
```