Bases de Dados

Introduction to SQL

BD 2018/2019

Departamento de Engenharia Informática
Instituto Superior Técnico

Slides baseados nos slides oficiais dos livros Database Management Systems, de Ramakrishnan e Gehrke e Database System Concepts, de Silberschatz, Korth e Sudarshan

Outline

Data Definition Language

Querying the Database

Examples

Outline

Tabular Data

- Example of tabular data in the relational model

Table: `customer`

`customer_name`	`customer_street`	`customer_city`
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

Database Used in the Examples

account(*account_number*, *branch_name*, *balance*)
branch(*branch_name*, *branch_city*, *assets*)
customer(*customer_name*, *customer_street*, *customer_city*)
loan(*loan_number*, *branch_name*, *amount*)
depositor(*customer_name*, *account_number*)
borrower(*customer_name*, *loan_number*)

Data Definition Language

Allows the specification of not only a set of relations (i.e. tables) 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.
- integer - Integer (a finite subset of the integers that is machine-dependent).
- numeric(p,d) - Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.
- And many more...

Creating Tables

- 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,
$constraint_1$, . . . , $constraint_k$
)

  - 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

create table *branch* **(**
branch_name **char(15),**
branch_city **char(30),**
assets **integer)**

Integrity Constraints

- Ensuring non-null values: `not null`
- Defining the identifier: `primary key` $(A_1, \ldots, A_n)$

```
create table branch (
    branch_name char(15),
    branch_city char(30),
    assets integer not null,
    primary key (branch_name)
)
```

Note: `primary key` also ensures that the attribute is not null

Deleting and Altering Tables

- The `drop table` command deletes all information about the dropped relation from the database.

  drop table *r*

- The `alter table` command is used to add/remove attributes to/from an existing relation:

  alter table *r* **add** *A D*

  alter table *r* **drop** *A*

  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.
  - Dropping of attributes is not supported by many databases

Simple Insertions

Add a new tuple to account:

```
insert into account
values ('A-9732', 'Perryridge', 1200)
```

or equivalently

```
insert into account(branch_name, balance, account_number)
values ('Perryridge', 1200, 'A-9732')
```

Outline

Basic Query Structure

- A typical SQL query has the form:

  select $A_1, A2, \ldots, A_n$
  from $r_1, r_2, \ldots, r_m$
  where P

  - A_i represents an attribute
  - r_i represents a relation
  - P is a predicate.

- The result of an SQL query is a relation

The `select` Clause

- The `select` clause list the attributes desired in the result of a query
- Example: find the names of all branches in the loan relation:

```
select branch_name
from loan
```

account(*account_number*, *branch_name*, *balance*)
branch(*branch_name*, *branch_city*, *assets*)
customer(*customer_name*, *customer_street*, *customer_city*)
loan(*loan_number*, *branch_name*, *amount*)
depositor(*customer_name*, *account_number*)
borrower(*customer_name*, *loan_number*)

Note: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)

The `select` Clause (cont.)

- SQL allows duplicates in relations as well as in query results
- To force the elimination of duplicates, insert the keyword `distinct` after select
- Example: find the names of all branches in the loan relations, and remove duplicates

```
select distinct branch_name
from loan
```

account(*account_number*, *branch_name*, *balance*)
branch(*branch_name*, *branch_city*, *assets*)
customer(*customer_name*, *customer_street*, *customer_city*)
loan(*loan_number*, *branch_name*, *amount*)
depositor(*customer_name*, *account_number*)
borrower(*customer_name*, *loan_number*)

The `select` Clause (cont.)

- An asterisk in the select clause denotes "all attributes"

```
select *
from loan
```

- The select clause can contain arithmetic expressions involving the operation, $+$, $-$, $*$, and $/$, and operating on constants or attributes of tuples

```
select loan_number, amount * 100
from loan
```

would return a relation where the value of the attribute amount is multiplied by 100.

The `where` Clause

- The `where` clause specifies conditions that the result must satisfy
- Example: to find all loan number for loans made at the Perryridge branch with loan amounts greater than $1200

```
select loan_number
from loan
where branch_name = 'Perryridge' and amount > 1200
```

- Results can be combined using `and`, `or`, and `not`
- Comparisons can be applied to results of arithmetic expressions

account(*account_number*, *branch_name*, *balance*)
branch(*branch_name*, *branch_city*, *assets*)
customer(*customer_name*, *customer_street*, *customer_city*)
loan(*loan_number*, *branch_name*, *amount*)
depositor(*customer_name*, *account_number*)
borrower(*customer_name*, *loan_number*)

The `where` Clause (cont.)

- SQL includes a `between` comparison operator
- Example: Find the loan number of those loans with loan amounts between \$90 000 and \$100 000 (that is, $\geq$ \$90 000 and $\leq$ \$100 000)

```
select loan_number
from loan
where amount between 90000 and 100000
```

account(*account_number*, *branch_name*, *balance*)
branch(*branch_name*, *branch_city*, *assets*)
customer(*customer_name*, *customer_street*, *customer_city*)
loan(*loan_number*, *branch_name*, *amount*)
depositor(*customer_name*, *account_number*)
borrower(*customer_name*, *loan_number*)

The `from` Clause

- The `from` clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation
- Example: find the Cartesian product $borrower \times loan$

```
select *
from borrower, loan
```

The `from` Clause

- The `from` clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation
- Example: find the Cartesian product $borrower \times loan$

```
select *
from borrower, loan
```

- Example: find the name, loan number and loan amount of all customers having a loan at the Perryridge branch

$$loan(loan\_number, branch\_name, amount)$$
$$borrower(customer\_name, loan\_number)$$

The `from` Clause

- The `from` clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation
- Example: find the Cartesian product $borrower \times loan$

```
select *
from borrower, loan
```

- Example: find the name, loan number and loan amount of all customers having a loan at the Perryridge branch

```
select customer_name, loan.loan_number, amount
from loan, borrower
where loan.loan_number = borrower.loan_number
    and branch_name = 'Perryridge'
```

Renaming

- SQL allows renaming relations and attributes using the as clause:

  old-name* as *new-name

- Example: find the name, loan number and loan amount of all customers; rename the column *loan_number* as *loan_id*

```
select customer_name, borrower.loan_number as loan_id,
    amount
from borrower, loan
where borrower.loan_number = loan.loan_number
```

Tuple Variables

- Tuple variables are defined in the `from` clause via the (optional) use of the as clause.
- Example: find the customer names and their loan numbers for all customers having a loan

```
select B.customer_name, B.loan_number, L.amount
from borrower as B, loan as L
where B.loan_number = L.loan_number
```

Tuple Variables

- Tuple variables are defined in the `from` clause via the (optional) use of the `as` clause.
- Example: find the customer names and their loan numbers for all customers having a loan

```
select B.customer_name, B.loan_number, L.amount
from borrower as B, loan as L
where B.loan_number = L.loan_number
```

- Example: find the names of all branches that have greater assets than some branch located in Brooklyn

$$branch(branch\_name, branch\_city, assets)$$

Tuple Variables

- Tuple variables are defined in the `from` clause via the (optional) use of the `as` clause.
- Example: find the customer names and their loan numbers for all customers having a loan

```
select B.customer_name, B.loan_number, L.amount
from borrower as B, loan as L
where B.loan_number = L.loan_number
```

- Example: find the names of all branches that have greater assets than some branch located in Brooklyn

```
select distinct T.branch_name
from branch as T, branch as S
where T.assets > S.assets and S.branch_city = 'Brooklyn'
```

String Operations

- The operator `like` uses patterns that are described using two special characters:
  - percent (%) - matches any substring.
  - underscore (_) - matches any character.
- Example: find the names of all customers whose street includes the substring "Main"

```
select customer_name
from customer
where customer_street like '%Main%'
```

- SQL supports many other string operations

Ordering the Display of Tuples

- List in alphabetic order the names of all customers having a loan in Perryridge branch

```
select distinct customer_name
from borrower B, loan L
where B.loan_number = L.loan_number
    and branch_name = 'Perryridge'
order by customer_name
```

- We may specify `desc` for descending order or `asc` for ascending order (the default)

```
order by customer_name desc
```

Outline

Example Queries

Get data from all customers

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get data from all customers

```
select * from customer
```

Example Queries

Get the name and city of all customers with a loan

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the name and city of all customers with a loan

```
select distinct c.customer_name, customer_city
from borrower b, customer c
where b.customer_name = c.customer_name
```

Example Queries

Get the name and city of all customers with a loan at the Perryridge branch

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the name and city of all customers with a loan at the Perryridge branch

```
select distinct c.customer_name, customer_city
from borrower b, customer c, loan l
where b.customer_name = c.customer_name
    and b.loan_number = l.loan_number
    and branch_name = 'Perryridge'
```

Example Queries

Get the number of all accounts with a balance between \$700 and \$900

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the number of all accounts with a balance between $700 and $900

```
select account_number
from account
where balance between 700 and 900
```

Example Queries

Get the name of all customers whose street name ends in 'Hill'

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the name of all customers whose street name ends in 'Hill'

```
select customer_name
from customer
where customer_street like '%Hill'
```

Example Queries

Get the name of all customers with both an account and a loan at the Perryridge branch

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the name of all customers with both an account and a loan at the Perryridge branch

```
select distinct b.customer_name
from borrower b, loan l, depositor d, account a
where b.loan_number = l.loan_number
    and b.customer_name = d.customer_name
    and d.account_number = a.account_number
    and a.branch_name = 'Perryridge'
    and l.branch_name = 'Perryridge'
```

Example Queries

Get the name of all clients with a loan at the Perryridge branch, sorted by customer name

$$account(account\_number, branch\_name, balance)$$
$$branch(branch\_name, branch\_city, assets)$$
$$customer(customer\_name, customer\_street, customer\_city)$$
$$loan(loan\_number, branch\_name, amount)$$
$$depositor(customer\_name, account\_number)$$
$$borrower(customer\_name, loan\_number)$$

Example Queries

Get the name of all clients with a loan at the Perryridge branch, sorted by customer name

```
select distinct customer_name
from borrower b, loan l
where b.loan_number = l.loan_number
    and l.branch_name = 'Perryridge'
order by b.customer_name
```

Questions?