

Principles of Database Systems With Internet and Java Applications

Today's Topic

Chapter 7: SQL, the Structured Query Language

Instructor's name and information goes here
Please see the notes pages for more information.

1

Chapter 7: SQL

- Standard Query Language
 - ANSI and ISO standard
 - SQL2 or SQL-92 is current standard
- SQL is a data manipulation language (DML) and a data definition language (DDL) and a programming language
- We can use SQL for
 - Logical database specification (database schema definitions)
 - Physical database specifications (indexes, etc.)
 - Querying database contents
 - Modifying database contents

2



Relational Operations in SQL

- **Select statement**
 - **select** <attribute names> **from** <tables>
where <condition>
- Projection in SQL using **select** clause
 - **Select** title **from** Movies
- Selection in SQL using **where** clause
 - **select** * **from** Customer **where** lastName = 'Doe'
 - **select distinct** lastName, firstName **from** Customer
 - no duplicates with **distinct**

3



Products and Joins in SQL

- Cartesian product in SQL using **from** clause
 - **Select** * **from** Employee, Timecard
- Join using **from** and **where** clauses
 - **Select** * **from** Employee, Timecard **where** Employee.ssn = Timecard.ssn
- Join using **join** and **on** (non-standard)
 - **Select** * **from** Employee **join** TimeCard **on** Employee.ssn = TimeCard.ssn

4



Nested Queries

- Nested select query
 - **Select** videoid, dateAcquired
from Videotape
where videoid in (
 select videoid **from** Rental
 where dateRented='1/1/99')
- compare with
 - **Select** v.videoid, dateAcquired
from Videotape v, Rental r
where v.videoid = r.videoid and
 dateRented='1/1/99')
- Same result?

5



Exists and Unique Queries

- find employees who have no time cards
 - **Select** firstName, lastName **from** Employee e
where not exists
 (**select** * **from** TimeCard t **where** e.ssn=t.ssn)
- Find managers who have exactly one time card
 - **select** firstName, lastName **from** Employee e
where unique
 (**select** * **from** TimeCard t **where** t.ssn=e.ssn)
and exists
 (**select** * **from** Store **where** ssn=manager)

6



Sets and nulls in where clauses

- Names of all stores that do not have managers
 - **select** name **from** Store
where manager is **null**
- All employees who work on project 1, 2 or 3
 - **select distinct** fname, lname **from** employee
where pno **in** (1,2,3)

7



Aggregate functions

- How many employees work at Store 3?
 - **select** count (*) **from** WorksAt **where** storeId=3
- What is the average hourly rate of all hourly employees
 - **select** average(hourlyRate) **from** HourlyEmployees
- How many different salaries are there?
 - **select** count (**distinct** salary) **from** SalariedEmployee
- What is the average salary in each store? or How many employees work at each store?
 - must apply average to the salaries of each group of store employees,
or count to each group of store employees!

8



Select Using Group by and Having

- **Group by** forms groups of rows with the same column values
- What is the average hourly rate by store?
 - **select** storeId, avg(hourlyRate)
from HourlyEmployee e, WorksAt w
where e.ssn = w.ssn
group by storeId
- How many employees work at each store?
 - **select** storeId, name, count (*)
from Store s, WorksAt w
where s.storeId = w.storeId
group by storeId, name
- **Having** filters the groups
 - **having** count (*)>2

9



Substrings, arithmetic and order

- Find a movie with 'Lion' in the title
 - **select** title **from** Movie **where** title **like** '%Lion%'
- List the monthly salaries of salaried employees who work in in store 3
 - **select** salary/12 **from** Employees e, WorksAt w
where e.ssn=w.ssn and storeId=3
- Give the list of employees in store 3, ordered by salary
 - **select** firstName, lastName
from Employees e, WorksAt w
where e.ssn=w.ssn and storeId=3

10



Modifying Content with SQL

- Insert queries
 - **insert into** Customer **values** (555, 'Yu', 'Jia', '540 Magnolia Hall', 'Tallahassee', 'FL', '32306')
 - **insert into** Customer (firstName, lastName, accountId) **values** ('Jia', 'Yu', 555)
- Update queries
 - **update** TimeCard **set** paid = true **where** paid = false
 - **update** HourlyEmployee **set** hourlyRate = hourlyRate * 1.1 **where** ssn = '145-09-0967'
- Samples in Access

11



Creating Pay Statements with SQL

- Find the number of hours worked for each employee entry
 - **select** TimeCard.ssn, **sum**((endTime-startTime)*24) **as** hoursWorked **from** TimeCard **where** paid=false **group by** ssn
- Create the Pay Statement entries for each Employee
 - **select** ssn, hourlyRate, hoursWorked, hoursWorked * hourlyRate **as** amountPaid, today **from** ...
- Insert into the PayStatement table
 - **Insert into** PayStatement **select** ...
- Look at the Access example in BigHit.mdb

12



Defining queries for the PayStatement

- A view is a named query
- **create view** EmployeeHours **as**
 - **select** TimeCard.ssn, **sum**((endTime-startTime)*24) **as** hoursWorked **from** TimeCard **where** paid=false **group by** ssn
- **create view** EmployeePay **as**
 - **select** ssn, hourlyRate, hoursWorked, hoursWorked * hourlyRate **as** amountPaid, today **from** EmployeeHours h, HourlyEmployee e **where** h.ssn=e.ssn
- **insert into** PayStatement **select * from** EmployeePay

13



Marking TimeCards as paid

- update TimeCard set paid = true
- update TimeCard set paid=true where paid=false
- update TimeCard set paid=true where ssn in (select ssn from EmployeePay)
- What happens if time cards added while pay statements are being created?

14



Delete Statements

- Delete all time cards for non-hourly employees
 - **delete from** Timecard **where not exists**
(select * **from** HourlyEmployee
where TimeCard.ssn = HourlyEmployee.ssn)
- More examples in BigHit Video Access database

15



Create Table Statement

- **create table** Customer (
 accountId **int**,
 lastName **varchar**(32),
 firstName **varchar**(32),
 street **varchar**(100),
 city **varchar**(32),
 state **char**(2),
 zipcode **varchar**(9)
)
- Note that SQL has specific types

16

Data types in SQL

Numeric types	integer	integer, int, smallint, long
	floating point	float, real, double precision
	formatted	decimal (i, j), dec (i, j)
Character-string types	fixed length	char (n), character (n)
	varying length	varchar (n), char varying (n), character varying (n)
Bit-string types	fixed length	bit (n)
	varying length	bit varying (n)
Date and time types		date, time, datetime, timestamp, time with time zone, interval
Large types	character	long varchar (n), clob, text
	binary	blob

17

Key Constraints in SQL

- Key declarations are part of *create table*
 - **create table** Store (
storeId **int primary key**,
 - **create table** Movie (
movieId **varchar(10) primary key**,
 - **create table** Rental (
accountId **int**,
videoId **varchar(10)**,
primary key (accountId, videoId)

18



Referential Integrity Constraints

- A relationship is implemented by attributes that reference the primary key of the related table
 - Enforcing referential integrity requires guaranteeing that there is a referenced object
 - An attempt to modify the relationship (insert, update or delete) is potential violation
- Declare foreign key constraints
 - **create table** Store (
 manager **int references** Employee
 - **create table** Rental (
 foreign key (accountId) **references**
 Customer(accountId)

19



Maintaining Referential Integrity

- What happens when an update violates referential integrity
 - update foreign key attribute
 - change catalog id of a video
 - insert new object
 - add a new video
 - delete related object
 - delete catalog entry
 - update primary key attribute
 - change catalog id of a video title
- Alternatives
 - propagate changes
 - set to null

20



Constraints on Values of Attributes

- Not null constraints
 - create table PreviousRental (
 accountId int not null references Customer,
 videoid int not null references Videotape,
 dateRented datetime not null,
 dateReturned datetime,
 cost real,
 primary key (accountId, videoid, dateRented))
- Check constraints
 - check (checkOut < dueDate)
 - check (answer in ('T','F'))
 - check (questionId in (select questionId from questions where quizId=...))

21



Strategies for Enforcing Constraints

- Enforce always
 - Never allow a violation of constraint
 - Suppose 2 rentals are recorded wrong
 - change the customerId of 2 records
 - some violation will result
- Enforce at end of transaction
 - Allow violations during updates, but check and enforce at the end of the process
- Leads us to consider
 - Chapter 14 Transactions in SQL
 - Allow cancellation of updates
 - Support concurrent access

22