

Chapter 3: Introduction to SQL

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan See www.db-book.com for conditions on re-use



SQL Parts

- DML -- provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- integrity the DDL includes commands for specifying integrity
- View definition -- The DDL includes commands for defining views.
- · Transaction control -includes commands for specifying the beginning and ending of transactions.
- Embedded SQL and dynamic SQL -- define how SQL statements can be embedded within general-purpose programming languages.
- Authorization includes commands for specifying access rights to



Create Table Construct

An SQL relation is defined using the create table command:

(A₁ D₁, A₂ D₂, ..., A_n D_n, (integrity-constraint₁),

(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 instructor (ID char(5) name varchar(20), dept_name varchar(20), salary numeric(8,2 numeric(8,2))



Outline

- Overview of The SQL Query Language
- SQL Data Definition
- Basic Query Structure of SQL Queries
- Additional Basic Operations
- Set Operations
- Null Values
- Aggregate Functions Nested Subqueries
- Modification of the Database

Data Definition Language

The SQL data-definition language (DDL) allows the specification of information about relations, including:

- . The type of values associated with each attribute.
- The 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.



Integrity Constraints in Create Table

- Types of integrity constraints
 - primary key $(A_1, ..., A_n)$
 - foreign key $(A_m, ..., A_n)$ references r
- not null
- SQL prevents any update to the database that violates an integrity constraint.
- Example:

create table instructor (char(5). name varchar(20) not null, dept_name varchar(20), salary numeric(8,2), primary key (ID),

foreign key (dept_name) references department);



History

- IBM Sequel language developed as part of System R project at the IBM San Jose Research Laboratory
- Renamed Structured Query Language (SQL)
- ANSI and ISO standard SQL:
- SQL-86
- SOL-89
- SQL-92
- SQL:1999 (language name became Y2K compliant!)
- SQL:2003
- · Commercial systems offer most, if not all, SQL-92 features, plus varying feature sets from later standards and special proprietary features
 - Not all examples here may work on your particular system.



Domain Types in SQL

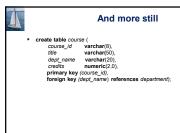
- char(n). Fixed length character string, with user-specified length n.
- varchar(n). Variable length character strings, with user-specified maximum length n.
- int. Integer (a finite subset of the integers that is machine-dependent).
- smallint. Small integer (a machine-dependent subset of the integer
- numeric(p,d). Fixed point number, with user-specified precision of p digits, with d digits to the right of decimal point. (ex., numeric(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- 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
- MySQL domain type
 - https://dev.mysql.com/doc/refman/8.0/en/data-types.html

And a Few More Relation Definitions

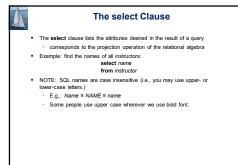
create table student (
ID varchar(5),
name varchar(20), not null,
dept_name varchar(20),
tot_cred numeric(30),
primary key (60),
foreign key (dept_name) references department);

create table takes (

course_id varchar(8), sec_id varchar(8), semester varchar(2). primary key (ID, course_id, sec_id, semester, year) , foreign key (ID) references student, foreign key (course_id, sec_id, semester, year) references section);



abase System Concepts - 7n Edition 3.10 @Silberschatz, Korth and Sudarsh



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

The query:

select ID, name, salary/12 from instructor

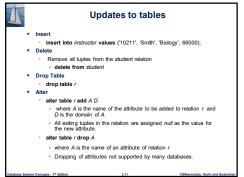
would return a relation that is the same as the instructor relation, except that the value of the attribute salary is divided by 12.

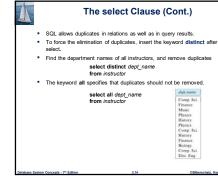
Can rename "salary/12 vising the as clause:

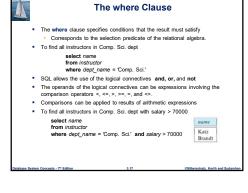
select ID, name, salary/12 as monthly_salary

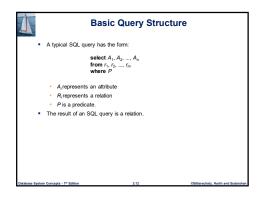
Select ID, name, salary/12 as monthly_salary

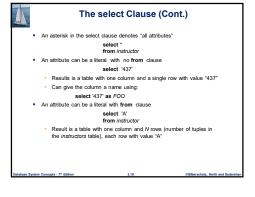
The select Clause (Cont.)

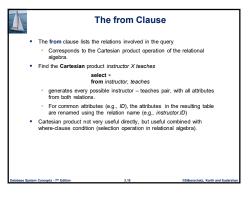














Examples

- instructor(ID.name, dept_name, salary) teaches(ID, course_id, sec_id, semester, year)
- Find the names of all instructors who have taught some course and the course id
 - select name, course_id from instructor , teaches where instructor.ID = teaches.ID

Find the names of all instructors in the Art department who have taught some course and the course_id

select name, course_id from instructor, teaches where instructor.ID = teaches.ID and instructor. dept_name = 'Art'

Srinivasan	CS-101	
Srinivasan	CS-315	
Srinivasan	CS-347	
Wu	FIN-201	
Mozart	MU-199	
Einstein	PHY-10	
El Said	HIS-351	
Katz	CS-101	
Katz	CS-319	
Crick	BIO-101	
Crick	BIO-301	
Brandt	CS-190	
Brandt	CS-190	
Brandt	CS-319	
Kim	EE-181	



String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator like uses patterns that are described using two special characters:
- percent (%). The % character matches any substring.
- underscore (_). The _ character matches any character.
- . Find the names of all instructors whose name includes the substring "dar". select name

from instructor where name like '%dar%'

Match the string "100%"

like '100 \%' escape "\'

in that above we use backslash (\) as the escape character.



Where Clause Predicates

- SQL includes a between comparison operator
- Example: Find the names of all instructors with salary between \$90,000 and \$100,000 (that is, ≥ \$90,000 and ≤ \$100,000)
 - select name

where salary between 90000 and 100000

- Tuple comparison
 - select name, course id from instructor, teaches

 - where (instructor.ID. dept_name) = (teaches.ID. 'Biology');
 - equivalent
 - select name, course_id

from instructor, teaches
where instructor.ID = teaches.ID

and instructor.dept_name='Biology';



The Rename Operation

- The SQL allows renaming relations and attributes using the as clause: old-name as new-name
- Find the names of all instructors who have a higher salary than some instructor in 'Comp. Sci'.
 - select distinct T name from instructor as T, instructor as S where T.salary > S.salary and S.dept_name = 'Comp. Sci.'
- Keyword as is optional and may be omitted instructor as T ≡ instructor T

String Operations (Cont.)

- Patterns are case sensitive
- Pattern matching examples:
- 'Intro%' matches any string beginning with "Intro".
- '%Comp%' matches any string containing "Comp" as a substring.
- · '___' matches any string of exactly three characters.
- · '___%' matches any string of at least three characters.
- . SQL supports a variety of string operations such as
- concatenation (using "||"), (CONCAT('abs','def')='absdef, in
- converting from upper to lower case (and vice versa)
- · LOWER(), UPPER()
- finding string length, extracting substrings, etc.
- https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

Set Operations

- Find courses that ran in Fall 2017 or in Spring 2018
 - (select course_id from section where sem = 'Fall' and year = 2017)
- (select course id from section where sem = 'Spring' and year = 2018)
- Find courses that ran in Fall 2017 and in Spring 2018 (MySQL not

(select course_id from section where sem = 'Fall' and year = 2017)

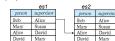
(select course_id from section where sem = 'Spring' and year = 2018)

- . Find courses that ran in Fall 2017 but not in Spring 2018 (MySQL not
- (select course_id from section where sem = 'Fall' and year = 2017)

(select course_id from section where sem = 'Spring' and year = 2018)

Self Join Example

Relation emp-super



- Find the supervisor of "Roh"
- Find the supervisor of the supervisor of "Bob"
- . Can you find ALL the supervisors (direct and indirect) of "Bob"?
- Select es2.supervisor from emp-super es1, emp-super es2 where es1.supervisor=es2.person and es1.person='Bob'



Ordering the Display of Tuples

- . List in alphabetic order the names of all instructors
 - select distinct name from instructor order by name
- . We may specify desc for descending order or asc for ascending order, for each attribute; ascending order is the default.
 - Example: order by name desc
- Can sort on multiple attributes
 - Example: order by dept_name, name

Set Operations (Cont.)

- Set operations union, intersect, and except
- Each of the above operations automatically eliminates duplicates
- To retain all duplicates use the
- union all.
- except all



Null Values

- . It is possible for tuples to have a null value, denoted by null, for some of
- null signifies an unknown value or that a value does not exist.
- . The result of any arithmetic expression involving null is null
- Example: 5 + null returns null
- The predicate is null can be used to check for null values.
- · Example: Find all instructors whose salary is null.

select name from instructor where salary is null

. The predicate is not null succeeds if the value on which it is applied is



Aggregate Functions Examples

- . Find the average salary of instructors in the Computer Science department select avg (salary)
 - from instructor where dept_name= 'Comp. Sci.';
- Find the total number of instructors who teach a course in the Spring 2018
 - select count (distinct ID) from teaches
 - where semester = 'Spring' and year = 2018;
- . Find the number of tuples in the course relation
 - select count (*) from course;



Aggregate Functions - Having Clause

· Find the names and average salaries of all departments whose average salary is greater than 42000

> select dept_name, avg (salary) as avg_salary from instructor group by dept_name having avg (salary) > 42000;

Note: predicates in the having clause are applied after the formation of groups whereas predicates in the where clause are applied before forming

select dept_name, avg (salary) as avg_salary from instructor where name <> 'Eric' group by dept_name having avg(salary) > 42000;



Null Values (Cont.)

- SQL treats as unknown the result of any comparison involving a null value (other than predicates is null and is not null).
- Example: 5 < null or null <> null or null = null
- . The predicate in a where clause can involve Boolean operations (and. or, not); thus the definitions of the Boolean operations need to be extended to deal with the value unknown.
 - and : (true and unknown) = unknown, (false and unknown) = false, (unknown and unknown) = unknown
 - or: (unknown or true) = true, (unknown or unknown) = unknown
- Result of where clause predicate is treated as false if it evaluates to

Aggregate Functions - Group By

- . Find the average salary of instructors in each department
 - select dept_name, avg (salary) as avg_salary from instructor group by dept_name;

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name avg_salary Biology Comp. Sci. Elec. Eng. 72000 77333 80000 Finance History 85000 61000 Music 40000



Nested Subqueries

- SQL provides a mechanism for the nesting of subqueries. A **subquery** is a **select-from-where** expression that is nested within another query.

. The nesting can be done in the following SQL query

 $\mathbf{select}\ A_1,\,A_2,\,...,\,A_n$ from r₁, r₂, ..., r_m where P

as follows:

- From clause: r_i can be replaced by any valid subquery
- Where clause: P can be replaced with an expression of the form: B < operation > (subquery)

B is an attribute and <operation> to be defined later.

Select clause:

 A_i can be replaced be a subquery that generates a single value.

Aggregate Functions

These functions operate on the multiset of values of a column of a relation, and return a value ava: average value

min: minimum value max: maximum value sum: sum of values count: number of values



Aggregation (Cont.)

- Attributes in select clause outside of aggregate functions must appear in
 - /* erroneous query */ select dept_name, ID, avg (salary) from instructor group by dept_name;