

CompSci 516 Data Intensive Computing Systems

Lecture 2 SQL

Instructor: Sudeepa Roy

Duke CS, Fall 2017

CompSci 516: Database Systems

1

Announcements

- If you are enrolled to the class, but have not received the email from Piazza, please send me an email
 - All notifications are being sent through piazza
- HW1 posted on sakai
 - Due on 09/21 (Thurs), 11:55 pm
 - Start early – no late days
- Mark your calendar for midterm days: 10/11, 11/29
 - Midterms won't be held later if you miss an exam
 - Missed exam policy: grade will be scaled after 20% deduction from the other exam (deduction waived with written letters from Duke authority)
 - Both exams missed -> incomplete grade

Duke CS, Fall 2017

CompSci 516: Database Systems

2

Recap: Lecture 1

- Why use a DBMS
- Structured data model: Relational data model
 - table, schema, instance, tuples, attributes
 - bag and set semantic
- Logical and physical data independence

Duke CS, Fall 2017

CompSci 516: Database Systems

3

Today's topic

- Overview of XML
- SQL in a nutshell
 - Reading material: [RG] Chapters 3 and 5
 - Additional reading for practice: [GUW] Chapter 6
- Try SQL from today's lecture on a toy dataset or DBLP dataset on PostGres!

Acknowledgement:

The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.

Duke CS, Fall 2017

CompSci 516: Database Systems

4

XML: an overview

Duke CS, Fall 2017

CompSci 516: Database Systems

5

Semi-structured Data and XML

- XML: Extensible Markup Language
- Will not be covered in detail in class, but many datasets available to download are in this form
 - You will download the DBLP dataset in XML format and transform into relational form (in HW1)
- Data does not have a fixed schema
 - "Attributes" are part of the data
 - The data is "self-describing"
 - Tree-structured

Duke CS, Fall 2017

CompSci 516: Database Systems

6

XML: Example

Attributes

Elements

```

<article mdate="2011-01-11" key="journals/acta/Saxena96">
  <author>Sanjeev Saxena</author>
  <title>Parallel Integer Sorting and Simulation Amongst CRCW
    Models.</title>
  <pages>607-619</pages>
  <year>1996</year>
  <volume>33</volume>
  <journal>Acta Inf.</journal>
  <number>7</number>
  <url>db/journals/acta/acta33.html#Saxena96</url>
  <ee>http://dx.doi.org/10.1007/BF03036466</ee>
</article>
    
```

Duke CS, Fall 2017 CompSci 516: Database Systems 7

Attribute vs. Elements

- Elements can be repeated and nested
- Attributes are unique and atomic

Duke CS, Fall 2017 CompSci 516: Database Systems 8

Why XML?

- Advantages/Disadvantages?

Which one is easier?

- XML (semi-structured) to relational (structured)
- or
- relational (structured) to XML (semi-structured)?

Duke CS, Fall 2017 CompSci 516: Database Systems 9

XML to Relational Model

- **Problem 1: Repeated attributes**

```

<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill
</book>
    
```

What is a good relational schema?

Duke CS, Fall 2017 CompSci 516: Database Systems 10

XML to Relational Model

- **Problem 1: Repeated attributes**

```

<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill</publisher>
</book>
    
```

Title	Publisher	Author1	Author2

What if the paper has a single author?

Duke CS, Fall 2017 CompSci 516: Database Systems 11

XML to Relational Model

- **Problem 1: Repeated attributes**

```

<book>
  <author>Garcia-Molina</author>
  <author>Ullman</author>
  <author>Widom</author>
  <title>Database Systems – The Complete Book</title>
  <publisher>Prentice Hall</publisher>
</book>
    
```

Duke CS, Fall 2017 CompSci 516: Database Systems 12

XML to Relational Model

Duke CS, Fall 2017

CompSci 516: Database Systems

13

XML to Relational Model

- Problem 2: Missing attributes

```
<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill
  <edition>Third</edition>
</book>
<book>
  <author>Garcia-Molina</author>
  <author>Ullman</author>
  <author>Widom</author>
  <title>Database Systems – The Complete
Book</title>
  <publisher>Prentice Hall</publisher>
</book>
```

Duke CS, Fall 2017

CompSci 516: Database Systems

14

Summary: Data Models

- Relational data model is the most standard for database managements
 - and is the main focus of this course
- Semi-structured model/XML is also used in practice – you will use them in hw assignments
- Unstructured data (text/photo/video) is unavoidable, but won't be covered in this class

Duke CS, Fall 2017

CompSci 516: Database Systems

15

SQL

Duke CS, Fall 2017

CompSci 516: Database Systems

16

Relational Query Languages

- A major strength of the relational model: supports simple, powerful querying of data.
- Queries can be written intuitively, and the DBMS is responsible for an efficient evaluation
 - The key: precise semantics for relational queries.
 - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

Duke CS, Fall 2017

CompSci 516: Database Systems

17

The SQL Query Language

- Developed by IBM (systemR) in the 1970s
- Need for a standard since it is used by many vendors
- Standards:
 - SQL-86
 - SQL-89 (minor revision)
 - SQL-92 (major revision)
 - SQL-99 (major extensions, current standard)

Duke CS, Fall 2017

CompSci 516: Database Systems

18

Purposes of SQL

- Data Manipulation Language (DML)
 - Querying: SELECT-FROM-WHERE
 - Modifying: INSERT/DELETE/UPDATE
- Data Definition Language (DDL)
 - CREATE/ALTER/DROP

Duke CS, Fall 2017

CompSci 516: Database Systems

19

The SQL Query Language

- To find all 18 year old students, we can write:

```

SELECT *
FROM Students S
WHERE S.age=18
    
```

all attributes

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```

SELECT S.name, S.login
    
```

Duke CS, Fall 2017

CompSci 516: Database Systems

20

Querying Multiple Relations

- What does the following query compute?

```

SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
    
```

Given the following instances of Enrolled and Students:

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get: ??

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Querying Multiple Relations

- What does the following query compute?

```

SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
    
```

Given the following instances of Enrolled and Students:

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Creating Relations in SQL

- Creates the "Students" relation
 - the type (domain) of each field is specified
 - enforced by the DBMS whenever tuples are added or modified
- As another example, the "Enrolled" table holds information about courses that students take

```

CREATE TABLE Students
(sid CHAR(20),
 name CHAR(20),
 login CHAR(10),
 age INTEGER,
 gpa REAL)
    
```

```

CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(20),
 grade CHAR(2))
    
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Students

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Enrolled

Duke CS, Fall 2017

CompSci 516: Database Systems

23

Destroying and Altering Relations

```

DROP TABLE Students
    
```

- Destroys the relation Students
 - The schema information *and* the tuples are deleted.

```

ALTER TABLE Students
ADD COLUMN firstYear: integer
    
```

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

Duke CS, Fall 2017

CompSci 516: Database Systems

24

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

Duke CS, Fall 2017

CompSci 516: Database Systems

25

Integrity Constraints (ICs)

- **IC: condition that must be true for any instance of the database**
 - e.g., domain constraints
 - ICs are specified when schema is defined
 - ICs are checked when relations are modified
- A **legal** instance of a relation is one that satisfies all specified ICs
 - DBMS will not allow illegal instances
- If the DBMS checks ICs, stored data is more faithful to real-world meaning
 - Avoids data entry errors, too!

Duke CS, Fall 2017

CompSci 516: Database Systems

26

Keys in a Database

- Key / Candidate Key
- Primary Key
- Super Key
- Foreign Key
- Primary key attributes are underlined in a schema
 - Person(pid, address, name)
 - Person2(address, name, age, job)

Duke CS, Fall 2017

CompSci 516: Database Systems

27

Primary Key Constraints

- A set of fields is a **key** for a relation if :
 1. No two distinct tuples can have same values in all key fields, and
 2. This is not true for any subset of the key
- Part 2 false? A **superkey**
- If there are > 1 keys for a relation, one of the keys is chosen (by DBA = DB admin) to be the **primary key**
 - E.g., sid is a key for Students
 - The set {sid, gpa} is a superkey.
- Is there any possible benefit to refer to a tuple using **primary key (than any key)?**

Duke CS, Fall 2017

CompSci 516: Database Systems

28

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.
- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY ???)
```

Duke CS, Fall 2017

CompSci 516: Database Systems

29

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

```
CREATE TABLE Enrolled
(sid CHAR(20)
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY ??)
```

- “For a given student and course, there is a single grade.”

• vs.

- “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY ???,
UNIQUE ???)
```

Duke CS, Fall 2017

CompSci 516: Database Systems

30

Foreign Keys, Referential Integrity

- **Foreign key** : Set of fields in one relation that is used to 'refer' to a tuple in another relation
 - Must correspond to primary key of the second relation
 - Like a 'logical pointer'
- E.g. **sid** is a foreign key referring to **Students**:
 - Enrolled(**sid**: string, cid: string, grade: string)
 - If all foreign key constraints are enforced, **referential integrity** is achieved
 - i.e., no dangling references

Duke CS, Fall 2017

CompSci 516: Database Systems

31

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses

```
CREATE TABLE Enrolled
(sid CHAR(20), cid CHAR(20), grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid) REFERENCES Students )
```

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Duke CS, Fall 2017

CompSci 516: Database Systems

32

Enforcing Referential Integrity

- Consider Students and Enrolled
 - sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted?
 - **Reject it!**
- What should be done if a Students tuple is **deleted**?
 - Three semantics allowed by SQL
 1. Also delete all Enrolled tuples that refer to it (cascade delete)
 2. Disallow deletion of a Students tuple that is referred to
 3. Set sid in Enrolled tuples that refer to it to a default sid
 4. (in addition in SQL): Set sid in Enrolled tuples that refer to it to a special value **null**, denoting 'unknown' or 'inapplicable'
- Similar if **primary key of Students tuple is updated**

Duke CS, Fall 2017

CompSci 516: Database Systems

33

Referential Integrity in SQL

- SQL:92 and SQL:1999 support all 4 options on deletes and updates.
 - Default is **NO ACTION** (delete/update is rejected)
 - **CASCADE** (also delete all tuples that refer to deleted tuple)
 - **SET NULL / SET DEFAULT** (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT )
```

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Where do ICs Come From?

- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations
- Can we infer ICs from an instance?
- Key and foreign key ICs are the most common; more general ICs supported too

Duke CS, Fall 2017

CompSci 516: Database Systems

35

Example Instances

- We will use these instances of the Sailors and Reserves relations in our examples
- If the key for the Reserves relation contained only the attributes **sid** and **bid**, how would the semantics differ?

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Basic SQL Query

```
SELECT [DISTINCT] <target-list>
FROM <relation-list>
WHERE <qualification>
```

- **relation-list** A list of relation names
 - possibly with a “range variable” after each name
- **target-list** A list of attributes of relations in relation-list
- **qualification** Comparisons
 - (Attr op const) or (Attr1 op Attr2)
 - where op is one of =, <, >, <=, >= combined using AND, OR and NOT
- **DISTINCT** is an optional keyword indicating that the answer should not contain duplicates
 - Default is that duplicates are not eliminated!

Duke CS, Fall 2017

CompSci 516: Database Systems

37

Conceptual Evaluation Strategy

```
SELECT [DISTINCT] <target-list>
FROM <relation-list>
WHERE <qualification>
```

- **Semantics** of an SQL query defined in terms of the following conceptual evaluation strategy:
 - Compute the cross-product of <relation-list>
 - Discard resulting tuples if they fail <qualifications>
 - Delete attributes that are not in <target-list>
 - If **DISTINCT** is specified, eliminate duplicate rows
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute the same answers

Duke CS, Fall 2017

CompSci 516: Database Systems

38

Example of Conceptual Evaluation

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor			
sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 1: Form **cross product** of Sailor and Reserves

Sailor				Reserves		
sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96
31	lubber	8	55	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Example of Conceptual Evaluation

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor			
sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 2: Discard tuples that do not satisfy <qualification>

Sailor				Reserves		
sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96
31	lubber	8	55	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Example of Conceptual Evaluation

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor			
sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 3: Select the specified attribute(s)

Sailor				Reserves		
sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96
31	lubber	8	55	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

A Note on “Range Variables”

- Really needed only if the same relation appears twice in the **FROM** clause
 - sometimes used as a short-name
- The previous query can also be written as:

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND bid=103
```

OR

```
SELECT sname
FROM Sailors, Reserves
WHERE Sailors.sid=Reserves.sid
AND bid=103
```

It is good style, however, to use range variables always!

Duke CS, Fall 2017

CompSci 516: Database Systems

42

Find sailor ids who've reserved at least one boat

```
SELECT ????
FROM Sailors S, Reserves R
WHERE S.sid=R.sid
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2017 CompSci 516: Database Systems 43

Find sailor ids who've reserved at least one boat

- Would adding DISTINCT to this query make a difference?

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2017 CompSci 516: Database Systems 44

Find sailors who've reserved at least one boat

- Would adding DISTINCT to this query make a difference?
- What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause?

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2017 CompSci 516: Database Systems 45

Joins

- Condition/Theta-Join
- Equi-Join
- Natural-Join
- (Left/Right/Full) Outer-Join

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2017 CompSci 516: Database Systems 46

Condition/Theta Join

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age >= 40
```

Form cross product, discard rows that do not satisfy the condition

sid	sname	rating	age	sid	bid	day	sid	bid	day
22	dustin	7	45	22	101	10/10/96	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96			
31	lubber	8	55	58	103	11/12/96			
58	rusty	10	35	22	101	10/10/96			
58	rusty	10	35	58	103	11/12/96			

Duke CS, Fall 2017 CompSci 516: Database Systems 47

Equi Join

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age = 45
```

A special case of theta join
Join condition only has equality predicate =

sid	sname	rating	age	sid	bid	day	sid	bid	day
22	dustin	7	45	22	101	10/10/96	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96			
31	lubber	8	55	58	103	11/12/96			
58	rusty	10	35	22	101	10/10/96			
58	rusty	10	35	58	103	11/12/96			

Duke CS, Fall 2017 CompSci 516: Database Systems 48

Natural Join

```
SELECT *
FROM Sailors S NATURAL JOIN Reserves R
```

A special case of equi join
Equality condition on ALL common predicates (sid)
Duplicate columns are eliminated

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	sname	rating	age	bid	day
22	dustin	7	45	101	10/10/96
22	dustin	7	45	103	11/12/96
31	lubber	8	55	101	10/10/96
31	lubber	8	55	103	11/12/96
58	rusty	10	35	101	10/10/96
58	rusty	10	35	103	11/12/96

Duke CS, Fall 2017

CompSci 516: Database Systems

49

Outer Join

```
SELECT S.sid, R.bid
FROM Sailors S LEFT OUTER JOIN Reserves R
ON S.sid=R.sid
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Preserves all tuples from the left table whether or not there is a match
if no match, fill attributes from right with null

Similarly RIGHT/FULL outer join

sid	bid
22	101
31	null
58	103

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2017

CompSci 516: Database Systems

50

Expressions and Strings

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching
- Find triples (of ages of sailors and two fields defined by expressions) for sailors
 - whose names begin and end with B and contain at least three characters
- LIKE is used for string matching. `_' stands for any one character and `%` stands for 0 or more arbitrary characters
 - You will need these often

Duke CS, Fall 2017

CompSci 516: Database Systems

51

Find sid's of sailors who've reserved a red or a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- Assume a Boats relation
- UNION: Can be used to compute the union of any two union-compatible sets of tuples
 - can themselves be the result of SQL queries
- If we replace OR by AND in the first version, what do we get?
- Also available: EXCEPT (What do we get if we replace UNION by EXCEPT?)

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Find sid's of sailors who've reserved a red and a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Find sid's of sailors who've reserved a red and a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- INTERSECT: Can be used to compute the intersection of any two union-compatible sets of tuples.
 - Included in the SQL/92 standard, but some systems don't support it

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Nested Queries - IN

Find names of sailors who've reserved boat #103:

```
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)
```

- A very powerful feature of SQL:
 - a WHERE/FROM/HAVING clause can itself contain an SQL query
- To find sailors who've not reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a **nested loops evaluation**
 - For each Sailors tuple, check the qualification by computing the subquery

Duke CS, Fall 2017

CompSci 516: Database Systems

55

Nested Queries with Correlation - EXISTS

Find names of sailors who've reserved boat #103:

- EXISTS is another set comparison operator, like IN
- Illustrates why, in general, subquery must be re-computed for each Sailors tuple

Duke CS, Fall 2017

CompSci 516: Database Systems

56

Nested Queries with Correlation - UNIQUE

Find names of sailors who've reserved boat #103:

- If UNIQUE is used, and * is replaced by *R.bid*, finds sailors with at most one reservation for boat #103
 - UNIQUE checks for duplicate tuples

Duke CS, Fall 2017

CompSci 516: Database Systems

57

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE
- Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL, *op* IN
 - where *op* : >, <, =, <=, >=
- Find sailors whose rating is greater than that of some sailor called Horatio
 - similarly ALL

Duke CS, Fall 2017

CompSci 516: Database Systems

58