CompSci 516

Database Systems

Lecture 2

SQL

(Incomplete Notes)

Instructor: Sudeepa Roy

Announcements

- If you are enrolled to the class, but have not received the email from Piazza, please send me an email
- If you are on the waitlist and want to enroll, please send me an email
- HW1 will be released soon (~tomorrow)
- TA office hours:
  - Yuchao: LSRC D309, Mondays 1:30-2:30 pm
  - Tianpeng: LSRC D344, Wednesdays 1:30-2:30 pm

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

Today's topic

- Overview of XML
- SQL in a nutshell
  - Reading material: [RG] Chapters 3 and 5
  - Additional reading for practice: [GUW] Chapter 6

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

XML: an overview

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

XML: Example

Attributes

```
<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>
```

Elements

Attribute vs. Elements

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

XML vs. Relational Databases

Which one is easier?

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

or

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

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?

XML to Relational Model

- Problem 1: Repeated attributes

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

Title	Publisher	Author1	Author2

What if the paper has a single author?

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>
     <pubisher>Prentice Hall</publisher>
</book>
```

Does not work

Title	Publisher	Author1	Author2

XML to Relational Model

Book

BookId	Title	Publisher
b1	Database Management Systems	McGraw Hill
b2	Database Systems – The Complete Book	Prentice Hall

BookAuthoredBy

BookId	Author
b1	Ramakrishnan
b1	Gehrke
b2	Garcia-Molina
b2	Ullman
b2	Widom

XML to Relational Model

- Problem 2: Missing attributes

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

BookId	Title	Publisher	Edition
b1	Database Management Systems	McGraw Hill	Third
b2	Database Systems – The Complete Book	Prentice Hall	null

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

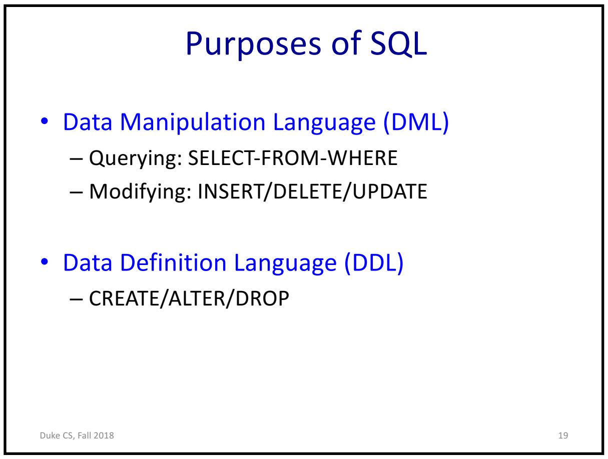
SQL
(Stuctured Query Language)

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
  - Based on a sound theory!
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

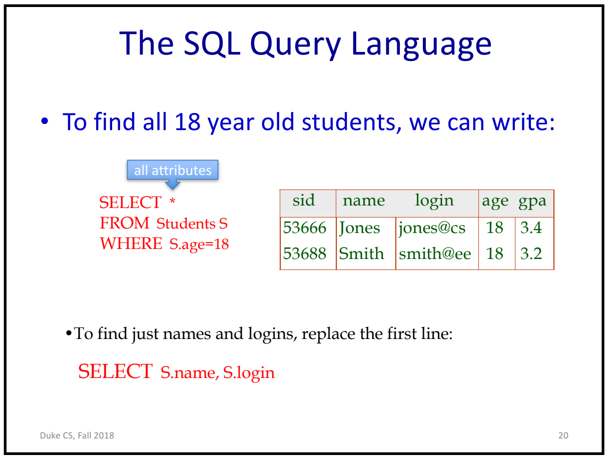
The SQL Query Language

- Developed by IBM (systemR) in the 1970s based on Ted Codd's relational model
  - First called "SEQUEL" (Structured English Query Language)
- First commercialized by Oracle (then Relational Software)in 1979
- Standards by ANSI and ISO since it is used by many vendors
  - SQL-86, -89 (minor revision), -92 (major revision), -96, -99 (major extensions), -03, -06, -08, -11, -16



Purposes of SQL

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



The SQL Query Language

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

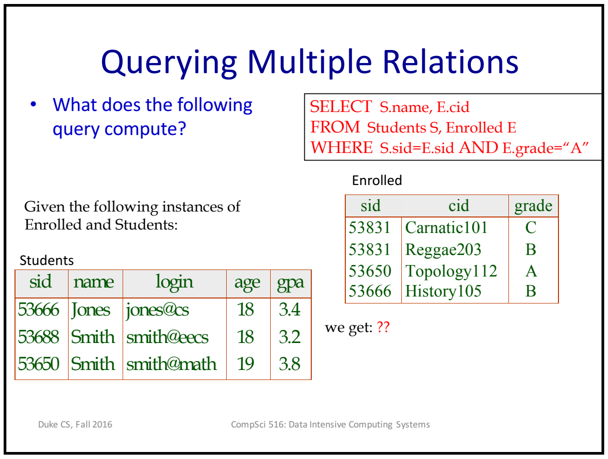
all attributes

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

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
```



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:

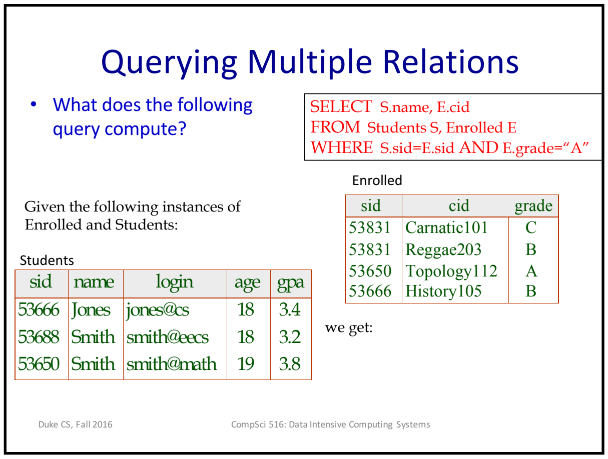
Enrolled

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

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

we get: ??



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:

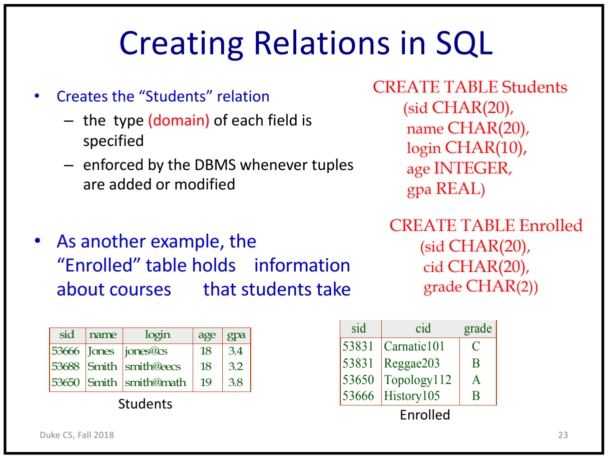
Enrolled

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

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

we get:



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

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

- As another example, the “Enrolled” table holds information about courses that students take

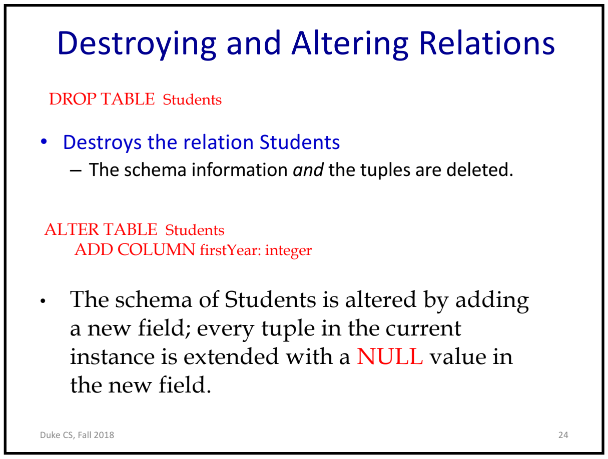
```
CREATE TABLE Enrolled
    (sid CHAR(20),
     cid CHAR(20),
     grade CHAR(2))
```

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



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.

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’
```

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!

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)

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.
- Any possible benefit to refer to a tuple using primary key (than any key)?

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  ???)
```

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.”

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."
- 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."

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."
- 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."

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."
- 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."
- Used carelessly, an IC can prevent the storage of database instances that arise in practice!

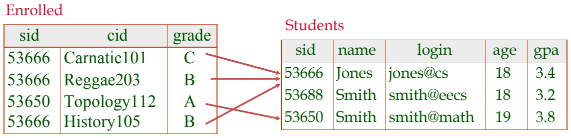
Foreign Keys, Referential Integrity

- 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

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 )
```



Enrolled

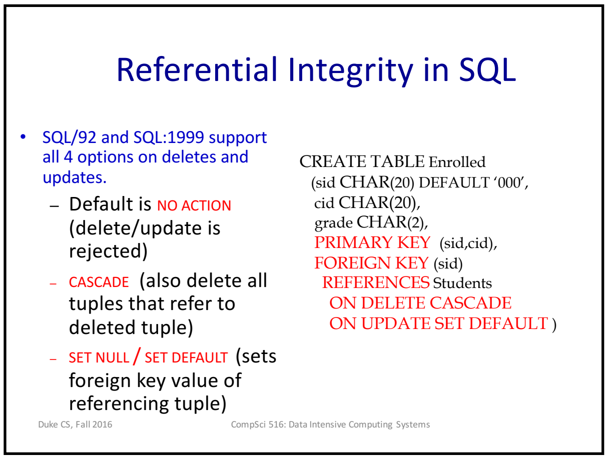
sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

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

Enforcing Referential Integrity

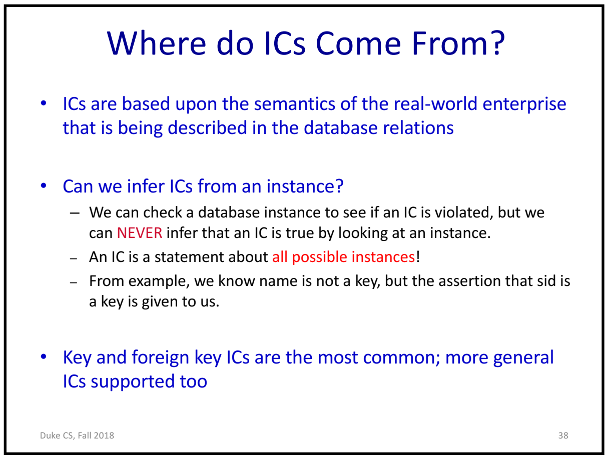
- Consider Students and Enrolled
  - sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted?
  - Reject it!
- What should be done if a Students tuple is deleted?
  - Three semantics allowed by SQL
    1. Also delete 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



Referential Integrity in SQL

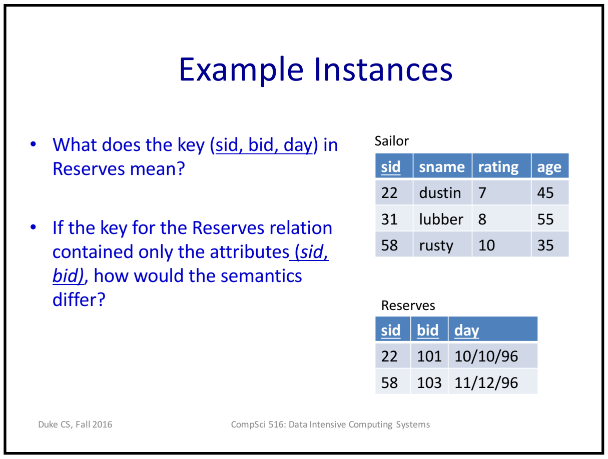
- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
  (sid CHAR(20) DEFAULT '000',
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
    REFERENCES Students
     ON DELETE CASCADE
     ON UPDATE SET DEFAULT )
```



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?
  - We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
  - An IC is a statement about all possible instances!
  - From example, we know name is not a key, but the assertion that sid is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too



Example Instances

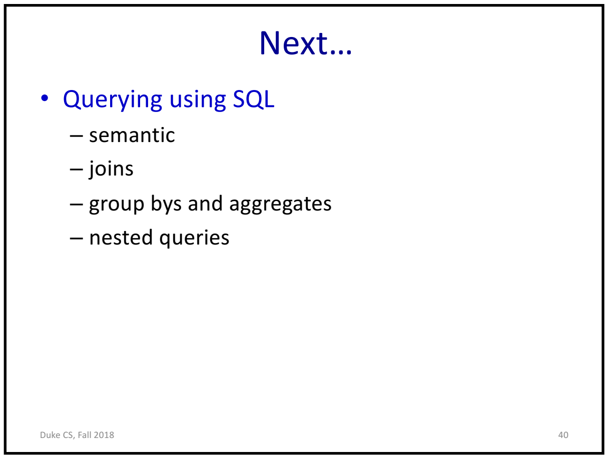
- What does the key (sid, bid, day) in Reserves mean?
- If the key for the Reserves relation contained only the attributes (sid, bid), how would the semantics differ?

Sailor

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

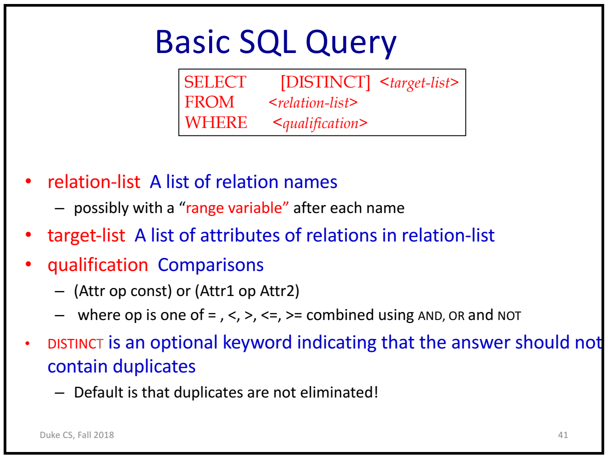
Reserves

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



Next…

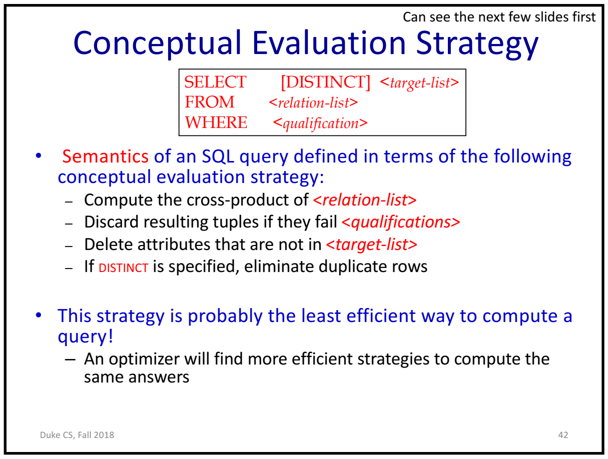
- Querying using SQL
  - semantic
  - joins
  - group bys and aggregates
  - nested queries



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!

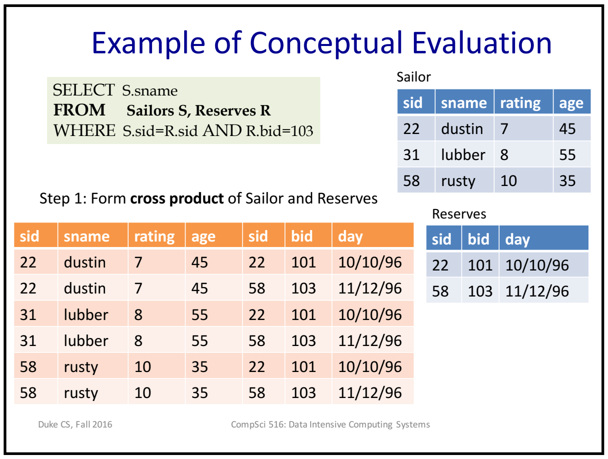


Can see the next few slides first

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



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

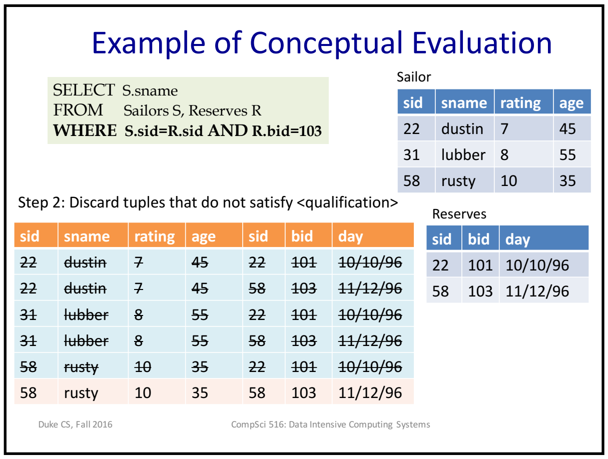
Step 1: Form **cross product** of Sailor and Reserves

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

Reserves

sid	bid	day
22	101	10/10/96
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

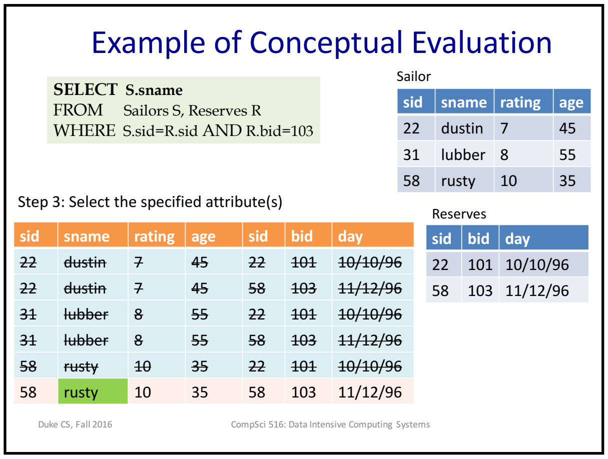
Step 2: Discard tuples that do not satisfy <qualification>

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

Reserves

sid	bid	day
22	101	10/10/96
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

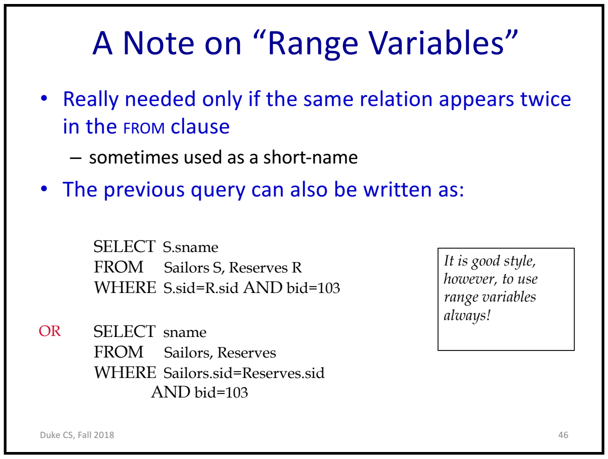
Step 3: Select the specified attribute(s)

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

Reserves

sid	bid	day
22	101	10/10/96
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:

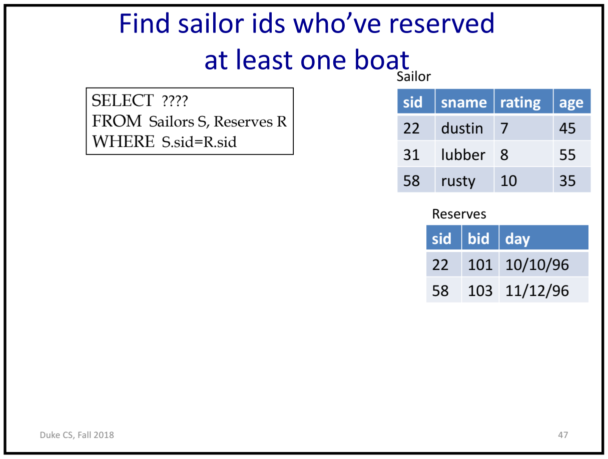
```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND bid=103
```

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 2018 46



Find sailor ids who've reserved at least one boat

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

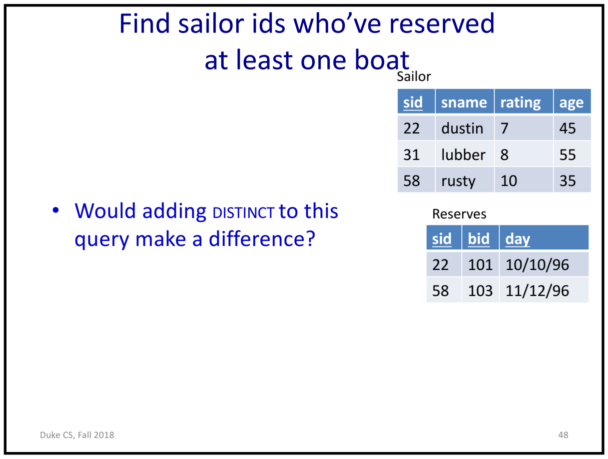
Sailor

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

Reserves

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

Duke CS, Fall 2018 47



Find sailor ids who've reserved at least one boat

Sailor

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

- Would adding DISTINCT to this query make a difference?

Reserves

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

Duke CS, Fall 2018 48

Find sailors who've reserved at least one boat

- Would adding DISTINCT to this query make a difference?

Sailor

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

Reserves

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

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

Condition/Theta Join

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

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

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

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~~

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

Equi Join

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

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

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

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~~

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

Natural Join

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

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

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

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

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

Outer Join

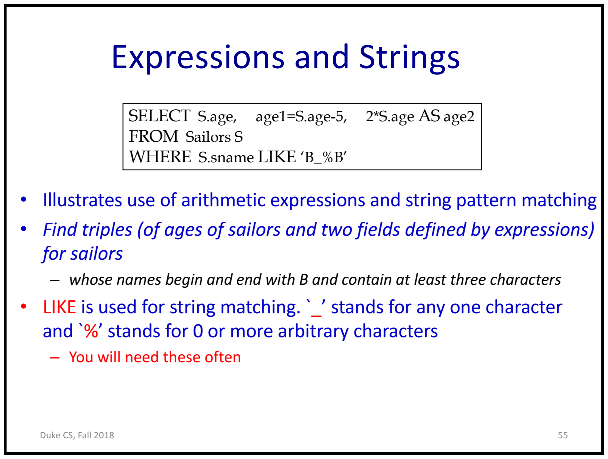
```
SELECT  S.sid, R. bid
FROM  Sailors S LEFT OUTER JOIN Reserves R
ON  S.sid=R.sid
```

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

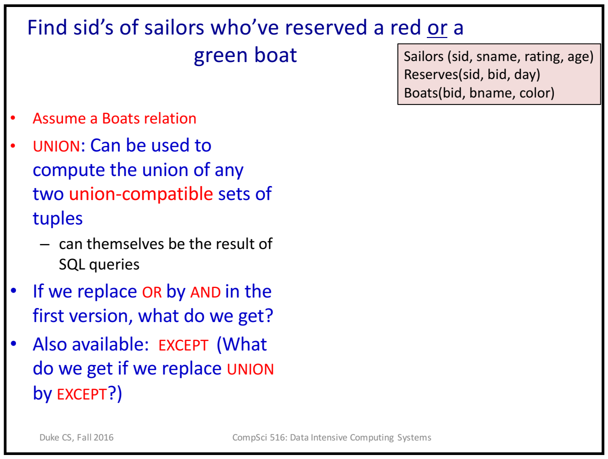
sid	bid	day
22	101	10/10/96
58	103	11/12/96



Expressions and Strings

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

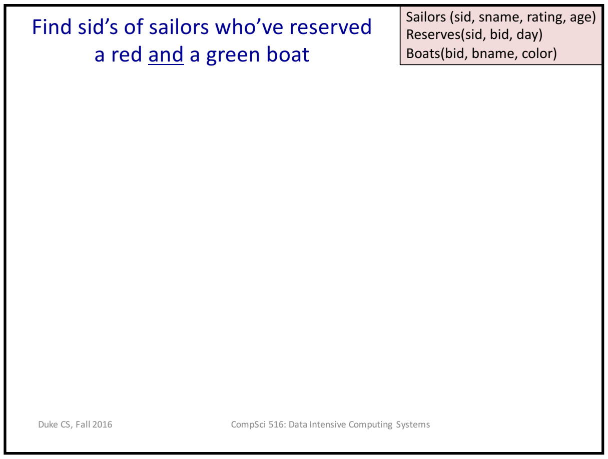
- Illustrates use of arithmetic expressions and string pattern matching
- *Find triples (of ages of sailors and two fields defined by expressions) for sailors*
  - *whose names begin and end with B and contain at least three characters*
- LIKE is used for string matching. `_' stands for any one character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



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

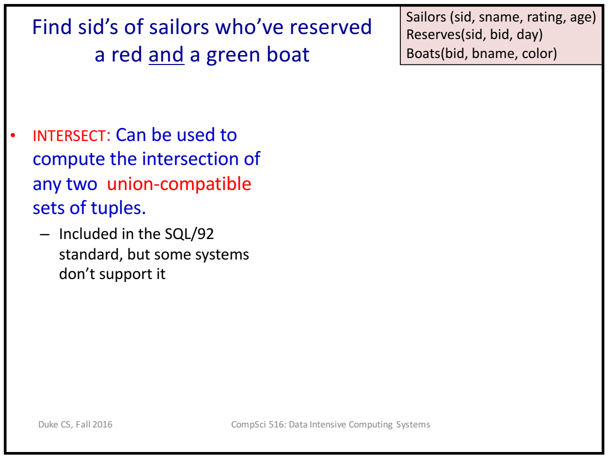
Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- 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?)



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

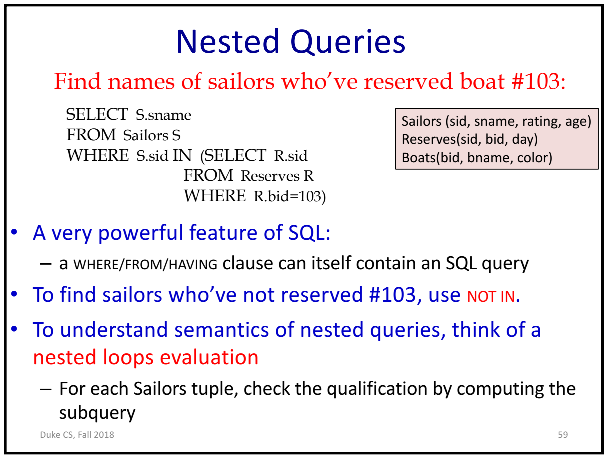
Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)



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



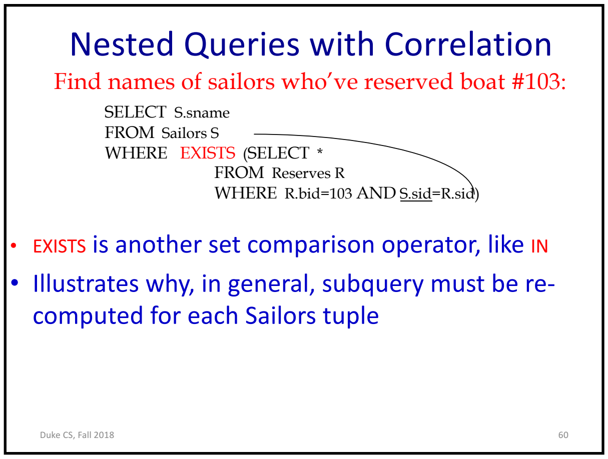
Nested Queries

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

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
                FROM Reserves R
                WHERE R.bid=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

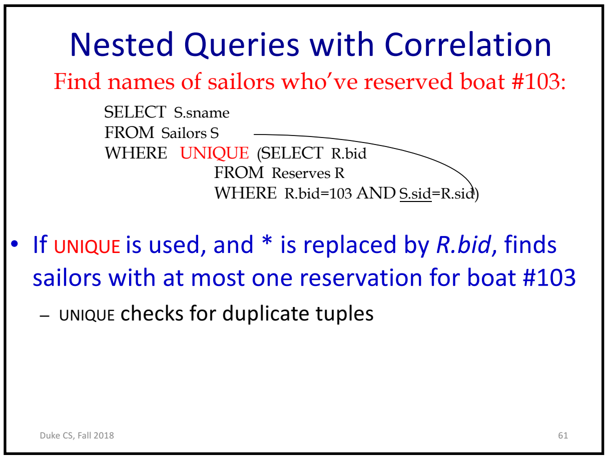


Nested Queries with Correlation

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

```
SELECT S.sname
FROM Sailors S
WHERE  EXISTS (SELECT *
               FROM Reserves R
               WHERE R.bid=103 AND S.sid=R.sid)
```

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

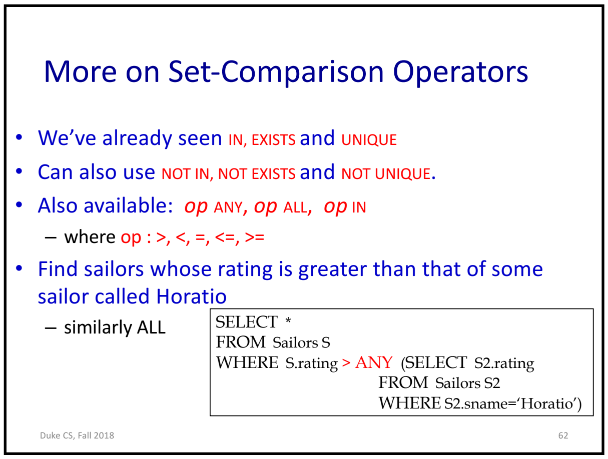


Nested Queries with Correlation

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

```
SELECT  S.sname
FROM  Sailors S
WHERE   UNIQUE  (SELECT  R.bid
                 FROM  Reserves R
                 WHERE  R.bid=103 AND S.sid=R.sid)
```

- 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



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

```
SELECT  *
FROM  Sailors S
WHERE  S.rating > ANY  (SELECT  S2.rating
                        FROM  Sailors S2
                        WHERE S2.sname='Horatio')
```