Tutorial 1

SQL

Shan Ning

September 26, 2021

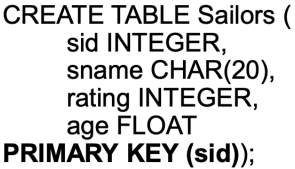
Overview

1. Terminology
2. Single Table Operation
3. Multiple Table Operation
4. Example

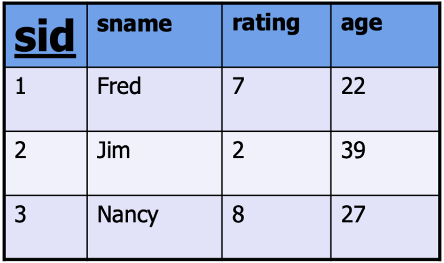
Terminology

- Database
- Relation: Schema and instance
- Attribute(Column, Field), Note: primary key
- Tuple(Record, Row)

Create Table

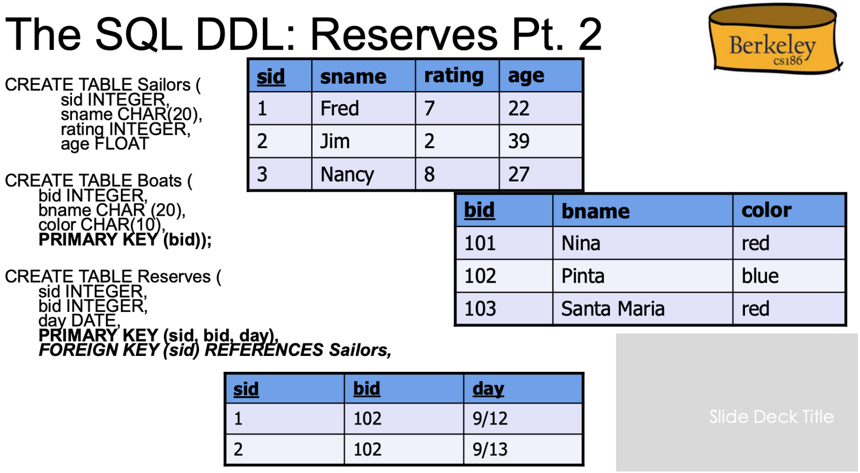


```
CREATE TABLE Sailors (
        sid INTEGER,
        sname CHAR(20),
        rating INTEGER,
        age FLOAT
PRIMARY KEY (sid));
```



<u>sid</u>	**sname**	**rating**	**age**
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Primary key & Foreign key



The SQL DDL: Reserves Pt. 2

```
CREATE TABLE Sailors (
          sid INTEGER,
          sname CHAR(20),
          rating INTEGER,
          age FLOAT

CREATE TABLE Boats (
      bid INTEGER,
      bname CHAR (20),
      color CHAR(10),
      PRIMARY KEY (bid));

CREATE TABLE Reserves (
      sid INTEGER,
      bid INTEGER,
      day DATE,
      PRIMARY KEY (sid, bid, day),
      FOREIGN KEY (sid) REFERENCES Sailors,
```

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

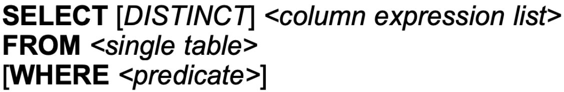
bid	bname	color
101	Nina	red
102	Pinta	blue
103	Santa Maria	red

sid	bid	day
1	102	9/12
2	102	9/13

Slide Deck Title

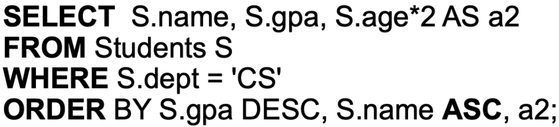
Query Pt.1

- select



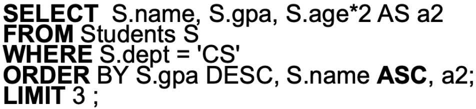
```
SELECT [DISTINCT] <column expression list>
FROM <single table>
[WHERE <predicate>]
```

- order by



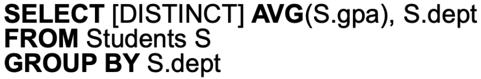
```
SELECT  S.name, S.gpa, S.age*2 AS a2
FROM Students S
WHERE S.dept = 'CS'
ORDER BY S.gpa DESC, S.name ASC, a2;
```

- limit

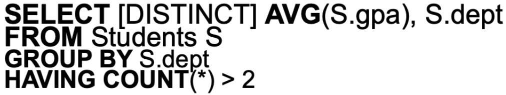


```
SELECT  S.name, S.gpa, S.age*2 AS a2
FROM Students S
WHERE S.dept = 'CS'
ORDER BY S.gpa DESC, S.name ASC, a2;
LIMIT 3 ;
```

Query Pt.2: Group by



```
SELECT [DISTINCT] AVG(S.gpa), S.dept
FROM Students S
GROUP BY S.dept
```

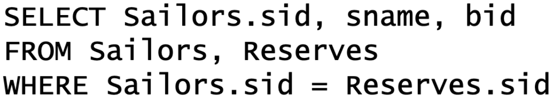


```
SELECT [DISTINCT] AVG(S.gpa), S.dept
FROM Students S
GROUP BY S.dept
HAVING COUNT(*) > 2
```

Query Summary

SELECT [DISTINCT] <*column expression list*>
FROM <*single table*>
[**WHERE** <*predicate*>]
[**GROUP BY** <*column list*>
[**HAVING** <*predicate*>]]
[**ORDER BY** <*column list*>]
[**LIMIT** <integer>];

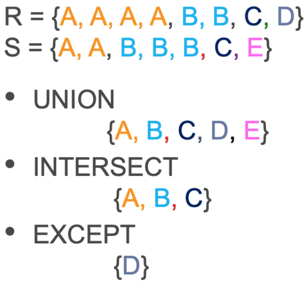
Cross Product



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

Use Boolean logic in WHERE

Set Semantics



R = {A, A, A, A, B, B, C, D}
S = {A, A, B, B, B, C, E}

- UNION

  {A, B, C, D, E}
- INTERSECT

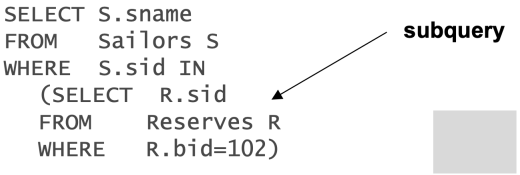
  {A, B, C}
- EXCEPT

  {D}

Note: 'ALL' multi-set semantics

Nested

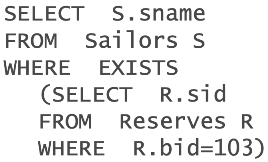
- IN



```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN
   (SELECT  R.sid
   FROM     Reserves R
   WHERE    R.bid=102)
```

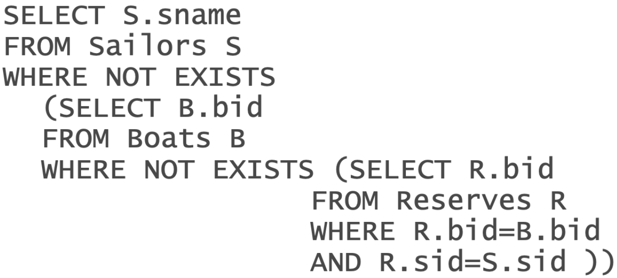
subquery

- Note: 'NOT IN' is similar
- EXISTS



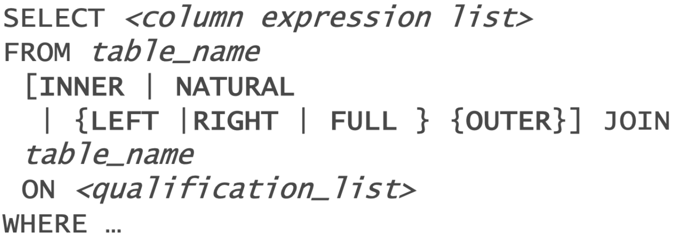
```
SELECT  S.sname
FROM  Sailors S
WHERE  EXISTS
   (SELECT  R.sid
   FROM  Reserves R
   WHERE  R.bid=103)
```

Nested Pt.2: Division



```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS
   (SELECT B.bid
   FROM Boats B
   WHERE NOT EXISTS (SELECT R.bid
                     FROM Reserves R
                     WHERE R.bid=B.bid
                     AND R.sid=S.sid ))
```

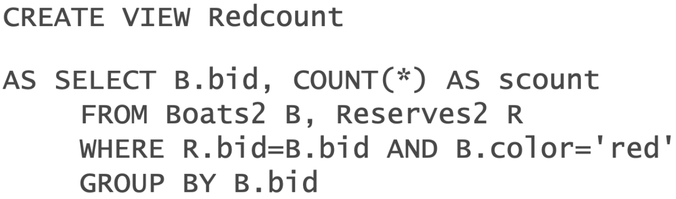
Join Operation



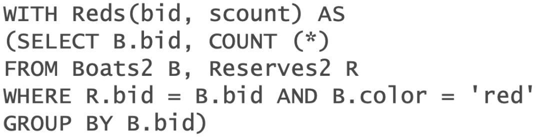
```
SELECT <column expression list>
FROM table_name
 [INNER | NATURAL
  | {LEFT |RIGHT | FULL } {OUTER}] JOIN
 table_name
 ON <qualification_list>
WHERE …
```

- Inner Join
- Outer Join: Returns all matched rows, and preserves all unmatched rows from the table on the left/right

Create VIEW



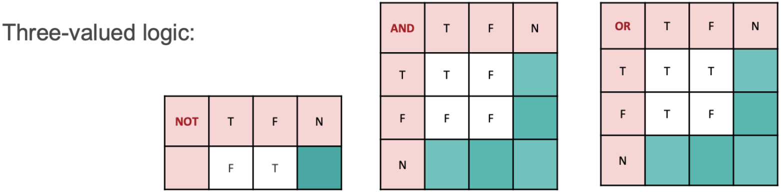
```
CREATE VIEW Redcount

AS SELECT B.bid, COUNT(*) AS scount
     FROM Boats2 B, Reserves2 R
     WHERE R.bid=B.bid AND B.color='red'
     GROUP BY B.bid
```



```
WITH Reds(bid, scount) AS
(SELECT B.bid, COUNT (*)
FROM Boats2 B, Reserves2 R
WHERE R.bid = B.bid AND B.color = 'red'
GROUP BY B.bid)
```

About Null

- (x op NULL) evaluates to NULL
- Do not output a tuple WHERE NULL
- NULL in Boolean logic



Three-valued logic:

NOT	T	F	N
	F	T	🟩

AND	T	F	N
T	T	F	🟩
F	F	F	🟩
N	🟩	🟩	🟩

OR	T	F	N
T	T	T	🟩
F	T	F	🟩
N	🟩	🟩	🟩

- Generally NULL **column values** are ignored by aggregate functions

Examples

The End