* Put up relation schemes from <u>4.1</u>

Relational Algebra Continued

Additional operators: Do <u>NOT</u> add any power to the language, just make query specification more succinct.

Intersection ($\cap$) : $r_1 \cap r_2$ = all tuples that are in r_1 and in r_2.

Say want students enrolled in 321 & 444:

$r_1 \cap r_2$

$r - s \equiv$ 

Note: $r \cap s \equiv r - (r_1 - s)$ $\quad r-(r-s) \equiv$  $\equiv r \cap s$

Theta Join ($\bowtie_\theta$) : combines selection & cartesian product into one operation.

$$\sigma_\theta (r \times s) = r \bowtie_\theta s \quad (\text{relation scheme} = R \cup S)$$

Ex: name of all students enrolled in 321

$$\pi_{s.name} (\text{students} \bowtie_\theta \text{enrolled})$$

where $\theta = (s.sid = e.sid) \wedge (e.cnum = 321)$

The theta join is very general. Usually use more specific joins:

Natural join ($\bowtie$) : Connect tuples from two relations that match on all common attributes. In the result common attributes are kept only once.

Ex: want list of all students and the classes they are taking:

Student = (name, sid, ssn, stat)
Enrolled (sid, cnum)

$r = \text{Student} \bowtie \text{Enrolled}$

Scheme (r) = (name, sid, ssn, status, cnum)

Note, common attributes kept only once

Say want just info on Sue:

$$\sigma_{name = Sue} (\text{Student} \bowtie \text{Enrolled})$$

How express as a single θ join?

$(\text{student} \bowtie_{\theta} \text{Enrolled})$ where $\theta = (s.sid = e.sid) \wedge (s.name = Sue)$

Formal definition of $\bowtie$:

let R = scheme of r
S = scheme of s

$A = R \cap S$ = { set of <u>attributes</u> in common

$$r \bowtie s = r \bowtie_{\theta} s$$

$$\text{where } \theta = (r.A_1 = s.A_1) \wedge (r.A_2 = s.A_2) \wedge \ldots (r.A_n = s.A_n)$$

$$\forall A_i \in R \cap S$$

We are usually only interested in equality of <u>one</u> attribute

Equi-join ($\underset{attr}{\bowtie}$) : Connect tuples from two relations that match on the specified attribute. Specified attribute kept once, other common attributes denote $r_1.attr$ + $r_2.attr$ in resultant scheme.

Ex: Find all names that are names of a student and a faculty member.

$$\Pi_{name} (student \underset{name}{\bowtie} faculty)$$

Q : would $\Pi_{name}(\text{student} \bowtie \text{faculty})$ give same answer?

student	name	sid	ssn	stat
	Bob	33	936	4
	Sue	19	854	3
	Deb	17	716	2
	Ulf	49	815	4

faculty	name	fid	ssn	salary
	Tom	18	321	50
	Sue	19	717	30
	Lisa	20	216	50

NO

$$\Pi_{name}(\text{student} \bowtie_{name} \text{faculty})$$

$$= \{ \text{Sue} \}$$

$$\Pi_{name}(\text{student} \bowtie \text{faculty})$$

$$= \{ \} \quad \text{empty set}$$

why? Because $\bowtie$ says __all__ attributes in common must be equal and student Sue has a different ssn than faculty Sue.

Grades(sid, cnum, semester, grade, fid)

Q : How get name and sid of all students who have taken a class by "Lent" ?

$$\pi_{S.name,\, sid}\, \sigma_{f.name = \text{"Lent"}} \left(\text{faculty} \underset{fid}{\bowtie} \text{Grades} \underset{sid}{\bowtie} \text{student} \right)$$

Q : what is the scheme of $\left(\text{faculty} \underset{fid}{\bowtie} \text{Grades} \underset{sid}{\bowtie} \text{student} \right)$?

(f.name , fid , f.ssn , salary , sid , cnum , semester , grade , s.name , s.ssn , status)

Must includ f.name + f.ssn to distinguish fro s.name + s.ssn

Q : would $\pi_{name,\, sid} \left(\text{faculty} \bowtie \text{Grades} \bowtie \text{student} \right)$ work ?

No ; would force equality of name + ssn also.

Q : would $\pi_{S.name,\, sid}\, \sigma_{f.name = \text{"Lent"}} \left(\text{grades} \underset{fid}{\bowtie} \text{faculty} \underset{sid}{\bowtie} \text{student} \right)$ work ?

Yes $\bowtie_\theta$ are associative.

Equi-join of 3 relations example

faculty (name fid ssn salary)
grades (sid cnum semester grade fid)
student (name sid ssn stat)

fac	name	fid	ssn	salary
	levt	7	999	10
	nicol	8	717	20
	park	2	871	30

student	name	sid	ssn	stat
	Bob	317	216	4
	Sue	389	285	4
	Mary	421	525	3
	Tom	516	219	2

grades	sid	cnum	semester	grade	fid
	317	100	92.1	B	7
	317	101	92.1	B	8
	421	201	92.1	A	2
	516	321	92.2	A	7
	317	321	92.2	B	7
	389	301	92.1	A	7
	317	444	91.3	A	2
	389	444	91.3	B	2
	421	444	91.3	C	2

1) $\pi_{s.name,\, sid}\left[\left(\left(\sigma_{f.name = lent}(faculty)\right) \bowtie_{fid} Grades\right) \bowtie_{sid} students\right]$

$\sigma_{f.name = lent}(faculty) =$

	fid		
lent	7	999	10

$r_1 = \left(\sigma_{fname}(faculty)\right) \bowtie_{fid} grades =$ (ignore semester & grades for brevity)

				sid	cnum
lent	7	999	10	317	100
"	"	"	"	516	321
"	"	"	"	317	321
lent	7	999	10	389	301

$r_1 \bowtie_{sid} students =$

fname	fid	fssn	salary	sid	cnum	s.name	s.ssn	stat
lent	7	999	10	317	100	Bob	216	4
|	|	|	|	516	321	Tom	219	2
|	|	|	|	317	321	Bob	216	4
lent	7	999	10	389	301	Sue	285	4

$\pi_{s.name,\, s.sid}\left(r_1 \bowtie_{sid} students\right) =$

Bob	317
Tom	516
Sue	389

5.5.c

2)

$$\pi_{s.name,\, sid}\ \sigma_{f.name = levt} \left[\text{faculty} \underset{fid}{\bowtie} \text{grades} \underset{sid}{\bowtie} \text{students} \right]$$

$\text{faculty} \underset{fid}{\bowtie} \text{grades} =$

(omit semester & grade for brevity)

name	fid	ssn	salary	sid	cnum
levt	7	999	10	317	100
				516	321
				317	321
levt	7	999	10	389	301
nicol	8	717	20	317	101
park	2	871	30	421	201
				317	444
				389	444
park	2	871	30	421	444

$\text{faculty} \underset{fid}{\bowtie} \text{grades} \underset{sid}{\bowtie} \text{students} =$

Same as above with appropriate student attributes added on.

Now $\sigma_{f.name = levt}$ eliminates all nicol & park tuples

then $\pi_{s.name,\, sid}$ give students who have taken a class taught by levt

What if I wanted name of all students who took a class by "lent" and have gotten an A in at least one of the classes?

How about:

$$\sigma_{grade = "A"} \; \pi_{s.name, sid} \; \sigma_{f.name = lent} \left[faculty \underset{fid}{\bowtie} grades \underset{sid}{\bowtie} students \right]$$

No : After the projection grade info has been lost

$\sigma_{grade = "A"}$ must be done before $\pi_{s.name, sid}$

Ex:

$$\pi_{s.name, sid} \; \sigma_{(f.name = lent) \wedge (grade = "A")} \left[faculty \underset{fid}{\bowtie} grades \underset{sid}{\bowtie} students \right]$$

Division ($\div$)

Ex: Find sids of all students have completed <u>all</u> courses offered in the past

$$S = \pi_{cnum}(grades)$$

$$r = \pi_{sid,\,cnum}(grades)$$

answer $= r \div S$

Schema $(r \div s) = R - S = \{sid\}$

Those students in r that have taken <u>all</u> courses in S

Built on basic operators

Idea: find all x values that are <u>disqualified</u> s.t. & subtract from A.

$$\text{disqualified} = \underbrace{\pi_{sid}\Big[\big(\underbrace{\pi_{sid}(r) \times S}_{\text{pair each sid with every cnum}}\big) - r\Big]}_{\text{gives sids of tuples that do not have every cnum}}$$

$$r \div s = \pi_{sid}(r) - \pi_{sid}\big[(\pi_{sid}(r) \times S) - r\big]$$

example

$s = \pi_{cnum}(grades) = \{100, 101, 200, 301\}$

$r = \pi_{sid,\,cnum}(grades) =$

sid	cnum
17	100
18	100
17	101
17	200
18	200
19	200
17	301
19	301

Is "18" $\in r \div s$?

No

Is "19" $\in r \div s$? No

Is "17" $\in r \div s$? Yes

* Can show. $R \div S$ is composed of 5 fundamental operators.

Another ÷ example

works_on (essn, pno, hours)   (essn: employee ssn; pno: project #)

employee (name, ssn, bdate, address, sex, salary)

→ Say want name of all employees who work on all the projects that "Poe" works on

→ create relation of project numbers that smith works on

1) ~~Smith~~ POE_pno $\leftarrow \pi_{pno}(\text{works\_on} \bowtie_{essn=ssn} \sigma_{name=poe}(\text{employee}))$

2) create relation of <pno, essn>

$$temp \leftarrow \pi_{pno,\, essn}(\text{works\_on})$$

3) Do division

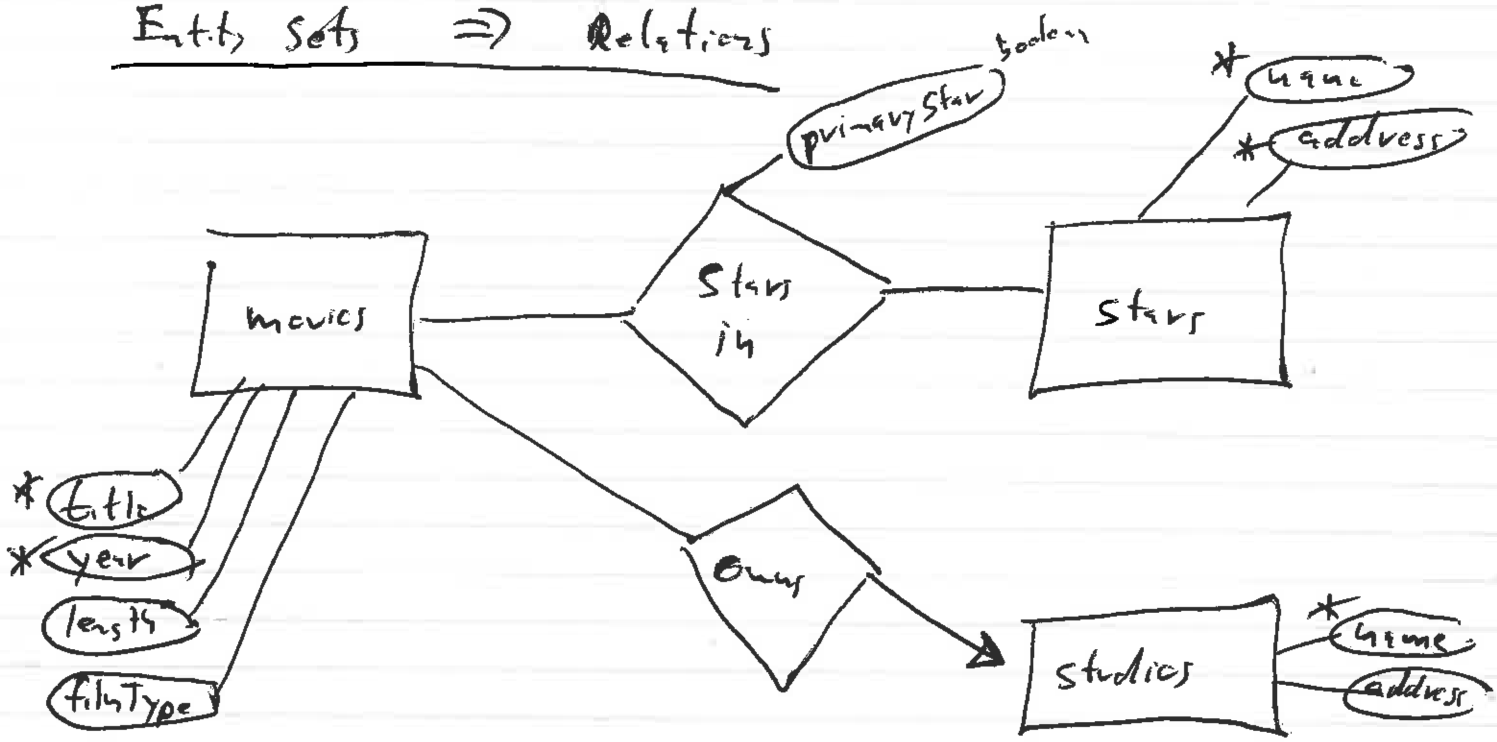
$$temp2 \leftarrow (temp \div POE\_pno)$$

Now have essn of all people who work on every project that smith does, get names

4) $result \leftarrow \pi_{name}(temp2 \bowtie employee)$

Transforming E/R → Relational Design

Entity Sets ⇒ Relations



⇒ Movies (title, year, length, filmtype)
Stars (name, address)
Studios (name, address) → Create table studios (
name varchar(40),
address varchar(255),
PRIMARY KEY (name));

Relationships ⇒ Relations

1) for each entity set involved we take its key attribute(s)

2) add any relationship attributes

⇒ StarsIn (title, year, name, address, primaryStar)

Owns (title, year, name)

* Def ~~but~~ should also specify foreign keys when done in oracle (SQL)

Create table StarsIn (

title varchar(30),

year ~~date~~ number,

name varchar(30),

address varchar(80),

primarystar char(3),

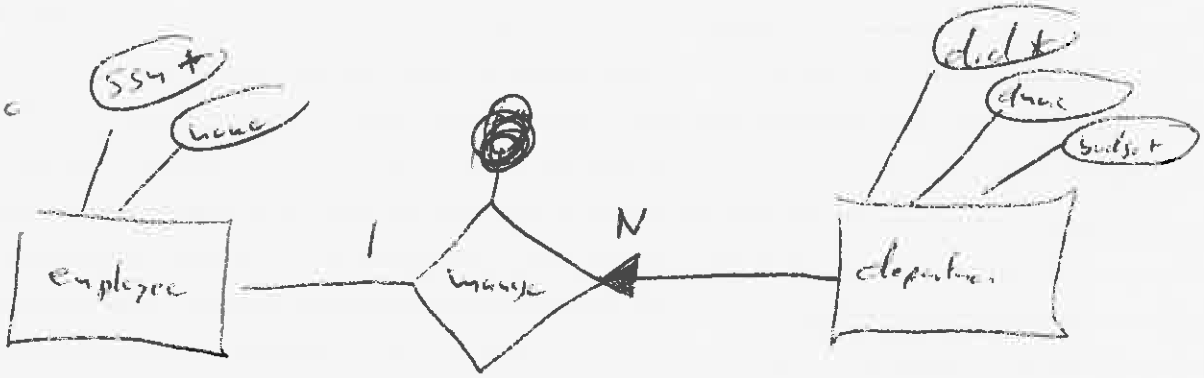
PRIMARY KEY (title, year, name, address),

→ FOREIGN KEY (title, year) REFERENCE movies,

→ " " (name, address) " stars);

Relationships that are N:1, 1:N, or 1:1 can be optimized:

Example



* Since a dept has at most one manager we can add the employee key to the dept relation & omit explicitly storing the manages info.

```
Create table department ( did      integer,
                          dname    varchar (40),
                          budget   real,
                          ssn      char (11),
                          PRIMARY key (did),
                          foreign key (ssn) reference employee);
```

- If relationship has attrs (ex "since") also add it to the department relationship

* In general, if 1:N just add the key from the "1" side into the ~~relation~~ entity set on the "N" side.

Further Examples

Sailors (sid, sname, rating)

Boats (bid, bname, color)

Reserve(sid, bid, date)

1) Find name of sailors that have reserved boat #2

$$\pi_{sname}\left(\sigma_{bid=2}\left(\text{Reserve} \bowtie_{sid} \text{Sailors}\right)\right)$$

$$\pi_{sname}\left(\text{Reserve} \bowtie_{\theta} \text{Sailors}\right)$$

where $\theta = (r.sid = s.sid) \wedge (r.bid = 2)$

2) Find name of sailors who have reserved a red boat.

$$\pi_{sname}\left(\text{Sailors} \bowtie_{sid} \left(\text{Reserve} \bowtie_{bid} \left(\sigma_{color=red}(\text{Boat})\right)\right)\right)$$

3) Find colors of boats reserved by Pat

$$\pi_{color}\left[\left(\sigma_{sname=Pat}(\text{Sailors})\right) \bowtie_{sid} \text{Reserve} \bowtie_{bid} \text{Boats}\right]$$

where can I move the σ to?

4) Find names of sailors who have reserved at least one boat

$$\Pi_{sname}\left[\text{Reserve} \bowtie_{sid} \text{Sailor}\right]$$

(division)

5) Find the names of the sailors who have reserved <u>all</u> boats

A) $$\Pi_{sname}\left[\left(\Pi_{sid,\,bid}(\text{Reserve}) \div \Pi_{bid}(\text{boats})\right) \bowtie_{sid} \text{Sailors}\right]$$

6) Find name of sailors who have reserved <u>both</u> a red and green boat.

(<u>Intersection</u>)

$$\Pi_{sname}\left[\left(\text{Sailors} \bowtie_{sid} \text{reserve} \bowtie_{bid} \left(\Pi_{color=red}(\text{boats})\right)\right) \cap \left(\text{Sailors} \bowtie_{sid} \text{reserve} \bowtie_{bid} \left(\Pi_{color=green}(\text{boats})\right)\right)\right]$$

7) Find name of sailors who have reserved <u>all</u> <u>red</u> boats:

Same as (5) except boats replaced with

$$\left(\sigma_{color=red}(\text{boats})\right)$$

8) Find names of sailors who have reserved a red or green boat.

Same as (6) except $\cup$ instead of $\cap$

Disadvantage of Relational Algebra:

It is a procedural language, i.e user must specify how system will get data.

Solution: Relation Calculus Languages

1) tuple relational calculus
2) QUEL
3) SQL

SQL

SQL developed at IBM San Jose (Almaden)
DML for system R, a research prototype

⇒ The primary DML for databases since IBM decided to use it.

SQL: Structured Query Language (also known as SeQueL)

general form : Select [Distinct] select-list
FROM from-list
where qualification

Select $A_1, A_2 \cdots A_n$
from $r_1, r_2, \cdots r_m$
where P

select ≡ project !! **Bad name**

$$\equiv \pi_{A_1, A_2 \cdots A_n}(\sigma_P(r_1 \times r_2 \times \cdots \times r_m))$$

Sailors (sid, sname, rating)
Boats (bid, bname, color)
Reserve (sid, bid, date)

7.2

9.2

Sailors (sid, sname, rating)

boats (bid, bname, color)

reserve (sid, bid, date)

1) Find the name of all sailors

π_{sname}(Sailors) $\Leftrightarrow$ select sname from sailors $\Leftrightarrow$ $\underbrace{\text{select S.sname from Sailors S}}_{\text{better because more readable}}$

2) Find unique names of sailors

select distinct S.sname
from Sailors S

* without distinct you get duplicates (multiset)

3) Find names of sailors whose rating is > 5

select S.sname
from sailors S
where S.rating > 5

3.5) Select *
from Sailors S
where S.rating > 5

good for interactive but poor style

4) Find name of sailors who have reserved boat # 2:

select sname
from sailors, reserve
where (sailors.sid = reserve.sid) and (bid = 2)

Can put reserve.bid but not needed since unambiguous
Best to be explicit + say reserve.bid or use tuple variable

Select S.sname, S.rating +1 As rating
FROM Sailors S, Reserve R1,
Where S.sid = R.sid AND R.bid = "17"

* This query reports the ~~ratings~~ name/ratings+1 of Sailors who have reserved boat "17"

⇒ Each item in qualification can be generalized also:

expr1 = expr2

Select S1.sname AS name1, S2.sname AS name2
FROM Sailors S1, Sailors S2
Where 2* S1.rating = S2.rating

* Reports pairs of sailor names for when 1st Sailor's rating is ½ the second sailors rating.

* Note: <u>two</u> tuple variables range over Sailors

Meaning of an SQL query:

1) compute X from the from-list
2) delete tuples that fail qualification condition
3) delete all columns not in the select-list
4) if DISTINCT specified remove duplicates

4.5) Find the names of sailors who have reserved at least one boat

```
Select S.name
From   Sailors S, Reserves R
Where  S.sid = R.sid
```

Expressions + Strings in the SELECT Command

SGL supports more general version of column list:

expression AS column-name

any standard arithmetic or string expression

rename old or create new column name

* Find names of sailors who have reserved at least two different boats for same day :

Select S.sname
From Sailors S, Reserve R1, Reserve R2
where (S.sid = R1.sid) AND (S.sid = R2.sid)
AND (R1.date = R2.date)
AND (R1.bid <> R2.bid)

leave off and ask if okay

String Pattern Matching : LIKE operator

% $\triangleq$ zero or more arbitrary chars
_ $\triangleq$ exactly one arbitrary char

Ex:

_AB% $\Leftrightarrow$ any 3 or more char string where 2nd & 3rd char are AB

Find all sailors whose name begins as "Scott"

Select *
From Sailors S
where S.name LIKE 'Scott%'

Set operations

(9.39)

(no 9.3)

∪ ∩ −

Union, Intersect + Except

- Find the name of sailors who have reserved a red or a green boat

```
SELECT S.name
From Sailors S, Boats B, Reserve R
where (S.sid = R.sid) and (R.bid = B.bid)
      and ( B.color = 'red' OR B.color = 'green')
```

- Find the name of sailors who have reserved a red and a green boat.

⇒ Can not just replace OR with and

⇓

```
Select S.name
From Sailors S, Boats B1, Reserve R1, Boats B2, Reserve R2
where S.sid = R.sid AND R1.bid = B1.bid
      AND S.sid = R2.Sid AND R2.bid = B2.bid
      AND B1.color = 'red' AND B2.color = green
```

UGLY, by using set operators this can be easier to express

~~5.4.4~~

9.3b

First, the "or" query:

```
Select S.sname
From Sailors S, Boats B, Reserve R
where S.sid = R.sid and R.bid = B.bid and B.color = 'red'
UNION
Select S2.sname
FROM Sailors S2, Boats B2, Reserve R2
where S2.sid = R2.sid and R2.bid = B2.bid and B2.color = 'green'
```

Now, how get the "red and green"

⇒ just replace UNION with INTERSECT

How get names of sailors who have reserved red boats but not green boats?

⇒ replace UNION with EXCEPT

* Union, Intersect, Except can be used on any two tables that are "union-compatible": have the same # of columns + the columns taken in order have the same domains.

~~9.4b~~

9.3c

Find all sids of sailors who have a rating of 10 or have reserved boat # 111

```
Select  S.sid
From    Sailors S
Where   S.rating = 10
UNION
Select  R.sid
From    Reserve R
Where   R.bid = 111
```

for $\cup, \cap, -$

* Unlike all other SQL commands, the default is to eliminate duplicates. If you want duplicates

Union All

7.3d

7.4

Nested Queries:

Often you want to ~~reference to~~ express a condition that is itself the result of a query

Example 1:

Find the names of sailors who have reserved boat #103

```
Select S.sname
From Sailors S
where S.sid IN ( select R.sid
                 From Reserve R
                 where R.bid = 103)
```

- Easy to modify to find sailors who have not reserved #103
  Just replace IN with NOT IN

Example 2:

Find the names of sailors who have reserved a red boat

```
Select S.sname
From Sailors S
where S.sid IN (select R.sid
                FROM Reserve R
                where R.bid IN (Select B.bid
                                From Boats b
                                where B.color = red)
```

1) What if I replaced outer IN ⇒ NOT IN ?

- set names of sailors who have not reserved a red boat

2) What if replace inner IN ⇒ NOT IN ?

- get names of sailor who have reserved a boat but ~~the boats~~ none of the boats reserved are red.

3) What if replace both IN ⇒ NOT IN?

- names of sailors who have not reserved ~~boats boats that are some color other than red.~~ a boat that is not red

Does it mean they reserved a red boat? ⇒ NO

Correlated Nested Queries:

Find names of sailors who have reserved boat #100:

```
Select  S.sname
From    Sailors S
where   EXISTS  ( Select *
                  From Reserve R
                  where R.bid = 100
                  and S.sid = R.sid)
```

correlation!

∀ tuple in S, checks to see if set of Reserve rows R s.t. r.bid = 100 + s.sid = r.sid is non empty.

⇒ replace EXISTS ⇒ with NOT EXISTS

⇒ name of sailors who have not reserved boat #100

UNIQUE : ~~replace~~ returns true if no duplicates
(note: returns true for empty set!)

NOT UNIQUE : there are duplicates

```
Select S.sname
From   Sailors S
where  NOT UNIQUE  (select *
                    From  Reserve R
                    where R.bid = 100
                      and S.sid = R.sid)
```

⇒ returns name of sailors who have reserved boat #100 at least twice

Set Operators   Exists, IN, UNIQUE

op ANY , op ALL

where op ∈ { <, <=, =, <>, >=, > }

Find sailors whose rating is better (>) Vish's :

Is this correct?
NO!

```
Select S.sid
From   Sailors S
where  S.rating > ANY (Select S2.rating
                       From  Sailors S2
                       where S2.sname = 'Vish')
```

9.4c

* Note, returns sids of sailors whose rating is greater than some sailor named Vish

If ~~you~~ there are multiple Vishs, and you want sailors whose rating is > all Vish's ratings:

~~Select~~ ⇒ replace ANY with ALL

What if no sailors named Vish?

a) > ANY ⇒ return false

(because never found a case where > was true)

b) > ALL ⇒ return true

~~because always [illegible] case~~ →

because > test was true for all returned rows.

Find the sailors with the highest rating.

```
Select S.sid
From Sailors S
Where S.rating >= ALL (Select S2.rating
                       From Sailors S2)
```

Note : IN ≡ = ANY

NOT IN ≡ <> ALL

9.4d

More Nested Queries

Find names of sailors who have reserved a red and green boat

```
Select  S.sname
FROM    Sailors S, Boats B, Reserve R
where   S.sid = R.sid  AND  R.bid = B.bid  AND  B.color = 'red'
        AND  S.sid  IN  ( Select  S2.sid
                          FROM    Sailors S2, Boats B2, Reserve R2
                          where   S2.sid = r2.sid  and
                                  R2.bid = r2.sid  and
                                  B2.color = 'green' )
```

All sailors who have reserved a red boat and have sailor ids ∈ { sailors who have reserved a green boat }

* This queries written using Intersect can be written using IN

How to do division:

Find the name of sailors who have reserved ALL boats.

```
Select  S.sname
From    Sailors S
where   NOT Exists ((  Select  B.bid
                       From    boats B )
                       EXCEPT
                       (Select  R.bid
                        From    Reserve  R
                        where   R.sid = S.sid))
```

Ordering display of results

```
select   name, sid
from     student
where    status = 3
order    by    name
```

The above query lists all juniors in alphabetical order.
If want reverse alphabetical order:

```
order by name desc
```

desc $\triangleq$ descending
asc $\triangleq$ ascending

May have duplicate name, if want secondary sort value:

```
select  name, sid
from    student
where   status = 3
order by name asc, sid desc
```

* sailors (sid, sname, rating, yrs-sailed, age) 9.6

SQL Aggregates
- avg min max sum count

1) select sum (rating), avg (rating); min (rating)
from sailors

2) Find average rating of sailors who have been sailing less than 2 years:

select avg (rating)
from sailors
where yrs_sailed < 2

3) Retrieve the number of sailors who have sailed for more than 5 years

select count (*)
from sailors
where yrs_sailed > 5

4) Retrieve the number of sailors who have reserved boat # 161 on ~~[illegible]~~ 013195
Select count (*)
from ~~sailors~~ reserve r
where ~~s.sid = r.sid and~~ bid = 161 and date = 013[illegible]

9.6b

5) Find the average age of Sailors with a rating ≥ 10

```
Select  AVG(S.age)
FROM    Sailors S
Where   S.rating = 10
```

6) Find the name + age of the oldest sailor:

```
Select  S.sname, MAX(S.age)
FROM    Sailors S
```

** Illegal

If the Select clause uses an aggregate, then it can only use aggregates, ~~attrs~~ or attributes from "group by"

```
Select  S.sname, S.age
FROM    Sailors S
Where   S.age = (Select MAX (S2.age)
                 FROM Sailors S2)
```

9.6€

7)

```
Select S.Sname
from Sailors S
where S.rating > Avg (S.rating)
```

NO

⇒

```
Select S.Sname
From Sailors S
where S.rating > (Select Avg (S.rating)
                  from Sailors )
```

~~7) Retrieve the name of sailors whose rating is above the average rating (over all sailors)~~

~~select sname~~
~~from sailors~~
~~where rating > AVG(rating)~~

<u>Grouping attributes</u>: when we want to apply the aggregates to subgroups of tuples.

8) Retrieve the age and average rating for sailors of each age

select age, avg(rating)
from sailors
group by age

Ex

sailors

• •	5	15
	4	21
	3	24
	6	28
	3	21
	5	21
	4	28
	1	24

15, 5
21, 4
24, 2
28, 5

General form

Select select-list
From from list
Where Q
Group by grouping list
Having group qualification

9) Find the age of the youngest sailor who is eligible to vote for each rating level with at least 2 sailors.

```
Select   S.rating, MIN (S.age)
From     Sailors S
Where    S.age >= 18
Group By S.rating
Having   Count (*) > 1
```

See 9.7c for example of how processed

10) For each red boat, find the # reservations

```
Select   B.bid, COUNT (*) As reserveCount
From     Boats B, Reserve R
Where    (R.bid = B.bid) and B.color = "red"
Group by B.bid
```

Assume following sailors instance:

sid	sname	rating	age
85	art	3	25.5
31	lubber	8	55.5
32	andy	8	25.5
95	bob	3	63.5
58	rusty	10	35.0
29	tina	1	33.0
71	jessica	10	16.0

Step 1: Create cross product

⇒ since only 1 relation ⇒ sailors

Step 2: apply where clause

⇒ eliminates jessica

Step 3: eliminate unneeded columns

Needed: anything in select, group by + having clauses

⇒ this eliminates sid, sname

Step 4: Sort ~~by~~ relation according to group by clause

1	33
3	25.5
3	63.5
8	55.5
8	25.5
10	35.0

(9.7d)

Step 5 apply having qualification

⇓

⇒ eliminate rows with 1 + 10

* Note order of <u>where</u> + <u>having</u> important

Step 6 apply the aggregate + selection clause to remaining groups

⇓

3	25.5
8	25.5

if having was applied before where jessie would have been included and would have been answer for ratings of {3, 8 + 10} even though there are NOT 2 or more 18 year olds in the group of "10s"

7.8

9.7e

11)

Select B.bid, Count (*) AS sailorcount
From ~~Sailors~~ S, Boats B, Reserve R
where ~~(S.sid = R.sid)~~ ~~and~~ (R.bid = B.bid)
Group By B.bid
Having B.color = "red"

Illegal

Can do by sayn
where (R.bid = B.bid) and (B.color =

* only columns that appear in group by clause can be in having clause unless they appear as arguments to an aggregate

12) Find the avg age of sailors for each rating level that has at least two sailors!

SELECT S.rating, AVG (S.age) AS avgage
FROM Sailors S
Group By S.rating
Having 1 < (Select COUNT(*)
From Sailors S2
where S.rating = S2.rating)

or

Having 1 < Count(*)

9.7f

Movie (title, year, length, inColor, studioName, producerC#)
MovieExec (name, address, cert# , netWorth)

1) Find the average net worth of all movie execs:

```
Select AVG(networth)
from MovieExec;
```

2) Find the total number of movie minutes for each studio (total length sum for each movie)

```
Select studioName, sum(length)
From Movie
Group by StudioName ;
```

3) Find the total length of film produced by each producer listed by name

```
Select name, Sum(length)
From MovieExec ~~as~~ ME , Movie M
Where M.producerC# = ME.cert#
Group by ME.name
```

9.79

what if two different producer have same name?

Select ME.name, ME.cert#, sum(length)
From MovieExec ME, Movie M
where M.producerC# = ME.cert#
Group by ME.name, ME.cert#

4) ~~Assume producer names are unique, we want to print total film lengths for those producer who have made at least one film before 1930~~

~~Select ME.name, sum(M.length)~~
~~From MovieExec ME, Movie M~~
~~where producerC# = cert#~~
~~Group by ME.name~~ No !
~~Having MIN(M.year) < 1930 ;~~

Data Definition Languages

SGL

- * • scheme for each relation
- * • domain of attribute values
- • which indices exist on each relation
- • security and authorization info for each relation
- • integrity constraints
- • physical storage structure of each relation

Consider only * items now

Define relation scheme (including domain of attributes) :

$$\text{create table } r\,(\underset{\text{attribute}}{A_1}\ \underset{\text{domain}}{D_1},\ A_2\ D_2,\ \ldots\ A_n\ D_n)$$

- create creates a new relation (empty)
- use insert to add tuples

drop table r

Drop table removes all data in r and all info about r

QUEL : similar commands exist for QUEL

⇒ parallel

QBE: Query By Example

QBE system developed at IBM T.J. Watson in early 70s. Main idea is to provide "user friendly" DML.

Queries expressed as examples:

System provide "skeleton table" which user modifies to show what is wanted

P. ≜ print this field
_X ≜ variable name X

student	name	sid	ssn	stat

↗ system supplies this skeleton

student	name	sid	ssn	stat
	P.			

⇒ Prints out name of all students.

⇒ QBE eliminates duplicates automatically, if want duplicates include must specify as:

P.all

students	name	sid	ssn	stat
	P.	P.	P.	P.

$\Updownarrow$

students	name	sid	ssn	stat
P.				

Print names of all juniors

students	name	sid	ssn	stat
	P.			=3

Variables used in joins:

what does this do?

students	name	sid	ssn	stat
	P.	_X		=2

grades	sid	cnum	grade
	_X	321	

Answer: prints out names of all sophmores enrolled in 321