Let's add the info from the Course table into the mix:

$$SG \underset{cnum}{\bowtie} Course$$

This gives us:

s.name	sg.sid *	s.ssn	s.status	cnum *	g.sem	g.grade	c.name	c.credits	c.dept	c.fid
Bob	99	121	4	314	92.1	A	architec	3	CS	3

So... we have a big Student-Grade-Course table. How do we get the answer we're looking for?

$$\pi_{s.name,\, s.sid}\left(\sigma_{c.fid = 9}\left(\left(\left(Student \bowtie Grade\right) \underset{cnum}{\bowtie} Course\right)\right)\right)$$

Q: would the following have worked instead?

$$\pi\left(\sigma\left(\left(Student \bowtie Course\right) \bowtie Grade\right)\right)$$

$\div$ (Division operator)

- Who are all students (by sid) who have taken <u>all</u> courses offered?

A: Grades ÷ Courses ★ this is not correct.... Why?

★ this operation is not implemented directly in SQL

$$R \div S = \pi_t[(\pi_t(R) \times S) - R]$$

where t is all attributes in R but not in S, and where attributes in S are a strict subset of attributes in R

Disadvantages of Relational Algebra

- It is a procedural language, i.e.: user has to specify how to get data they're looking for.

Solution: Non-procedural languages

1) Tuple Relational Calculus
2) QUEL
3) SQL

Final Relational Algebra Examples

Q1: Find SSN of all students who got an A in the class named "Simulations."

Student (name, sid, ssn, status) ✓
Faculty (name, fid, ssn, salary) ✗
Course (name, credits, dept, cnum, fid) ✓
Grade (sid, cnum, semester, grade) ✓
Enrolled (sid, cnum) ✗

Step 2 (join tables):

$$SGC = (Student \underset{sid}{\bowtie} Grade \underset{cnum}{\bowtie} Course)$$

Step 3 (narrow down rows, σ, then attributes, π):

$$\pi_{s.ssn}\left(\sigma_{c.name = \text{"Simulations"} \wedge g.grade = A}(SGC)\right)$$

Q2: The names of students that got an A from faculty named "Park"?

Student (name, sid, ssn, status) ✓
Faculty (name, fid, ssn, salary) ✓
Course (name, credits, dept, cnum, fid) ✓
Grade (sid, cnum, semester, grade) ✓
Enrolled (sid, cnum) ✗

Step 1: 

Step 2: Student × Grade (sid)
Faculty × Course (fid)
Grade × Course (cnum)

$$(\text{Student} \overset{}{\bowtie}_{\text{sid}} \text{Grade} \overset{}{\bowtie}_{\text{cnum}} \text{Course} \overset{}{\bowtie}_{\text{fid}} \text{Faculty})$$

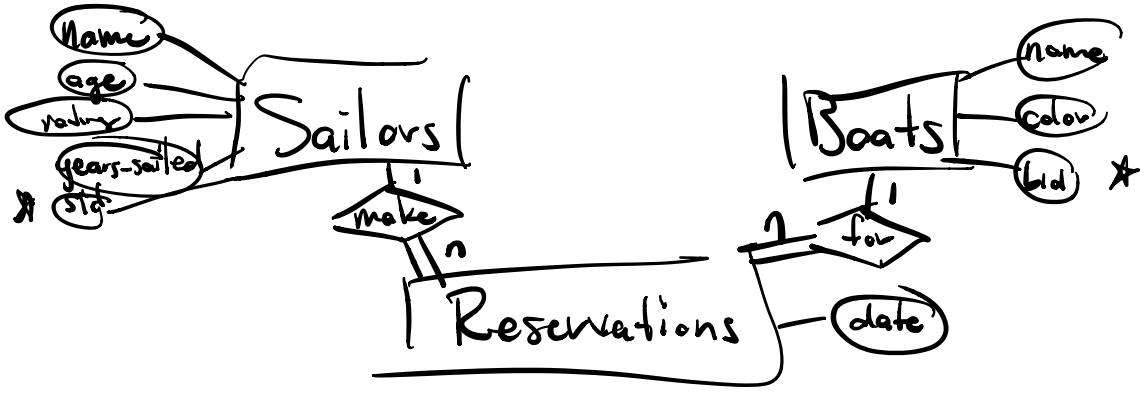
$$=$$

SGCF

Step 3:

$$\Pi_{\text{S.name}}\left(\sigma_{\text{f.name} = \text{Park} \wedge \text{grade.grade} = \text{A}}\left(\text{SGCF}\right)\right)$$

Schema for SQL Examples



Relational Schema:

sailors (name, age, rating, sid, years_sailed)
boats (bid, name, color)
reservations (sid, bid, date)

```
CREATE TABLE sailors (
    sid INT,
    name VARCHAR(40),
    rating INT,
    age INT,
    years_sailed INT,
    PRIMARY KEY (sid));

CREATE TABLE boats (
    bid INT,
    name VARCHAR(40),
    color VARCHAR(20),
    PRIMARY KEY (bid));
```

```
CREATE TABLE reservations (
    sid INT,
    bid INT,
    date DATE,
    PRIMARY KEY (sid, bid, date),
    FOREIGN KEY (sid) REFERENCES
        sailors (sid),
    FOREIGN KEY (bid) REFERENCES
        boats (bid));
```

SQL

Structured Query Language (SeQuel)

general form:
```
SELECT [DISTINCT] select_list
FROM from_list
WHERE qualification;
```

```
SELECT a1, a2, a3 ....
FROM r1, r2, r3 ....
WHERE p;
```

→ SELECT ≡ Projection (π) ★

$\pi_{a_1, a_2 \ldots}(\sigma_p(r_1 \times r_2 \times r_3 \ldots))$

Schema for examples:

Sailors (sid, name, rating, age, yrs_sailed)

boats (bid, name, color)

reservations (sid, bid, date)

Q1) What are the names of all the sailors?

$\pi_{name}(sailors)$ ⇒

```
SELECT name
FROM sailors;
```

⇓

```
SELECT s.name
FROM sailors s;
```