Structured Query Language
SQL Es-Que-El or Sequel

Didn't we already talk about SQL?

Two sublanguages

DDL Data Definition Language

define and modify schema (physical, logical, view)

CREATE TABLE, Integrity Constraints

DML Data Manipulation Language

get and modify data

simple SELECT, INSERT, DELETE

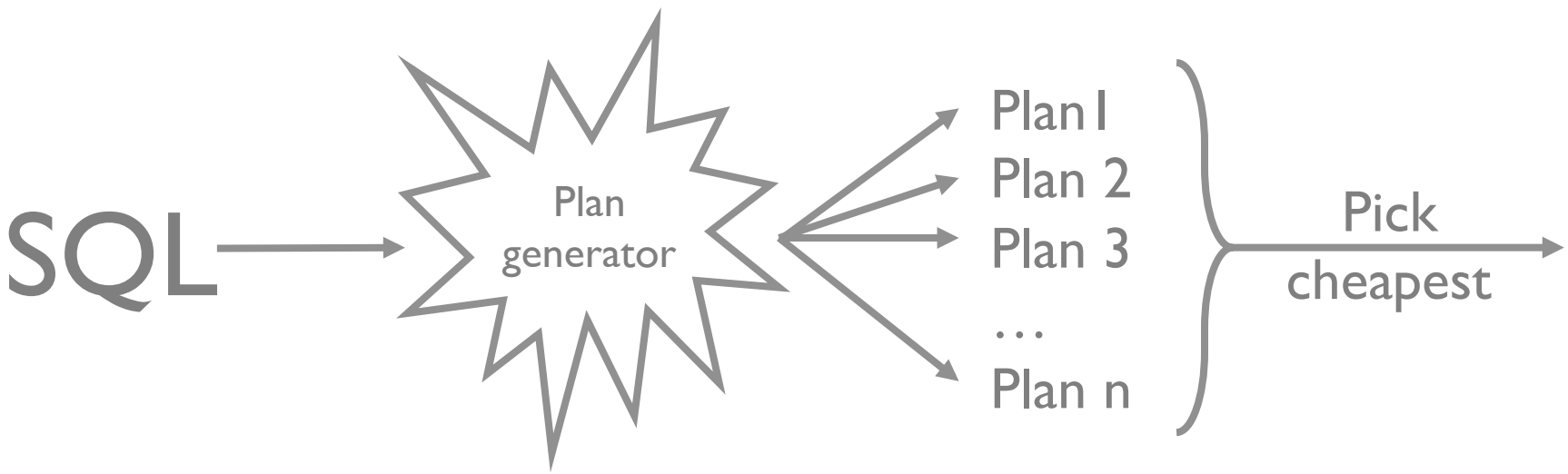
human-readable language

DBMS (tries to) execute efficiently

Key: precise query semantics

Reorder/modify queries while answers stay same

DBMS estimates costs for different evaluation plans



SQL: Extended Relational Algebra

Multisets rather than sets

- Relations can contain duplicates (unless constrained)
- Order doesn't matter

NULLs

Aggregates

Today's Database

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Boats

bid	name	color
101	Legacy	red
102	Melon	blue
103	Mars	red

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14

Is Reserves table correct?

Today's Database

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Boats

bid	name	color
101	Legacy	red
102	Melon	blue
103	Mars	red

Reserves

sid	bid	**day**
1	102	9/12
2	102	9/13
2	103	9/14

Is Reserves table correct?
Day should be part of key

Today's Database

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Boats

bid	name	color
101	Legacy	red
102	Melon	blue
103	Mars	red

Reserves

sid	bid	**day**
1	102	9/12
2	102	9/13
2	103	9/14
2	103	9/15

Is Reserves table correct?
Day should be part of key

<30 year old sailors

```
SELECT *
FROM Sailors
WHERE age < 30
```

sid	name	rating	age
1	Eugene	7	22
3	Ken	8	27

```
SELECT name, age
FROM Sailors
WHERE age < 30
```

name	age
Eugene	22
Ken	27

<30 year old sailors

```
SELECT *
FROM Sailors
WHERE age < 30
```

$\sigma_{age<30}$ (Sailors)

```
SELECT name, age
FROM Sailors
WHERE age < 30
```

$\pi_{name,\ age}$ ($\sigma_{age<30}$ (Sailors))

Who has reserved boat 102?

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14

Who has reserved boat 102?

```
SELECT S.name
FROM   Sailors AS S, Reserves AS R
WHERE  S.sid = R.sid AND R.bid = 102
```

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14

name
Eugene
Luis

Who reserved boat 102?

```
SELECT S.name
FROM   Sailors AS S, Reserves AS R
WHERE  S.sid = R.sid AND R.bid = 102
```

$$\pi_{name}\,(\sigma_{bid=2}(\text{Sailors} \bowtie_{sid} \text{Reserves}))$$

(equi-join)

Who has reserved boat 102?

```
SELECT S.name
FROM   Sailors AS S, Reserves AS R
WHERE  S.sid = R.sid AND R.bid = 102
```

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14
1	**102**	**9/15**

name
Eugene
Luis
Eugene

DISTINCT: unique rows / set

Reserves

sid	bid	day
1	102	9/12
2	102	9/13
2	103	9/14

```
SELECT  bid
FROM    Reserves
```

bid
102
102
103

```
SELECT  DISTINCT bid
FROM    Reserves
```

bid
102
103

Structure of a SQL Query

DISTINCT
Optional: Remove duplicates (set)
Default: duplicates permitted (multiset)

target-list
List of expressions over attrs of tables in relation-list

SELECT [DISTINCT] *target-list*
FROM *relation-list*
WHERE *qualification*

relation-list
List of relation names
Can define aliases "AS X"

qualification
Boolean expressions
- Combined w/ AND,OR,NOT
- attr *op* const
- $attr_1$ *op* $attr_2$
- op is =, <, >, !=, etc

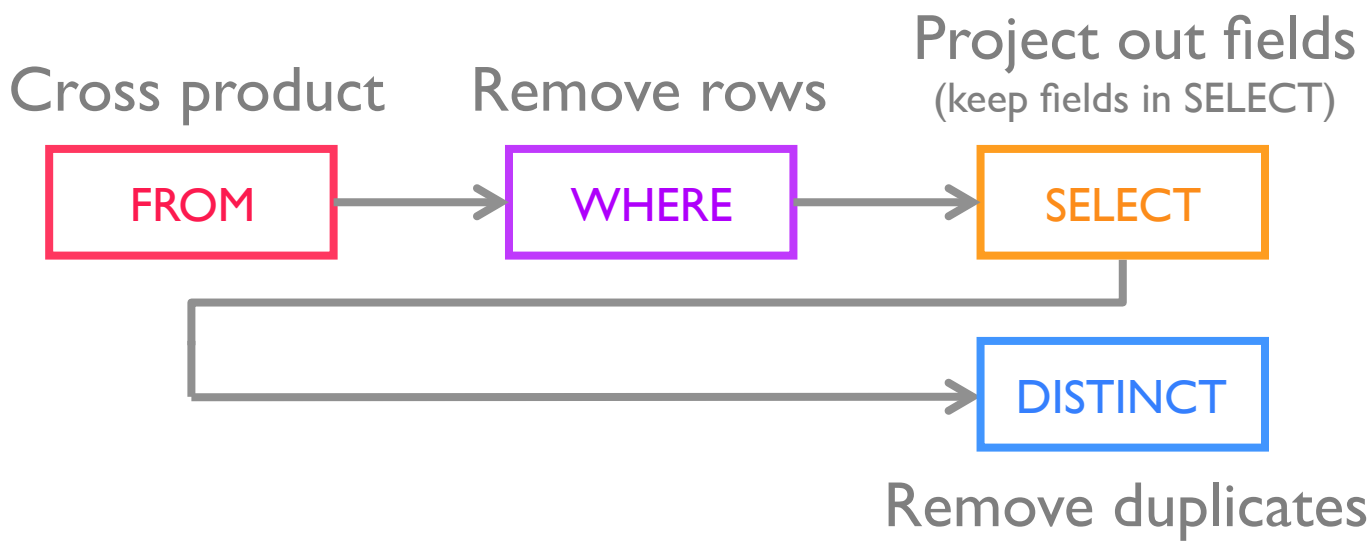
Semantics

SELECT	[DISTINCT] *target-list*
FROM	*relation-list*
WHERE	*qualification*

FROM	compute cross product of relations
WHERE	remove tuples that fail qualifications
SELECT	remove fields not in target-list
DISTINCT	remove duplicate rows

Conceptual Query Evaluation

SELECT	[DISTINCT] *target-list*
FROM	*relation-list*
WHERE	*qualification*
GROUP BY	*grouping-list*
HAVING	*group-qualification*



Not how actually executed! Above is likely very slow

Sailors that reserved 1+ boats

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

Would DISTINCT change anything in this query?

Sailors.sid is a primary key

What if SELECT clause was SELECT S.name?

Sailors that reserved 1+ boats

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

Table Alias (AS, Range Variables)

Disambiguate relations

same table used multiple times (self join)

```
SELECT   sid
FROM     Sailors, Sailors
WHERE    age > age
```

```
SELECT   S1.sid
FROM     Sailors AS S1, Sailors AS S2
WHERE    S1.age > S2.age
```

Table Alias (AS, Range Variables)

Disambiguate relations

same table used multiple times (self join)

```
SELECT   sid
FROM     Sailors, Sailors
WHERE    age > age
```

```
SELECT   S1.name, S1.age, S2.name, S2.age
FROM     Sailors AS S1, Sailors AS S2
WHERE    S1.age > S2.age
```

Expressions (Math)

```
SELECT   S.age, S.age – 5 AS age2, 2*S.age AS age3
FROM     Sailors AS S
WHERE    S.name = ‘eugene’
```

```
SELECT   S1.name AS name1, S2.name AS name2
FROM     Sailors AS S1, Sailors AS S2
WHERE    S1.rating*2 = S2.rating - 1
```

Expressions (Strings)

```
SELECT   S.name
FROM     Sailors AS S
WHERE    S.name LIKE ‘e_%’
```

Strings quoted with single quotes:‘ (identifiers: double quote)
If you need an embedded quote: use two: 'this is "quoted"'

‘_’ any one character (• in regex)

‘%’ 0 or more characters of any kind (•* in regex)

Most DBMSes have rich string manipulation support e.g., regex

PostgreSQL documentation

http://www.postgresql.org/docs/9.3/static/functions-string.html

Expressions (Date/Time)

```
SELECT   R.sid
FROM     Reserves AS R
WHERE    now() - R.date < interval '1 day'
```

TIMESTAMP, DATE, TIME types

Values quoted: '2016-02-16', 'Feb-16-2016', '4:05 PM'

now() returns timestamp at start of transaction

DBMSes provide rich time manipulation support

exact support may vary by vender

Postgresql Documentation

http://www.postgresql.org/docs/9.3/static/functions-datetime.html

Expressions

Constant	1, ‘hello’, 7.85
Col reference	Sailors.name
Arithmetic	Sailors.sid * 10
Unary operators	NOT
Binary operators	AND, OR, <, =, >=
Function	abs(), sqrt(), ...
Casting	1.7::int, ‘10-12-2015’::date

UNION, INTERSECT, EXCEPT

Algebra: $\cup$, $\cap$, -

Combine results from two queries:

SELECT [query1] UNION SELECT [query2]

By default: *distinct results*! (set semantics)
(operator) ALL: Keep duplicates: multi-set

sid of Sailors that reserved red or blue boat

```
SELECT   DISTINCT R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND
         (B.color = ‘red’ OR B.color = ‘blue’)
```

OR

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘red’
UNION
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red or blue boat

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND
         (B.color = ‘red’ OR B.color = ‘blue’)
```

OR

```
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘red’
UNION ALL
SELECT   R.sid
FROM     Boats B, Reserves R
WHERE    B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

```
SELECT    R.sid
FROM      Boats B, Reserves R
WHERE     B.bid = R.bid AND
          (B.color = ‘red’ AND B.color = ‘blue’)
```

```
SELECT    R.sid
FROM      Boats B, Reserves R
WHERE     B.bid = R.bid AND B.color = ‘red’
INTERSECT
SELECT    R.sid
FROM      Boats B, Reserves R
WHERE     B.bid = R.bid AND B.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R.sid
FROM     Boats B1, Reserves R1
WHERE
         B1.bid = R1.bid AND

         B1.color = ‘red’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   DISTINCT R.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE
         B1.bid = R1.bid AND

         B1.color = ‘red’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   R.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE
         B1.bid = R1.bid AND
         B2.bid = R2.bid AND
         B1.color = ‘red’ AND B2.color = ‘blue’
```

sid of Sailors that reserved red and blue boat

Can use self-join instead

```
SELECT   R.sid
FROM     Boats B1, Reserves R1, Boats B2,Reserves R2
WHERE    R1.sid = R2.sid AND
         B1.bid = R1.bid AND
         B2.bid = R2.bid AND
         B1.color = ‘red’ AND B2.color = ‘blue’
```

sids of sailors that haven't reserved a boat

```
SELECT   S.sid
FROM     Sailors S

EXCEPT

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

Nested Queries

```
SELECT   S.sid
FROM     Sailors S
WHERE    S.sid IN (SELECT    R.sid
                   FROM      Reserves R
                   WHERE     R.bid = 101)
```

Many clauses can contain SQL queries

WHERE, FROM, HAVING, SELECT

Conceptual model:

```
for each Sailors tuple
   run the subquery and evaluate qualification
```

Nested Query vs Join

```
SELECT   S.sid
FROM     Sailors S
WHERE    S.sid IN (SELECT    R.sid
                   FROM      Reserves R
                   WHERE     R.bid = 101)
```

```
SELECT   S.sid
FROM     Sailors S, Reserves R
WHERE    S.sid = R.sid AND R.bid = 101
```

What if a student reserved a boat more than once?

Nested: No duplicates **Join: Duplicates**

SET Comparison Operators

x IN *r*: True if value *x* appears in *r*

EXISTS *r*: True if relation *r* is not empty (NOT EXISTS)

x (operator) ANY r: True if x (operator) is true for any row in r

E.g. x IN r is equivalent to x = ANY r

x (operator) ALL r: True if x (operator) is true for all rows in r

E.g. x NOT IN r is equivalent to x <> ALL r

Reference outer table in nested query

```
SELECT   S.sid
FROM     Sailors S
WHERE    EXISTS (SELECT   *
                 FROM     Reserves R
                 WHERE    R.bid = 101 AND
                          S.sid = R.sid)
```

Outer table referenced in nested query

Conceptual model:

```
for each Sailors tuple
   run the subquery and evaluate qualification
```

Sailors whose rating is greater than any sailor named “Bobby”

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ANY (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = ‘Bobby’)
```

How are these different?

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ANY (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = 'Bobby')
```

```
SELECT S1.name
FROM   Sailors S1
WHERE  S1.rating > ALL (SELECT   S2.rating
                        FROM     Sailors S2
                        WHERE    S2.name = 'Bobby')
```

Rewrite INTERSECT using IN

```
SELECT  S.sid
FROM    Sailors S
WHERE   S.rating > 2

INTERSECT

SELECT  R.sid
FROM    Reserves R
```

```
SELECT  S.sid
FROM    Sailors S
WHERE   S.rating > 2 AND
        S.sid IN (
    SELECT  R.sid
    FROM    Reserves R
)
```

Similar trick for EXCEPT → NOT IN

What if want *names* instead of sids?

Names are not unique!