ICS 321 Spring 2012
The Database Language SQL (i)

Asst. Prof. Lipyeow Lim

Information & Computer Science Department

University of Hawaii at Manoa

Example Relations

- Sailors(
    sid: integer,
    sname: string,
    rating: integer,
    age: real)
- Boats(
    bid: integer,
    bname: string,
    color: string)
- Reserves(
    sid: integer,
    bid: string,
    day: date)

R1

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

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

B1

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	green
104	Marine	Red

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!

Example Q1

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

Without range variables

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

- Range variables really needed only if the same relation appears twice in the FROM clause.

- Good style to always use range variables

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following *conceptual* evaluation strategy:
    1. Compute the cross-product of *relation-list*.
    2. Discard resulting tuples if they fail *qualifications*.
    3. Delete attributes that are not in *target-list*.
    4. 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 Q1: conceptual evaluation

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

Conceptual Evaluation Steps:
1. Compute cross-product
2. Discard disqualified tuples
3. Delete unwanted attributes
4. If DISTINCT is specified, eliminate duplicate rows.

S.sid	sname	rating	age	R.sid	bid	day
22	Dustin	7	45	22	101	10/10/96
22	Dustin	7	45	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

S.sid	sname	rating	age	R.sid	bid	day
58	Rusty	10	35.0	58	103	11/12/96

sname
Rusty

Q2: Find sailors who've reserved at least one boat

R1

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

```
SELECT S1.sid
FROM    Sailors S1, Reserves R1
WHERE  S1.sid=R1.sid
```

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

- Would adding DISTINCT to this query make a difference?
- What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause? Would adding DISTINCT to this variant of the query make a difference?

Q3: Find the colors of boats reserved by Lubber

SELECT B1.color
FROM Sailors S1, Reserves R1, Boats B1
WHERE S1.sid=R1.sid
AND R1.bid=B1.bid
AND S1.sname='Lubber'

R1

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

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

B1

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	green
104	Marine	Red

Expressions

- WHERE-qualification can contain expressions
- SELECT-list can also contain arithmetic or string expressions over the column names
- Example: compute a new ``age adjusted'' rating for each sailor whose rating satisfies a special formula

```
SELECT  S1.sname,
        S1.rating * S1.age / 100
              AS NewRating
FROM    Sailors S1
WHERE   S1.rating – 5.0 > S1.age /
12.0
```

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

NULLs

SELECT S1.sname,
FROM   Sailors S1
WHERE S1.rating – 5.0 > 0

S1

sid	sname	rating	age
22	Dustin	NULL	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

- The result of any arithmetic operator +,-,/,× involving a NULL is always NULL

- The result of any comparison operator like =,>,< is always UNKNOWN

The "UNKNOWN" truth-value

X	Y	X AND Y	X OR Y	NOT X
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

- If TRUE = 1, False = 0, UNKNOWN=0.5
  - AND : min, OR : max, NOT : 1-v

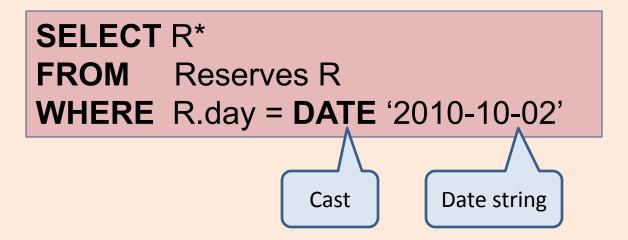# Strings & Pattern Matching

- String comparisons via the comparisons operators (<, >, =, etc), but take note of collations
  - i.e. determines the ordering. Lexicographic, languages etc
- SQL supports pattern matching via the **LIKE** operator and wildcards
  - ``%'' : zero or more arbitrary chars
  - ``_'' : any one char

SELECT S1.sname, S1.rating
FROM   Sailors S1
WHERE S1.sname **LIKE** `L_%'

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Date, Time, Timestamp

SELECT R*
FROM   Reserves R
WHERE R.day = **DATE** '2010-10-02'

Cast

Date string

- Dates and time constants are specified using strings and "cast" into the date/time datatypes using functions.

TIME '15:00:02.5'
TIMESTAMP '2010-10-02 15:00:02'

Ordering the Output

SELECT S1.sname, S1.rating
FROM   Sailors S1
ORDER BY S1.rating **DESC**

- ORDER BY clause sorts the result of the SQL query according to the given column(s).

S1

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

sname	rating
Rusty	10
Lubber	8
Dustin	7