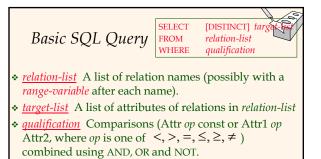
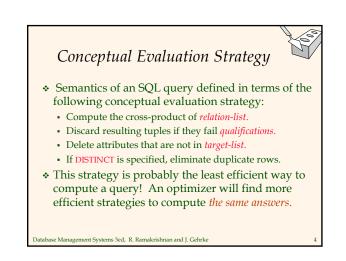


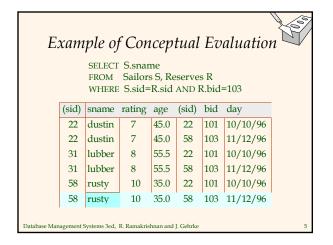
Example Instanc	<u>sid</u> 22 58	<u>bid</u> 101 103	10/1	ay 0/96 2/96	0	
<ul> <li>instances of the Sailors and Reserves relations in our examples.</li> <li>If the key for the Reserves relation S2 contained only the attributes <i>sid</i> and <i>bid</i>, how would the semantics differ?</li> </ul>	<u>sid</u> 22 31 58 <u>sid</u> 28 31 44 58	snan dust lubb rusty snan yupp lubb gupp rusty	in er me ra oy ber oy	ting 7 8 10 ting 9 8 5 10	age 45.0 55.5 35.0 age 35.0 55.5 35.0 35.0 35.0	
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke 2						

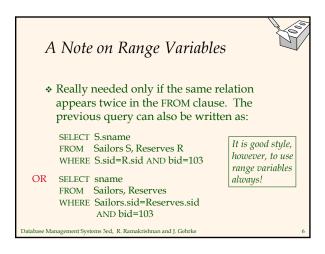


 DISTINCT is an optional keyword indicating that the answer should not contain duplicates. Default is that duplicates are <u>not</u> eliminated!

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke







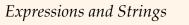
#### Find sailors who've reserved at least one boat

No series and series a

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

- 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?

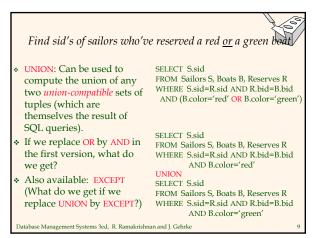
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

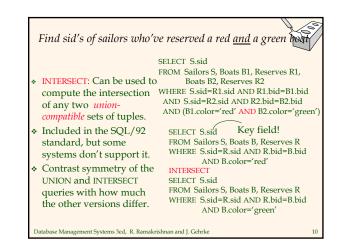


SELECT S.age, age1=S.age-5, 2\*S.age AS age2 FROM Sailors S WHERE S.sname LIKE 'B %B'

- Illustrates use of arithmetic expressions and string pattern matching: Find triples (of ages of sailors and two fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.
- \* AS and = are two ways to name fields in result.
- LIKE is used for string matching. `\_' stands for any one character and `%' stands for 0 or more arbitrary characters.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



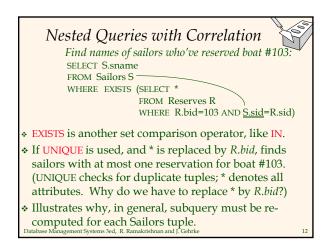


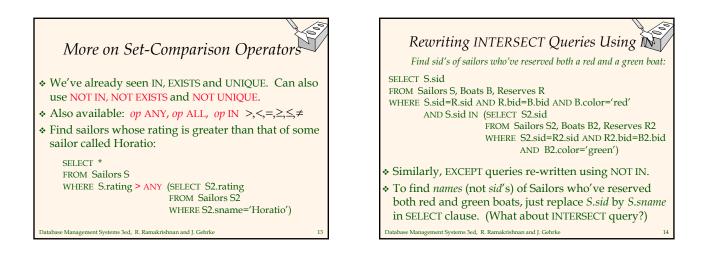
#### Nested Queries

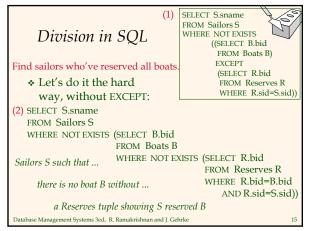


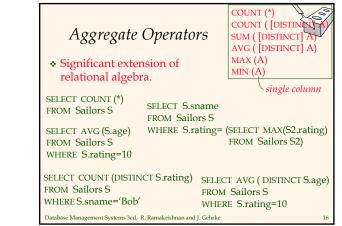
Find names of sailors who've reserved boat #103 SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R WHERE R.bid=103)

- A very powerful feature of SQL: a WHERE clause can itself contain an SQL query! (Actually, so can FROM and HAVING clauses.)
- ✤ To find sailors who've not reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a <u>nested loops</u> evaluation: For each Sailors tuple, check the qualification by computing the subquery. Database Management Systems 3ed, R. Ramakrishnan and J. Cehrke









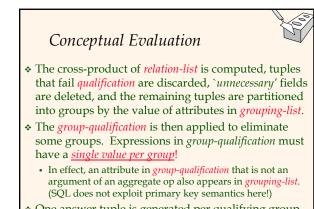


### Queries With GROUP BY and HAVING

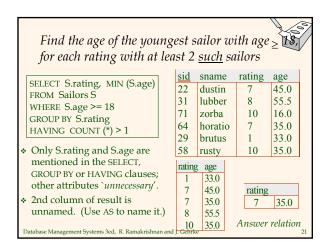
SELECT	[DISTINCT] target-list
FROM	relation-list
WHERE	qualification
GROUP BY	grouping-list
HAVING	group-qualification

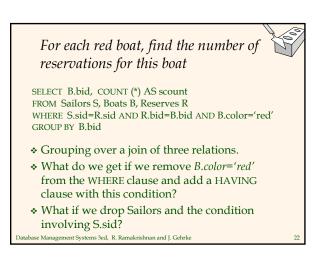
- The target-list contains (i) attribute names (ii) terms with aggregate operations (e.g., MIN (S.age)).
  - The <u>attribute list (i)</u> must be a subset of *grouping-list*. Intuitively, each answer tuple corresponds to a *group*, and these attributes must have a single value per group. (A *group* is a set of tuples that have the same value for all attributes in *grouping-list*.)

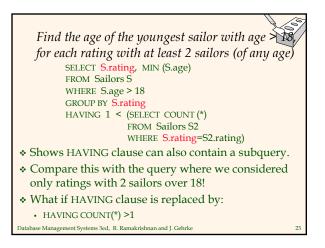
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

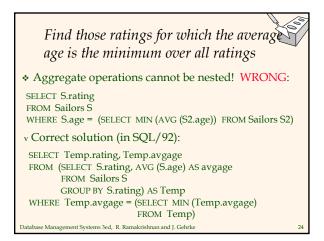


One answer tuple is generated per qualifying group. Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke









# Null Values

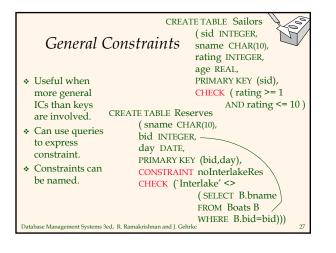


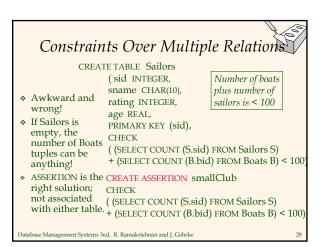
- Field values in a tuple are sometimes *unknown* (e.g., a rating has not been assigned) or *inapplicable* (e.g., no spouse's name).
  - SQL provides a special value <u>null</u> for such situations.
- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not null.
  - Is *rating*>8 true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
  - We need a <u>3-valued logic</u> (true, false and *unknown*).
  - Meaning of constructs must be defined carefully. (e.g., WHERE clause eliminates rows that don't evaluate to true.)
    New operators (in particular, *outer joins*) possible/needed.
- Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

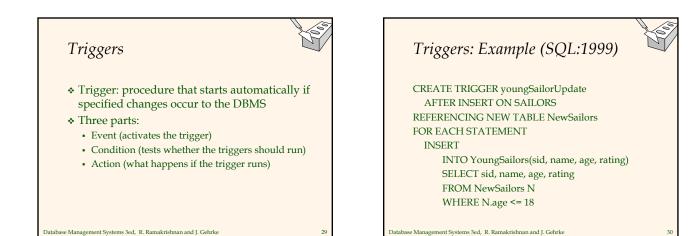
# Integrity Constraints (Review)

- An IC describes conditions that every *legal instance* of a relation must satisfy.
  - Inserts/deletes/updates that violate IC's are disallowed.
  - Can be used to ensure application semantics (e.g., *sid* is a key), or prevent inconsistencies (e.g., *sname* has to be a string, *age* must be < 200)</li>
- <u>Types of IC's</u>: Domain constraints, primary key constraints, foreign key constraints, general constraints.
  - *Domain constraints*: Field values must be of right type. Always enforced.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke







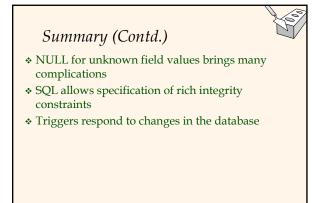
# Summary



31

- SQL was an important factor in the early acceptance of the relational model; more natural than earlier, procedural query languages.
- Relationally complete; in fact, significantly more expressive power than relational algebra.
- Even queries that can be expressed in RA can often be expressed more naturally in SQL.
- Many alternative ways to write a query; optimizer should look for most efficient evaluation plan.
  - In practice, users need to be aware of how queries are optimized and evaluated for best results.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke