
Let's start

SCC.201

Database Management Systems

2023 – (CONT.) Week 4 – Relational Algebra – Schema
Refinement

Uraz C Turker & Ricki Boswell

Previously:

- Multivalued attribute:

-
-
-
-
-
-
-
-
-
-

Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 -
 -
 -
 -
- -
 -
 -
 -

Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- Prime attribute:
 - A part of a (composite) key.
-
-

Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- Prime attribute:
 - A part of a (composite) key.
- Non-prime attribute:
 - An attribute that is not a part of any keys.

<u>Supplier</u>	<u>Part#</u>	Location	Stock
-----------------	--------------	----------	-------

Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- Prime attribute:
 - A part of a (composite) key.
- Non-prime attribute:
 - An attribute that is not a part of any keys.



Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 - Prime attribute:
 - A part of a (composite) key.
 - Non-prime attribute:
 - An attribute that is not a part of any keys.
- 1st Normal Form
 - No multivalued attributes
 -
 -
 -
 -



Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 - Prime attribute:
 - A part of a (composite) key.
 - Non-prime attribute:
 - An attribute that is not a part of any keys.
- 1st Normal Form
 - No multivalued attributes
 - 2nd Normal Form
 -
 -
 -



Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- **Prime attribute:**
 - A part of a (composite) key.
- **Non-prime attribute:**
 - An attribute that is not a part of any keys.
- 1st Normal Form
 - No multivalued attributes
- 2nd Normal Form
 - In 1st Normal form, and no part of a key determines a non-prime attribute.



Previously:

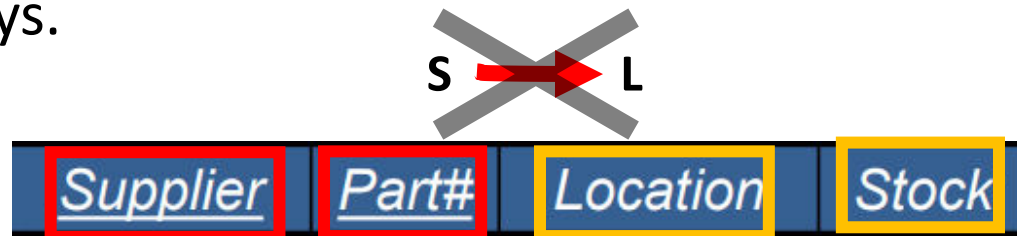
- **Multivalued attribute:**
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 - **Prime attribute:**
 - A part of a (composite) key.
 - **Non-prime attribute:**
 - An attribute that is not a part of any keys.
- 1st Normal Form
 - No multivalued attributes
 - 2nd Normal Form
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
 -
 -

S → L



Previously:

- **Multivalued attribute:**
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- **Prime attribute:**
 - A part of a (composite) key.
- **Non-prime attribute:**
 - An attribute that is not a part of any keys.



- **1st Normal Form**
 - No multivalued attributes
- **2nd Normal Form**
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
-
-

Previously:

- **Multivalued attribute:**
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 - **Prime attribute:**
 - A part of a (composite) key.
 - **Non-prime attribute:**
 - An attribute that is not a part of any keys.
- **1st Normal Form**
 - No multivalued attributes
 - **2nd Normal Form**
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
 - **3rd Normal Form**
 -



Previously:

- **Multivalued attribute:**
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
 - **Prime attribute:**
 - A part of a (composite) key.
 - **Non-prime attribute:**
 - An attribute that is not a part of any keys.
- **1st Normal Form**
 - No multivalued attributes
 - **2nd Normal Form**
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
 - **3rd Normal Form**
 - In 2nd Normal form, and no non-prime attribute determines a non-prime attribute (only keys can determine!).



Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- Prime attribute:
 - A part of a (composite) key.
- Non-prime attribute:
 - An attribute that is not a part of any keys.

L → St



- 1st Normal Form
 - No multivalued attributes
- 2nd Normal Form
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
- 3rd Normal Form
 - In 2nd Normal form, and no non-prime attribute determines a non-prime attribute (only keys can determine!).

Previously:

- Multivalued attribute:
 - An attribute that can hold a set of values, i.e. a set of phone numbers.
- Prime attribute:
 - A part of a (composite) key.
- Non-prime attribute:
 - An attribute that is not a part of any keys.



- 1st Normal Form
 - No multivalued attributes
- 2nd Normal Form
 - In 1st Normal form, and no part of a key determines a non-prime attribute.
- 3rd Normal Form
 - In 2nd Normal form, and no non-prime attribute determines a non-prime attribute (only keys can determine!).

Boyce-Codd Normal Form (BCNF)

- Relation R with FDs F is in BCNF if, for all $X \rightarrow A$ in F^+
 - $A \in X$ (called a *trivial* FD), or
 - X contains a key for R. (i.e., X is a superkey)
- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
 - No dependency in R that can be predicted using FDs alone.
 - If example relation is in BCNF and X is a key, the 2 tuples must be identical (since X is a key).
 - To check we must know all keys.

X	Y	A
x	y1	a
x	y2	?

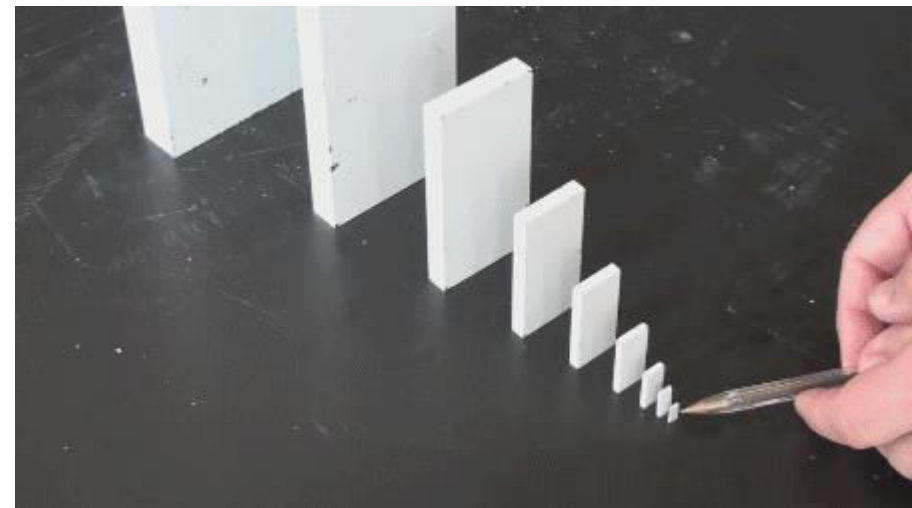
F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$.
 -
 -
 -
 -
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$.
 - Whenever we are to optimise a relation, we have to **analyse** the FDs to reveal **hidden** dependencies.

-
-
-
-
-
-
-
-
-
-



F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$.
 - Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 -
 -
 -
 -
 -
 -
 -
 -
 -
 -



F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$.
 - Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 -
 -
 -
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$
 - Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 -
 -
 -
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$.
 - Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 -
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$.
 - Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 -
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$.
 - When ever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 - If $BC \rightarrow DEFG$ then $ABC \rightarrow ADEFG$?
 -
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$.
 - When ever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 - If $BC \rightarrow DEFG$ then $ABC \rightarrow ADEFG$?
 - If $ABC \rightarrow ADEFG$ then $ABBC$ \rightarrow $ABCDEF$?
 -
 -
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$.
 - When ever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 - If $BC \rightarrow DEFG$ then $ABC \rightarrow ADEFG$?
 - If $ABC \rightarrow ADEFG$ then $ABBCC \rightarrow ABCDEFG$?
 - So $ABC \rightarrow ABCDEFG$?
 - So **ABC** is a key... What else?
 -
 -

F^+ what is going on here?

-
- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
 - Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$, $ABC \rightarrow ABCDEFG$.
 - When ever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
 - This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 - If $BC \rightarrow DEFG$ then $ABC \rightarrow ADEFG$?
 - If $ABC \rightarrow ADEFG$ then $ABBCC \rightarrow ABCDEFG$?
 - So $ABC \rightarrow ABCDEFG$?
 - So ABC is a key... What else?
 -
 -

F^+ what is going on here?

- Let us assume that we are given a relation $R=\{A,B,C,D,E,F,G\}$
- Also let us assume that we are given a set of FDs
 - $A \rightarrow B$, $B \rightarrow C$, $BC \rightarrow D$, $D \rightarrow EFG$, $A \rightarrow C$, $BC \rightarrow DEFG$, $ABC \rightarrow ABCDEFG$, $A \rightarrow ABCDEFG$.
- When ever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done intuitively:
 - In the above $A \rightarrow B$, $B \rightarrow C$ this implicitly means that $A \rightarrow C$?
 - If $BC \rightarrow D$ and if $D \rightarrow EFG$ then $BC \rightarrow DEFG$?
 - If $BC \rightarrow DEFG$ then $ABC \rightarrow ADEFG$?
 - If $ABC \rightarrow ADEFG$ then $ABBCC \rightarrow ABCDEFG$?
 - So $ABC \rightarrow ABCDEFG$?
 - So ABC is a key... What else?
 - $A \rightarrow B$ and $A \rightarrow C$ so $A \rightarrow ABC$
 - So $A \rightarrow ABCDEFG$ and A is a key.



Reasoning About FDs

- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = *closure of F* is the set of all FDs that are implied by F .
 -
 -
 -
 -

Reasoning About FDs

- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = *closure of F* is the set of all FDs that are implied by F .
- Armstrong's Axioms (X, Y, Z are sets of attributes):
 -
 -
 -
-

Reasoning About FDs

- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = closure of F is the set of all FDs that are implied by F .
- Armstrong's Axioms (X, Y, Z are sets of attributes):
 - Reflexivity: If $X \in Y$, then $Y \rightarrow X$ (a trivial FD) "IF X is IN Y" comment 😊
 -
 -
-

Reasoning About FDs

-
- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = closure of F is the set of all FDs that are implied by F .
 - Armstrong's Axioms (X, Y, Z are sets of attributes):
 - Reflexivity: If $X \in Y$, then $Y \rightarrow X$ (a trivial FD) "IF X is IN Y" comment ☺
 - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
 -
 -

Reasoning About FDs

-
- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = closure of F is the set of all FDs that are implied by F .
 - Armstrong's Axioms (X, Y, Z are sets of attributes):
 - Reflexivity: If $X \in Y$, then $Y \rightarrow X$ (a trivial FD) "IF X is IN Y" comment 😊
 - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow Z$ for any Z
 - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
 -
 -
 - These are *sound* and *complete* inference rules for FDs!

Reasoning About FDs

-
- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = closure of F is the set of all FDs that are implied by F .
 - Armstrong's Axioms (X, Y, Z are sets of attributes):
 - Reflexivity: If $X \in Y$, then $Y \rightarrow X$ (a trivial FD) "IF X is IN Y" comment 😊
 - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow Z$ for any Z
 - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
 - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
 -
 - These are *sound* and *complete* inference rules for FDs!

Reasoning About FDs

-
- A FD f is implied by a set of FDs F if f holds whenever all FDs in F hold.
 - F^+ = closure of F is the set of all FDs that are implied by F .
 - Armstrong's Axioms (X, Y, Z are sets of attributes):
 - Reflexivity: If $X \in Y$, then $Y \rightarrow X$ (a trivial FD) "IF X is IN Y" comment 😊
 - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow Z$ for any Z
 - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
 - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
 - Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
 - These are *sound* and *complete* inference rules for FDs!

Reasoning About FDs

For example, in the above schema

$S N \rightarrow S$ is a trivial FD
since $\{S, N\}$ is a superset of $\{S\}$

S	N	L	R	W	H

Reasoning About FDs

S	N	L	R	W	H

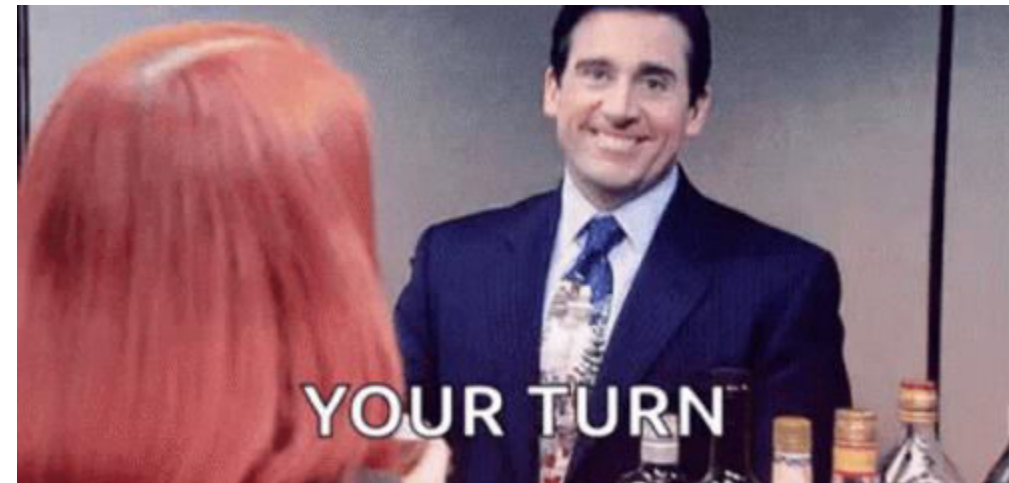
For example, in the given schema

If $S N \rightarrow R W$, then $S N L \rightarrow R W L$ (by augmentation)

If $S \rightarrow R$ and $R \rightarrow W$, then $S \rightarrow W$ (by transitivity)

Reasoning About FDs

-
- Example: $Contracts(cid, sid, jid, did, pid, qty, value)$, and given FDs:
 - C is the key: $C \rightarrow CSJDPQV$
 - Project purchases each part using single contract: $JP \rightarrow C$
 - Dept purchases at most one part from a supplier: $SD \rightarrow P$ then **prove that SDJ is a KEY:**



Normalisation (3 STEPS)

For a given FD F we have to find all possible Keys using F^+ Closure.

Normalisation (3 STEPS)

For a given FD F we have to find all possible Keys using F^+ Closure.

And then using the Normalisation rules we decide on the Normalisation level.

Normalisation (3 STEPS)

For a given FD F we have to find all possible Keys using F^+ Closure.

And then using the Normalisation rules we decide on the Normalisation level.

If required level is not satisfied we decompose the relation according to the FDs that prevents required Normal Form

Normal Forms Contd.

- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E \}$
 - What is the maximum Normalisation level?
 -
 -
 -
 -

Normal Forms Contd.

- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E \}$
 - What is the maximum Normalisation level?
- $AC \rightarrow BE \dots \dots AC \rightarrow B, AC \rightarrow E$ (decomposition)
-
-
-

Normal Forms Contd.

- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E \}$
 - What is the maximum Normalisation level?
- $AC \rightarrow BE \dots \dots AC \rightarrow B, AC \rightarrow E$ (decomposition)
-
-
-

Normal Forms Contd.

- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E \}$
 - What is the maximum Normalisation level?
- $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
- $ACC \rightarrow BC \dots AC \rightarrow BC$ (Augmentation)
-
-

Normal Forms Contd.

- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC \}$
 - What is the maximum Normalisation level?
- $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
- $ACC \rightarrow BC \dots AC \rightarrow BC$ (Augmentation)
-
-

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC \}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$ (Transitivity)
 -

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D \}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$ (Transitivity)
 -

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D \}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, **by union and augmentation**
 $ABC \rightarrow ABCDE$.

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{ BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D \}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are **PRIME ATTRIBUTES** & $\{B, D, E\}$ are **NONPRIME ATTRIBUTES**.

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Observation1: **No single prime attribute determines a nonprime attribute.**

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Observation1: **No single prime attribute determines a nonprime attribute, so in 2NF.**

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $AC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Observation2: **B determines E, so Relation is not in 3NF.**

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Observation3: **B determines E, so Relation is not in 3NF and so not in BCNF.**

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Answer is 2NF.

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Solution:

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $ACC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $ABC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.
- Solution:** As $B \rightarrow E$ nonprime attribute implies nonprime attribute. We decompose $ABCDE$ to $ABCD$, and BE .

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?

- $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$

- $AC \rightarrow BC \dots AC \rightarrow BC$

- $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$

- AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $AC \rightarrow ABCDE$.

- $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.

Solution: However we have FD $BC \rightarrow D$. BC is not a key. So we will decompose ABCD to ABC and BCD.

Normal Forms Contd.

-
- Ex: $R = ABCDE$, $F = \{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E, AC \rightarrow B, AC \rightarrow E, AC \rightarrow BC, AC \rightarrow D\}$
 - What is the maximum Normalisation level?
 - $AC \rightarrow BE \dots AC \rightarrow B, AC \rightarrow E$
 - $AC \rightarrow BC \dots AC \rightarrow BC$
 - $BC \rightarrow D \dots AC \rightarrow BC \rightarrow D \dots AC \rightarrow D$
 - AC is the candidate key as $AC \rightarrow BE, AC \rightarrow D$, by union and augmentation $AC \rightarrow ABCDE$.
 - $\{A, C\}$ are PRIME ATTRIBUTES & $\{B, D, E\}$ are NONPRIME ATTRIBUTES.
- Solution:** Finally we have three tables ABC, BE, and BCD. This is in 3NF, and BCNF.

Summary

- Relational algebra and Normal forms are quite important apparatuses to optimise
 - Database (Normal forms)
 - Querying (Relational algebra)
- We have seen four normal forms
 - 1st, 2nd, 3rd, and Boyce-Codd normal forms.
- Each Normalisation level guarantees a level of non-redundancy.
- Decomposition is done on FDs.

Let's start

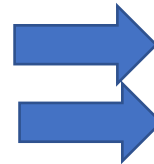
SCC.201

Database Management Systems

2023 - Week 5 – SQL and JDBC
Uraz C Turker & Ricki Boswell

What will you learn today?

- Advanced SQL queries
- Connection with JDBC.



Curriculum Design: Outline Syllabus

This module builds upon knowledge gained in Part I by providing a theoretical background to the design, implementation and use of database management systems, both for data designers and application developers. It takes into account all relevant aspects related to information security in the design, development and use of database systems. The course consists of a number of related sections, which range from single lectures to multi-lecture streams, depending on the required depth of coverage. The sections are as follows.

Introduction : we begin with a brief history of how the need for database management systems (DBMS) grew over time and how they are applied in day to day scenarios.

Database Design: before making use of a DBMS, we must capture our requirements : what data do we actually wish to model? We make use of the Extended Entity-Relationship (EER) model which is both a technique and a notation for designing the data in a DBMS independent way.

The Relational Model: now the de-facto standard for DBMS, this was a revolutionary step taken in 1970. We extensively examine the Model, looking at relational database systems, the model itself and the normalisation process, the relational algebra (the mathematical theory that underpins the model), the three schema architecture and schema definition in SQL. Finally, we look at how we can map the EER model into an equivalent Relational Model. The resultant database is then examined in terms of access rights and privileges.

A (re)Introduction to SQL: SQL is the de-facto standard for DBMS query languages. We look at both the DDL (data definition language) and DML (data manipulation language). We introduce the use of views, a powerful mechanism for providing privacy and security. We look at the Discretionary Access Control (DAC) features that allow the granting and withholding of access rights and privileges.

Accessing relational DBMS via Java: we explore the facilities of the JDBC and show how we can write applications in Java which connect with a relational DBMS (in practice, MySQL).

The Physical Model: as Computer Scientists, our students need an awareness of the techniques that allow rapid access to stored data. In this section, we examine the physical data organisation and associated access methods. We show under what circumstances the organisations can be applied, and we look at how queries can be optimised.

Transaction processing and concurrency control: a huge part of DBMS in practice is the need to support transactions and concurrency, allowing huge numbers of users to access the DBMS at any one time while still ensuring the consistency of the data. This stream examines the problems and solutions in depth.

Lecture 1	Introduction to the module, Why do we need Databases? Entity Relationship Model
Lecture 2	Entity Relationship Model (ERM) (cont.)
Lab	A gentle start to the ER diagrams.
Lecture 1	Relational Model (RM)
Lecture 2	ER to RM
Lab	ER diagrams.
Lecture 1	Relational Model To SQL & SQL scripting
Lecture 2	Review
Lab	ER to Relational Model.
Lecture 1	Relational Algebra
Lecture 2	Functional Dependencies + 1st, 2nd Normal Forms
Lab	Relational Algebra + SQL + Relational Model To SQL.
Lecture 1	3dr and Boycott normal forms. Advanced SQL queries.
Lecture 2	JDBC
Lab	Functional dependencies and Normalisation.
Lecture 1	Physical Storage - record files
Lecture 2	Storage - secondary files
Lab	Functional dependencies and Normalisation + JDBC Example.
Lecture 1	Record Search - B-Trees
Lecture 2	Search - Hashing
Lab	Working on project
Lecture 1	Access Routines
Lecture 2	Query Optimisation
Lab	Project Grade Phase 1
Lecture 1	Concurrency - Transaction Processing
Lecture 2	Locking
Lab	Project Grade Phase 2
Lecture 1	Advanced SQL - schemas, views, access control
Lecture 2	Review and recap?
Lab	Project Grade Phase 3

Find sid's of sailors who've reserved a red and a green boat

- **AND** : Used to compute the set intersection of two *union-compatible* sets of tuples

B

<u>BID</u>	Colr
b1	red
b2	grn

R

SID	<u>BID</u>
s1	b1
s1	b2
s2	b1

B1 X R1

BID	Colr	SID	BID
b1	red	s1	b1
b1	red	s1	b2
b1	red	s2	b1
b2	grn	s1	b1
b2	grn	s1	b2
b2	grn	s2	b1

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid
AND (B.color='red' AND B.color='green')
```



Find sid's of sailors who've reserved a red and a green boat

$$\rho(Tmp1, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie Reserves))$$

$$\rho(Tmp2, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie Reserves))$$

$Tmp1 \cap Tmp2$

B1

<u>BID</u>	Colr
b1	red
b2	grn

R1

SID	<u>BID</u>
s1	b1
s1	b2
s2	b1

B2

<u>BID</u>	Colr
b1	red
b2	grn

R2

SID	<u>BID</u>
s1	b1
s1	b2
s2	b1

B1 X R1

BID	Colr	SID	BID
b1	red	s1	b1
b1	red	s1	b2
b1	red	s2	b1
b2	grn	s1	b1
b2	grn	s1	b2
b2	grn	s2	b1

B2 X R2

BID	Colr	SID	BID
b1	red	s1	b1
b1	red	s1	b2
b1	red	s2	b1
b2	grn	s1	b1
b2	grn	s1	b2
b2	grn	s2	b1

Tmp1

s1
s2

Tmp2



Find sid's of sailors who've reserved a red and a green boat



-
- **INTERSECT**: Can be used to compute the intersection of any two *union-compatible* sets of tuples.
 - Included in the SQL/92 standard, but some systems don't support it.
 - Contrast symmetry of the UNION and INTERSECT queries with how much the other versions differ.

```
SELECT R.sid
FROM Boats B1, Reserves R1,
     Boats B2, Reserves R2
WHERE R1.sid = R2.sid AND
     R1.bid=B1.bid AND R2.bid=B2.bid
     AND (B1.color='red' AND B2.color='green')
```

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid
     AND B.color='red'
INTERSECT
SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid=B.bid
     AND B.color='green'
```

Nested Queries

Find names of sailors who've reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid (ONLY ONE Colum)
                FROM Reserves R
                WHERE R.bid=103)
```

- WHERE clause can itself contain an SQL query! (As well as FROM and HAVING clauses which we will see later on.)

Nested Queries

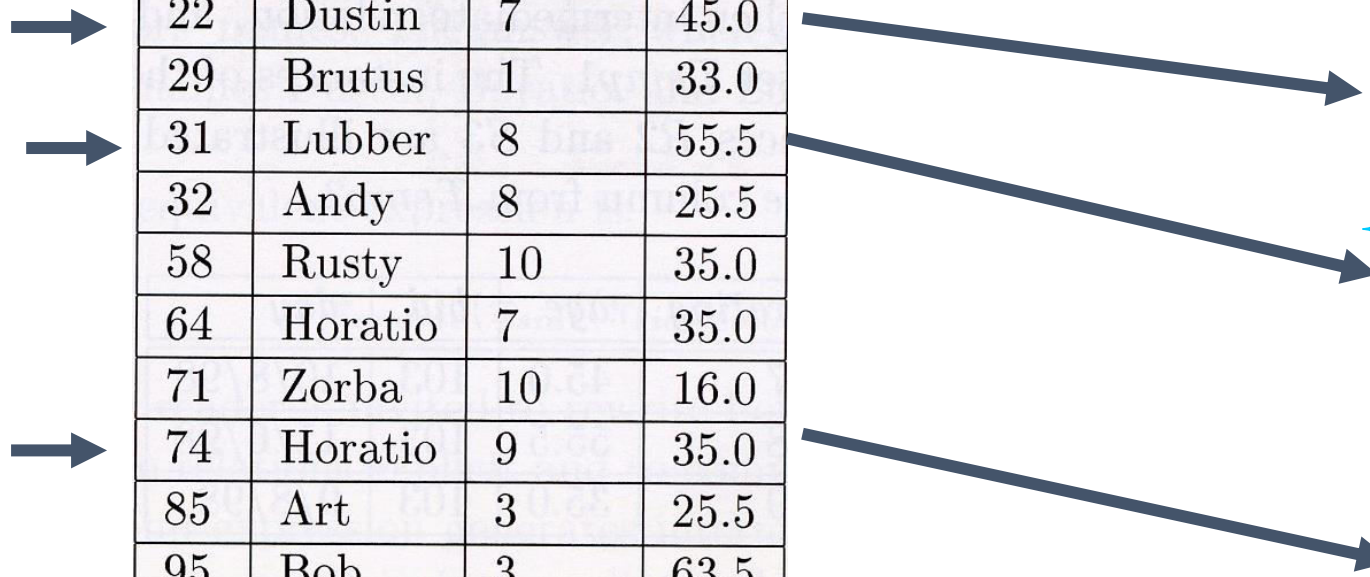
```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=103)
```



Find names of sailors who've reserved boat #103:

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



Nested Queries

*Find names of sailors
who did NOT reserve boat #103:*

```
SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN (SELECT R.sid
                    FROM Reserves R
                    WHERE R.bid=103)
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5




<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



Nested Queries with Correlation

- EXISTS returns true **TRUE** if the set, is nonempty. EXISTS operator is another set comparison operator, like IN.

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
              FROM Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

*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)
```

- To understand semantics of correlated queries, think of a *nested loops* evaluation:
For each Sailors tuple, check the qualification by computing the subquery.


```
For (i=1...10) do {
    For (j=1...5) do {
        y=x+1;
    }
}
```


Nested Queries with Correlation

Find names of sailors who reserved a boat at most once

- UNIQUE construct can be used.
- UNIQUE checks for duplicate tuples. Returns TRUE if the corresponding set **does not** contain duplicates.

```
SELECT S.sname
FROM Sailors S
WHERE UNIQUE (SELECT R.bid
              FROM Reserves R
              WHERE S.sid=R.sid)
```



<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE. Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL
- **Find sailors whose rating is greater than that of some sailor called Horatio:**

>, <, =, ≥, ≤, ≠, <>

ANY (returns true if there exist tuples (returned from nested WHERE clause) obey the condition!)
 ALL (returns true if all tuples (returned from nested WHERE clause) obey the condition!)

```
SELECT *
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                     FROM Sailors S2
                     WHERE S2.sname='Horatio')
```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Division in SQL

Find sailors who've reserved all boats.

```

SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS
    ((SELECT B.bid
      FROM Boats B)
     EXCEPT
     (SELECT R.bid
      FROM Reserves R
      WHERE R.sid = S.sid))
  
```

A
-
B

EXISTS (.. Is nonempty ?)

NOT EXISTS (.. Is empty ?)

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.15 An Instance S3 of Sailors

Figure 4.16 An Instance R2 of Reserves

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

THINK ABOUT HOW YOU CAN WRITE THE RELATIONAL ALGEBRA VERSION WITHOUT USING THE DIVISION OPERATOR

Aggregate Operators

```
COUNT (*)  
COUNT ([DISTINCT] A)  
SUM ([DISTINCT] A)  
AVG ([DISTINCT] A)  
MAX (A)  
MIN (A)
```

- Significant extension of relational algebra.
- They are used to write statistical queries
- Mainly used for reporting, such as
 - the total sales in 2004,
 - average, max, min income of employees
 - Total number of employees hired/fired in 2004

Aggregate Operators

The total number of sailors in the club?

```
SELECT COUNT (*)
FROM Sailors S
```

```
COUNT (*)
COUNT ( [DISTINCT] A)
SUM ( [DISTINCT] A)
AVG ( [DISTINCT] A)
MAX (A)
MIN (A)
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R2* of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B1* of Boats

Aggregate Operators

Average age of sailors in the club
Whose rating is 10?

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

- COUNT (*)
- COUNT ([DISTINCT] A)
- SUM ([DISTINCT] A)
- AVG ([DISTINCT] A)
- MAX (A)
- MIN (A)

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

Aggregate Operators

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

Average distinct ages of sailors in the club
Whose rating is 10?

```
SELECT AVG (DISTINCT S.age)
FROM Sailors S
WHERE S.rating=10
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

Aggregate Operators

Names of sailors whose rating is equal to the maximum rating in the club.

```
SELECT S.sname
FROM Sailors S
WHERE S.rating= (SELECT MAX(S2.rating)
                FROM Sailors S2)
```

COUNT (*)
 COUNT ([DISTINCT] A)
 SUM ([DISTINCT] A)
 AVG ([DISTINCT] A)
 MAX (A)
 MIN (A)

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

Aggregate Operators

Number of sailors whose rating is equal to the maximum rating in the club.

```
SELECT COUNT(S.sid)
FROM Sailors S
WHERE S.rating= (SELECT MAX(S2.rating)
                FROM Sailors S2)
```

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.15 An Instance S3 of Sailors

Figure 4.16 An Instance R2 of Reserves

Figure 4.17 An Instance B1 of Boats

Aggregate Operators

How many different ratings are there in the club?

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

```
SELECT COUNT (S.rating)
FROM Sailors S
```

Above query is not correct. Think why!

```
SELECT COUNT (DISTINCT S.rating)
FROM Sailors S
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

Find name and age of the oldest sailor(s)

- This query is correct and it is allowed in the SQL/92 standard, but is not supported in some systems.

```
SELECT S.sname, S.age
FROM Sailors S
WHERE S.age =
      (SELECT MAX (S2.age)
       FROM Sailors S2)
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

Find name and age of the oldest sailor(s)

- This query is valid for all systems .

```
SELECT S.sname, S.age
FROM Sailors S
WHERE (SELECT MAX (S2.age)
      FROM Sailors S2)
      = S.age
```

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance R2 of Reserves

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

GROUP BY and HAVING

- So far, we've applied aggregate operators to all (qualifying) tuples. Sometimes, we want to apply them to each of several *groups* of tuples.

Consider: *Find the age of the youngest sailor for each rating level.*

In general, we don't know how many rating levels exist and what the rating values for these levels are!

- Suppose we know that rating values go from 1 to 10; we can write 10 queries that look like this (!):

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance S3 of Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance I22 of Reserves

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance B1 of Boats

For $i = 1, 2, \dots, 10$:

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = i
```

Queries With GROUP BY and HAVING

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

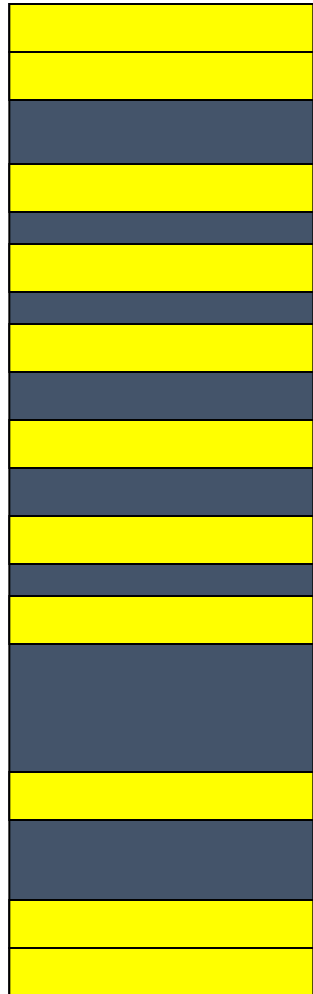
- The *target-list* contains (i) attribute list(ii) terms with aggregate operations (e.g., MIN (*S.age*)).

Conceptual Evaluation

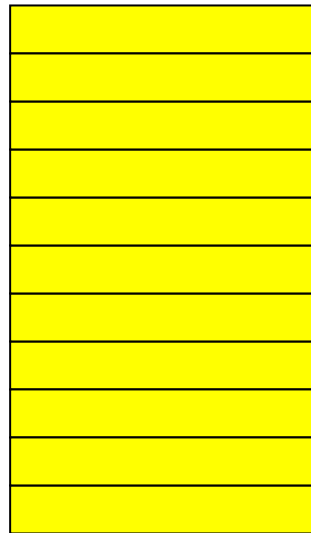
- The cross-product of *relation-list* is computed, tuples that fail *qualification* are discarded, 'unnecessary' fields are deleted, and the remaining tuples are partitioned into groups by the value of attributes in *grouping-list*.
- The *group-qualification* is then applied to eliminate some groups. Expressions in *group-qualification* must have a single value per group!
- One answer tuple is generated per qualifying group.

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

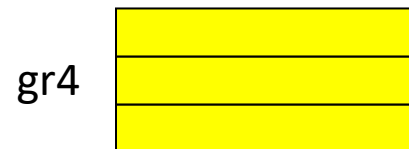
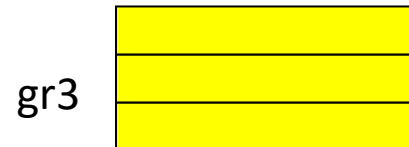
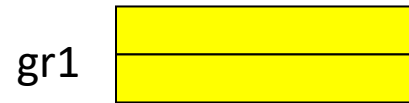
Conceptual Evaluation



```
SELECT target-list
FROM relation-list
WHERE qualification
```

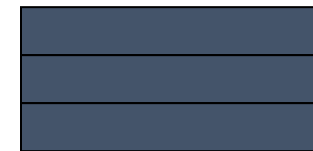


```
GROUP BY grouping-list
```



```
HAVING group-qualification
```

RESULT



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


Conceptual Evaluation

```
SELECT S.rating  
FROM Sailors S  
WHERE S.age >= 18
```

Age = 20
Age = 25
Age = 19
Age=70
Age = 60
Age = 40
Age = 33
Age = 32
Age = 18
Age = 22
Age = 39

Rating = 4
Rating = 2
Rating = 3
Rating=2
Rating=3
Rating=5
Rating=1
Rating=4
Rating=3
Rating=4
Rating=1

GROUP BY
S.rating

gr1	Rating=1
	Rating=1
gr2	Rating=2
	Rating=2
gr3	Rating=3
	Rating=3
	Rating=3
gr4	Rating=4
	Rating=4
	Rating=4
gr5	Rating=5

HAVING COUNT (*) > 1

RESULT

Rating = 1
Rating = 2
Rating = 3
Rating = 4

```
SELECT S.rating  
FROM Sailors S  
WHERE S.age >= 18  
GROUP BY S.rating  
HAVING COUNT (*) > 1
```

Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 such sailors

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

- Only S.rating and S.age are mentioned in the SELECT, GROUP BY or HAVING clauses;
- 2nd column of result is unnamed. (Use AS to name it.)

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
71	zorba	10	16.0
64	horatio	7	35.0
29	brutus	1	33.0
58	rusty	10	35.0

rating	age
1	33.0
7	45.0
7	35.0
8	55.5
10	35.0

rating	
7	35.0

Answer relation

Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 such sailors

```
SELECT S.rating, ag=MIN (S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
71	zorba	10	16.0
64	horatio	7	35.0
29	brutus	1	33.0
58	rusty	10	35.0

rating	age
1	33.0
7	45.0
7	35.0
8	55.5
10	35.0

rating	ag
7	35.0

Answer relation

- Only S.rating and S.age are mentioned in the SELECT, GROUP BY or HAVING clauses;
- 2nd column of result is unnamed. (Use AS to name it.)