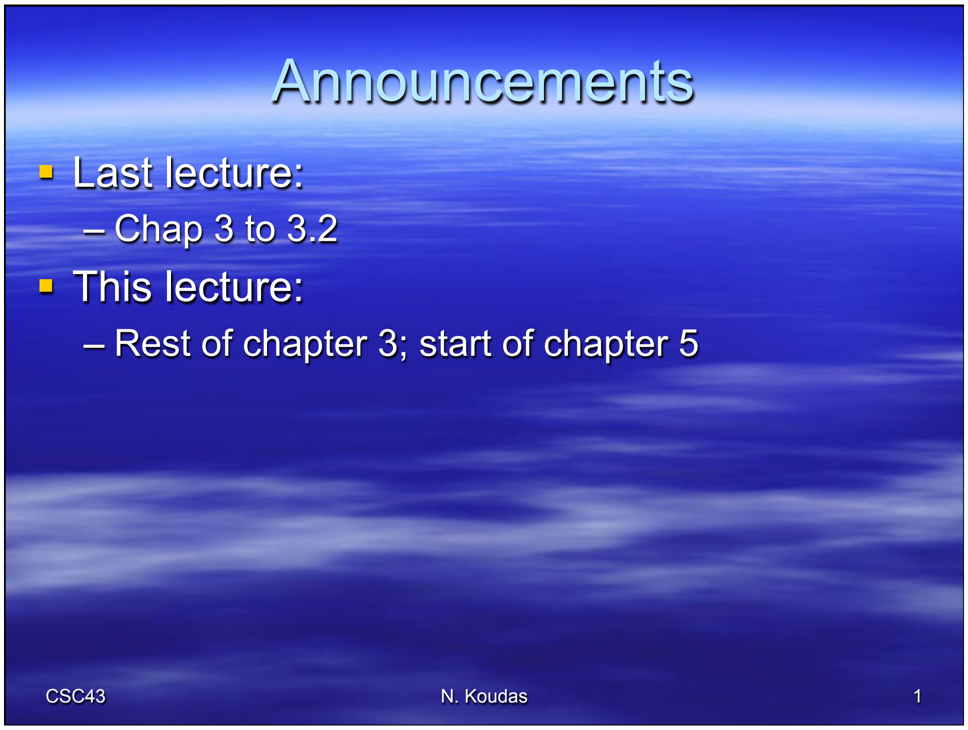
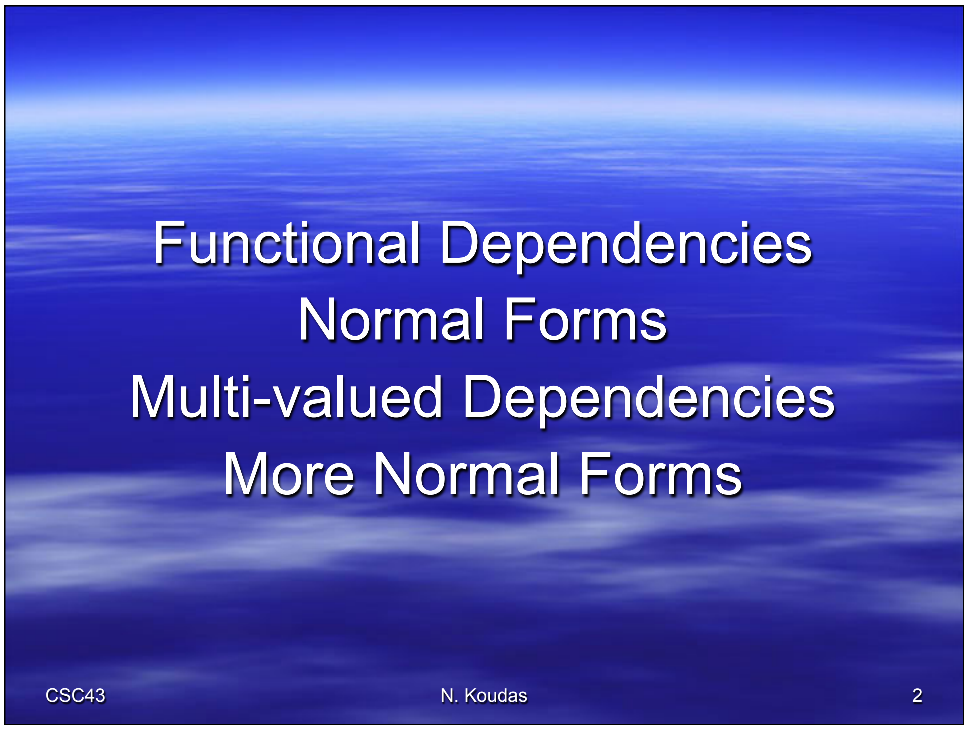


Announcements

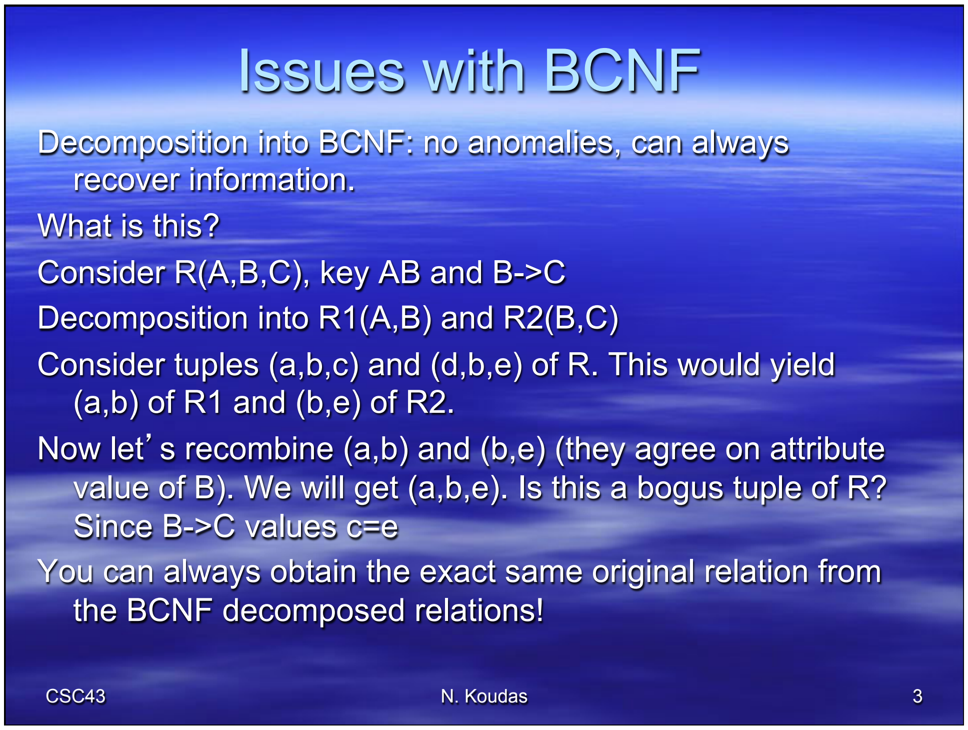
- Last lecture:
  - – Chap 3 to 3.2
- This lecture:
  - – Rest of chapter 3; start of chapter 5

CSC43 N. Koudas 1



Functional Dependencies
Normal Forms
Multi-valued Dependencies
More Normal Forms

CSC43 N. Koudas 2



Issues with BCNF

Decomposition into BCNF: no anomalies, can always recover information.

What is this?

Consider R(A,B,C), key AB and B->C

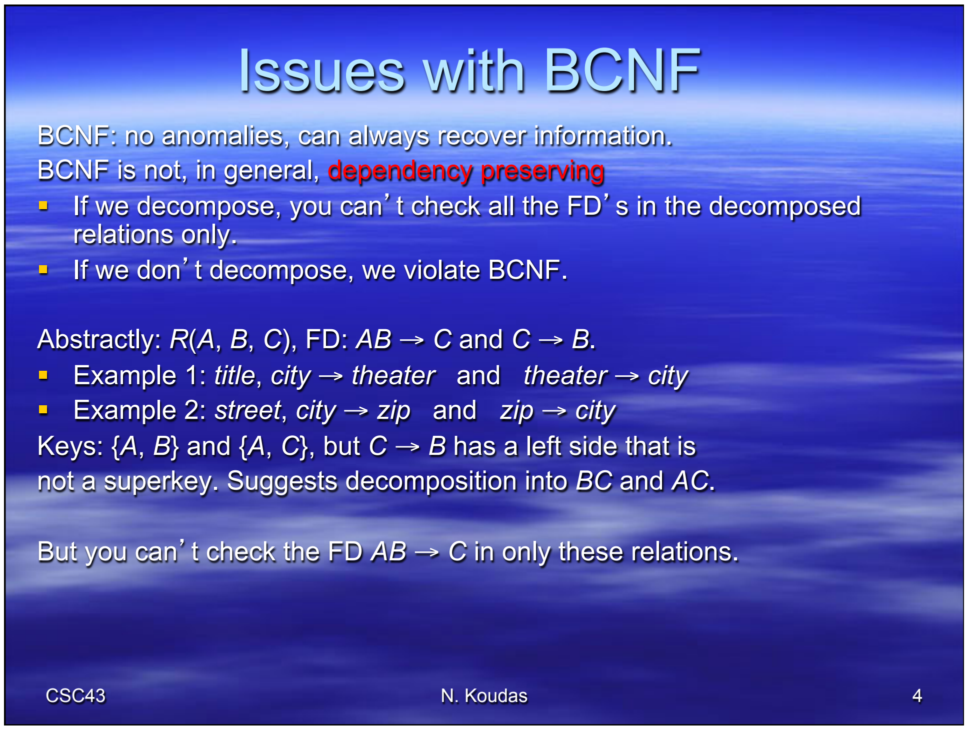
Decomposition into R1(A,B) and R2(B,C)

Consider tuples (a,b,c) and (d,b,e) of R. This would yield (a,b) of R1 and (b,e) of R2.

Now let’s recombine (a,b) and (b,e) (they agree on attribute value of B). We will get (a,b,e). Is this a bogus tuple of R? Since B->C values c=e

You can always obtain the exact same original relation from the BCNF decomposed relations!

CSC43 N. Koudas 3



Issues with BCNF

BCNF: no anomalies, can always recover information.

BCNF is not, in general, dependency preserving

- If we decompose, you can’t check all the FD’s in the decomposed relations only.
- If we don’t decompose, we violate BCNF.

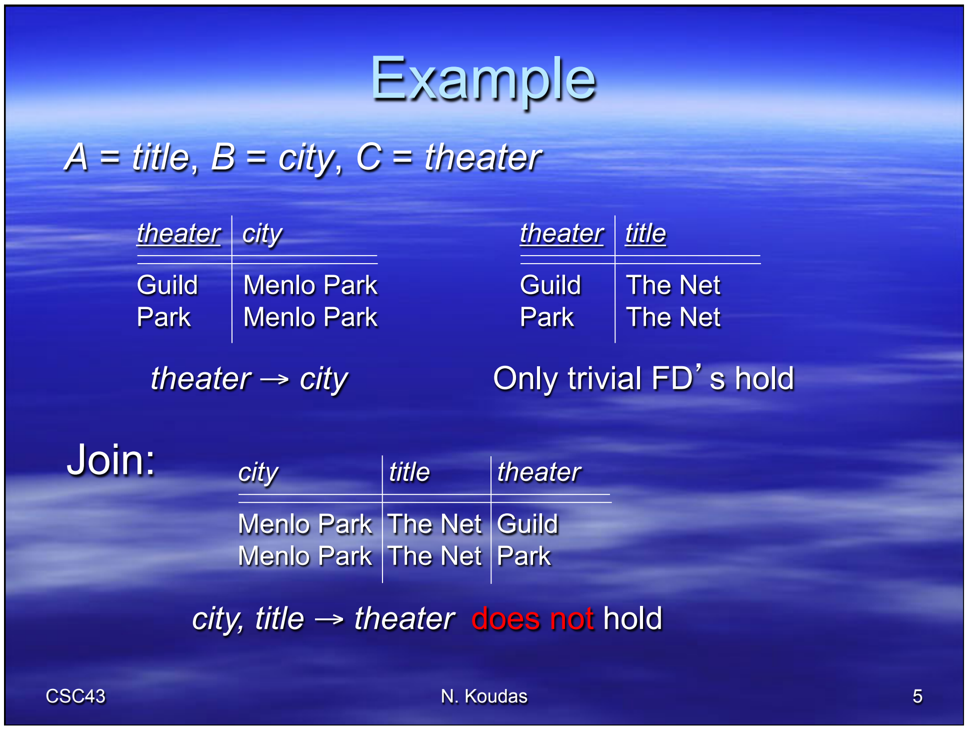
Abstractly: $R(A, B, C)$, FD: $AB \rightarrow C$ and $C \rightarrow B$.

- Example 1: *title*, *city* → *theater* and *theater* → *city*
- Example 2: *street*, *city* → *zip* and *zip* → *city*

Keys: $\{A, B\}$ and $\{A, C\}$, but $C \rightarrow B$ has a left side that is not a superkey. Suggests decomposition into *BC* and *AC*.

But you can’t check the FD $AB \rightarrow C$ in only these relations.

CSC43 N. Koudas 4



Example

A = title, B = city, C = theater

theater	*city*
Guild	Menlo Park
Park	Menlo Park

theater	*title*
Guild	The Net
Park	The Net

theater → *city*

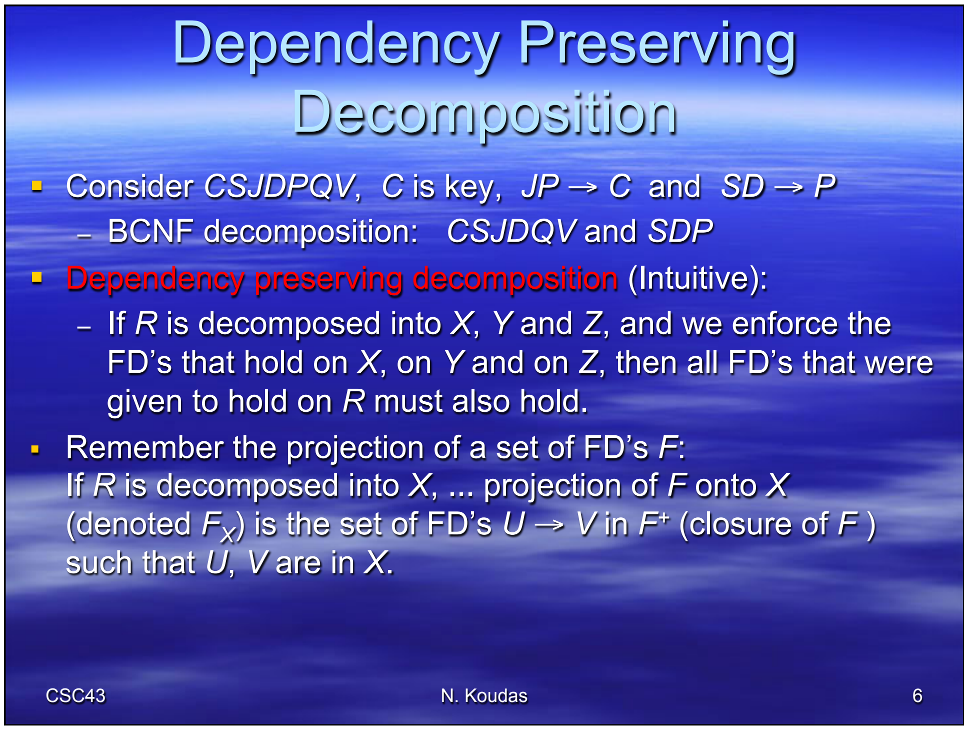
Only trivial FD's hold

Join:

city	*title*	*theater*
Menlo Park	The Net	Guild
Menlo Park	The Net	Park

city, title → *theater* does not hold

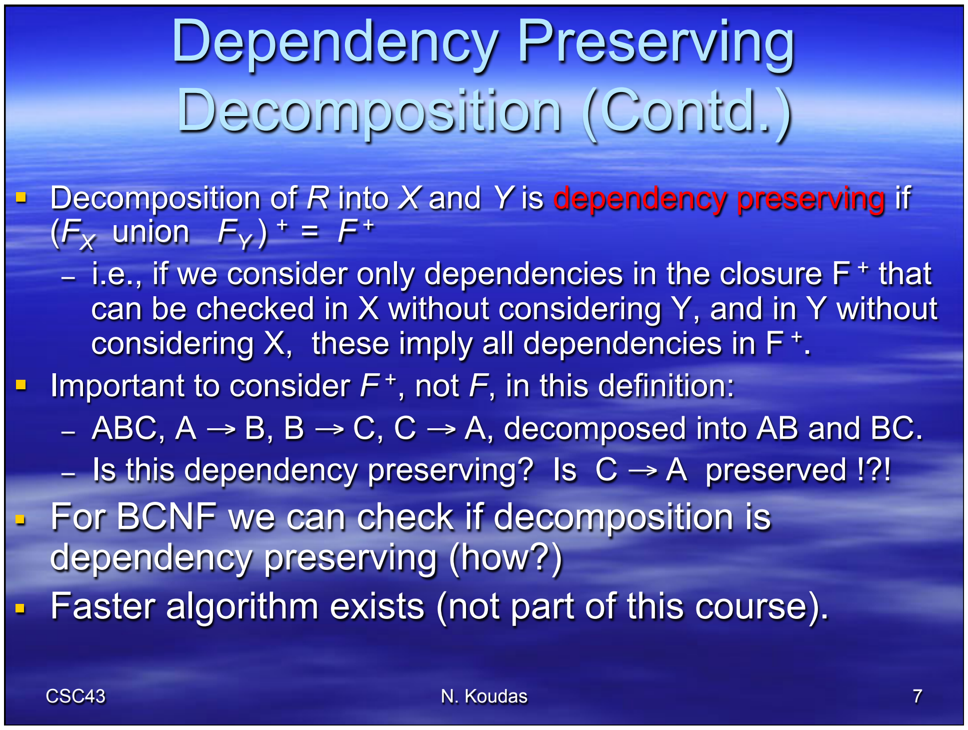
CSC43 N. Koudas 5



Dependency Preserving Decomposition

- Consider *CSJDPQV*, *C* is key, *JP* → *C* and *SD* → *P*
  - BCNF decomposition: *CSJDQV* and *SDP*
- Dependency preserving decomposition (Intuitive):
  - If *R* is decomposed into *X*, *Y* and *Z*, and we enforce the FD's that hold on *X*, on *Y* and on *Z*, then all FD's that were given to hold on *R* must also hold.
- Remember the projection of a set of FD's *F*:
  If *R* is decomposed into *X*, ... projection of *F* onto *X* (denoted F_X) is the set of FD's *U* → *V* in F^+ (closure of *F*) such that *U*, *V* are in *X*.

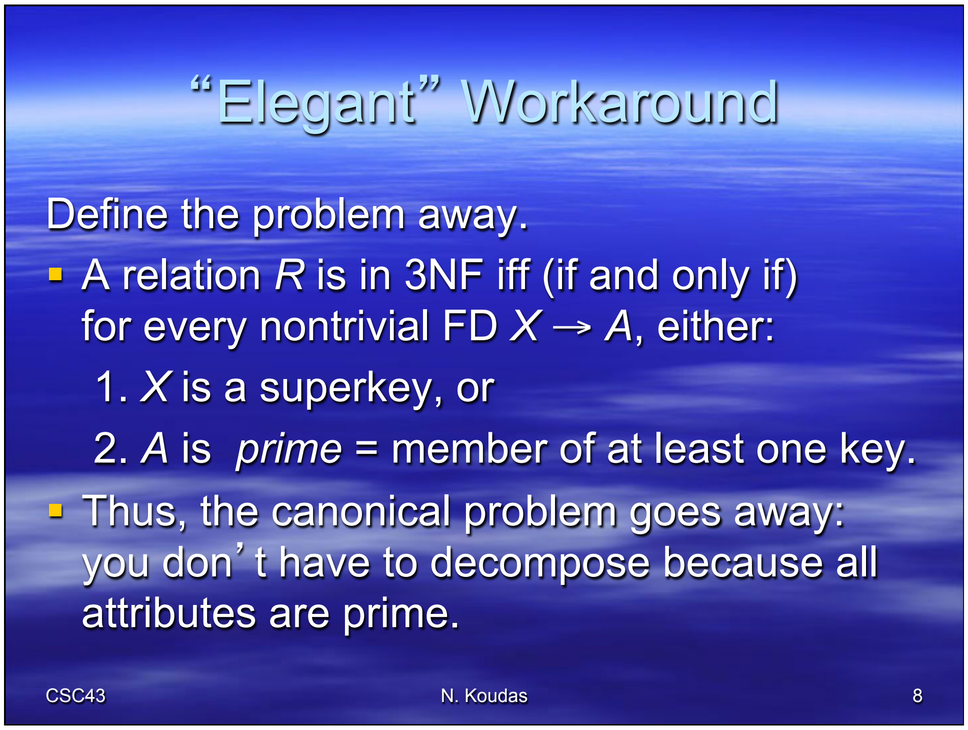
CSC43 N. Koudas 6



Dependency Preserving Decomposition (Contd.)

- Decomposition of *R* into *X* and *Y* is dependency preserving if $(F_X \text{ union } F_Y)^+ = F^+$
  - i.e., if we consider only dependencies in the closure F^+ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F^+.
- Important to consider F^+, not *F*, in this definition:
  - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, decomposed into AB and BC.
  - Is this dependency preserving? Is $C \rightarrow A$ preserved !?!
- For BCNF we can check if decomposition is dependency preserving (how?)
- Faster algorithm exists (not part of this course).

CSC43 N. Koudas 7

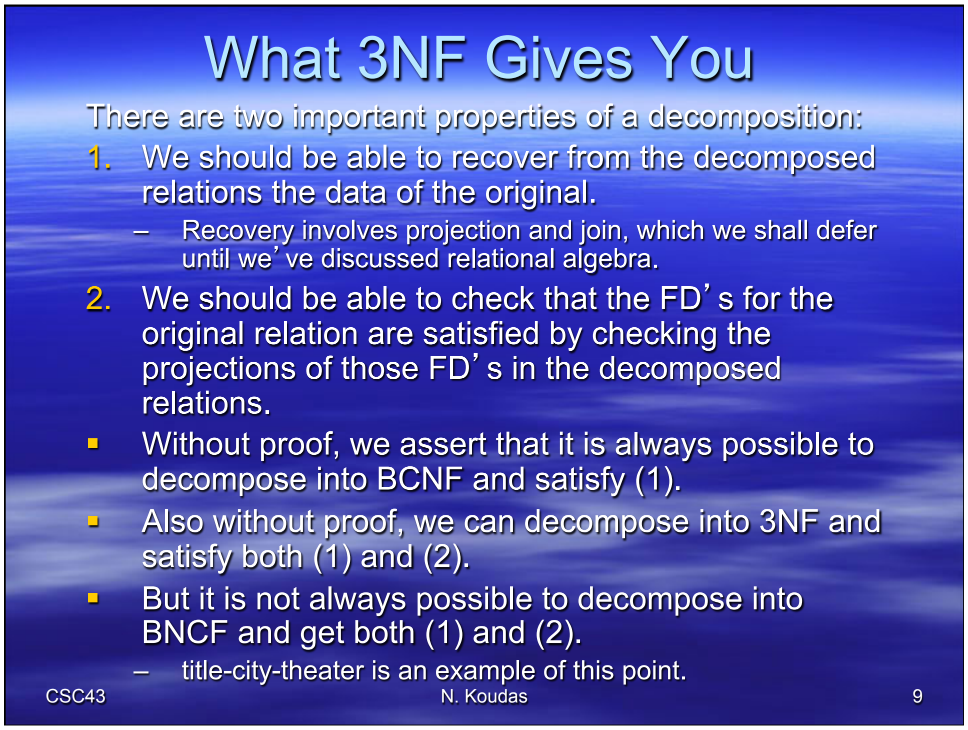


"Elegant" Workaround

Define the problem away.

- A relation *R* is in 3NF iff (if and only if) for every nontrivial FD $X \rightarrow A$, either:
  1. *X* is a superkey, or
  2. *A* is *prime* = member of at least one key.
- Thus, the canonical problem goes away: you don't have to decompose because all attributes are prime.

CSC43 N. Koudas 8



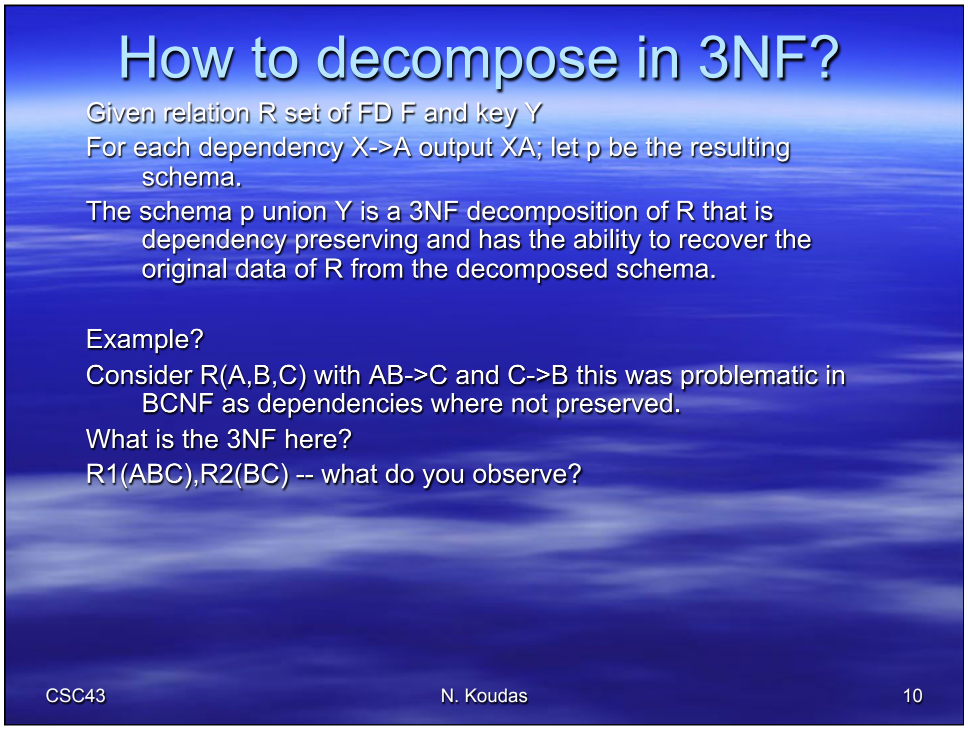
What 3NF Gives You

There are two important properties of a decomposition:

1. We should be able to recover from the decomposed relations the data of the original.
   - Recovery involves projection and join, which we shall defer until we' ve discussed relational algebra.
2. We should be able to check that the FD' s for the original relation are satisfied by checking the projections of those FD' s in the decomposed relations.

- Without proof, we assert that it is always possible to decompose into BCNF and satisfy (1).
- Also without proof, we can decompose into 3NF and satisfy both (1) and (2).
- But it is not always possible to decompose into BNCF and get both (1) and (2).
  - title-city-theater is an example of this point.

CSC43 N. Koudas 9



How to decompose in 3NF?

Given relation R set of FD F and key Y

For each dependency X->A output XA; let p be the resulting schema.

The schema p union Y is a 3NF decomposition of R that is dependency preserving and has the ability to recover the original data of R from the decomposed schema.

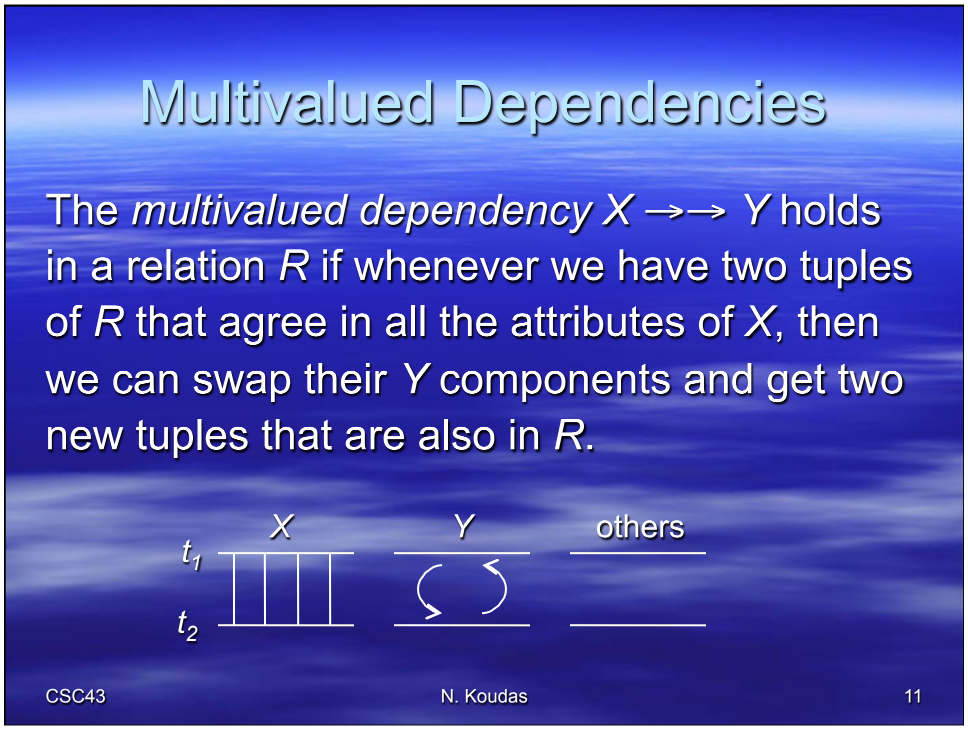
Example?

Consider R(A,B,C) with AB->C and C->B this was problematic in BCNF as dependencies where not preserved.

What is the 3NF here?

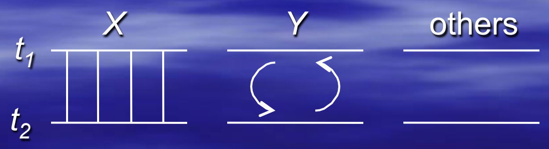
R1(ABC),R2(BC) -- what do you observe?

CSC43 N. Koudas 10

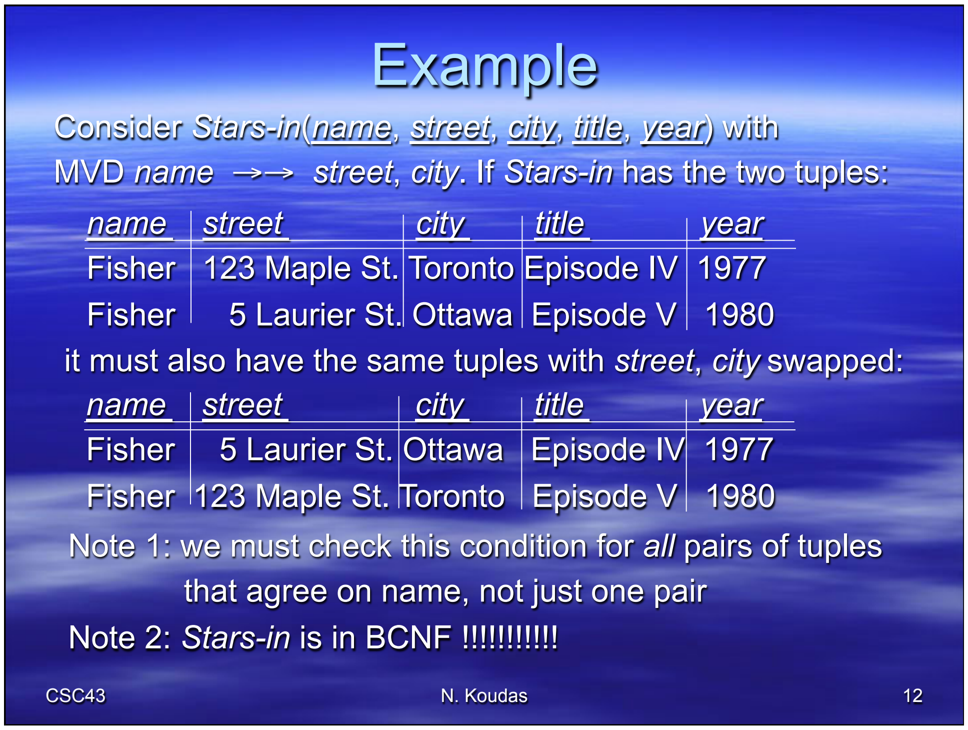


Multivalued Dependencies

The *multivalued dependency* *X* →→ *Y* holds in a relation *R* if whenever we have two tuples of *R* that agree in all the attributes of *X*, then we can swap their *Y* components and get two new tuples that are also in *R*.



CSC43 N. Koudas 11



Example

Consider *Stars-in*(*name*, *street*, *city*, *title*, *year*) with MVD *name* →→ *street*, *city*. If *Stars-in* has the two tuples:

name	*street*	*city*	*title*	*year*
Fisher	123 Maple St.	Toronto	Episode IV	1977
Fisher	5 Laurier St.	Ottawa	Episode V	1980

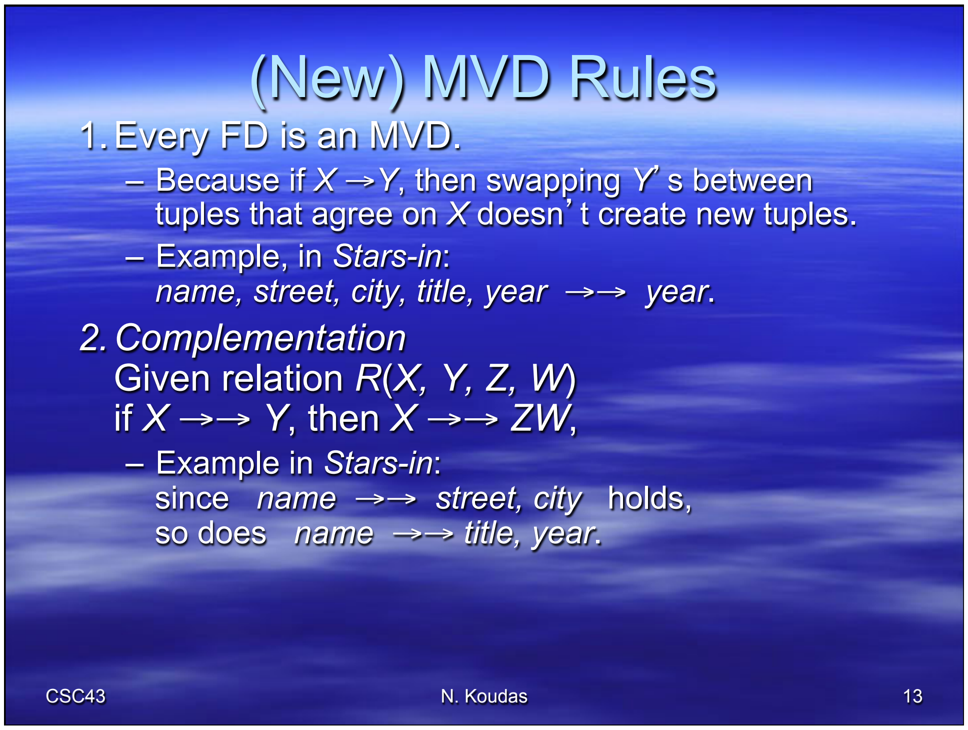
it must also have the same tuples with *street*, *city* swapped:

name	*street*	*city*	*title*	*year*
Fisher	5 Laurier St.	Ottawa	Episode IV	1977
Fisher	123 Maple St.	Toronto	Episode V	1980

Note 1: we must check this condition for *all* pairs of tuples that agree on name, not just one pair

Note 2: *Stars-in* is in BCNF !!!!!!!!!!

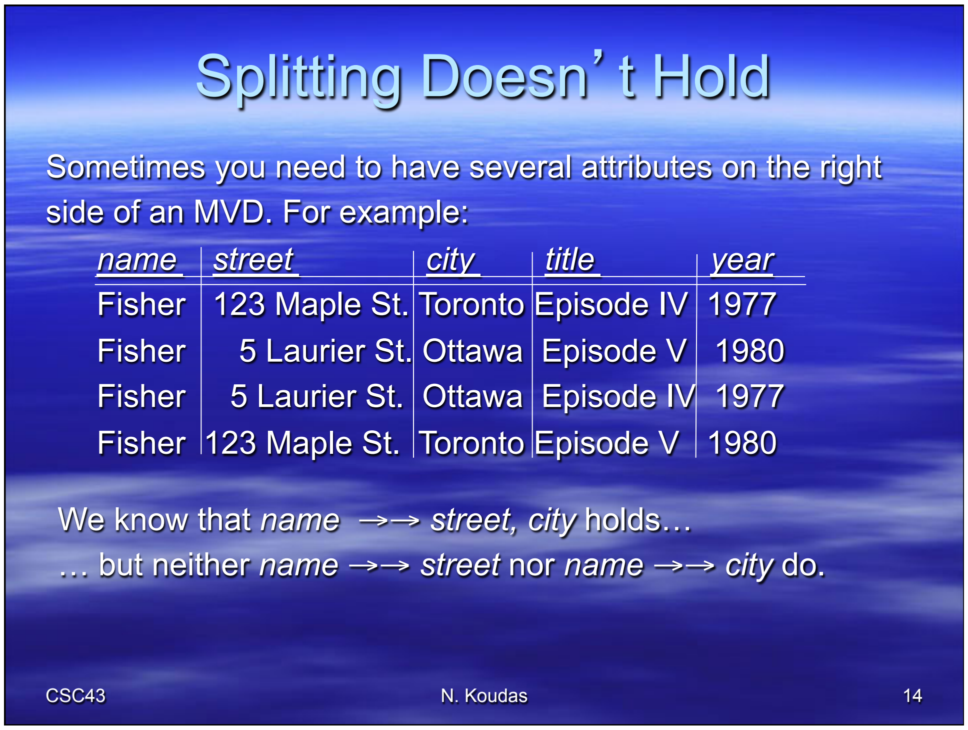
CSC43 N. Koudas 12



(New) MVD Rules

1. Every FD is an MVD.
   - Because if $X \rightarrow Y$, then swapping Y's between tuples that agree on X doesn't create new tuples.
   - Example, in *Stars-in*: *name, street, city, title, year* $\rightarrow\rightarrow$ *year*.
2. *Complementation*
   Given relation $R(X, Y, Z, W)$
   if $X \rightarrow\rightarrow Y$, then $X \rightarrow\rightarrow ZW$,
   - Example in *Stars-in*:
     since *name* $\rightarrow\rightarrow$ *street, city* holds,
     so does *name* $\rightarrow\rightarrow$ *title, year*.

CSC43 N. Koudas 13



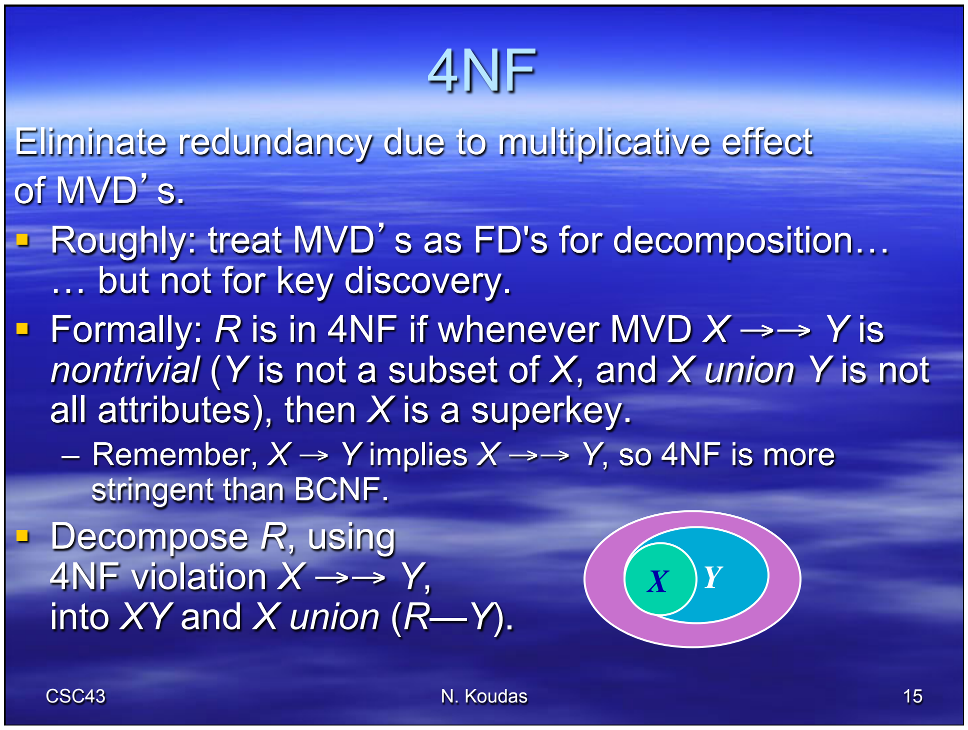
Splitting Doesn't Hold

Sometimes you need to have several attributes on the right side of an MVD. For example:

name	*street*	*city*	*title*	*year*
Fisher	123 Maple St.	Toronto	Episode IV	1977
Fisher	5 Laurier St.	Ottawa	Episode V	1980
Fisher	5 Laurier St.	Ottawa	Episode IV	1977
Fisher	123 Maple St.	Toronto	Episode V	1980

We know that *name* $\rightarrow\rightarrow$ *street, city* holds…
… but neither *name* $\rightarrow\rightarrow$ *street* nor *name* $\rightarrow\rightarrow$ *city* do.

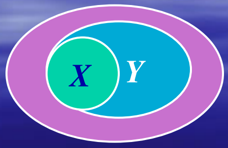
CSC43 N. Koudas 14



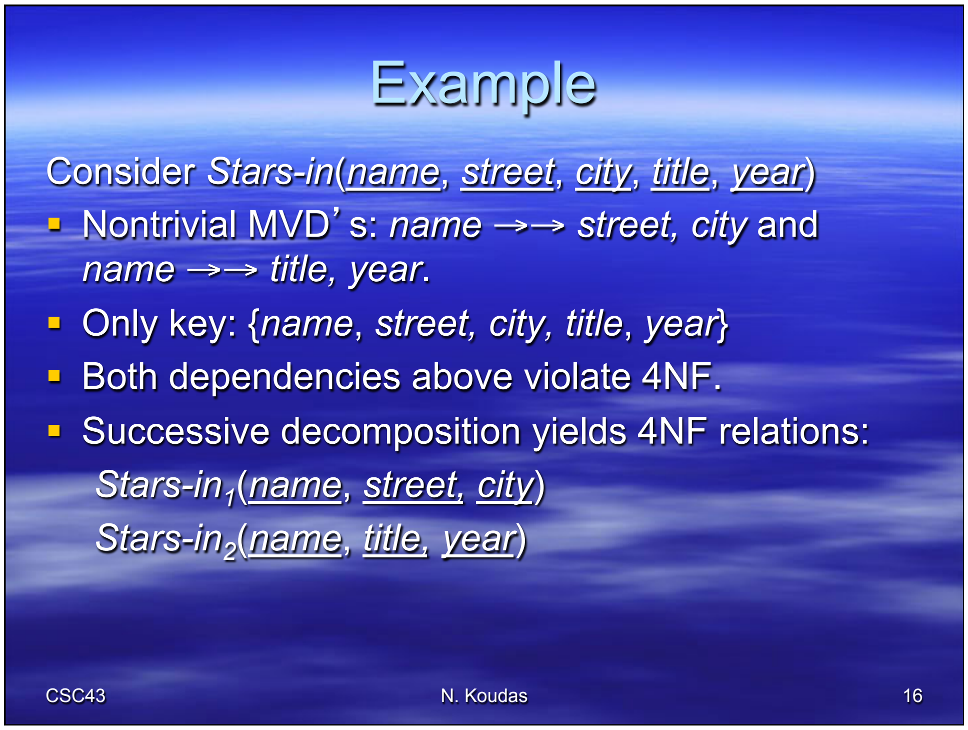
4NF

Eliminate redundancy due to multiplicative effect of MVD's.

- Roughly: treat MVD's as FD's for decomposition… … but not for key discovery.
- Formally: *R* is in 4NF if whenever MVD $X \rightarrow\rightarrow Y$ is *nontrivial* (*Y* is not a subset of *X*, and *X union Y* is not all attributes), then *X* is a superkey.
  - Remember, $X \rightarrow Y$ implies $X \rightarrow\rightarrow Y$, so 4NF is more stringent than BCNF.
- Decompose *R*, using 4NF violation $X \rightarrow\rightarrow Y$, into *XY* and *X union* (*R*—*Y*).



CSC43 N. Koudas 15



Example

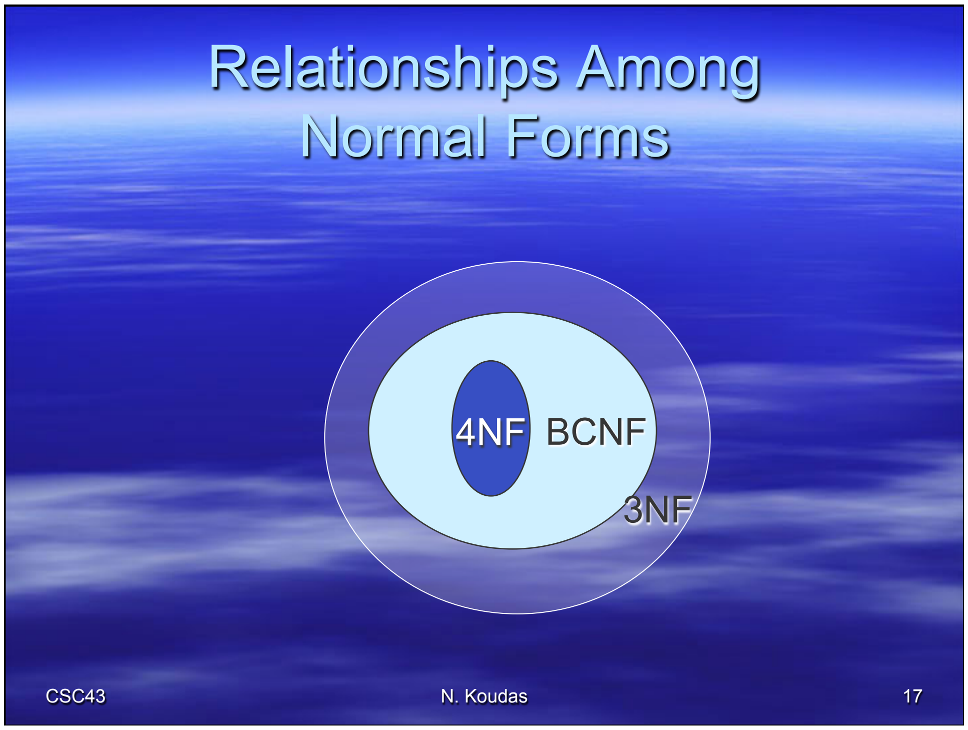
Consider *Stars-in*(<u>*name*</u>, <u>*street*</u>, <u>*city*</u>, <u>*title*</u>, <u>*year*</u>)

- Nontrivial MVD's: *name* $\rightarrow\rightarrow$ *street, city* and *name* $\rightarrow\rightarrow$ *title, year*.
- Only key: {*name*, *street, city, title*, *year*}
- Both dependencies above violate 4NF.
- Successive decomposition yields 4NF relations:

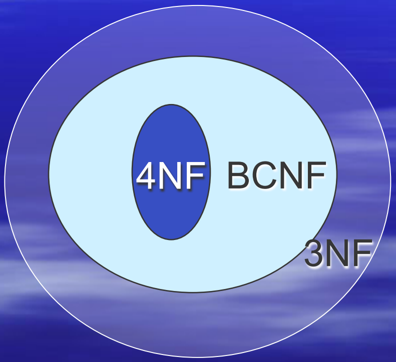
  Stars-in$_1$(<u>*name*</u>, <u>*street*</u>, <u>*city*</u>)

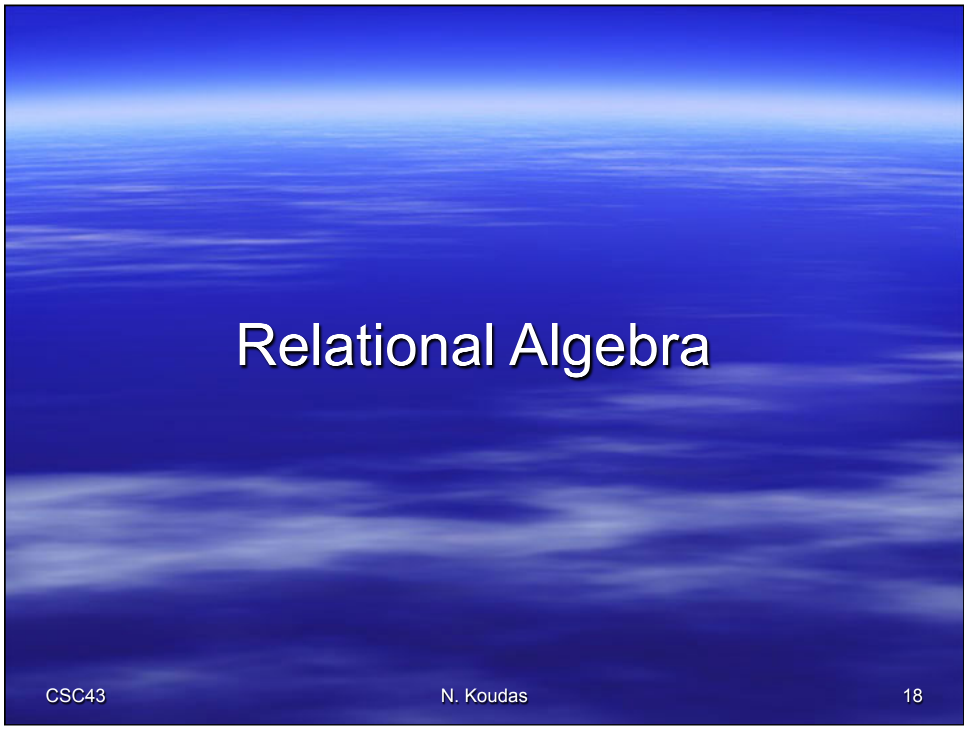
  Stars-in$_2$(<u>*name*</u>, <u>*title*</u>, <u>*year*</u>)

CSC43 N. Koudas 16

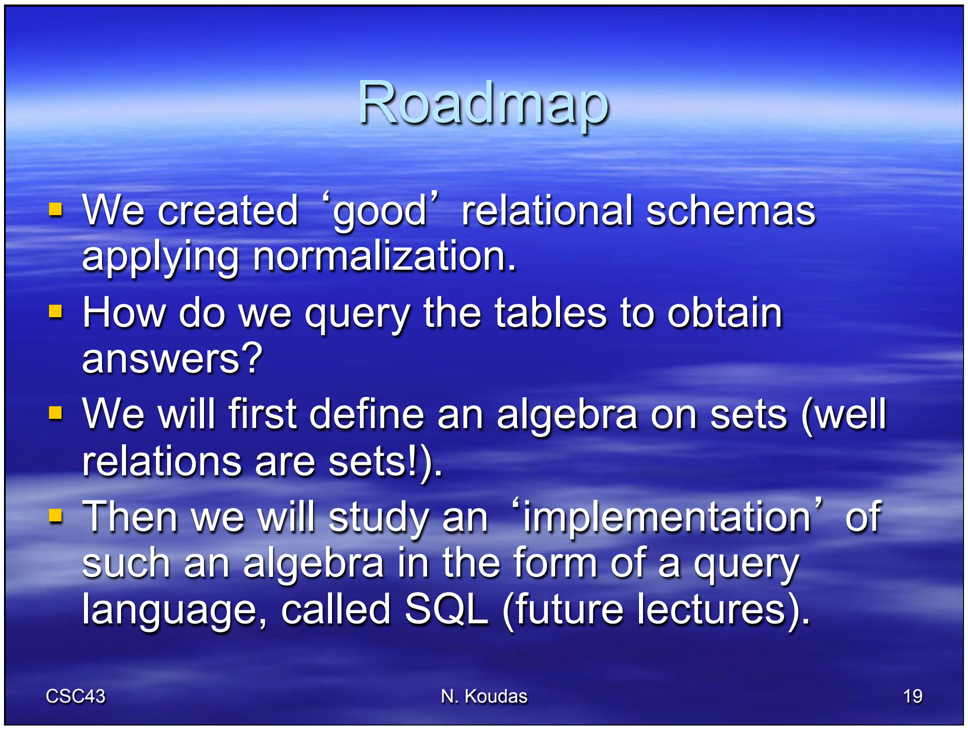


Relationships Among Normal Forms





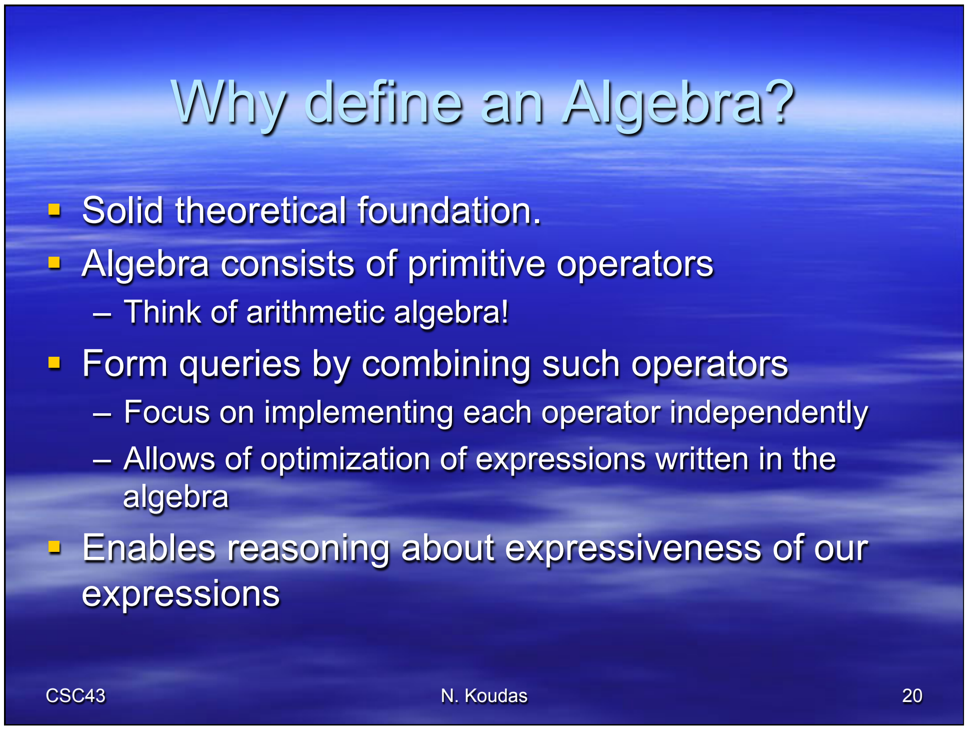
Relational Algebra



Roadmap

- We created 'good' relational schemas applying normalization.
- How do we query the tables to obtain answers?
- We will first define an algebra on sets (well relations are sets!).
- Then we will study an 'implementation' of such an algebra in the form of a query language, called SQL (future lectures).

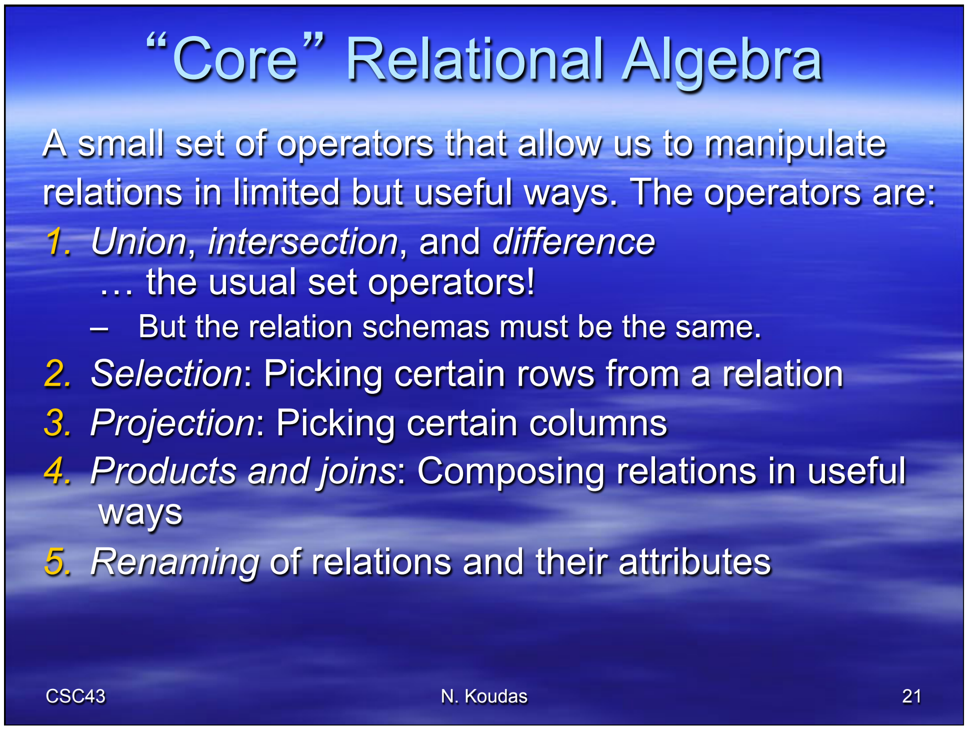
CSC43 N. Koudas 19



Why define an Algebra?

- Solid theoretical foundation.
- Algebra consists of primitive operators
  - Think of arithmetic algebra!
- Form queries by combining such operators
  - Focus on implementing each operator independently
  - Allows of optimization of expressions written in the algebra
- Enables reasoning about expressiveness of our expressions

CSC43 N. Koudas 20

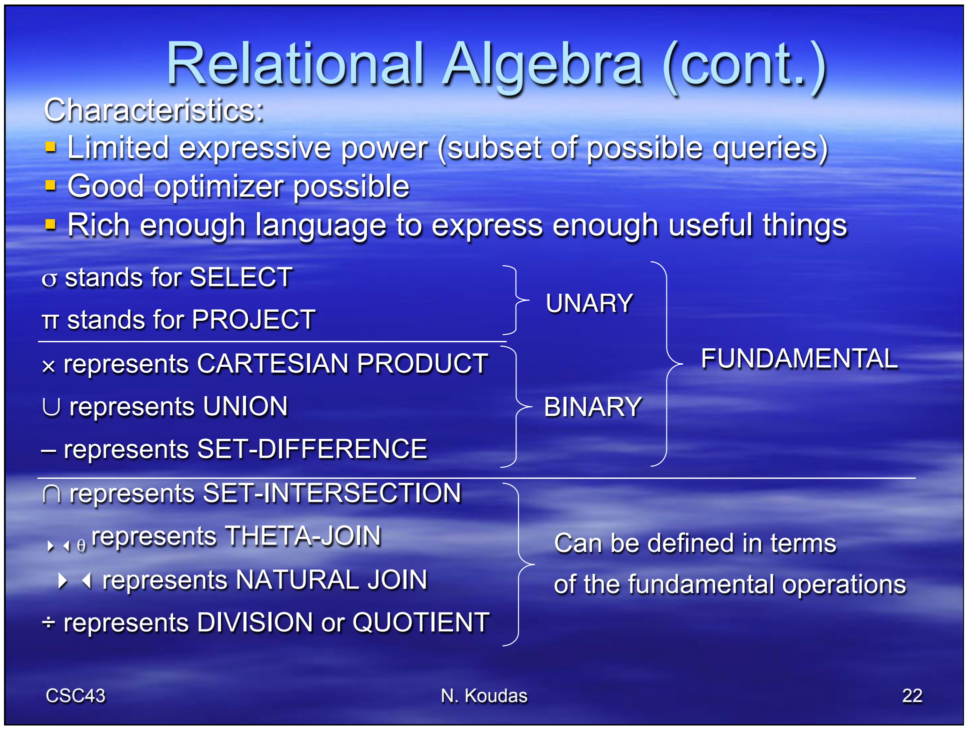


“Core” Relational Algebra

A small set of operators that allow us to manipulate relations in limited but useful ways. The operators are:

1. *Union*, *intersection*, and *difference*
   … the usual set operators!
   – But the relation schemas must be the same.
2. *Selection*: Picking certain rows from a relation
3. *Projection*: Picking certain columns
4. *Products and joins*: Composing relations in useful ways
5. *Renaming* of relations and their attributes

CSC43 N. Koudas 21



Relational Algebra (cont.)

Characteristics:

- Limited expressive power (subset of possible queries)
- Good optimizer possible
- Rich enough language to express enough useful things

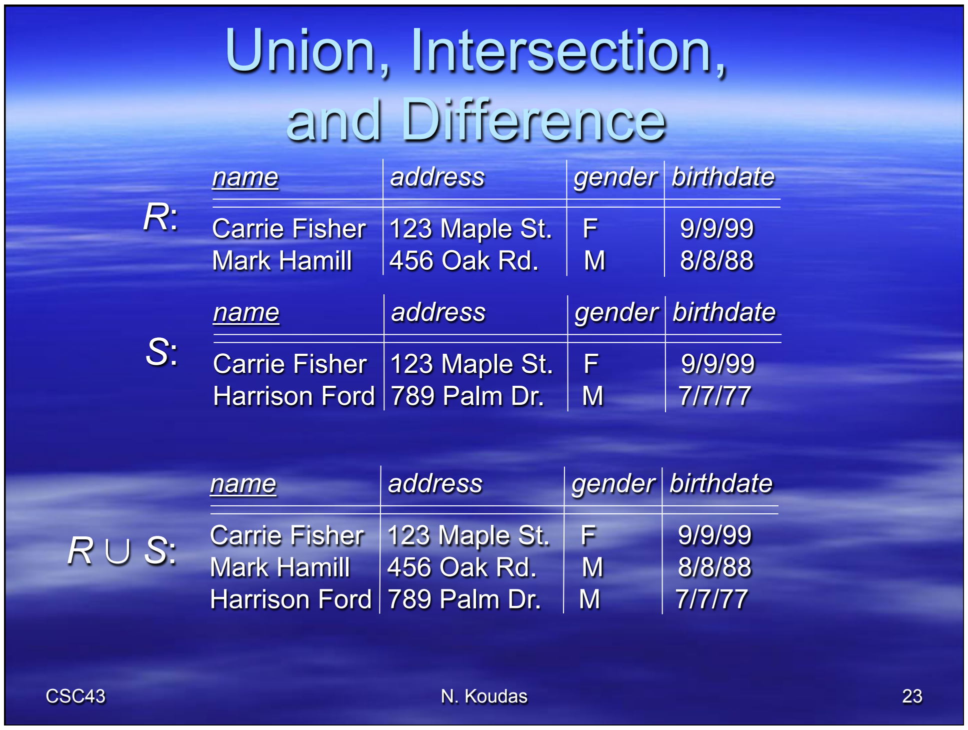
σ stands for SELECT
π stands for PROJECT
} UNARY

$\times$ represents CARTESIAN PRODUCT
$\cup$ represents UNION
$-$ represents SET-DIFFERENCE
} BINARY

} FUNDAMENTAL

$\cap$ represents SET-INTERSECTION
$\bowtie_\theta$ represents THETA-JOIN
$\bowtie$ represents NATURAL JOIN
$\div$ represents DIVISION or QUOTIENT
} Can be defined in terms of the fundamental operations

CSC43 N. Koudas 22



Union, Intersection, and Difference

R:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99
Mark Hamill	456 Oak Rd.	M	8/8/88

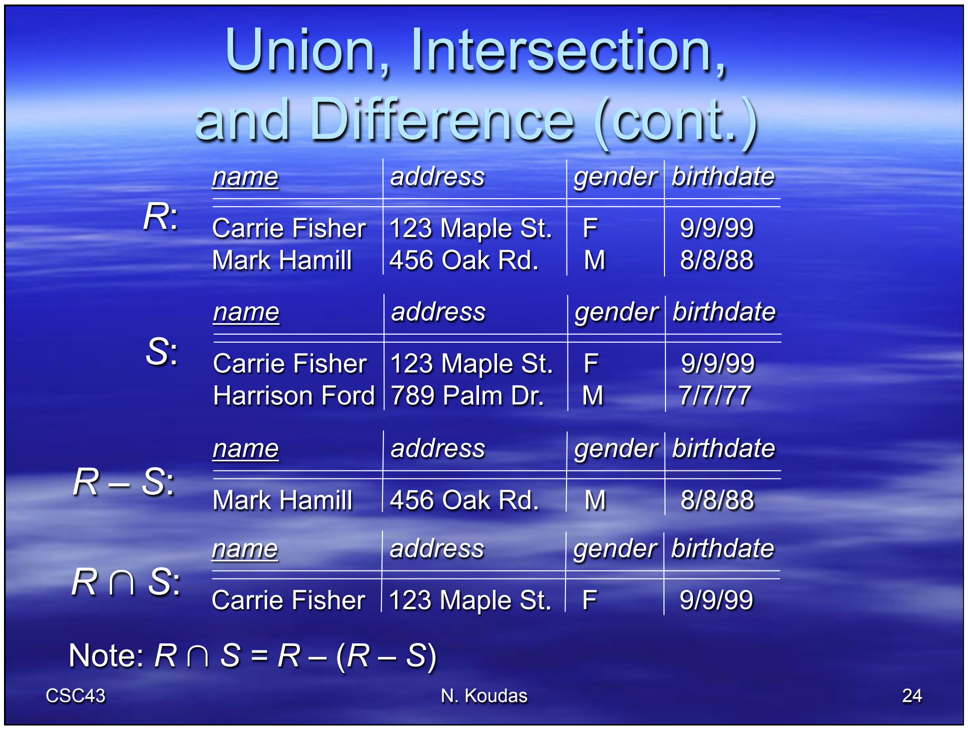
S:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99
Harrison Ford	789 Palm Dr.	M	7/7/77

$R \cup S$:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99
Mark Hamill	456 Oak Rd.	M	8/8/88
Harrison Ford	789 Palm Dr.	M	7/7/77

CSC43 N. Koudas 23



Union, Intersection, and Difference (cont.)

R:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99
Mark Hamill	456 Oak Rd.	M	8/8/88

S:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99
Harrison Ford	789 Palm Dr.	M	7/7/77

$R - S$:

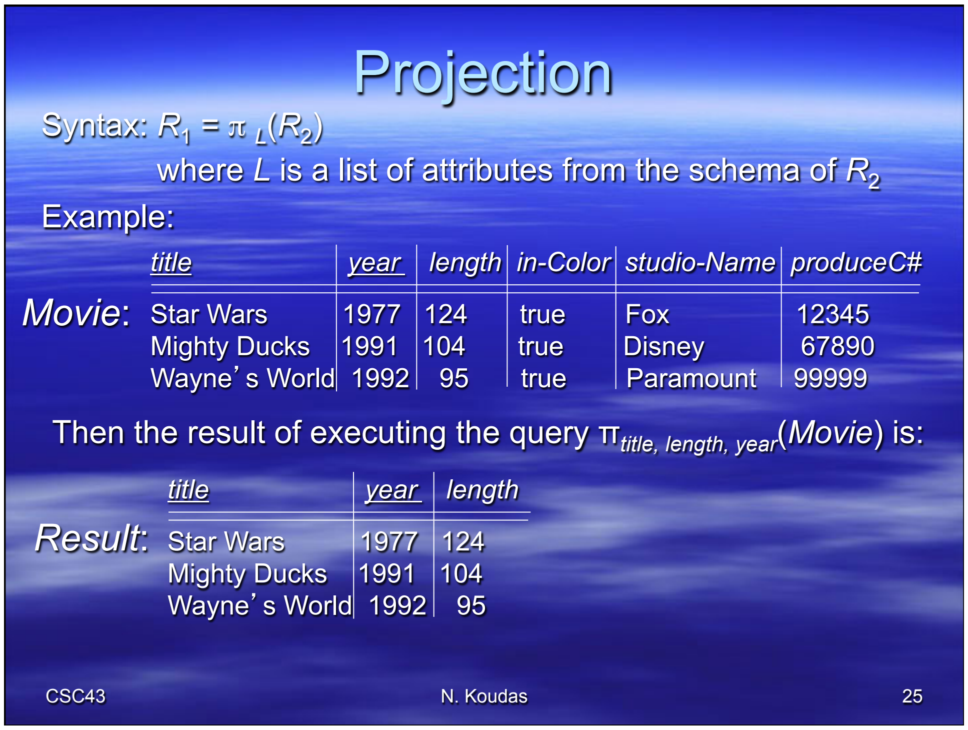
name	address	gender	birthdate
Mark Hamill	456 Oak Rd.	M	8/8/88

$R \cap S$:

name	address	gender	birthdate
Carrie Fisher	123 Maple St.	F	9/9/99

Note: $R \cap S = R - (R - S)$

CSC43 N. Koudas 24



Projection

Syntax: $R_1 = \pi_L(R_2)$

where L is a list of attributes from the schema of R_2

Example:

Movie:

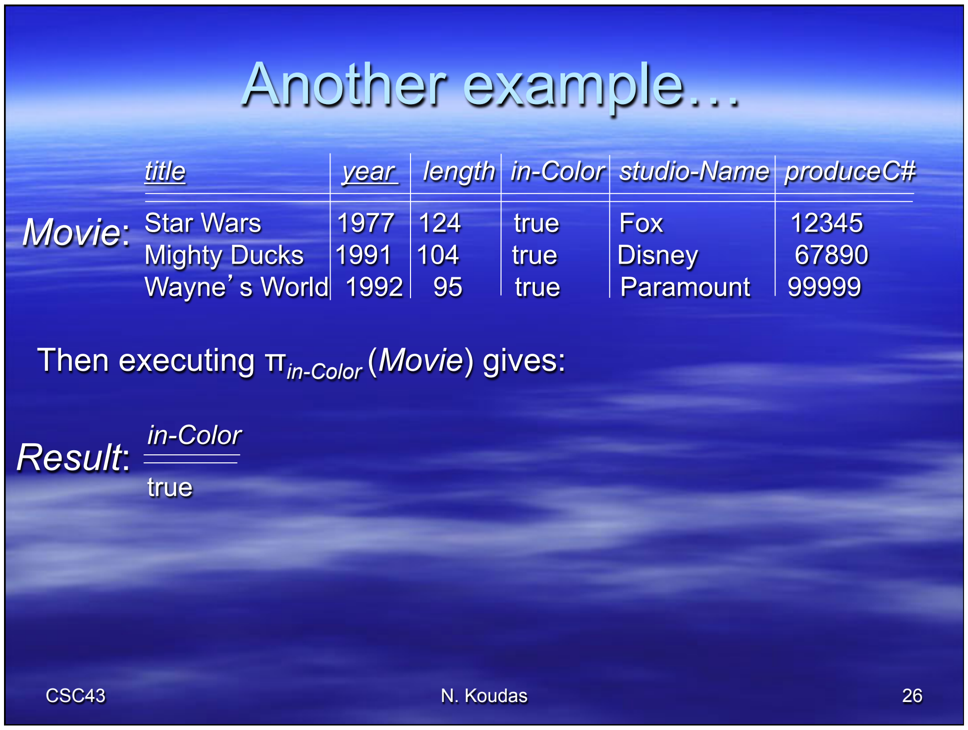
title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890
Wayne's World	1992	95	true	Paramount	99999

Then the result of executing the query $\pi_{title,\ length,\ year}(Movie)$ is:

Result:

title	*year*	*length*
Star Wars	1977	124
Mighty Ducks	1991	104
Wayne's World	1992	95

CSC43 N. Koudas 25



Another example…

Movie:

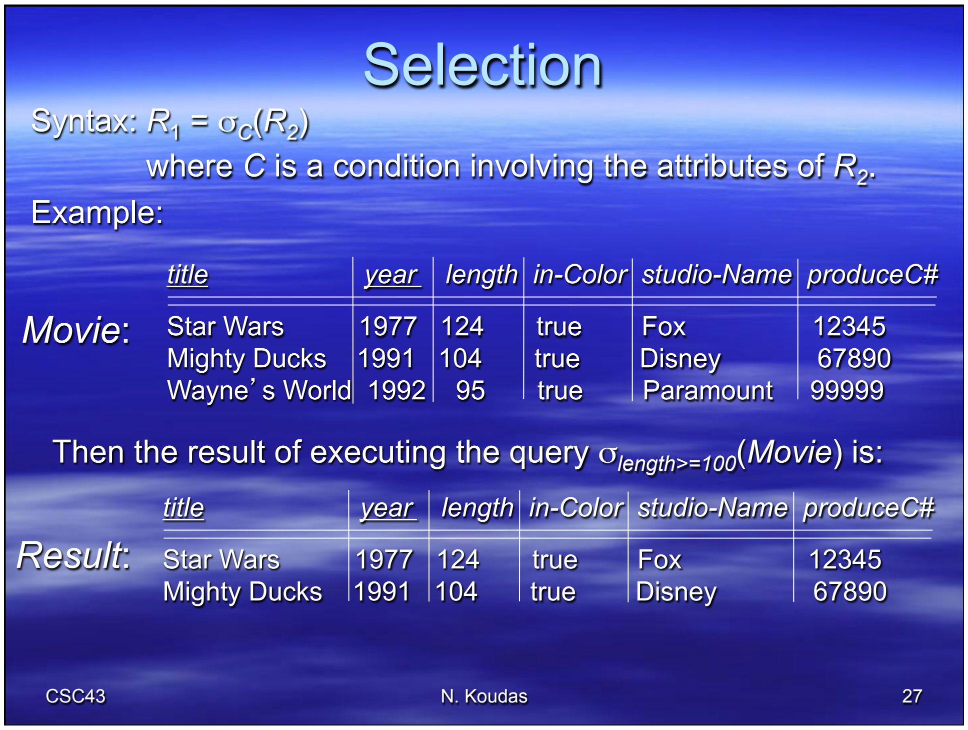
title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890
Wayne's World	1992	95	true	Paramount	99999

Then executing $\pi_{in\text{-}Color}(Movie)$ gives:

Result:

in-Color
true

CSC43 N. Koudas 26



Selection

Syntax: $R_1 = \sigma_C(R_2)$

where C is a condition involving the attributes of R_2.

Example:

Movie:

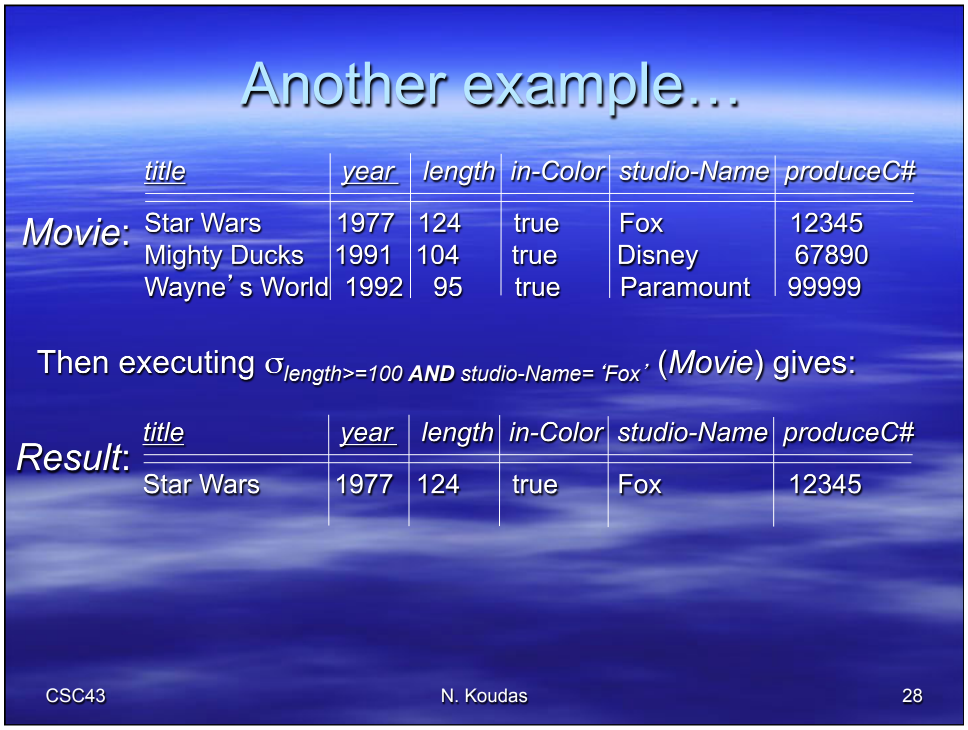
title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890
Wayne's World	1992	95	true	Paramount	99999

Then the result of executing the query $\sigma_{length>=100}$(*Movie*) is:

Result:

title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890

CSC43 N. Koudas 27



Another example…

Movie:

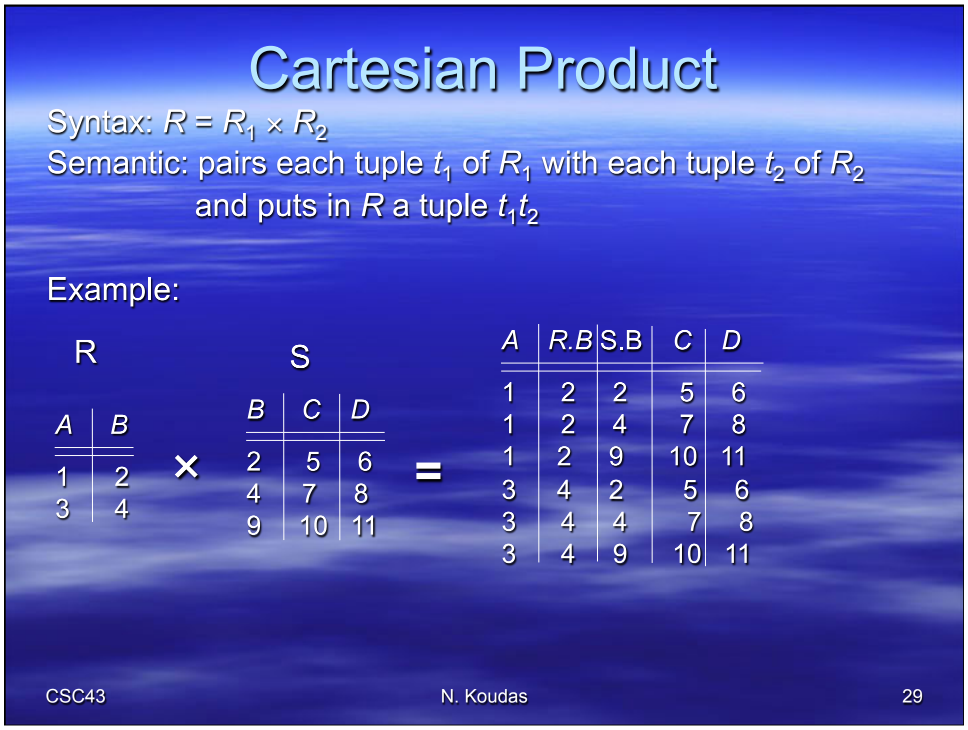
title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345
Mighty Ducks	1991	104	true	Disney	67890
Wayne's World	1992	95	true	Paramount	99999

Then executing $\sigma_{length>=100 \textbf{ AND } studio\text{-}Name= \text{'Fox'}}$ (*Movie*) gives:

Result:

title	*year*	*length*	*in-Color*	*studio-Name*	*produceC#*
Star Wars	1977	124	true	Fox	12345

CSC43 N. Koudas 28



Cartesian Product

Syntax: $R = R_1 \times R_2$

Semantic: pairs each tuple t_1 of R_1 with each tuple t_2 of R_2 and puts in R a tuple t_1t_2

Example:

R

A	*B*
1	2
3	4

×

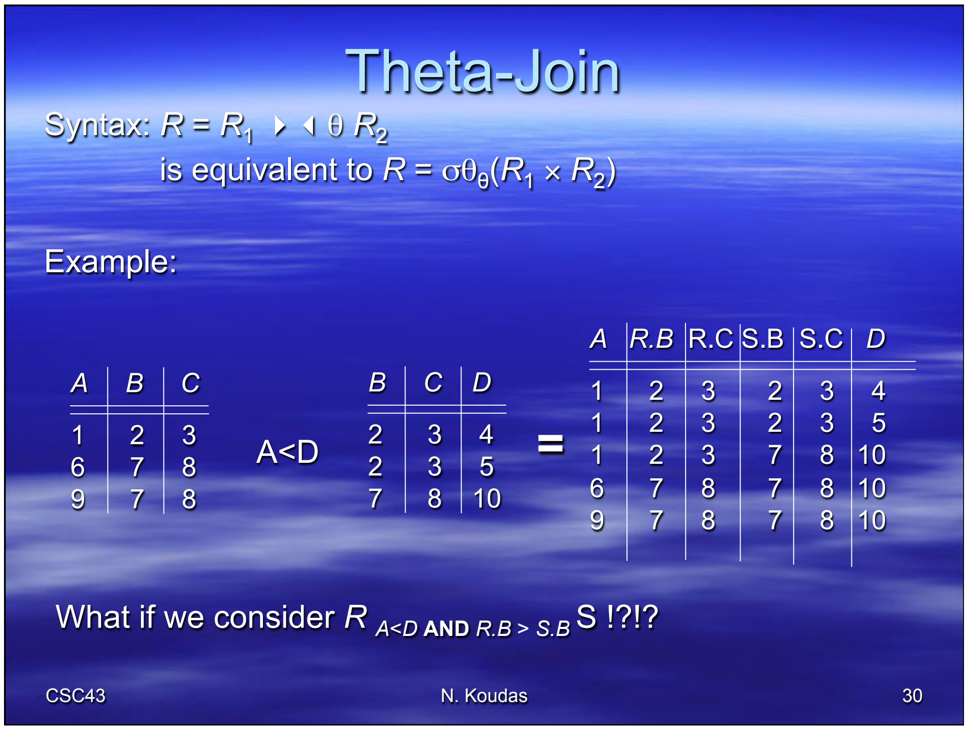
S

B	*C*	*D*
2	5	6
4	7	8
9	10	11

=

A	*R.B*	S.B	*C*	*D*
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

CSC43 N. Koudas 29



Theta-Join

Syntax: $R = R_1 \bowtie_\theta R_2$

is equivalent to $R = \sigma\theta_\theta(R_1 \times R_2)$

Example:

A	*B*	*C*
1	2	3
6	7	8
9	7	8

A<D

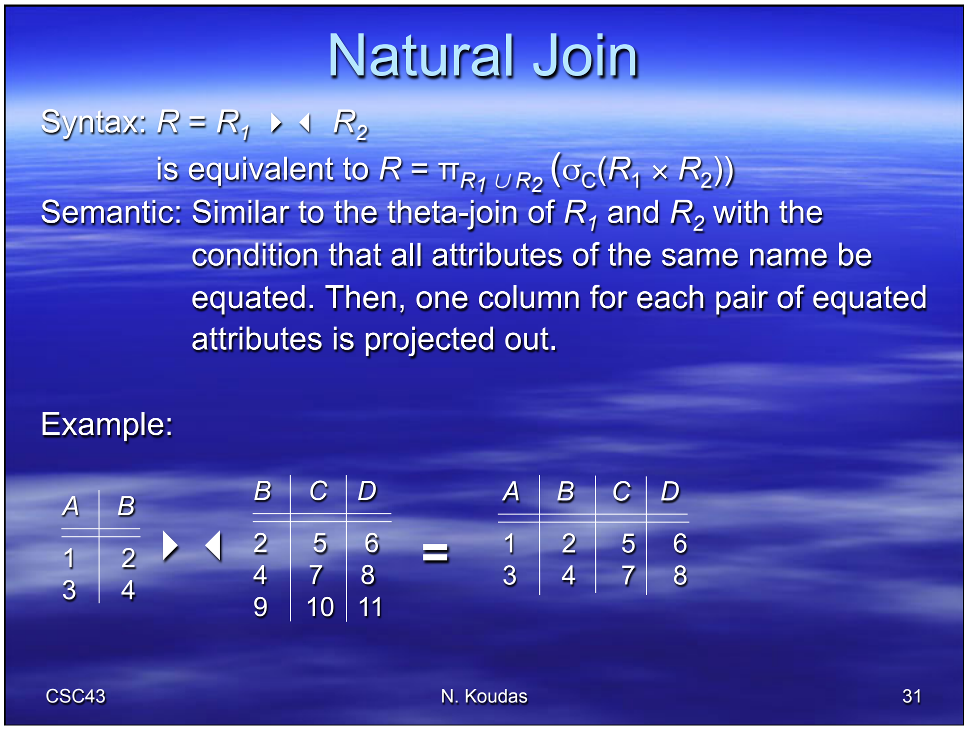
B	*C*	*D*
2	3	4
2	3	5
7	8	10

=

A	*R.B*	R.C	S.B	S.C	*D*
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

What if we consider $R_{A<D \textbf{ AND } R.B > S.B}$ S !?!?

CSC43 N. Koudas 30



Natural Join

Syntax: $R = R_1 \bowtie R_2$

is equivalent to $R = \pi_{R_1 \cup R_2}(\sigma_C(R_1 \times R_2))$

Semantic: Similar to the theta-join of R_1 and R_2 with the condition that all attributes of the same name be equated. Then, one column for each pair of equated attributes is projected out.

Example:

A	*B*
1	2
3	4

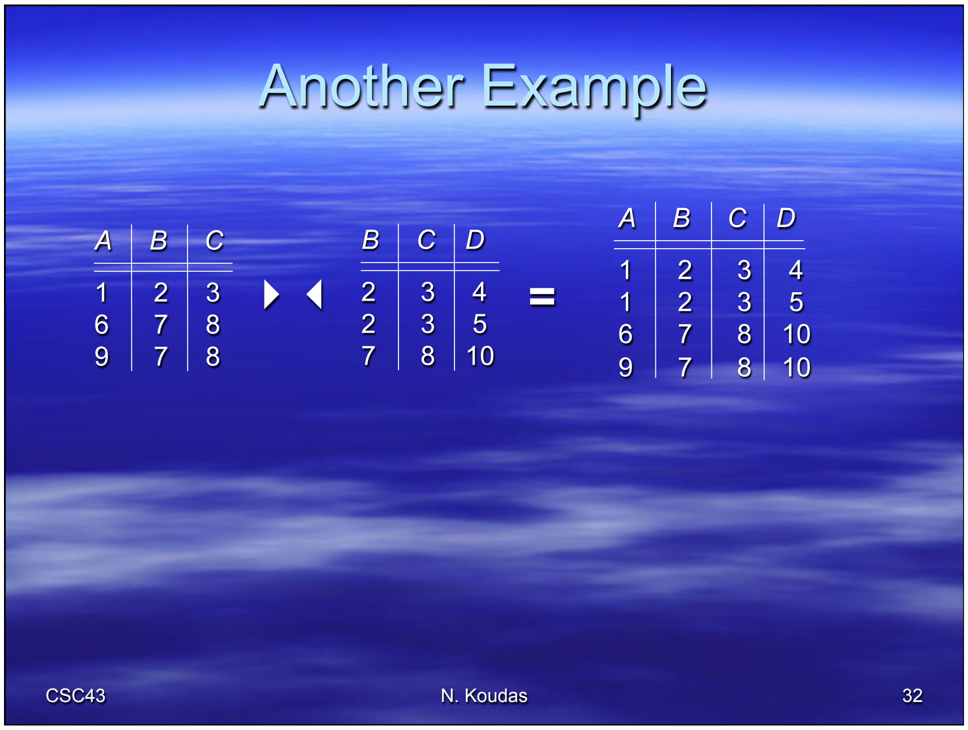
$\bowtie$

B	*C*	*D*
2	5	6
4	7	8
9	10	11

=

A	*B*	*C*	*D*
1	2	5	6
3	4	7	8

CSC43 N. Koudas 31



Another Example

A	*B*	*C*
1	2	3
6	7	8
9	7	8

$\bowtie$

B	*C*	*D*
2	3	4
2	3	5
7	8	10

=

A	*B*	*C*	*D*
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

CSC43 N. Koudas 32

That’s it for today…