CS5300
Database Systems

Relational Algebra

A.R. Hurson
323 CS Building
hurson@mst.edu

Database Systems

- This module is intended to introduce:
  - relational algebra as the backbone of relational model, and
  - set of operations defined within the scope of relational model.

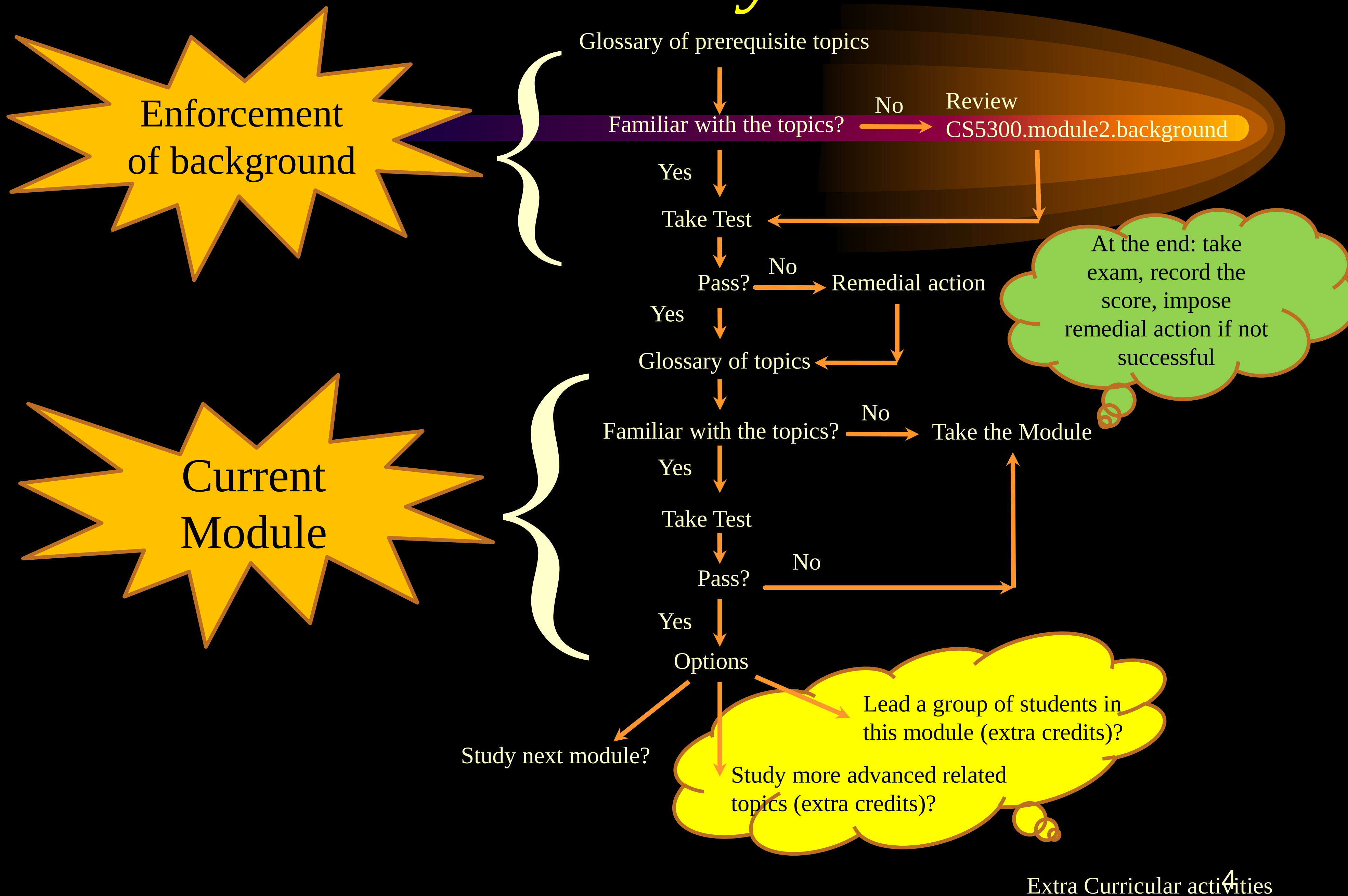
Database Systems

Note, this unit will be covered in two lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS5300.module3
2) Study the supplement module (supplement CS5300.module2)
3) Act as a helper to help other students in studying CS5300.module2

Note, options 2 and 3 have extra credits as noted in course outline.

Database Systems



Database Systems

- You are expected to be familiar with:
  - Relational data model
  - Basic operations in relational model
- If not, you need to study CS5300.module2.background

Database Systems

- Relational Algebra
  - Relational model is based on set theory. A relation is simply a set.
  - The relational algebra is a set of high level operators that operate on relations. Each relational operator assumes one or two relations as input (unary or binary operators) and produces a relation as output. Relational algebra is composed of eight operators divided into two groups.

Database Systems

- Relational Algebra
  - The traditional set operations:

    UNION, INTERSECTION, DIFFERENCE, and CARTESIAN PRODUCT.
  - The special relational operations:

    RESTRICT, PROJECT, JOIN, and DIVIDE.

Database Systems

- Relational Algebra
  - Two relations are called Union-compatible if and only if they have identical headings, more precisely;
    - They have the same set of attributes, and
    - Corresponding attributes are defined on the same domain.
  - Union, Intersection, and Difference require their operands to be union-compatible.

Database Systems

- Set Operation
  - Union: The union of two union-compatible relations *A* and *B* is a relation with the same heading as *A* and *B* (same schema) and a body consisting of the set of all tuples *t* belonging to either *A* or *B* (or both).

$$A \cup B = \{t \mid t \in A \text{ or } t \in B\}$$

Database Systems

- Set Operation
  - Intersection: Intersection of two union-compatible relations *A* and *B* is a relation with the same heading as each of *A* and *B* (same schema) and with a body consisting of the set of the tuples *t* belonging to both *A* and *B*.

$$A \cap B = \{t \mid t \in A \text{ and } t \in B\}$$

Database Systems

- Set Operation
  - Difference: Difference between two union-compatible relations *A* and *B* is a relation with the same heading as *A* and *B* (same schema) and with a body consisting of the set of tuples belonging to *A* and not to *B*.

$$A - B = \{t \mid t \in A \text{ and } t \notin B\}$$

Database Systems

- Set Operation
  - Cartesian Product: Cartesian product of two product-compatible (disjoint headings) relations *A* and *B*, is a relation with a heading that is the coalescing of the headings of *A* and *B* (concatenated schemas) and a body consisting of the set of tuples *t* such that *t* is the concatenation of a tuple *a* from *A* and a tuple *b* from *B*.

$$A \text{ X } B = \{<a,b> \mid a \in A \text{ and } b \in B\}$$

Database Systems

- Assume the following two relations:

A

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris

Database Systems

$A \cup B =$

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London
S2	Jones	10	Paris

$A \cap B =$

S#	SNAME	STATUS	CITY
S1	Smith	20	London

$A — B =$

S#	SNAME	STATUS	CITY
S4	Clark	20	London

Database Systems

- Set Operation
  - Union, Intersect, and Product are associative operations;

$$A \cup (B \cup C) \equiv (A \cup B) \cup C$$

  - Union, Intersect, and Product are commutative operations;

$$A \cup B \equiv B \cup A$$

  - however, $A - B \neq B - A$

Database Systems

- Special Relational Operations
  - Restriction (Select): Let $\theta \in \{=, <, >, \neq, \ldots\}$, the θ-restriction of relation *A* on attributes *X* and *Y* is a relation with the same heading as *A* and a body consisting of the set of all tuples *t* in *A* that satisfy $X\ \theta\ Y$.

$$\sigma_{X\ \theta\ Y}(A) = \{t \mid t \in A \text{ and } (t_x \theta\ t_y \text{ is true})\}$$

  - θ-restriction effectively yields a horizontal subset of a given relation.

Database Systems

S

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

P

P#	Pname	Color	Weight	City
P_1	Nut	Red	12	London
P_2	Bolt	Green	17	Paris
P_3	Screw	Blue	17	Rome
P_4	Screw	Red	14	London
P_5	Cam	Blue	12	Paris
P_6	Cog	Red	19	London

Database Systems

$\sigma_{city=\text{“London”}}(S) =$

S#	Sname	Status	City
S_1	Smith	20	London
S_4	Clark	20	London

$\sigma_{Weight\ <\ 14}(P) =$

P#	Pname	Color	Weight	City
P_1	Nut	Red	12	London
P_5	Cam	Blue	12	Paris

Database Systems

Special Relational Operations

- Projection: Projection of a relation *A* on attributes *X*, *Y*, ….., is a relation with the heading of (X, Y, …) and a body consisting of all tuples (X:x, Y:y, …) such that a tuple *t* in *A* with the same X-value x, Y-value y, …

$$\Pi_{(X,Y,\ldots)}(A) = \{r \mid r_x = t_x,\ r_y = t_y, \ldots \text{ and } t \in A\}$$

- Projection operator effectively yields vertical slice of a given relation.

Database Systems

$\Pi_{(City)}(S) =$

City
London
Paris
Athens

$\Pi_{(Color,\ City)}(P) =$

Color	City
Red	London
Green	Paris
Blue	Rome
Blue	Paris

Database Systems

- Special Relational Operations
  - Join (restrictive Cartesian product): Join operation comes in several variations:
    - Natural Join
    - θ-Join
    - Outer Join

$$A \bowtie_c B = \sigma_c(A \times B)$$

Database Systems

- Natural Join: Natural join of two relations *A* and *B* is a relation consisting of tuples *t* = <*a*, *b*> such that *a* is a *tuple* of *A* and *b* is a tuple of *B* and the common attribute values of *A* and *B* are equal — Equijoin on all common fields.

$$A \bowtie_c B = \{ \langle a,b \rangle \mid a \in A,\ b \in B, \text{ and } a_c = b_c \}$$

- Natural join is both associative and commutative.

Database Systems

- θ-Join: Let relations A and B be product-compatible (no attribute names in common) and let $\theta \in \{<, >, \neq, \ldots\}$, then the θ-Join of relation A on attribute X with relation B on attribute Y is a relation consist of tuples t from Cartesian product of A and B such that $t_x\ \theta\ t_y$ is true.

$$A \bowtie_{\theta} B = \{ <a,b> \mid <a,b> \in A \times B, \text{ and } a_X\ \theta\ b_Y \text{ is true} \}$$

Database Systems

$A \bowtie_{S.City=P.City} B$

S#	Sname	Status	S.City	P#	Pname	Color	Weight	P.City
S_1	Smith	20	London	P_1	Nut	Red	12	London
S_1	Smith	20	London	P_4	Screw	Red	14	London
S_1	Smith	20	London	P_6	Cog	Red	19	London
S_2	Jones	10	Paris	P_2	Bolt	Green	17	Paris
S_2	Jones	10	Paris	P_5	Cam	Blue	12	Paris
S_3	Blake	30	Paris	P_2	Bolt	Green	17	Paris
S_3	Blake	30	Paris	P_5	Cam	Blue	12	Paris
S_4	Clark	20	London	P_1	Nut	Red	12	London
S_4	Clark	20	London	P_4	Screw	Red	14	London
S_4	Clark	20	London	P_6	Cog	Red	19	London

Database Systems

- Division: Let *A* and *B* be two relations where set of *B* attributes are included in the one of *A* — A (X_1, X_2, …., X_m, Y_1, Y_2, …, Y_n) and B (Y_1, Y_2, …, Y_n), then A ÷ B is a relation with the heading (X) and a body of the set of all tuples (X;x) such that (X:x, Y:y) is in *A* for all tuples (Y:y) in *B*.

$$A \div B = \{ \langle a\ (X) \rangle \mid \exists \langle a,b \rangle \in A ,\ \forall \langle b \rangle \in B \}$$

Database Systems

- Consider the following relations:

A

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

B

P#
P1

B’

P#
P2
P4

B”

P#
P1
P2
P3
P4
P5
P6

Database Systems

A ÷ B =

S#
S1
S2

A ÷ B’ =

S#
S1
S4

A ÷ B” =

S#
S1

A

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

B

P#
P1

B’

P#
P2
P4

B”

P#
P1
P2
P3
P4
P5
P6

Database Systems

Missing Information

- The problem of missing or unknown information is a very important data base issue. Information is very often incomplete. So we need a way of defining such incompleteness and a way to manipulate incomplete data bases. Discussion of the use of the Universal relation assumption has also increased interest in dealing with null values.
- A special column value called null ($\perp$) is used to represent incomplete data.

Database Systems

◆ Missing Information

- A partial tuple is one that contains one or more null values (⊥). A tuple *t* is said to be total (t↓) if and only if it contains no null values. This definition can be extended to relations as well.
- A relation *A* is total (A ↓) if all of its tuples are total.
- Relational operations using a search condition to select tuples will be basically affected by the incomplete data.

Database Systems

◆ Missing Information

- Codd proposed the concept of three-valued logic as the basis to manipulate null values.
- Three-valued logic

NOT	
T	F
ω	ω
F	T

AND	T	ω	F
T	T	ω	F
ω	ω	ω	F
F	F	F	F

OR	T	ω	F
T	T	T	T
ω	T	ω	ω
F	T	ω	F

Database Systems

- Maybe Algebra
  - A truth-valued (logic) expression has the value of ω if and only if:
    - Each occurrence of a null value in the expression can be replaced by a non-null value so as to yield the truth value T for the expression.
    - Each occurrence of a null value in the expression can be replaced by a non-null value so as to yield the truth value F for the expression.

Database Systems

Maybe Algebra

- Assume the following two relations

r(R)

A	B	C_r
a_1	b_1	1
a_2	b_2	$\perp$
a_3	b_3	2

s(S)

C_s	D
1	d_1
$\perp$	d_2

Database Systems

- True Select *r* when *C=1*:

A	B	C_r
a_1	b_1	1

r(R)

A	B	C_r
a_1	b_1	1
a_2	b_2	$\perp$
a_3	b_3	2

- Maybe Select *r* when *C=1*:

A	B	C_r
a_2	b_2	$\perp$

Database Systems

- True Join *r* and *s* over *C*:

A	B	C_r	C_s	D
a_1	b_1	1	1	d_1

- Maybe Join *r* and *s* over *C*:

A	B	C_r	C_s	D
a_1	b_1	1	$\perp$	d_2
a_2	b_2	$\perp$	1	d_1
a_2	b_2	$\perp$	$\perp$	d_2
a_3	b_3	2	$\perp$	d_2

r(R)

A	B	C_r
a_1	b_1	1
a_2	b_2	$\perp$
a_3	b_3	2

s(S)

C_s	D
1	d_1
$\perp$	d_2

Database Systems

- Maybe Division: Assume the following two relations:

r(R)

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_1	b_2	c_2
a_2	b_2	$\perp$

s(S)

B	C
b_1	c_1
b_2	c_2

Database Systems

- True Division ($r \div s$):

A
a_1

r(R)

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_1	b_2	c_2
a_2	b_2	$\perp$

- Maybe Division ($r \div s$):

A
a_2

s(S)

B	C
b_1	c_1
b_2	c_2

Database Systems

- A somewhat different views of the null value is taken for the remaining relational algebra operations. For these operations, null values in different tuples are interpreted as being equivalent. Therefore, there is no changing in these operations over their use in traditional relational algebra.

r(R)

A	B	C
a_1	$\perp$	c_1
a_1	$\perp$	c_2
a_2	b_2	$\perp$
a_3	b_3	c_3

$\Pi_{(A,B)} r$

A	B
a_1	$\perp$
a_2	b_2
a_3	b_3

Database Systems

- Missing Information
  - Concept of null values can be used to bring the so-called dangling tuples into the join process and make relations union-compatible.

Database Systems

- Outer Join ⓣ: This is a direct consequence of dealing with incompleteness. In outer join, tuples in the join relations that do not match the join condition, will be extended by nulls and appear in the resultant relation.
- There are several variation of outer join:
  - Left outer join
  - right outer join
  - full outer join

Database Systems

- Outer Join r(A,B) and s(C,D) over $B\ \theta\ C$ (B and C are defined over the same domain and θ is the test condition) is defined as follows:

$$r \ \theta\ s = T \cup (R_1 \ X \ (C : \perp, D : \perp)) \cup ((A : \perp, B : \perp) \ X \ S_1)$$

Where:

T is the true Join of r and s over $B\ \theta\ C$,

$R_1 = r — \Pi_{(A,B)}(T)$

$S_1 = s — \Pi_{(C,D)}(T)$

Database Systems

- Outer Join: An example, assume θ is equality, and

r

A	B
a_1	b_1
a_2	b_2
a_3	b_3

s

C	D
b_1	d_1
b_2	d_2
b_4	d_3

Database Systems

$r \, \theta \, s = T \cup (R_1 \times (C:\perp, D:\perp)) \cup ((A:\perp, B:\perp) \times S_1)$

$R_1 = r - \Pi_{(A,B)}(T)$

$S_1 = s - \Pi_{(C,D)}(T)$

T

A	B	C	D
a_1	b_1	b_1	d_1
a_2	b_2	b_2	d_2

R_1

A	B
a_3	b_3

S_1

C	D
b_4	d_3

$r \, \theta \, s$

A	B	C	D
a_1	b_1	b_1	d_1
a_2	b_2	b_2	d_2
a_3	b_3	$\perp$	$\perp$
$\perp$	$\perp$	b_4	d_3

Database Systems

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

Database Systems

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Burton	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	11	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

Boats

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

Database Systems

Reserved

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

Database Systems

- Find names of sailors who have reserved a red boat

$$\Pi_{sname}((\ \sigma_{color=\text{“red”}} Boats) \bowtie Reserves \bowtie Sailors)$$

sname
Dustin
Lubber
Horatio

Database Systems

- A more efficient solution would do the projections as early as possible:

$$\Pi_{sname}(\Pi_{sid}(\ (\Pi_{bid}\ (\sigma_{color=\text{“red”}}\text{Boats})\ )\ \bowtie\ \text{Reserves})\bowtie\ \text{Sailors})$$

Database Systems

Questions

- Find sailors that have reserved a green boat and a red boat.
  - Identify sailors reserving red boats
  - Identify sailors reserving green boats
  - Intersect the sets and determine the sailors name
- What are the names of the sailors with the lowest rating (whatever that rating is).
  - Compare the rating of each sailor with that of every other sailor
  - Result is the set of sailors that never come out on top in one of these comparisons.