COSI 127b
Introduction to Database Systems

Lecture 11: E/R (cont), Theoretical DB Design

What a DBMS Manages

1. Data Organization
   - Logical: Relational Data Model, Database Design Techniques
2. Data Retrieval
   - Logical: Query Languages: RA, TRC, SQL
3. Data Integrity
   - Logical: Transactions, Integrity Constraints

Review: Good DB Design

Three Approaches:

1. Ad hoc:
   - use Entity-Relationship Model to model data requirements
   - translate ER design into relational schema

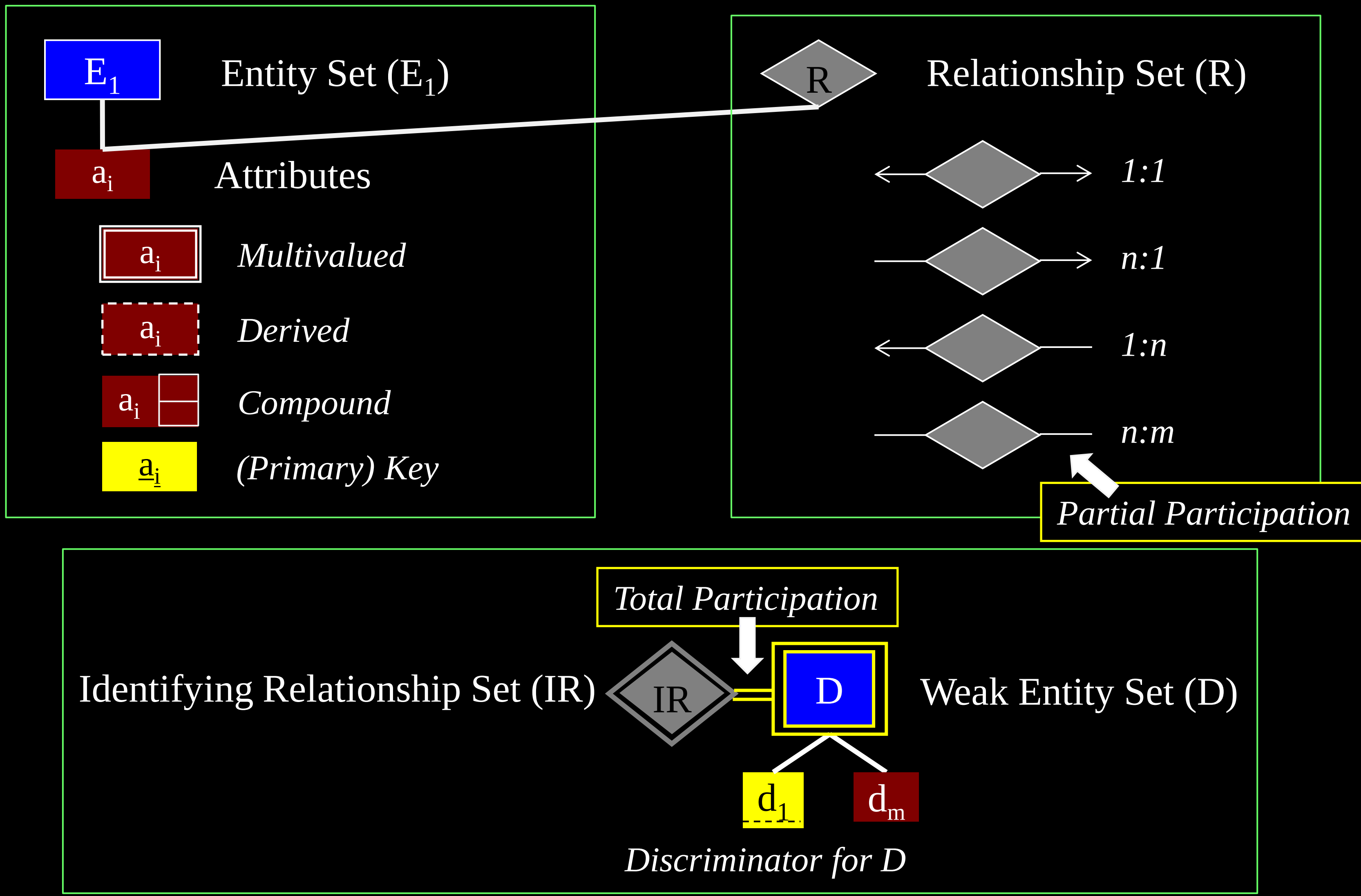
   Issue: How to tell if design is "good"?

2. Theoretical:
   - construct universal relations (e.g., `Borrower-All`)
   - decompose above using known functional dependencies

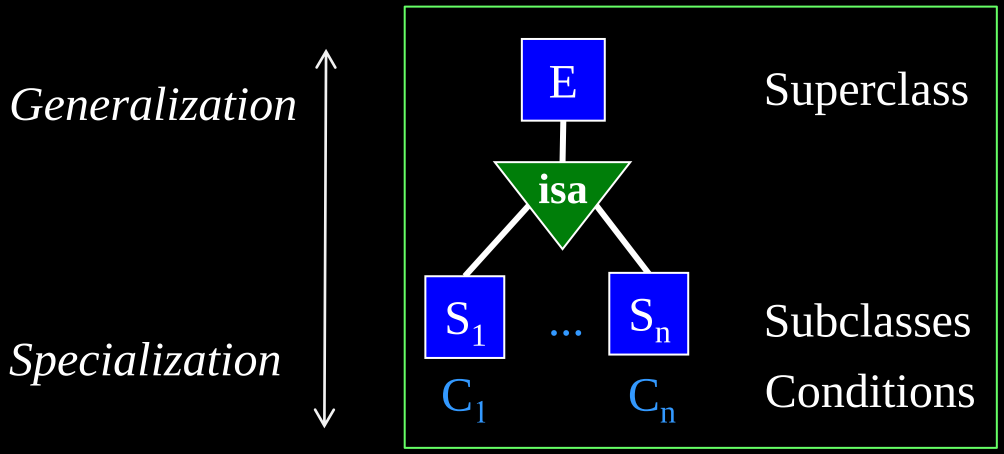
   Issue: Time-Consuming and Complex

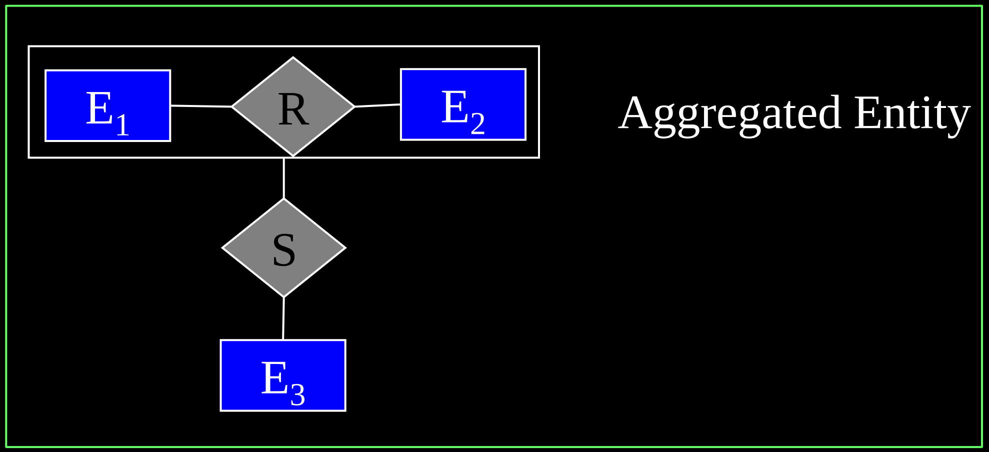
3. Practical:
   - use ER Model to produce 1st cut DB design
   - use FDs to refine and verify

Review: E/R Cheat Sheat

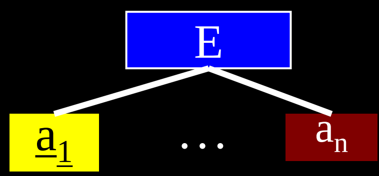


Review: E/R Cheat Sheat

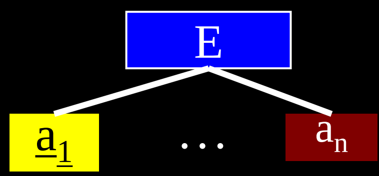




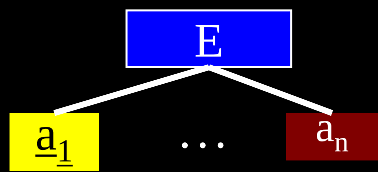
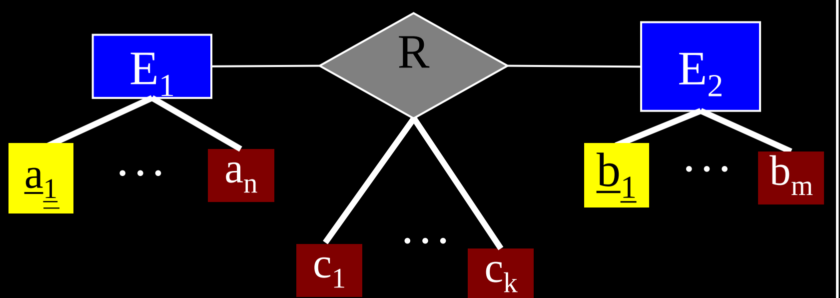
E/R Diagrams and Relations

E/R	Relational Schema
Entity Sets 	

E/R Diagrams and Relations

E/R	Relational Schema
Entity Sets 	`E = (`$\underline{a_1}$`, …, `a_n`)`

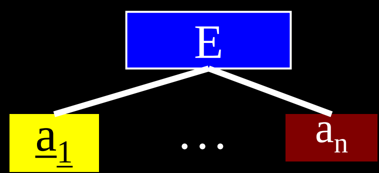
E/R Diagrams and Relations

E/R	Relational Schema
Entity Sets 	$E = (\underline{a_1}, \ldots, a_n)$
Relationship Sets 	

E/R Diagrams and Relations

<table>
<tr><th>E/R</th><th>Relational Schema</th></tr>
<tr><td>

Entity Sets

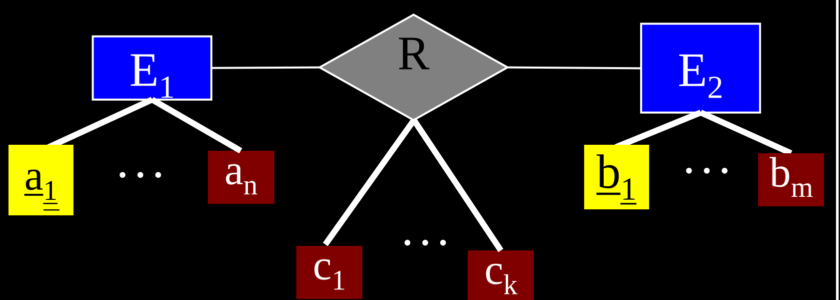


</td><td>

$E = (\underline{a_1}, \ldots, a_n)$

</td></tr>
<tr><td>

Relationship Sets



</td><td>

$R = (\underline{a_1, b_1}, c_1, \ldots, c_n)$

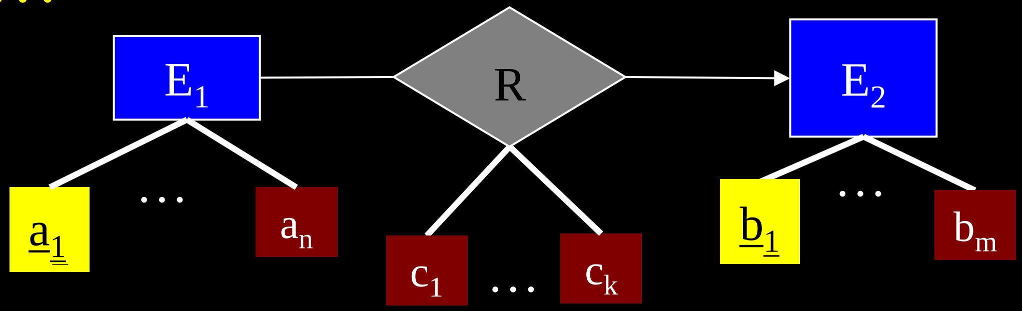
a_1: E_1's key
b_1: E_2's key
$c_1, \ldots, c_k$: attributes of R

</td></tr>
</table>

Not the whole story for Relationship Sets …

E/R Diagrams and Relations

What about…



Could have: $\texttt{R = (}\underline{\texttt{a}_1\texttt{,b}_1}\texttt{,c}_1\texttt{,…,c}_k\texttt{)}$ but…

- a_1 is a key for $\texttt{E}_1\texttt{ = (}\underline{\texttt{a}_1}\texttt{,…,a}_n\texttt{)}$
- a_1 is also a key for $\texttt{R}$

Instead:

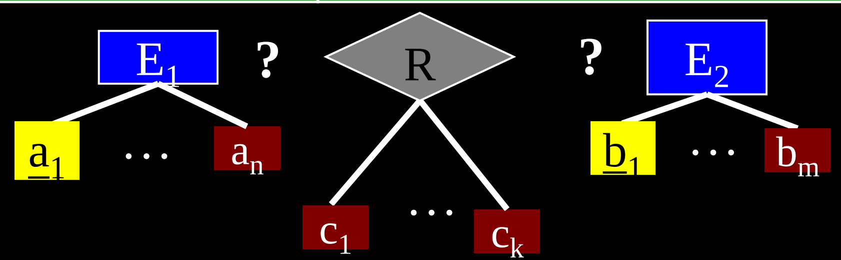
- Ignore $\texttt{R}$
- Add $\texttt{b}_1\texttt{,c}_1\texttt{,…,c}_k$ to $\texttt{E}_1$ instead (i.e: $\texttt{E}_1\texttt{ = (}\underline{\texttt{a}_1}\texttt{,…,a}_n\texttt{,b}_1\texttt{,c}_1\texttt{,…,c}_k\texttt{)}$)

__Rule of Thumb__

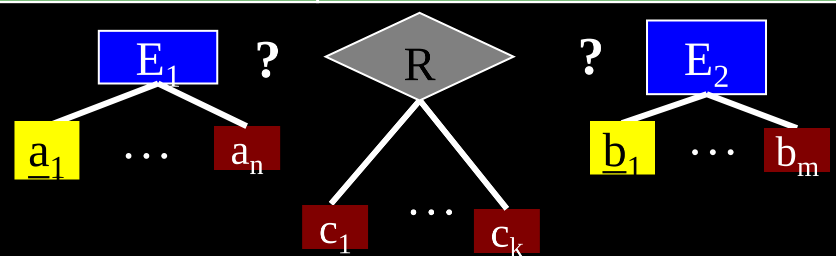
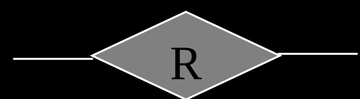
Fewer tables good, as long as no redundancy

E/R Diagrams and Relations

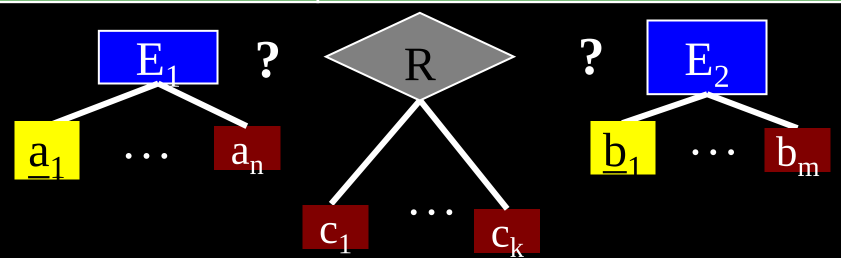
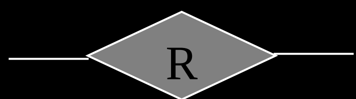
Relationship Cardinality	Relational Schema



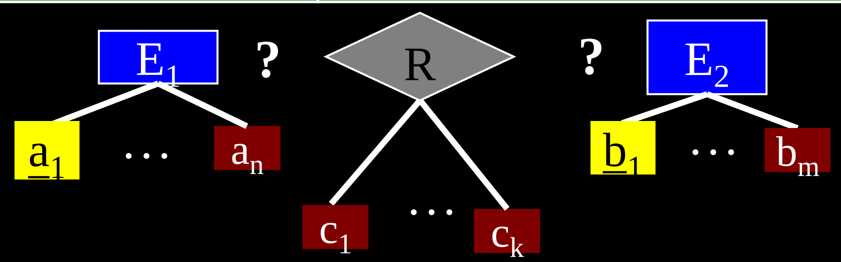
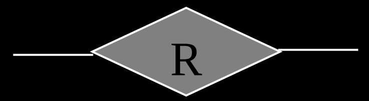
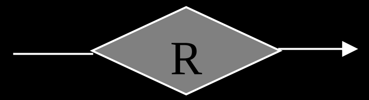
E/R Diagrams and Relations

<table>
<tr><th>Relationship Cardinality</th><th>Relational Schema</th></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>n:m

</td><td></td></tr>
</table>

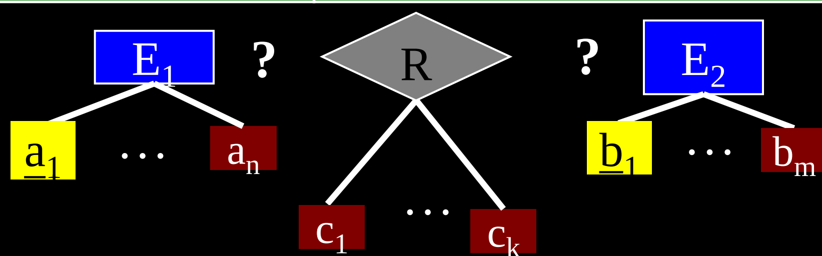
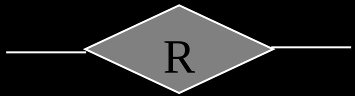
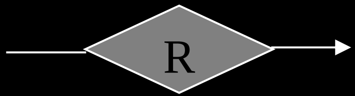
E/R Diagrams and Relations

<table>
<tr><th>Relationship Cardinality</th><th>Relational Schema</th></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>n:m

</td><td>$E_1 = (\underline{a_1}, \ldots, a_n)$
$E_2 = (\underline{b_1}, \ldots, b_m)$
$R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$</td></tr>
</table>

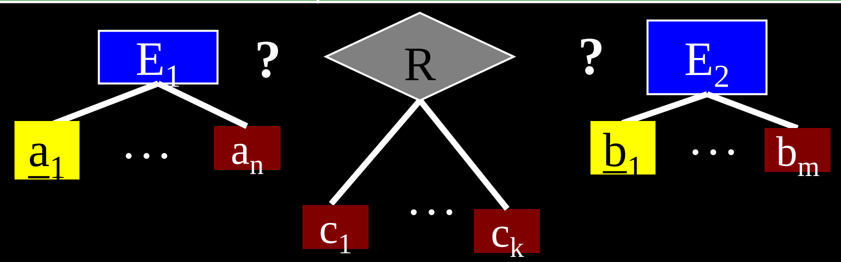
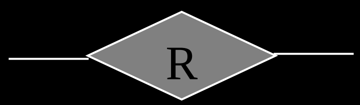
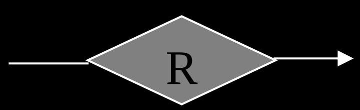
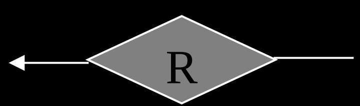
E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	

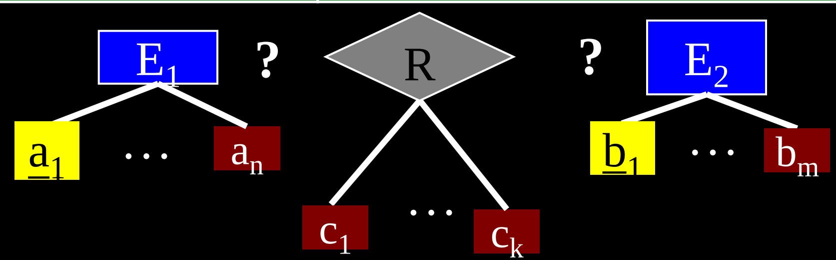
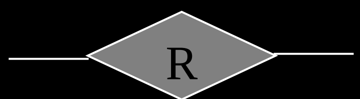
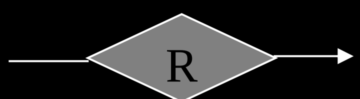
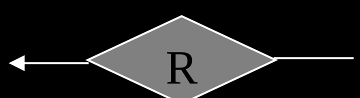
E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_k)$ $E_2 = (\underline{b_1}, \ldots, b_m)$

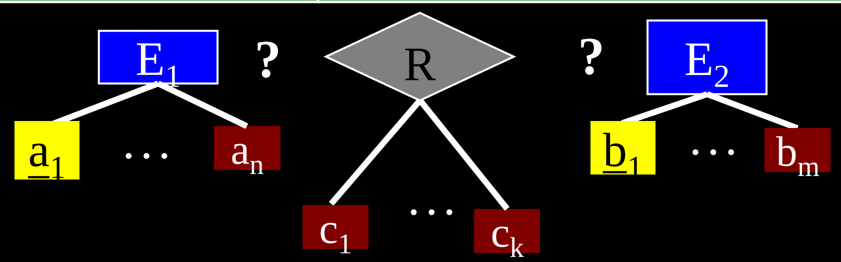
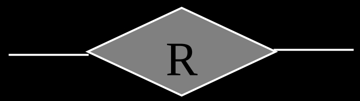
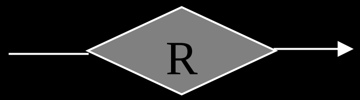
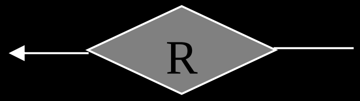
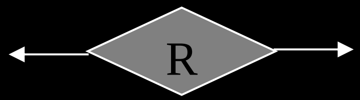
E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_k)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n 	

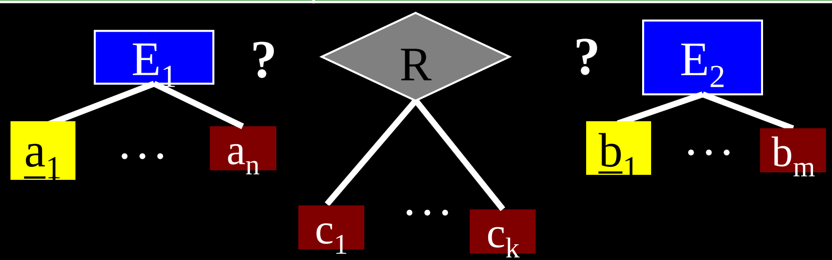
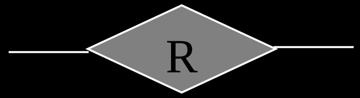
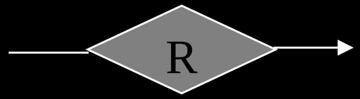
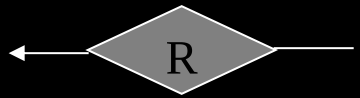
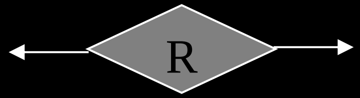
E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_{n,} b_1, c_1, \ldots, c_k)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_{m,}, a_1, c_1, \ldots, c_k)$

E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_k)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m, a_1, c_1, \ldots, c_k)$
1:1 	

E/R Diagrams and Relations

Relationship Cardinality	Relational Schema
	
n:m 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m)$ $R = (\underline{a_1, b_1}, c_1, \ldots, c_k)$
n:1 	$E_1 = (\underline{a_1}, \ldots, a_n, b_1, c_1, \ldots, c_k)$ $E_2 = (\underline{b_1}, \ldots, b_m)$
1:n 	$E_1 = (\underline{a_1}, \ldots, a_n)$ $E_2 = (\underline{b_1}, \ldots, b_m, a_1, c_1, \ldots, c_k)$
1:1 	Treat as n:1 or 1:n

Exercise: Draw the ER diagram for...

5

Account

bname	acct_no	balance
Downtown	A-101	500
Mianus	A-215	700
Perry	A-102	400
R.H.	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750

6

Depositor

cname	acct_no
Johnson	A-101
Smith	A-215
Hayes	A-102
Turner	A-305
Johnson	A-201
Jones	A-217
Lindsay	A-222

1

Customer

cname	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

Branch 2

bname	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

Borrower 4

cname	lno
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17
Adams	L-16

Loan 3

bname	lno	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

Exercise: Draw the ER diagram for...

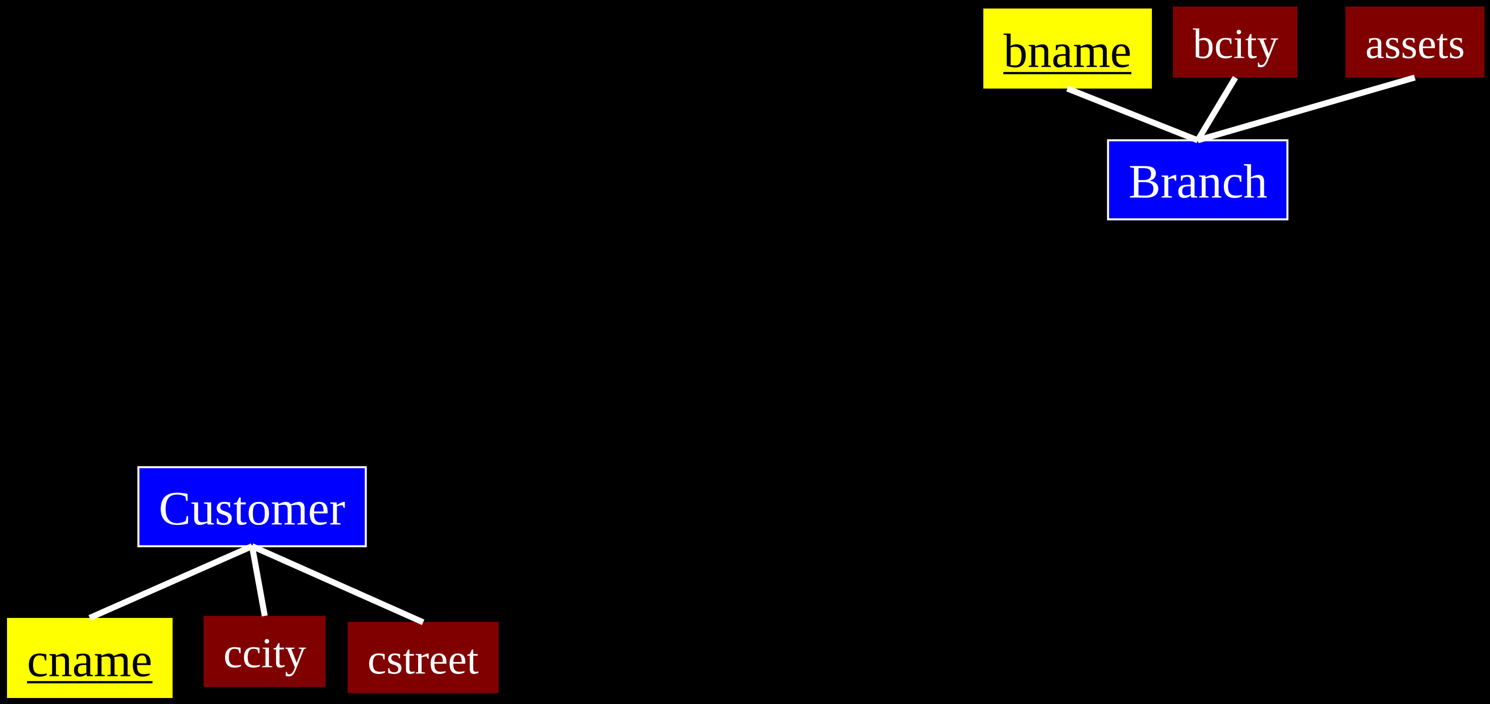
Branch		
bname	**bcity**	**assets**
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

2

Customer		
cname	**cstreet**	**ccity**
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

1

Exercise: Draw the ER diagram for...



Exercise: Draw the ER diagram for...

Branch		
<u>bname</u>	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

2

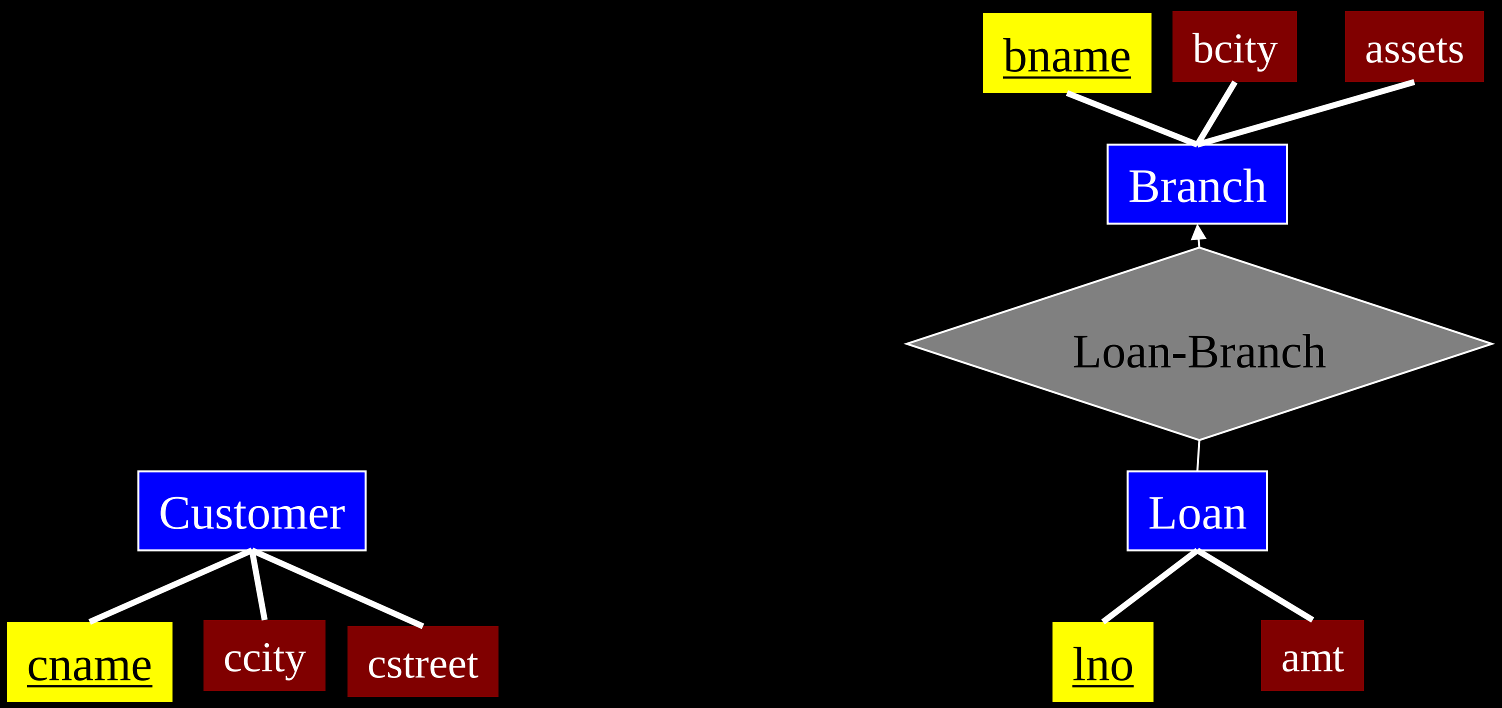
1

Customer		
<u>cname</u>	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

Loan		
bname	<u>lno</u>	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

3

Exercise: Draw the ER diagram for...



Exercise: Draw the ER diagram for...

2

Branch		
<u>bname</u>	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

4

Borrower	
cname	lno
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17
Adams	L-16

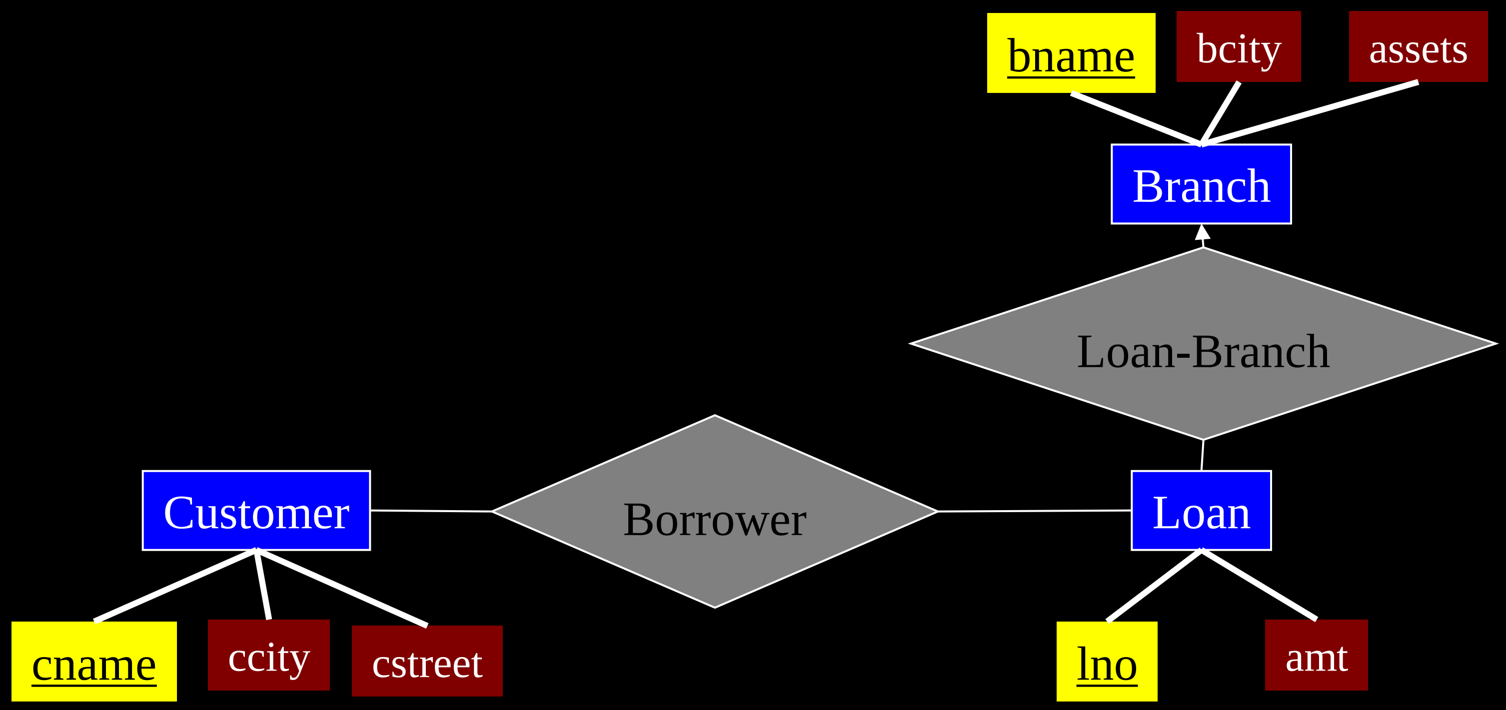
1

Customer		
<u>cname</u>	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

3

Loan		
bname	<u>lno</u>	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

Exercise: Draw the ER diagram for...



Exercise: Draw the ER diagram for...

5

Account		
bname	acct_no	balance
Downtown	A-101	500
Mianus	A-215	700
Perry	A-102	400
R.H.	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750

2

Branch		
bname	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

4

Borrower	
cname	lno
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17
Adams	L-16

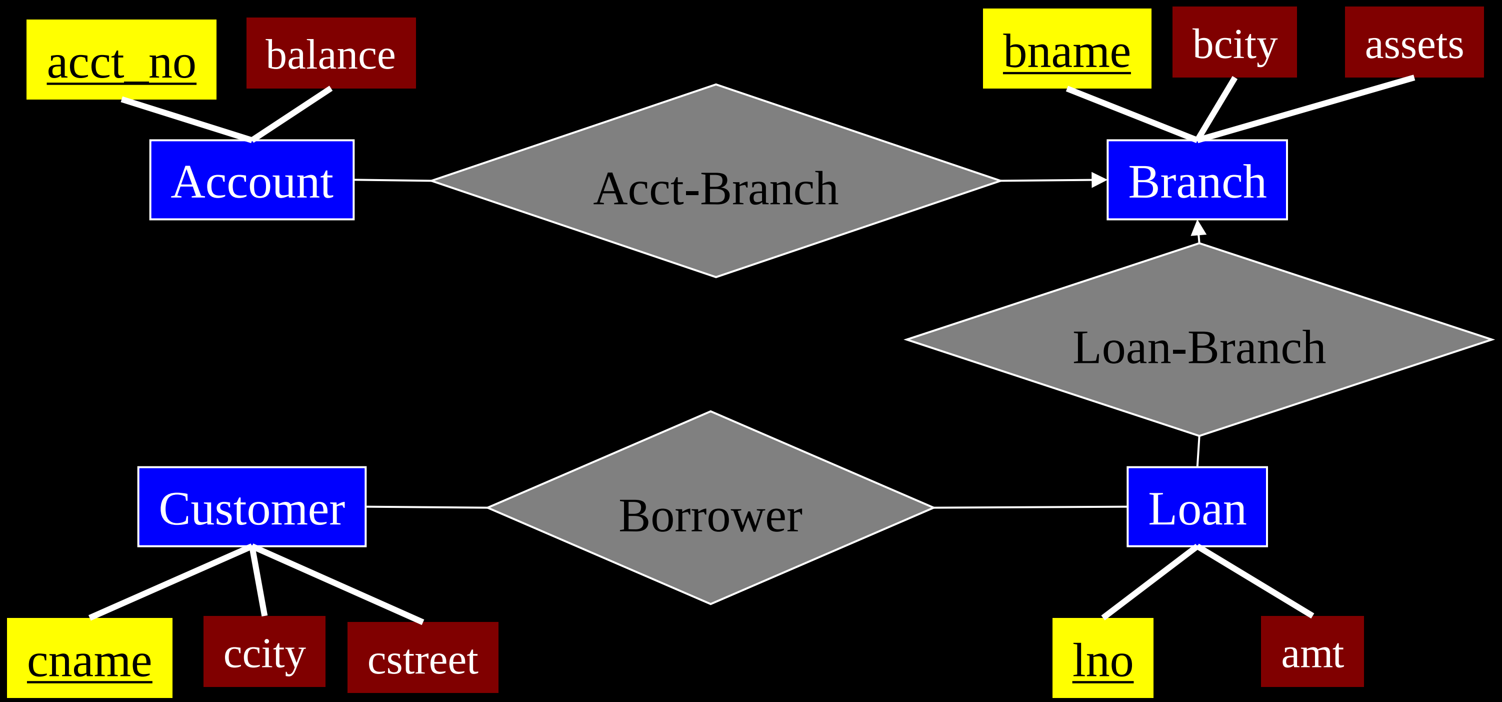
1

Customer		
cname	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

3

Loan		
bname	lno	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

Exercise: Draw the ER diagram for...



Exercise: Draw the ER diagram for...

5

Account		
bname	acct_no	balance
Downtown	A-101	500
Mianus	A-215	700
Perry	A-102	400
R.H.	A-305	350
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	750

6

Depositor	
cname	acct_no
Johnson	A-101
Smith	A-215
Hayes	A-102
Turner	A-305
Johnson	A-201
Jones	A-217
Lindsay	A-222

1

Customer		
cname	cstreet	ccity
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stanford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stanford

2

Branch		
bname	bcity	assets
Downtown	Brooklyn	9M
Redwood	Palo Alto	2.1M
Perry	Horseneck	1.7M
Mianus	Horseneck	0.4M
R.H.	Horseneck	8M
Pownel	Bennington	0.3M
N. Town	Rye	3.7M
Brighton	Brooklyn	7.1M

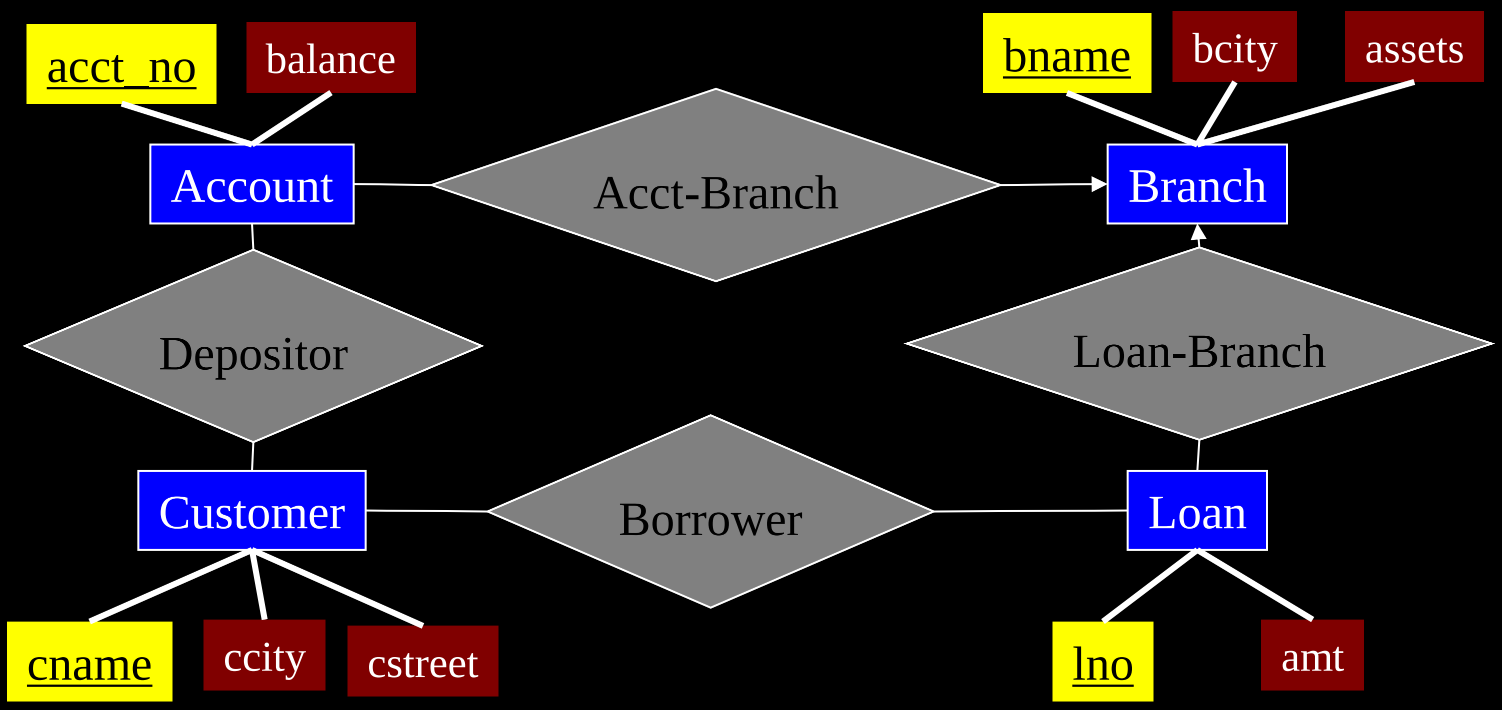
4

Borrower	
cname	lno
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17
Adams	L-16

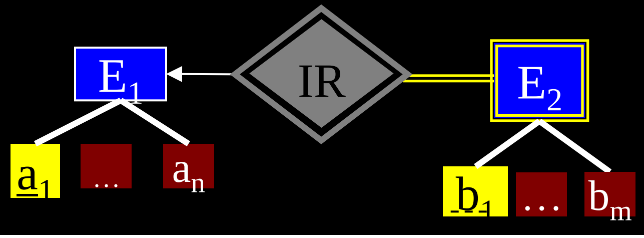
3

Loan		
bname	lno	amt
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
R.H.	L-11	900
Perry	L-16	1300

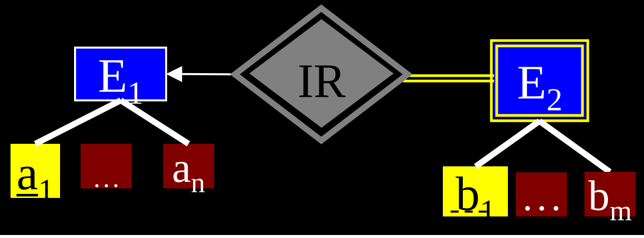
Exercise: Draw the ER diagram for...



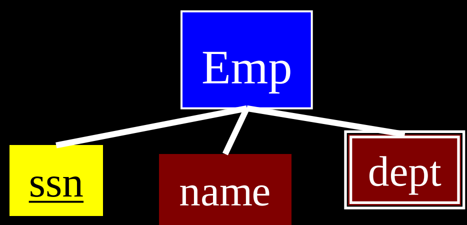
E/R Diagrams and Relations

E/R	Relational Schema
Weak Entity Sets	
	

E/R Diagrams and Relations

<table>
<tr><th>E/R</th><th>Relational Schema</th></tr>
<tr><td colspan="2">Weak Entity Sets</td></tr>
<tr><td>
</td><td>$E_1 = (\underline{a_1}, \ldots, a_n)$
$E_2 = (\underline{a_1, b_1}, \ldots, b_m)$</td></tr>
</table>

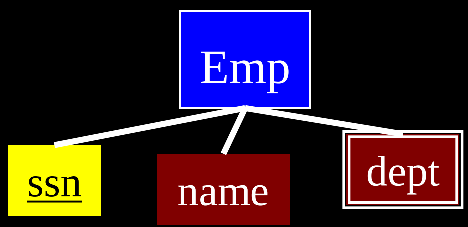
E/R Diagrams and Relations

E/R	Relational Schema
Multivalued Attributes	
	

E/R Diagrams and Relations

E/R	Relational Schema

Multivalued Attributes




```
Emp        = (ssn,name)
Emp-Depts  = (ssn,dept)
```

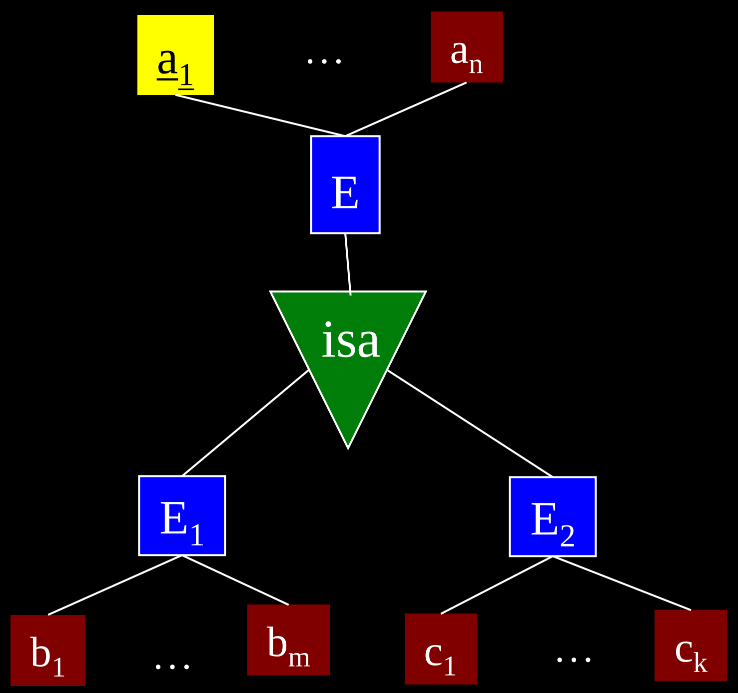
ssn	name
001	Smith
…	…

Emp

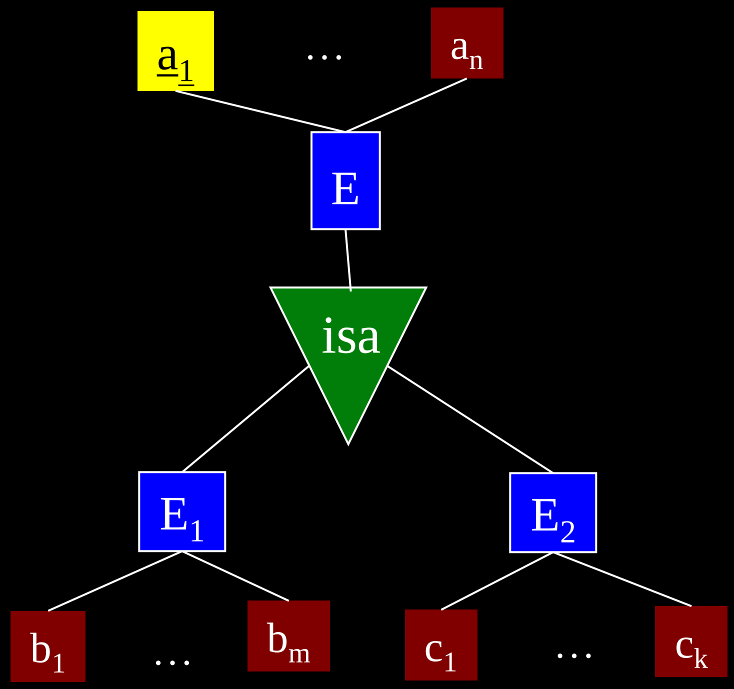
ssn	dept
001	Acct
001	Sales
…	…

Emp-Depts

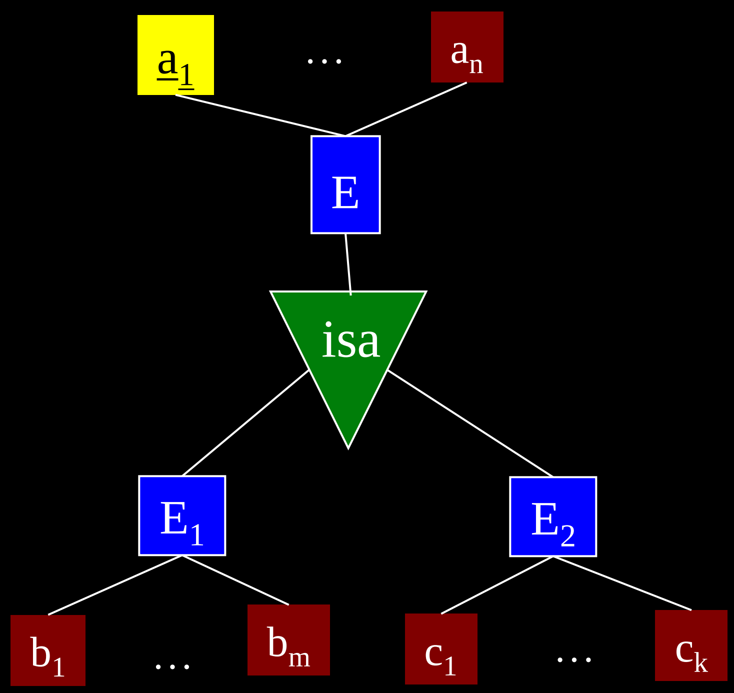
E/R Diagrams and Relations

E/R	Relational Schema
Subclasses	
	

E/R Diagrams and Relations

E/R	Relational Schema
Subclasses	
	***Method 1:*** $E = (\underline{a_1}, \ldots, a_n)$ $E_1 = (\underline{a_1}, b_1, \ldots, b_m)$ $E_2 = (\underline{a_1}, c_1, \ldots, c_k)$

E/R Diagrams and Relations

E/R	Relational Schema
Subclasses	
	***Method 1:*** $E = (\underline{a_1}, \ldots, a_n)$ $E_1 = (\underline{a_1}, b_1, \ldots, b_m)$ $E_2 = (\underline{a_1}, c_1, \ldots, c_k)$ ***Method 2:*** $E_1 = (\underline{a_1}, \ldots, a_n, b_1, \ldots, b_m)$ $E_2 = (\underline{a_1}, \ldots, a_n, c_1, \ldots, c_k)$

E/R Diagrams and Relations

Subclasses example:

Method 1:

```
Account   =  (acct_no, balance)
SAccount  =  (acct_no, interest)
CAccount  =  (acct_no, overdraft)
```

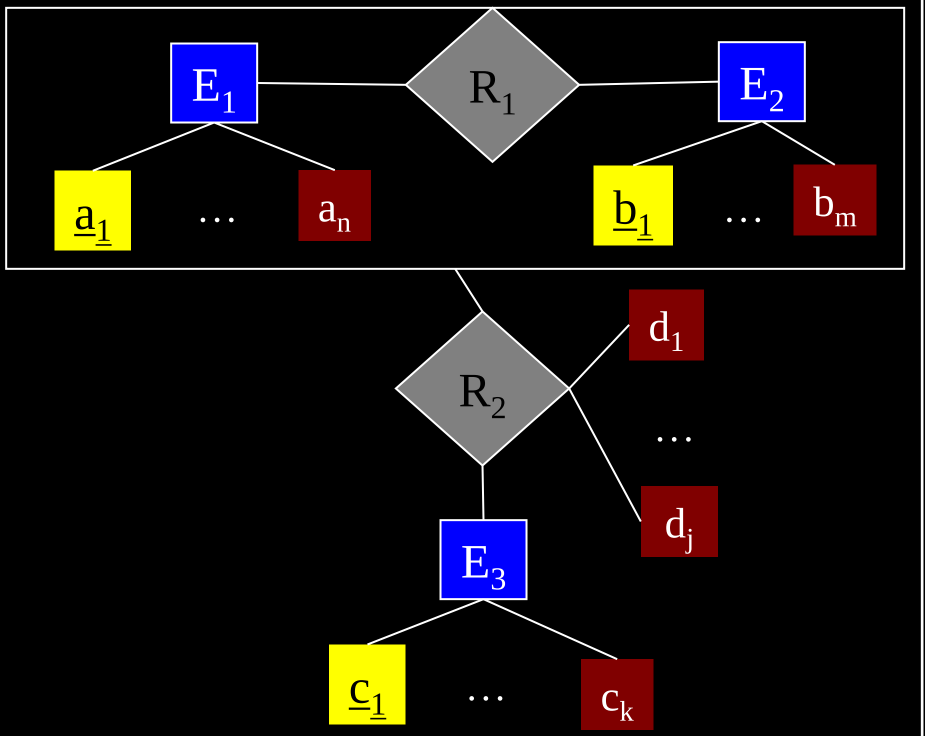
Method 2:

```
SAccount  =  (acct_no, balance, interest)
CAccount  =  (acct_no, balance, overdraft)
```

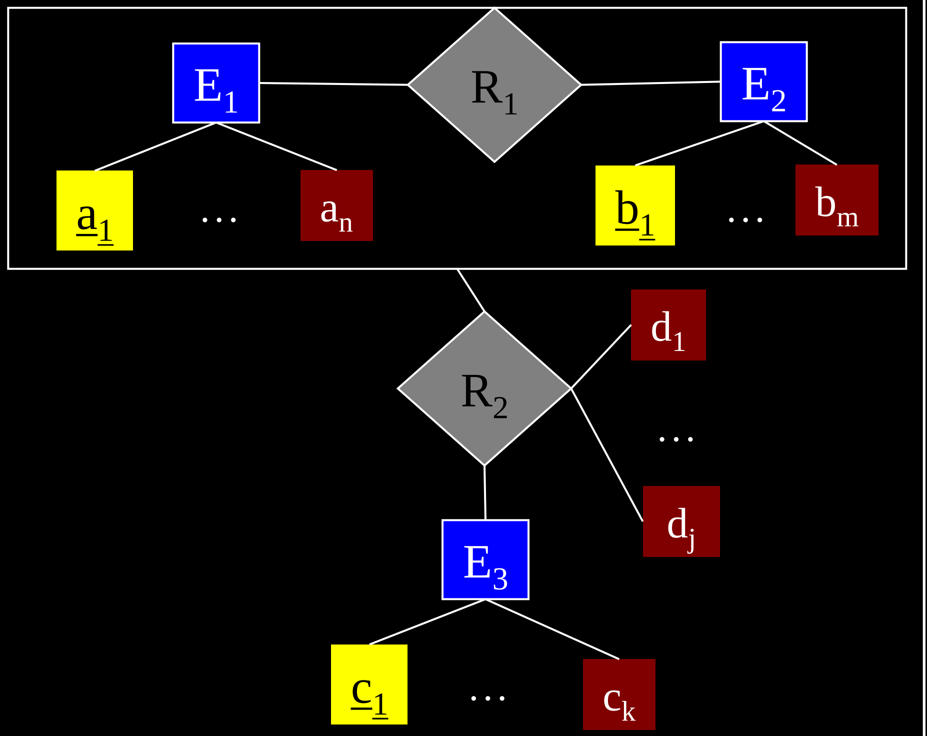
Q: When is method 2 not possible?

A: When subclassing is partial

E/R Diagrams and Relations

E/R	Relational Schema
Aggregation	
	

E/R Diagrams and Relations

E/R	Relational Schema
Aggregation	
	E_1, E_2, E_3, R_1 as before $R_2 = (\underline{c_1, a_1, b_1}, d_1, \ldots, d_j)$

Good DB Design

Three Approaches:

1. Ad hoc:
   - use Entity-Relationship Model to model data requirements
   - translate ER design into relational schema

   Issue: How to tell if design is "good"?

2. Theoretical:
   - construct universal relations (e.g., `Borrower-All`)
   - decompose above using known functional dependencies

   Issue: Time-Consuming and Complex

3. Practical:
   - use ER Model to produce 1st cut DB design
   - use FDs to refine and verify

Good DB Design

Three Approaches:

1. Ad hoc:
   - use Entity-Relationship Model to model data requirements
   - translate ER design into relational schema

   Issue: How to tell if design is "good"?

2. Theoretical:
   - construct universal relations (e.g., `Borrower-All`)
   - decompose above using known functional dependencies

   Issue: Time-Consuming and Complex

3. Practical:
   - use ER Model to produce 1st cut DB design
   - use FDs to refine and verify

Review: Functional Dependencies

In General:

$A_1, \ldots, A_n \rightarrow B$

Informally:

If 2 tuples agree on their values for $A_1, \ldots, A_n$,
then they will also agree on their values for B

Formally:

$$\boxed{\forall t, u\ (t[A_1] = u[A_1] \wedge \ldots \wedge t[A_n] = u[A_n]) \Rightarrow t[B] = u[B])}$$

Review: Deriving FDs

FD Sources:

1. Key Constraints (e.g.: `bname` → `Branch`)
2. Known "many-to-one" (n::1) relationships
   - e.g.: `beer` → `manufacturer`, `beer` → `price`
3. Laws of Physics
   - e.g.: `time, room` → `course`
4. Trial-and-error
   - given `R = (A,B,C)`, see which of the following make sense:

A → B	A → C	B → A
B → C	C → A	C → B
AB → C	AC → B	BC → A

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

e.g., **lno** → **bname** + **bname** → **bcity** implies

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

e.g., **lno** → **bname** + **bname** → **bcity** implies **lno** → **bcity**

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Q: Is {**lno** → **bname**, **bname** → **bcity**}

equivalent to

{**lno** → **bname**, **bname** → **bcity**, $\boxed{\textbf{lno} \rightarrow \textbf{bcity}}$} ?

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

{`lno` → `bname`, `bname` → `bcity`}

$\equiv$

{`lno` → `bname`, `bname` → `bcity`, `lno` → `bcity`}

$\overset{?}{\equiv}$

{`lno` → `bname bcity`, `bname` → `bcity`, `lno` → `bcity`}

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

{`lno` → `bname`, `bname` → `bcity`}

$\equiv$

{`lno` → `bname`, `bname` → `bcity`, `lno` → `bcity`}

$\equiv$

{`lno` → `bname bcity`, `bname` → `bcity`, `lno` → `bcity`}

$\overset{?}{\equiv}$

{`lno` → `bname bcity`, `bname` → `bcity`, `lno` → `lno`}

Deriving FDs: FD Closures (F^+)

Idea: Some FDs are implied by others

{`lno` → `bname`, `bname` → `bcity`}

≡

{`lno` → `bname`, `bname` → `bcity`, `lno` → `bcity`}

≡

{`lno` → `bname bcity`, `bname` → `bcity`, `lno` → `bcity`}

≡

{`lno` → `bname bcity`, `bname` → `bcity`, `lno` → `lno`}

Deriving FDs: FD Closures (F^+)

Given FD sets over R, F and G, how to decide if $F \equiv G$?

- Idea: Compare sets of FDs that F, G imply (closures)

$$\boxed{F \equiv G \textit{ if and only if } F^+ = G^+}$$

E.g., Given:

$R = (A, B, C, D, E, H)$

$F = \boxed{\{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}}$

$G = \boxed{\{A \to BH,\ B \to CE,\ D \to B\}}$

Is $F \equiv G$? (Is $F^+ = G^+$?)

Computing FD Closures

Algorithm 1: Using Attribute Closures

- Z^+ = set of attributes determined by <u>set of attributes</u>, Z
- can use attribute closures to compute F^+ by determining:

$$Z \rightarrow Z^+$$

for all subsets of attributes, Z

Example:

for $F = \{ AC \rightarrow B,\ B \rightarrow A \}$:

compute $F^+ = \{ A \rightarrow A^+,\ B \rightarrow B^+,\ C \rightarrow C^+,$
$AB \rightarrow AB^+,\ AC \rightarrow AC^+,\ BC \rightarrow BC^+,$
$ABC \rightarrow ABC^+ \}$

Computing FD Closures

Algorithm 1: Using Attribute Closures

- Z^+ = set of attributes determined by set of attributes, Z
- can use attribute closures to compute F^+ by determining:

$$Z \rightarrow Z^+$$

for all subsets of attributes, Z

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {**A** → **BC**, **B** → **CE**, **A** → **E**, **AC** → **H**, **D** → **B**}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD^+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD$^+$*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD$^+$*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD$^+$*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺? (the closure of {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD^+? *(the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,**B**}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B}
2	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD^+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺? (the closure of {C,D} wrt F)

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,**E**}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD⁺*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD^{+}*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD$^+$*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

What is CD^+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

What is CD^+*? (the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, B → CE, A → E, AC → H, D → B}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F affect the result?

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,**B**}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,**E**}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F affect the result?

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Attribute Closures

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

Does the order of FD's in F *affect the result?*

Att-Closure ({C,D},F):

Iteration #	result
0	{C,D}
1	{C,D,B}
2	{C,D,B,E}
3	{C,D,B,E}

Iteration #	result
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

A: No, but may change the # of passes of the algorithm required to reach "stability".

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

$F = \{A \rightarrow BC,\ D \rightarrow B,\ A \rightarrow E,\ AC \rightarrow H,\ B \rightarrow CE\}$

What is ACD^+*? (the closure of* {A,C,D} *wrt* F*)*

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD+*? (the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	result
0	{A,C,D}

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD$^+$? *(the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	**result**
0	{A,C,D}
1	

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
    FOR EACH functional dependency in F, X → Y DO
      IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD$^+$*? (the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	**result**
0	{A,C,D}
1	{A,C,D,B,E,H}

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD+*? (the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	**result**
0	{A,C,D}
1	{A,C,D,B,E,H}
2	

Exercise

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD⁺*? (the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	**result**
0	{A,C,D}
1	{A,C,D,B,E,H}
2	{A,C,D,B,E,H}

After iteration 1,

ACD⁺ = R.

Must be in stable state

Aside: Attribute Closures and Keys

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is ACD^+*? (the closure of* {A,C,D} *wrt* F*)*

Att-Closure ({A,C,D},F):

Iteration #	result
0	{A,C,D}
1	{A,C,D,B,E,H}
2	{A,C,D,B,E,H}

After iteration 1,
ACD^+ = R.
Must be in stable state

Aside: Attribute Closures and Keys

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

*Is {A,C,D} (ACD) a **superkey** of R = (A,B,C,D,E,H)?*

A: *Yes, because* $ACD^+ \to R$

*Is {A,C,D} (ACD) a **candidate key** of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $ACD^+ \to R$ — *must be a superkey*
2. $CD^+ \not\to R$
3. $AC^+ \not\to R$
4. $AD^+ \not\to R$

must be minimal (subtracting any attribute makes it not a key)

We Answered this Earlier

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is CD^+? *(the closure of* {C,D} *wrt* F*)*

Att-Closure ({C,D},F):

Iteration #	**result**
0	{C,D}
1	{C,D,B,E}
2	{C,D,B,E}

Aside: Attribute Closures and Keys

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

Is {A,C,D} (ACD) a ***superkey*** *of R = (A,B,C,D,E,H)?*

A: *Yes, because* $ACD^+ \to R$

Is {A,C,D} (ACD) a ***candidate key*** *of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $ACD^+ \to R$ ✓ *must be a superkey*
2. $CD^+ \not\to R$ ✓
3. $AC^+ \not\to R$
4. $AD^+ \not\to R$

$\left.\begin{array}{l} 2.\ CD^+ \not\to R \\ 3.\ AC^+ \not\to R \\ 4.\ AD^+ \not\to R \end{array}\right\}$ *must be minimal (subtracting any attribute makes it not a key)*

Aside: Attribute Closures and Keys

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is AC+*? (the closure of* {A,C} *wrt* F*)*

Att-Closure ({A,C},F):

Iteration #	result
0	{A,C}

Aside: Attribute Closures and Keys

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

*Is {A,C,D} (ACD) a **superkey** of R = (A,B,C,D,E,H)?*

A: *Yes, because* $ACD^+ \rightarrow R$

*Is {A,C,D} (ACD) a **candidate key** of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

✓ 1. $ACD^+ \rightarrow R$ *must be a superkey*

✓ 2. $CD^+ \not\rightarrow R$

✓ 3. $AC^+ \not\rightarrow R$

4. $AD^+ \not\rightarrow R$

must be minimal (subtracting any attribute makes it not a key)

Aside: Attribute Closures and Keys

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {A → BC, D → B, A → E, AC → H, B → CE}

What is AD^+*? (the closure of* {A,D} *wrt* F*)*

Att-Closure ({A,D},F):

Iteration #	**result**
0	{A,D}

Aside: Attribute Closures and Keys

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

*Is {A,C,D} (ACD) a **superkey** of R = (A,B,C,D,E,H)?*

A: *Yes, because* $ACD^+ \to R$

*Is {A,C,D} (ACD) a **candidate key** of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $ACD^+ \to R$ — *must be a superkey*
2. $CD^+ \not\to R$
3. $AC^+ \not\to R$
4. $AD^+ \not\to R$

must be minimal (subtracting any attribute makes it not a key)

Therefore, ACD is not a candidate key of R

Exercise

$$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$$

*Is {A,D} (AD) a **candidate key** of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $AD^+ \to R$ *must be a superkey*
2. $A^+ \not\to R$
3. $D^+ \not\to R$

must be minimal (subtracting any attribute makes it not a key)

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

Exercise Solution

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {**A** → **BC**, **D** → **B**, **A** → **E**, **AC** → **H**, **B** → **CE**}

What is A$^+$*? (the closure of* {A} *wrt* F*)*

Att-Closure ({A},F):

Iteration #	**result**
0	{A}
1	{A,B,C,E,H}
2	{A,B,C,E,H}

$A^+ \neq R$

Exercise Solution

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

Is {A,D} (AD) a ***candidate key*** *of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $AD^+ \rightarrow R$ *must be a superkey*
2. $A^+ \not\rightarrow R$
3. $D^+ \not\rightarrow R$

must be minimal (subtracting any attribute makes it not a key)

Exercise Solution

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
     FOR EACH functional dependency in F, X → Y DO
       IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

F = {**A** → **BC**, **D** → **B**, **A** → **E**, **AC** → **H**, **B** → **CE**}

What is D+*? (the closure of* {D} *wrt* F*)*

Att-Closure ({D},F):

Iteration #	**result**
0	{D}
1	{D,B}
2	{D,B,C,E}
3	{D,B,C,E}

$D^+ \neq R$

Exercise Solution

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

*Is {A,D} (AD) a **candidate key** of R = (A,B,C,D,E,H)?*

A: *Conditions that must be true for the answer to be yes:*

1. $AD^+ \to R$ — *must be a superkey*
2. $A^+ \not\to R$
3. $D^+ \not\to R$

(2 and 3:) *must be minimal (subtracting any attribute makes it not a key)*

Therefore, AD is a candidate key of R

Deriving FDs: FD Closures (F^+)

$F \equiv G$ *if and only if* $F^+ = G^+$

E.g., Given:

$R = (A, B, C, D, E, H)$

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

$G = \{A \rightarrow BH,\ B \rightarrow CE,\ D \rightarrow B\}$

Is $F \equiv G$? (Is $F^+ = G^+$?)

Computing FD Closures

Algorithm 1: Using Attribute Closures

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
   Result ← Result ∪ {Z → Att-Closure(Z,F)}
  RETURN Result
END
```

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
   FOR EACH functional dependency in F, X → Y DO
     IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Result = {}

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Result =
 $\{\{A \rightarrow A+\} \cup \{B \rightarrow B+\} \cup \{C \rightarrow C+\} \cup \{D \rightarrow D+\} \cup \{E \rightarrow E+\} \cup \{H \rightarrow H+\} \cup \ldots$

Computing F^+ With Attribute Closures

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Result =

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪ ...

Computing F$^+$ With Attribute Closures

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Result =

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪ {ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪ {ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪ {BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪ {BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪ {CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪ ...

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

Result =

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪ {ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪ {ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪ {BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪ {BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪ {CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
{ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪ {ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
{ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪ {ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
{BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪ {BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪...

Computing F+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

```
Result =
 {{A → A+} ∪ {B → B+} ∪{C → C+} ∪ {D → D+} ∪ {E → E+} ∪{H → H+} ∪
 {AB → (AB)+} ∪ {AC → (AC)+} ∪{AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
 {BC → (BC)+} ∪ {BD → (BD)+} ∪{BE → (BE)+} ∪ {BH → (BH)+} ∪{CD → CDBE} ∪
 {CE → (CE)+} ∪{CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪{EH → (EH)+} ∪
 {ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪{ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
 {ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪{ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
 {ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪{BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
 {BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪{BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
 {CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪{CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
 {ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪{ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
 {ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪{ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
 {BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪{BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
 {ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪{BCDEH → (BCDEH)+} ∪ ...
```

Computing F$^+$ With Attribute Closures

F = {**A** → **BC**, **B** → **CE**, **A** → **E**, **AC** → **H**, **D** → **B**}

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

```
Result =
 {{A → A+} ∪ {B → B+} ∪{C → C+} ∪ {D → D+} ∪ {E → E+} ∪{H → H+} ∪
 {AB → (AB)+} ∪ {AC → (AC)+} ∪{AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
 {BC → (BC)+} ∪ {BD → (BD)+} ∪{BE → (BE)+} ∪ {BH → (BH)+} ∪{CD → CDBE} ∪
 {CE → (CE)+} ∪{CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪{EH → (EH)+} ∪
 {ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪{ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
 {ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪{ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
 {ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪{BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
 {BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪{BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
 {CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪{CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
 {ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪{ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
 {ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪{ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
 {BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪{BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
 {ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪{BCDEH → (BCDEH)+} ∪
 {ABCDEH → (ABCDEH)+}}
```

Computing F^+ With Attribute Closures

F = {A → BC, B → CE, A → E, AC → H, D → B}

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

```
Result =
 {{A → A+} ∪ {B → B+} ∪{C → C+} ∪ {D → D+} ∪ {E → E+} ∪{H → H+} ∪
 {AB → (AB)+} ∪ {AC → (AC)+} ∪{AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
 {BC → (BC)+} ∪ {BD → (BD)+} ∪{BE → (BE)+} ∪ {BH → (BH)+} ∪{CD → CDBE} ∪
 {CE → (CE)+} ∪{CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪{EH → (EH)+} ∪
 {ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪{ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
 {ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪{ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
 {ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪{BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
 {BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪{BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
 {CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪{CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
 {ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪{ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
 {ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪{ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
 {BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪{BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
 {ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪{BCDEH → (BCDEH)+} ∪
 {ABCDEH → (ABCDEH)+}}
```

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

$F^+ =$

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪ {ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪ {ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪ {BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪ {BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪ {CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
{ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪ {ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
{ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪ {ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
{BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪ {BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
{ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪ {BCDEH → (BCDEH)+} ∪
{ABCDEH → (ABCDEH)+}}

Computing F^+ With Attribute Closures

$G = \{A \rightarrow BH,\ B \rightarrow CE,\ D \rightarrow B\}$

$G^+ =$

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪ {ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪ {ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪ {BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪ {BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪ {CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
{ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪ {ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
{ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪ {ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
{BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪ {BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
{ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪ {BCDEH → (BCDEH)+} ∪
{ABCDEH → (ABCDEH)+}}

Computing FD Closures

Algorithm 1: Using Attribute Closures

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
   Result ← Result ∪ {Z → Att-Closure(Z,F)}
  RETURN Result
END
```

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
   FOR EACH functional dependency in F, X → Y DO
     IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $\mathbf{X} \rightarrow \mathbf{Y}$
2. *Augmentation*
   - if $X \rightarrow Y$ then $\mathbf{WX} \rightarrow \mathbf{WY}$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{Z}$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{YZ}$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $\mathbf{WX} \rightarrow \mathbf{Z}$

Armstrong's Axioms

Reflexivity: if $\mathtt{Y} \subseteq \mathtt{X}$ then $\mathbf{X} \rightarrow \mathbf{Y}$

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- justifies trivial FDs: e.g.,
  - $\{\mathtt{lno}\} \subseteq \{\mathtt{lno}\}$ implies:

Armstrong's Axioms

Reflexivity: if $Y \subseteq X$ then **X** $\rightarrow$ **Y**

Borrower-All								
lno	cname	cstreet	ccity	bname	amt	bcity	assets	*lno*
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M	L-17
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M	L-23
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M	L-15
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M	L-17
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M	L-93
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M	L-11
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M	L-16

- justifies trivial FDs: e.g.,
  - {lno} $\subseteq$ {lno} implies: **lno** $\rightarrow$ **lno**

Armstrong's Axioms

Reflexivity: if Y $\subseteq$ X then **X** $\rightarrow$ **Y**

Borrower-All								
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**	***lno***
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M	L-17
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M	L-23
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M	L-15
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M	L-17
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M	L-93
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M	L-11
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M	L-16

- justifies trivial FDs: e.g.,
  - {lno} $\subseteq$ {lno} implies: **lno** $\rightarrow$ **lno**
  - {lno} $\subseteq$ {lno, cname} implies: **lno cname** $\rightarrow$ **lno**

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $\mathbf{X} \rightarrow \mathbf{Y}$
2. *Augmentation*
   - if $X \rightarrow Y$ then $\mathbf{WX} \rightarrow \mathbf{WY}$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{Z}$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{YZ}$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $\mathbf{WX} \rightarrow \mathbf{Z}$

Armstrong's Axioms

Augmentation: if **X** → **Y** then **WX** → **WY**

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **bname** → **bcity** implies:

Armstrong's Axioms

Augmentation: if **X** → **Y** then **WX** → **WY**

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **bname** → **bcity** implies: **cname bname** → **cname bcity**

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $X \rightarrow Y$
2. *Augmentation*
   - if $X \rightarrow Y$ then $WX \rightarrow WY$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$

Armstrong's Axioms

Transitivity: if **X** → **Y** and **Y** → **Z** then **X** → **Z**

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **lno** → **bname** and **bname** → **bcity** implies:

Armstrong's Axioms

Transitivity: if **X** → **Y** and **Y** → **Z** then **X** → **Z**

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **lno** → **bname** and **bname** → **bcity** implies: **lno** → **bcity**

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $\mathbf{X} \rightarrow \mathbf{Y}$
2. *Augmentation*
   - if $X \rightarrow Y$ then $\mathbf{WX} \rightarrow \mathbf{WY}$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{Z}$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{YZ}$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $\mathbf{WX} \rightarrow \mathbf{Z}$

Armstrong's Axioms

Union: if $X \to Y$ and $X \to Z$ then $X \to YZ$

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - `lno` → `bname` and `lno` → `amt` implies:

Armstrong's Axioms

Union: if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - lno → bname and lno → amt implies: lno → bname amt

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $X \to Y$
2. *Augmentation*
   - if $X \to Y$ then $WX \to WY$
3. *Transitivity*
   - if $X \to Y$ and $Y \to Z$ then $X \to Z$
4. *Union*
   - if $X \to Y$ and $X \to Z$ then $X \to YZ$
5. *Decomposition*
   - if $X \to YZ$ then $X \to Y$ and $X \to Z$
6. *Pseudotransitivity*
   - if $X \to Y$ and $WY \to Z$ then $WX \to Z$

Armstrong's Axioms

Decomposition: if $\mathbf{X} \rightarrow \mathbf{YZ}$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - `bname` → `bcity assets` implies:

Armstrong's Axioms

Decomposition: if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - `bname` → `bcity assets` implies: `bname` → `bcity`, `bname` → `assets`

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $\mathbf{X} \rightarrow \mathbf{Y}$
2. *Augmentation*
   - if $X \rightarrow Y$ then $\mathbf{WX} \rightarrow \mathbf{WY}$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{Z}$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{YZ}$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $\mathbf{WX} \rightarrow \mathbf{Z}$

Armstrong's Axioms

Pseudotransitivity: if **X** → **Y** and **WY** → **Z** then **WX** → **Z**

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

Borrower-All							
lno	cname	cstreet	ccity	bname	amt	bcity	assets
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **bcity assets** → **bname** and **cname bname** → **lno** implies:

Armstrong's Axioms

Pseudotransitivity: if **X** → **Y** and **WY** → **Z** then **WX** → **Z**

Borrower-All							
lno	**cname**	**cstreet**	**ccity**	**bname**	**amt**	**bcity**	**assets**
L-17	Jones	Main	Harrison	Downtown	1000	Brooklyn	9M
L-23	Smith	North	Rye	Redwood	2000	Palo Alto	2.1M
L-15	Hayes	Main	Harrison	Perry	1500	Horseneck	1.7M
L-17	Jackson	Senator	Brooklyn	Downtown	1000	Brooklyn	9M
L-93	Curry	Walnut	Stanford	Mianus	500	Horseneck	0.4M
L-11	Smith	North	Rye	R.H.	900	Horseneck	8M
L-16	Adams	Spring	Pittsfield	Perry	1300	Horseneck	1.7M

- e.g.,
  - **bcity assets** → **bname** and **cname bname** → **lno** implies:

    cname bcity assets → **lno**

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $\mathbf{X} \rightarrow \mathbf{Y}$
2. *Augmentation*
   - if $X \rightarrow Y$ then $\mathbf{WX} \rightarrow \mathbf{WY}$
3. *Transitivity*
   - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{Z}$
4. *Union*
   - if $X \rightarrow Y$ and $X \rightarrow Z$ then $\mathbf{X} \rightarrow \mathbf{YZ}$
5. *Decomposition*
   - if $X \rightarrow YZ$ then $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$
6. *Pseudotransitivity*
   - if $X \rightarrow Y$ and $WY \rightarrow Z$ then $\mathbf{WX} \rightarrow \mathbf{Z}$

Computing FD Closures

Algorithm 2: Using Armstrong's Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
   IF for any of Armstrong's Axioms (if A then B),
        A matches part of Result THEN
            Result ← Result ∪ B
  RETURN Result
END
```

Computing $\mathtt{F}^+$ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

```
F   =  {        A → BC,
                B → CE,
                A → E,
                AC → H,
                D → B          }
```

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

F⁺ = { (1) A → BC,
(2) B → CE,
(3) A → E,
(4) AC → H,
(5) D → B,

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

F^+ = { (1) A → BC,
(2) B → CE,
(3) A → E,
(4) AC → H,
(5) D → B,

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

F^+ = {
(1) A → BC,
(2) B → CE,
(3) A → E,
(4) AC → H,
(5) D → B,
(6) A → B, decomposition (1)
(7) A → C, decomposition (1)

Decomposition
if **X** → **YZ** then **X** → **Y** and **X** → **Z**

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

Transitivity

if **X** → **Y** and **Y** → **Z**, then **X** → **Z**

```
F+  =  {   (1) A → BC,
           (2) B → CE,
           (3) A → E,
           (4) AC → H,
           (5) D → B,
           (6) A → B,          decomposition (1)
           (7) A → C,          decomposition (1)
           (8) A → CE,         transitivity (6),(2)
```

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
    IF for any of Armstrong's Axioms (if A then B),
         A matches part of Result THEN
           Result ← Result ∪ B
  RETURN Result
END
```

Decomposition

if **X** → **YZ** then **X** → **Y** and **X** → **Z**

F^+ = {

(1)	A → BC,	
(2)	B → CE,	
(3)	A → E,	
(4)	AC → H,	
(5)	D → B,	
(6)	A → B,	decomposition (1)
(7)	A → C,	decomposition (1)
(8)	A → CE,	transitivity (6),(2)
(9)	B → C,	decomposition (2)
(10)	B → E,	decomposition (2)

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
    IF for any of Armstrong's Axioms (if A then B),
         A matches part of Result THEN
           Result ← Result ∪ B
  RETURN Result
END
```

Pseudotransitivity

if **X** → **Y** and **WY** → **Z**, then **WX** → **Z**

F^+	=	{	(1) A → BC,	
			(2) B → CE,	
			(3) A → E,	
			(4) AC → H,	
			(5) D → B,	
			(6) A → B,	decomposition (1)
			(7) A → C,	decomposition (1)
			(8) A → CE,	transitivity (6),(2)
			(9) B → C,	decomposition (2)
			(10) B → E,	decomposition (2)
			(11) A → H,	pseudotransitivity (7),(4) …}

Computing F^+ With Attribute Closures

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
     Result ← Result ∪ {Z → Att-Closure (Z,F)}
  RETURN Result
END
```

$F^+ =$

{{A → A+} ∪ {B → B+} ∪ {C → C+} ∪ {D → D+} ∪ {E → E+} ∪ {H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪ {AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪ {BE → (BE)+} ∪ {BH → (BH)+} ∪ {CD → CDBE} ∪
{CE → (CE)+} ∪ {CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪ {EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪ {ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪ {ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪ {BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪ {BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪ {CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
{ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪ {ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
{ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪ {ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
{BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪ {BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
{ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪ {BCDEH → (BCDEH)+} ∪
{ABCDEH → (ABCDEH)+}}

Computing F^+ With Armstrongs Axioms

$F = \{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}$

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

F^+ =

{{A → A+} ∪ {B → B+} ∪{C → C+} ∪ {D → D+} ∪ {E → E+} ∪{H → H+} ∪
{AB → (AB)+} ∪ {AC → (AC)+} ∪{AD → (AD)+} ∪ {AE → (AE+} ∪ {AH → (AH)+} ∪
{BC → (BC)+} ∪ {BD → (BD)+} ∪{BE → (BE)+} ∪ {BH → (BH)+} ∪{CD → CDBE} ∪
{CE → (CE)+} ∪{CH → (CH)+} ∪ {DE → (DE)+} ∪ {DH → (DH)+} ∪{EH → (EH)+} ∪
{ABC → (ABC)+} ∪ {ABD → (ABD)+} ∪{ABE → (ABE)+} ∪ {ABH → (ABH)+} ∪
{ACD → ACDBEH} ∪ {ACE → (ACE)+} ∪{ACH → (ACH)+} ∪ {ADE → (ADE)+} ∪
{ADH → (ADH)+} ∪ {AEH → (AEH)+} ∪{BCD → (BCD)+} ∪ {BCE → (BCE)+} ∪
{BCH → (BCH)+} ∪ {BDE → (BDE)+} ∪{BDH → (BDH)+} ∪ {BEH → (BEH)+} ∪
{CDE → (CDE)+} ∪ {CDH → (CDH)+} ∪{CEH → (CEH)+} ∪ {DEH → (DEH)+} ∪
{ABCD → (ABCD)+} ∪ {ABCE → (ABCE)+} ∪{ABCH → (ABCH)+} ∪ {ABDE → (ABDE)+} ∪
{ABEH → (ABEH)+} ∪ {ACDE → (ACDE)+} ∪{ACEH → (ACEH)+} ∪ {ADEH → (ADEH)+} ∪
{BCDE → (BCDE)+} ∪ {BCDH → (BCDH)+} ∪{BDEH → (BDEH)+} ∪ {CDEH → (CDEH)+} ∪
{ABCDE → (ABCDE)+} ∪ {ABCDH → (ABCDH)+} ∪{BCDEH → (BCDEH)+} ∪
{ABCDEH → (ABCDEH)+}}

Computing F^+ With Armstrongs Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
     IF for any of Armstrong's Axioms (if A then B),
          A matches part of Result THEN
             Result ← Result ∪ B
  RETURN Result
END
```

Usually use Armstrong's Axioms selectively to make a point

```
F+   =  {    (1)  A → BC,
             (2)  B → CE,
             (3)  A → E,
             (4)  AC → H,
             (5)  D → B,
             (6)  A → B,          decomposition (1)
             (7)  A → C,          decomposition (1)
             (8)  A → CE,         transitivity (6),(2)
             (9)  B → C,          decomposition (2)
             (10)  B → E,         decomposition (2)
             (11)  A → H,         pseudotransitivity (7),(4)  …}
```

Selective use of Armstrong's Axioms

Another way to show that $F^+ = G^+$ ($F \equiv G$):

$$F = \boxed{\{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}}$$

$$G = \boxed{\{A \to BH,\ B \to CE,\ D \to B\}}$$

Strategy: User Armstrong's Axioms to show:

1. $G \subseteq F^+$, and
2. $F \subseteq G^+$

Intuition:

$$\left.\begin{array}{l} G^+ \subseteq F^+ \\ F^+ \subseteq G^+ \end{array}\right\} \Rightarrow F^+ = G^+$$

Selective use of Armstrong's Axioms

Another way to show that $F^+ = G^+$ ($F \equiv G$):

$F = \{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}$

$G = \{A \to BH,\ B \to CE,\ D \to B\}$

Strategy: User Armstrong's Axioms to show:

1. $G \subseteq F^+$, and
2. $F \subseteq G^+$

Selective use of Armstrong's Axioms

Another way to show that $F^+ = G^+$ ($F \equiv G$):

$$F = \boxed{\{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}}$$

$$G = \boxed{\{A \rightarrow BH,\ B \rightarrow CE,\ D \rightarrow B\}}$$

Strategy: User Armstrong's Axioms to show...

1. $f \in G \Rightarrow f \in F^+$, and
2. $f \in F \Rightarrow f \in G^+$

Selective use of Armstrong's Axioms

Another way to show that $F^+ = G^+$ $(F \equiv G)$:

$F = \boxed{\{A \rightarrow BC,\ B \rightarrow CE,\ A \rightarrow E,\ AC \rightarrow H,\ D \rightarrow B\}}$

$G = \boxed{\{A \rightarrow BH,\ B \rightarrow CE,\ D \rightarrow B\}}$

Strategy: User Armstrong's Axioms to show...

1. $f \in G \Rightarrow f \in F^+$

To prove (1), must show:

a) '$A \rightarrow BH$' $\in F^+$

b) '$B \rightarrow CE$' $\in F^+$

c) '$D \rightarrow B$' $\in F^+$

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '**A** → **BH**' ∈ F^+

Proof:

F^+ = { 1. **A** → **BC**,
2. **B** → **CE**,
3. **A** → **E**,
4. **AC** → **H**,
5. **D** → **B**,

. . . }

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '$A \rightarrow BH$' $\in F^+$

<u>Proof</u>:

F^+ = { 1. $A \rightarrow BC$,
2. $B \rightarrow CE$,
3. $A \rightarrow E$,
4. $AC \rightarrow H$,
5. $D \rightarrow B$,
6. $A \rightarrow C$, decomposition (1)
7. $A \rightarrow B$, decomposition (1)

... }

Decomposition

if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '$A \rightarrow BH$' $\in F^+$

Proof:

Pseudotransitivity

if $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

F^+ = { 1. $A \rightarrow BC$,
2. $B \rightarrow CE$,
3. $A \rightarrow E$,
4. $AC \rightarrow H$,
5. $D \rightarrow B$,
6. $A \rightarrow C$, decomposition (1)
7. $A \rightarrow B$, decomposition (1)
8. $A \rightarrow H$, pseudotransitivity (6,4)
... }

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '$\mathbf{A} \rightarrow \mathbf{BH}$' $\in F^+$

Proof:

$F^+ = \{$	1.	$\mathbf{A} \rightarrow \mathbf{BC}$,	
	2.	$\mathbf{B} \rightarrow \mathbf{CE}$,	
	3.	$\mathbf{A} \rightarrow \mathbf{E}$,	
	4.	$\mathbf{AC} \rightarrow \mathbf{H}$,	
	5.	$\mathbf{D} \rightarrow \mathbf{B}$,	
	6.	$\mathbf{A} \rightarrow \mathbf{C}$,	decomposition (1)
	7.	$\mathbf{A} \rightarrow \mathbf{B}$,	decomposition (1)
	8.	$\mathbf{A} \rightarrow \mathbf{H}$,	pseudotransitivity (6,4)
	9.	$\mathbf{A} \rightarrow \mathbf{BH}$,	union (7,8) ... $\}$

Union

if $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{X} \rightarrow \mathbf{Z}$ then $\mathbf{X} \rightarrow \mathbf{YZ}$

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '$\mathbf{A} \rightarrow \mathbf{BH}$' $\in F^+$

Proof:

```
F+ = { 1. A → BC,
       2. B → CE,
       3. A → E,
       4. AC → H,
       5. D → B,
       6. A → C,        decomposition (1)
       7. A → B,        decomposition (1)
       8. A → H,        pseudotransitivity (6,4)
       9. A → BH,       union (7,8) ... }
```

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) $'A \rightarrow BH' \in F^+$ (9)

b) $'B \rightarrow CE' \in F^+$

c) $'D \rightarrow B' \in F^+$

Proof:

$F^+ = \{$ 1. $A \rightarrow BC$,
2. $B \rightarrow CE$,
3. $A \rightarrow E$,
4. $AC \rightarrow H$,
5. $D \rightarrow B$,
6. $A \rightarrow C$, decomposition (1)
7. $A \rightarrow B$, decomposition (1)
8. $A \rightarrow H$, pseudotransitivity (6,4)
9. $A \rightarrow BH$, union (7,8) ... $\}$

Selective use of Armstrong's Axioms

Strategy: User Armstrong's Axioms to show...

To prove (1), must show:

a) '$A \rightarrow BH$' $\in F^+$ (9)

b) '$B \rightarrow CE$' $\in F^+$ (2)

c) '$D \rightarrow B$' $\in F^+$ (5)

<u>Proof</u>:

F^+ = { 1. $A \rightarrow BC$,
2. $B \rightarrow CE$,
3. $A \rightarrow E$,
4. $AC \rightarrow H$,
5. $D \rightarrow B$,
6. $A \rightarrow C$, decomposition (1)
7. $A \rightarrow B$, decomposition (1)
8. $A \rightarrow H$, pseudotransitivity (6,4)
9. $A \rightarrow BH$, union (7,8) ... }

Selective use of Armstrong's Axioms

Another way to show that $F^+ = G^+$ ($F \equiv G$):

$F = \boxed{\{A \to BC,\ B \to CE,\ A \to E,\ AC \to H,\ D \to B\}}$

$G = \boxed{\{A \to BH,\ B \to CE,\ D \to B\}}$

Strategy: User Armstrong's Axioms to show...

✓ 1. $f \in G \Rightarrow f \in F^+$, and

2. $f \in F \Rightarrow f \in G^+$