CS2102: Database Systems

T03: Entity Relationship Model

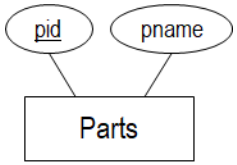
Question 1

Question 1

Entity Sets

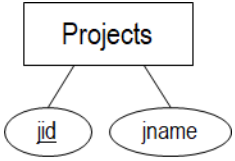
Parts

- *Parts(<u>pid</u>, pname)*



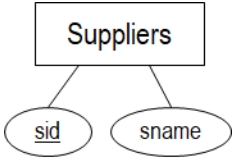
Projects

- *Projects(<u>jid</u>, jname)*



Suppliers

- *Suppliers(<u>sid</u>, sname)*



Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

Entity Sets

Convention

- We will use the name *p*, *p1*, *p2*, ... for parts
- We will use the name *j*, *j1*, *j2*, ... for projects
- We will use the name *s*, *s1*, *s2*, ... for suppliers
- We will use _ *(underscore)* or - *(minus)* to indicate *any value* *(i.e., we don't care what the value is)*
- We will use tuple *(X, Y, ...)* to indicate an entry in entity and/or relationship sets
  - May use other tuple as well for other values

Discussions

> We will limit our discussions to the following constraints:
>
> 1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
> 2. Can we know which supplier supplies which parts to which projects
> 3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

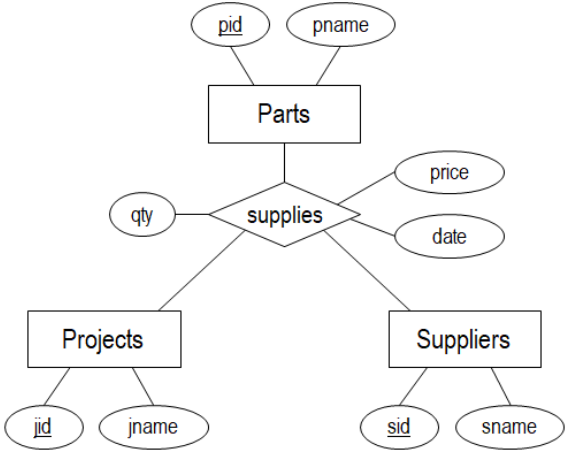
Question 1

Q1A

Design A

Discuss

(*5 minutes*)



Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

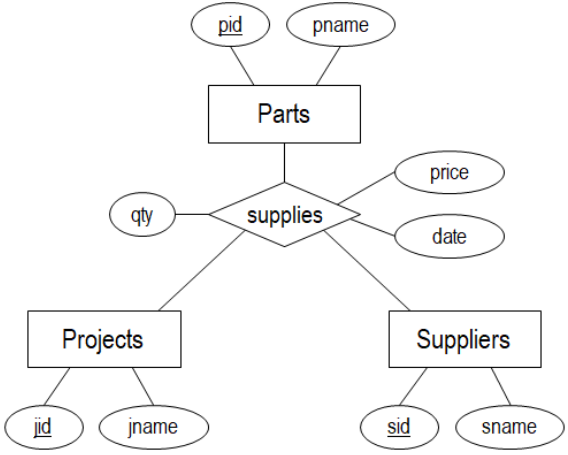
Question 1

Q1A

Design A

Constraint #1

> How to check?



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. Can we know which supplier supplies which parts to which projects
> 3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

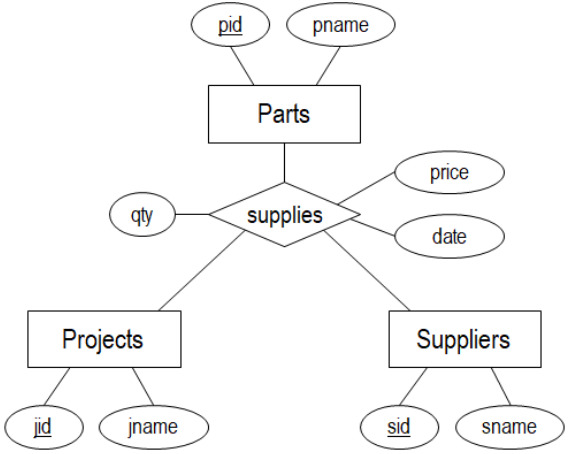
Q1A

Design A

Constraint #1

Is this a valid data?

pid	sid	jid	qty	price	date
p	s	j1	-	10	-
p	s	j2	-	20	-



Discussions

We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

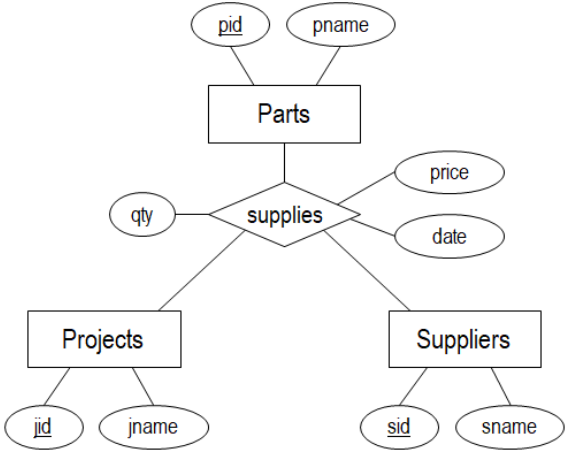
Q1A

Design A

Constraint #1

Is this a valid data?

pid	sid	jid	qty	price	date
p	s	j1	-	10	-
p	s	j2	-	20	-



Discussions

We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

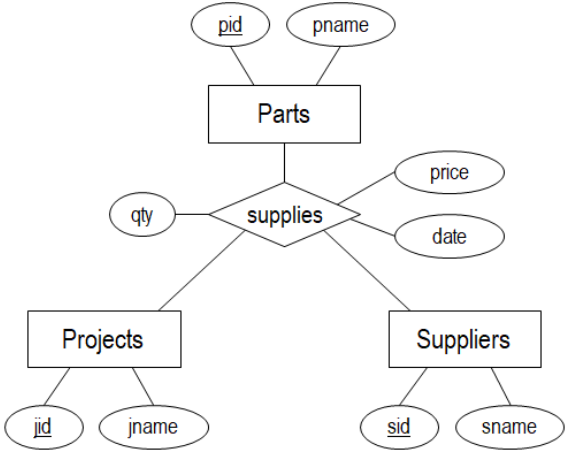
Question 1

Q1A

Design A

Constraint #2

> How to check?



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

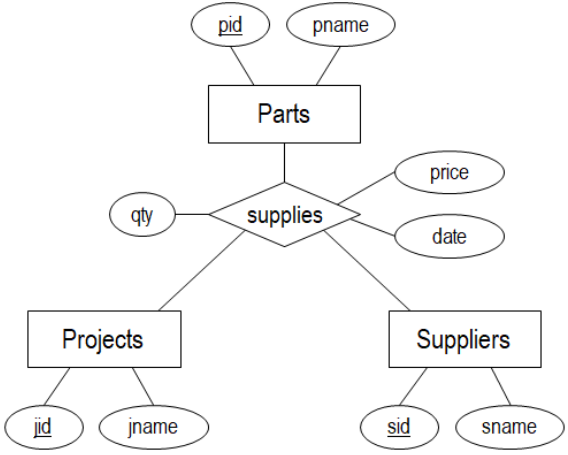
Q1A

Design A

Constraint #2

Who supplies *p1* to *j1*?

pid	sid	jid	qty	price	date
p2	s1	j2	-	-	-
p1	s1	j1	-	-	-
p1	s2	j1	-	-	-



Discussions

We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. **Can we know which supplier supplies which parts to which projects**
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

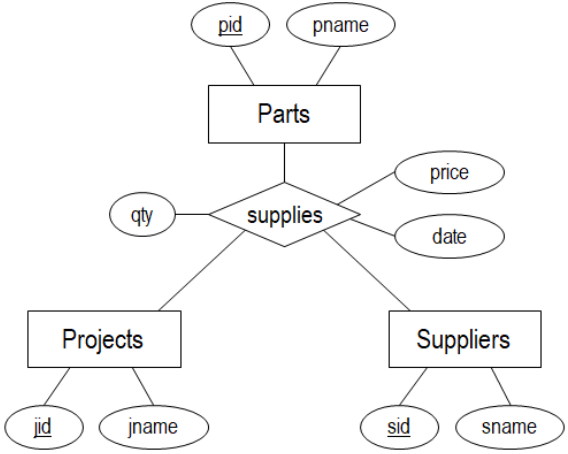
Q1A

Design A

Constraint #2

Who supplies *p1* to *j1*?

pid	sid	jid	qty	price	date
p2	s1	j2	-	-	-
p1	s1	j1	-	-	-
p1	s2	j1	-	-	-



Discussions

We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. **Can we know which supplier supplies which parts to which projects**
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

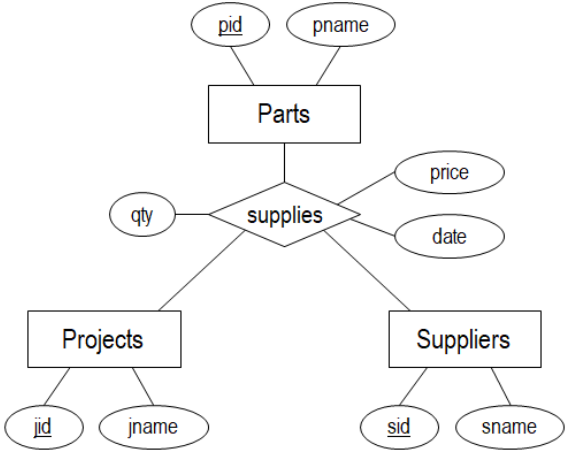
Question 1

Q1A

Design A

Constraint #2

> It is always possible to do this because the triple *(p, s, j)* always means that the part *p* is supplied to project *j* by *s* *(there can be many supplier but we know all suppliers)*



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

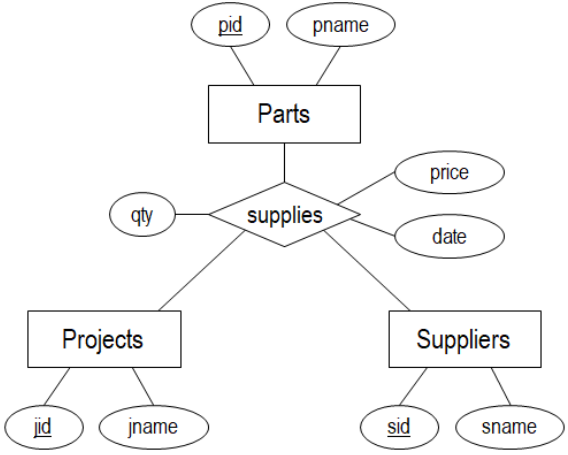
Question 1

Q1A

Design A

Constraint #3

> How to check?



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

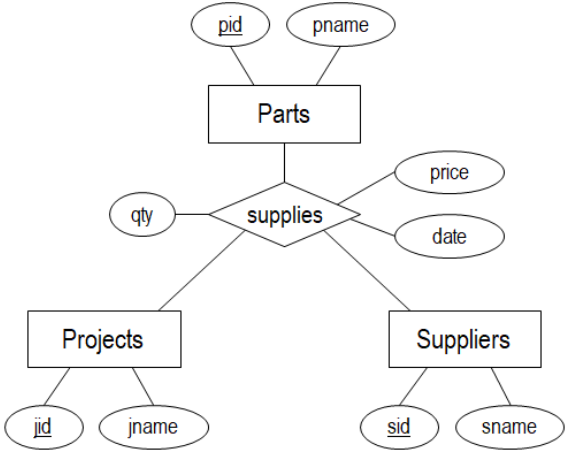
Question 1

Q1A

Design A

Constraint #3

> If there is no supplier, we require *(p,* NULL*, j). Can we have such triple?*
>
> **Hint:** Check the translation to schema



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

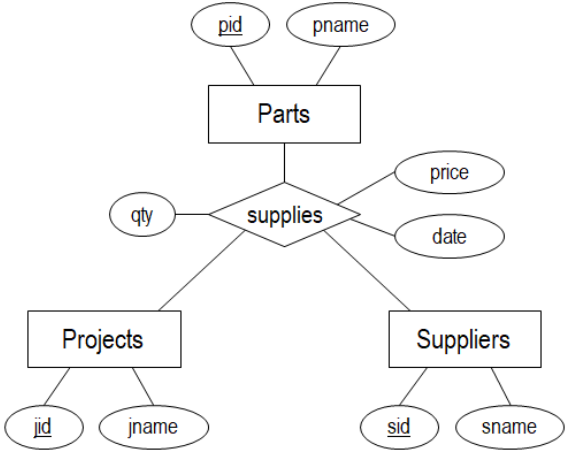
Question 1

Q1A

Design A

Constraint #3

> If there is no supplier, we require *(p,* NULL*, j)*. ~~*Can we have such triple?*~~
>
> **NO:** The primary key of *supplies* are *{pid, sid, jid}*



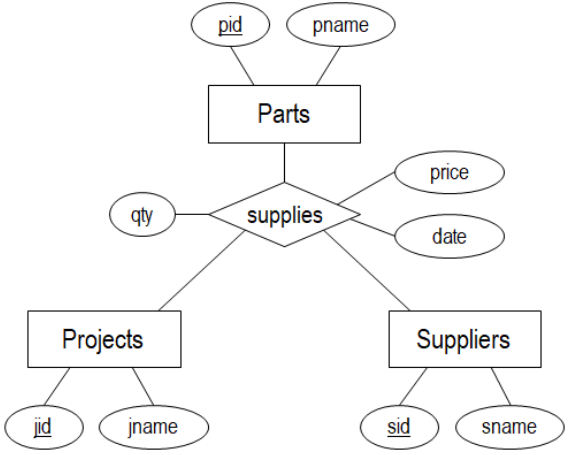
Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

Question 1

Q1A

Design A



Discussions

We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. **Can we know which supplier supplies which parts to which projects**
3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

Question 1

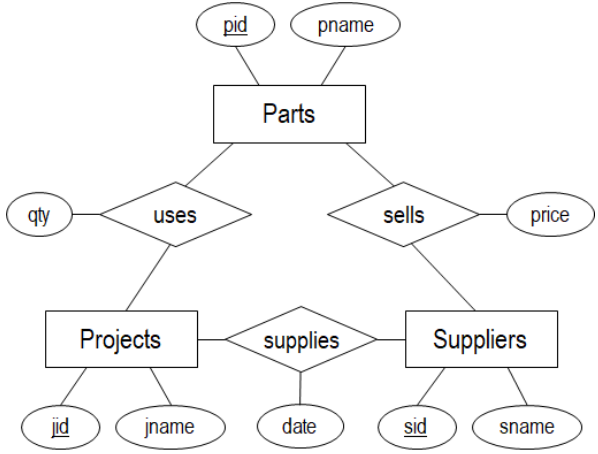
Entity
Q1A
- Design A
- Design B
- Design C
Q1B

Q1A

Design B

Discuss

(*5 minutes*)



Discussions

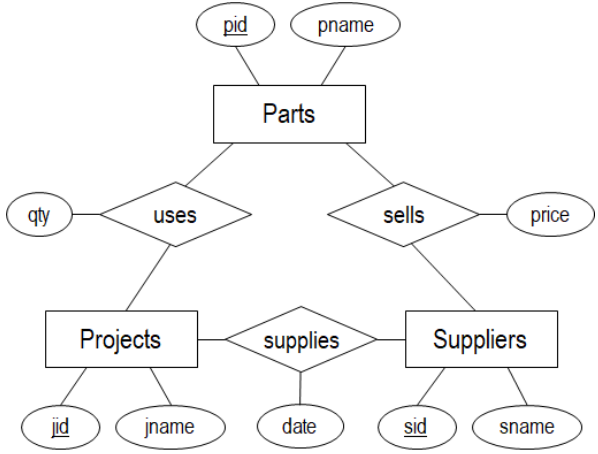
We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

Q1A

Design B



1. **NO**
   - Since *(s, p)* is *unique* in *sells*
2. **NO**
   - Will discuss this further
3. **YES**
   - We can have *(p, j)* in *uses* without any *(p, s)* in *sells*
   - Because they are on different relationship set!

Discussions

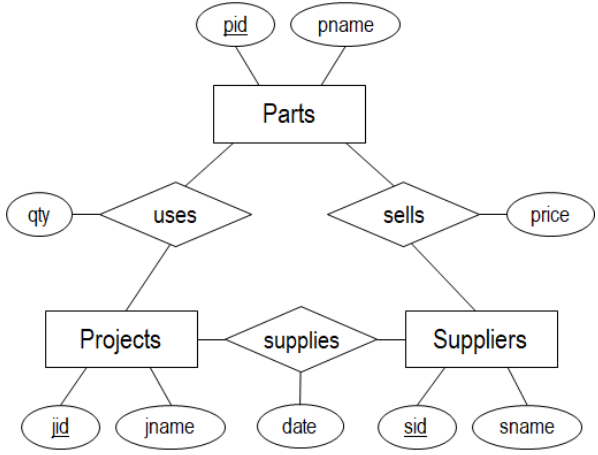
We will limit our discussions to the following constraints:

1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
2. **Can we know which supplier supplies which parts to which projects**
3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

Question 1

Q1A

Design B



Counter-Example

Assume the following. What would the entries look like?

- *s1* sells *p1* and *p2* but supplies only *p1* to *j*
- *s2* sells *p1* and *p2* but supplies only *p2* to *j*

Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. **Can we know which supplier supplies which parts to which projects**
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

Entity
Q1A
- Design A
- Design B
- Design C
Q1B

Q1A

Design B

uses

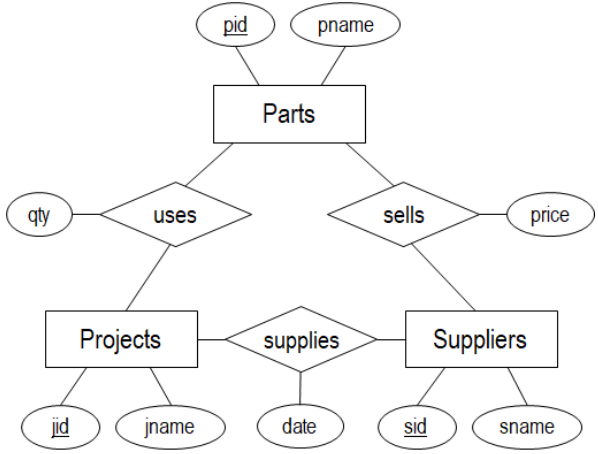
pid	jid	-
p1	j	-
p2	j	-

sells

pid	sid	-
p1	s1	-
p1	s2	-
p2	s1	-
p2	s2	-

supplies

jid	sid	-
j	s1	-
j	s2	-



Can you still tell now?

Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. **Can we know which supplier supplies which parts to which projects**
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

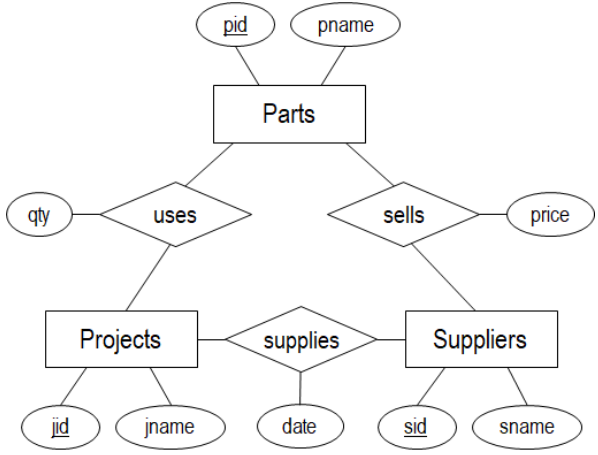
Question 1

Q1A

Design B

uses ⋈ sells ⋈ supplies

pid	jid	sid	-
p1	j	s1	-
p1	j	s2	-
p2	j	s1	-
p2	j	s2	-



We lost the information

Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. **Can we know which supplier supplies which parts to which projects**
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

Question 1

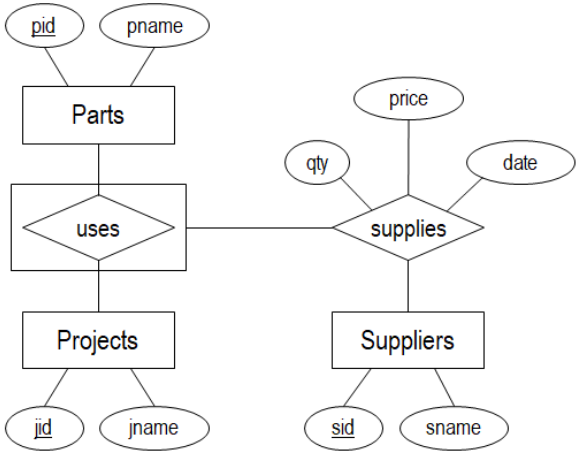
Entity
Q1A
- Design A
- Design B
- Design C
Q1B

Q1A

Design C

Discuss

(*5 minutes*)



Discussions

We will limit our discussions to the following constraints:

1. Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2
2. Can we know which supplier supplies which parts to which projects
3. Can a project *J* require parts *P* that currently has no supplier *S* that can supply them

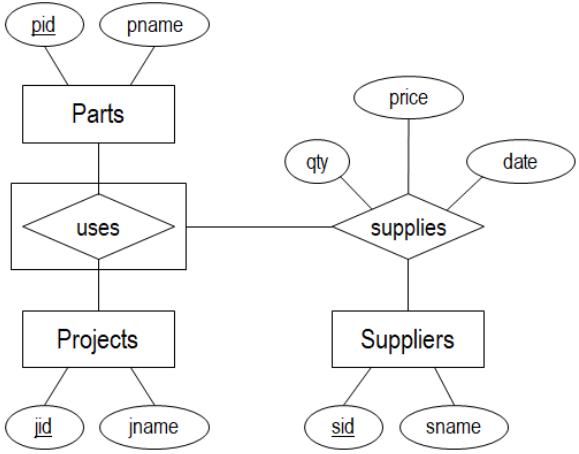
Question 1

Q1A

Design C

1. **YES**
2. **YES**
3. **YES**

> Can you explain why?



Discussions

> We will limit our discussions to the following constraints:
>
> 1. **Can a supplier *S* sells a part *P* for $x to project J_1 but $y to project J_2**
> 2. **Can we know which supplier supplies which parts to which projects**
> 3. **Can a project *J* require parts *P* that currently has no supplier *S* that can supply them**

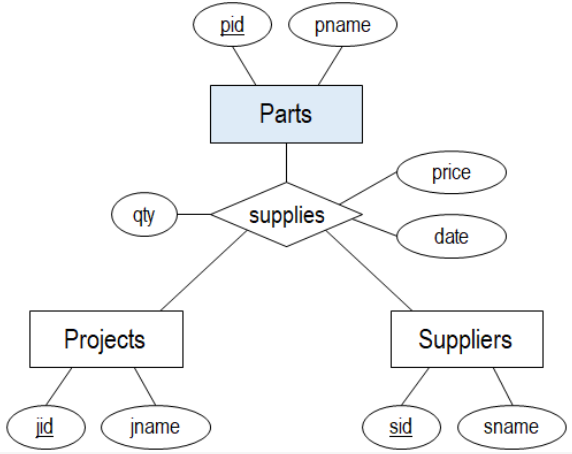
Question 1

Q1B

Design A

Notes

- Use of `NOT NULL` requires more information about the requirement



Parts

```sql
CREATE TABLE Parts (
  pid    INTEGER PRIMARY KEY,
  pname  TEXT
);
```

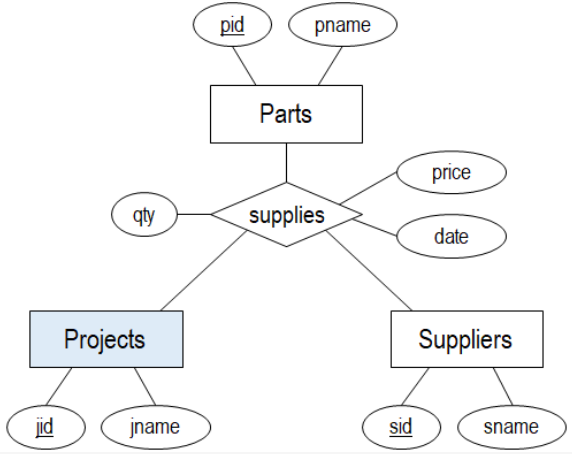
Question 1

Q1B

Design A

Notes

- Use of `NOT NULL` requires more information about the requirement



Projects

```
CREATE TABLE Projects (
  jid    INTEGER PRIMARY KEY,
  jname  TEXT
);
```

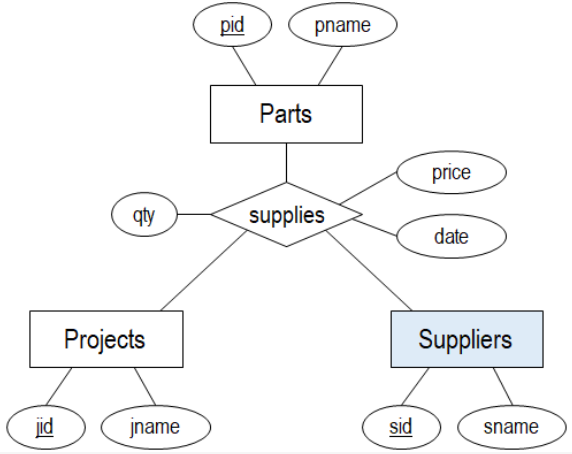
Question 1

Q1B

Design A

Notes

- Use of `NOT NULL` requires more information about the requirement



Suppliers

```
CREATE TABLE Suppliers (
  sid    INTEGER PRIMARY KEY,
  sname  TEXT
);
```

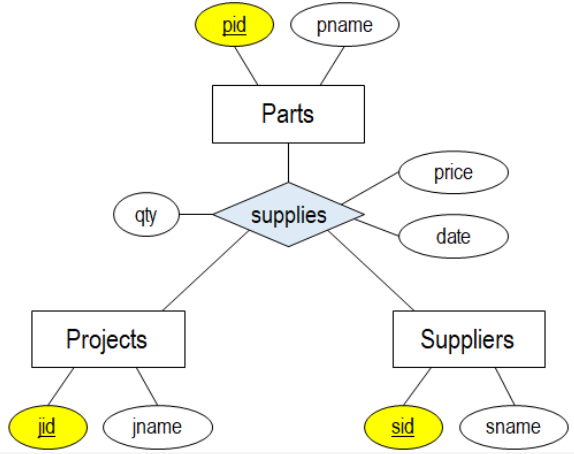
Question 1

Q1B

Design A

Notes

- May want to add `CHECK (qty >= 0)`
- Price may include "*cents*"



Supplies

```
CREATE TABLE supplies (
  pid    INTEGER REFERENCES Parts (pid),
  sid    INTEGER REFERENCES Suppliers (sid),
  jid    INTEGER REFERENCES Projects (jid),
  qty    INTEGER,
  price  NUMERIC,
  date   DATE,
  PRIMARY KEY (pid, sid, jid)
);
```

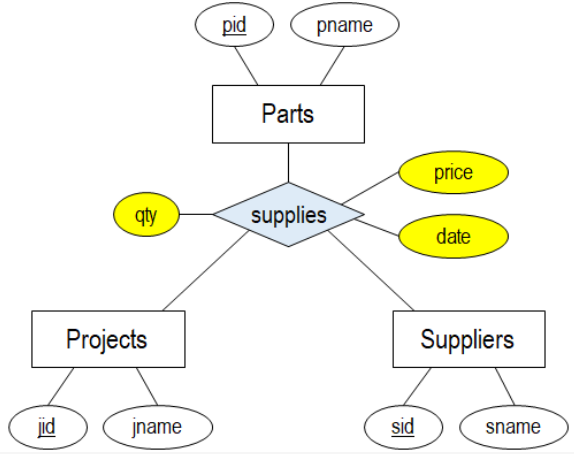
Question 1

Q1B

Design A

Notes

- May want to add `CHECK (qty >= 0)`
- Price may include "*cents*"



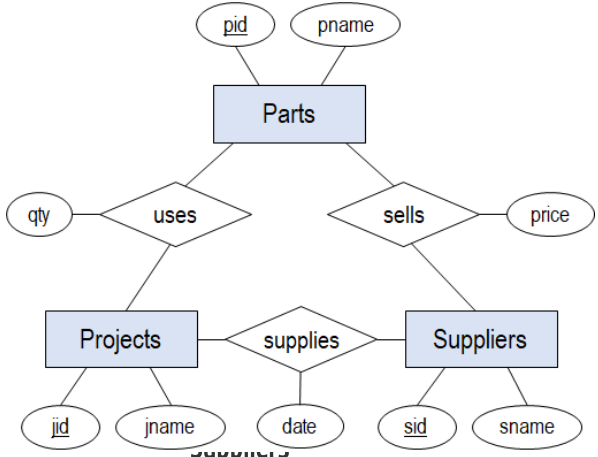
Supplies

```
CREATE TABLE supplies (
  pid    INTEGER REFERENCES Parts (pid),
  sid    INTEGER REFERENCES Suppliers (sid),
  jid    INTEGER REFERENCES Projects (jid),
  qty    INTEGER,
  price  NUMERIC,
  date   DATE,
  PRIMARY KEY (pid, sid, jid)
);
```

Question 1

Q1B

Design B



Notes

- *Parts* is the same as before
- *Projects* is the same as before
- *Suppliers* is the same as before

Parts

```
CREATE TABLE Parts (
  pid    INTEGER,
  pname  TEXT,
  PRIMARY KEY (pid)
);
```

Projects

```
CREATE TABLE Projects (
  jid    INTEGER,
  jname  TEXT,
  PRIMARY KEY (jid)
);
```

Suppliers

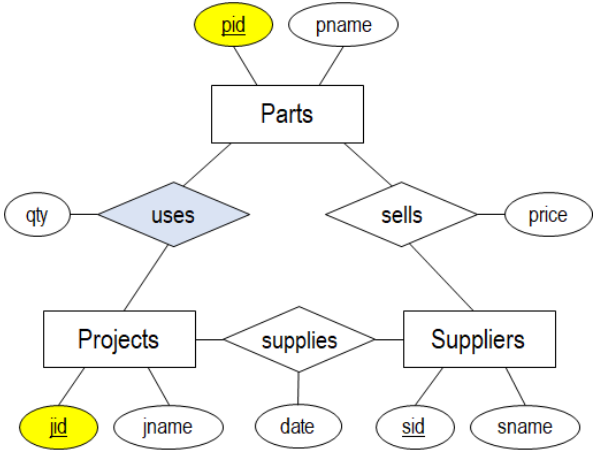
```
CREATE TABLE Suppliers (
  sid    INTEGER,
  sname  TEXT,
  PRIMARY KEY (sid)
);
```

[#]Alternative primary key constraint using *table constraint*.

Question 1

Q1B

Design B



Notes

- May want to add `CHECK (qty >= 0)`

Uses

```
CREATE TABLE uses (
  pid   INTEGER REFERENCES Parts,
  jid   INTEGER REFERENCES Projects,
  qty   INTEGER,
  PRIMARY KEY (pid, jid)
);
```

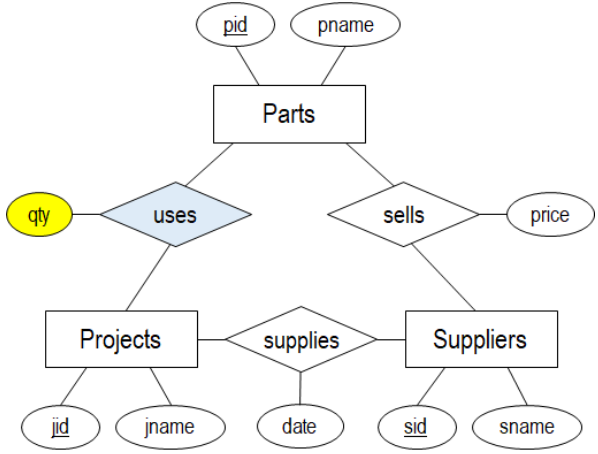
Question 1

Q1B

Design B

Notes

- May want to add `CHECK (qty >= 0)`



Uses

```
CREATE TABLE uses (
  pid   INTEGER REFERENCES Parts,
  jid   INTEGER REFERENCES Projects,
  qty   INTEGER,
  PRIMARY KEY (pid, jid)
);
```

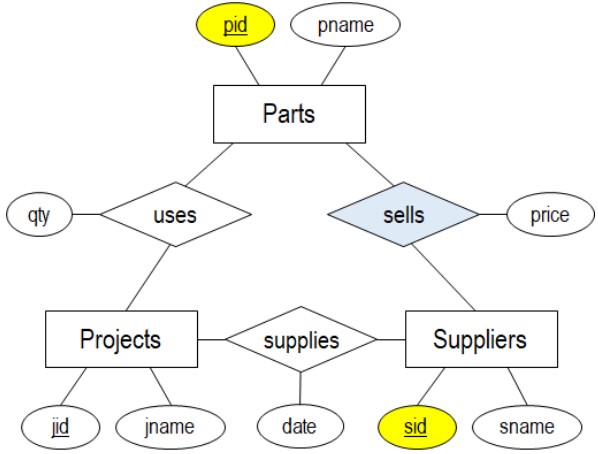
Question 1

Q1B

Design B

Notes

- Price may include "*cents*"



Sells

```
CREATE TABLE sells (
  pid   INTEGER REFERENCES Parts,
  sid   INTEGER REFERENCES Suppliers,
  price NUMERIC,
  PRIMARY KEY (pid, sid)
);
```

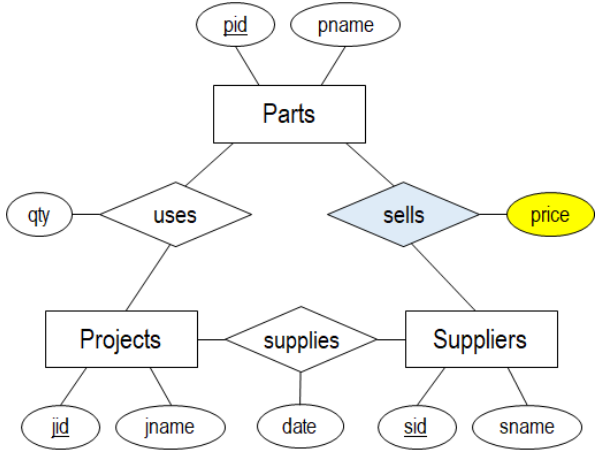
Question 1

Q1B

Design B

Notes

- Price may include "*cents*"



Sells

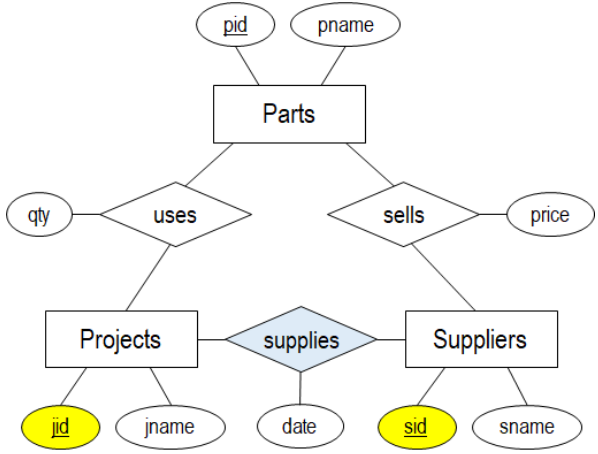
```
CREATE TABLE sells (
  pid   INTEGER REFERENCES Parts,
  sid   INTEGER REFERENCES Suppliers,
  price NUMERIC,
  PRIMARY KEY (pid, sid)
);
```

Question 1

Q1B

Design B

Notes



Supplies

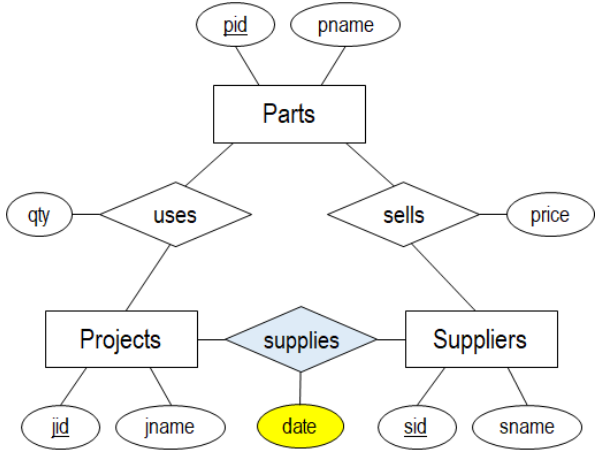
```
CREATE TABLE supplies (
  jid   INTEGER REFERENCES Projects,
  sid   INTEGER REFERENCES Suppliers,
  date  DATE,
  PRIMARY KEY (jid, sid)
);
```

Question 1

Q1B

Design B

Notes



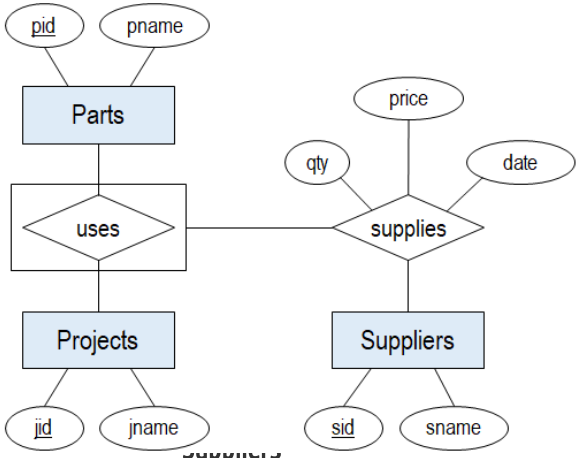
Supplies

```
CREATE TABLE supplies (
  jid   INTEGER REFERENCES Projects,
  sid   INTEGER REFERENCES Suppliers,
  date  DATE,
  PRIMARY KEY (jid, sid)
);
```

Question 1

Q1B

Design C



Notes

- *Parts* is the same as before
- *Projects* is the same as before
- *Suppliers* is the same as before

Parts

```
CREATE TABLE Parts (
  pid    INTEGER,
  pname  TEXT,
  PRIMARY KEY (pid)
);
```

Projects

```
CREATE TABLE Projects (
  jid    INTEGER,
  jname  TEXT,
  PRIMARY KEY (jid)
);
```

Suppliers

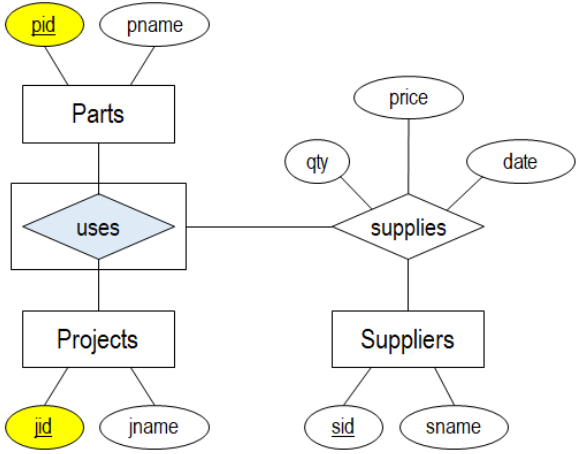
```
CREATE TABLE Suppliers (
  sid    INTEGER,
  sname  TEXT,
  PRIMARY KEY (sid)
);
```

[#]Alternative primary key constraint using *table constraint*.

Question 1

Q1B

Design C



Notes

- *uses* has to be created before *supplies* *(why?)*

Uses

```
CREATE TABLE uses (
  pid   INTEGER REFERENCES Parts,
  jid   INTEGER REFERENCES Projects,
  PRIMARY KEY (pid, jid)
);
```

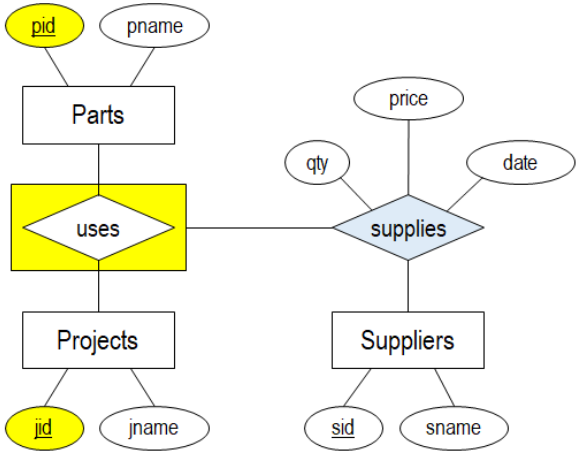
Question 1

Q1B

Design C

Notes

- Why do we need both *pid* and *jid* in *supplies*?



Supplies

```
CREATE TABLE supplies (
  pid   INTEGER,
  jid   INTEGER,
  sid   INTEGER REFERENCES Suppliers,
    :   -- qty, price, date omitted due to space constraint
  PRIMARY KEY (pid, jid, sid),
  FOREIGN KEY (pid, jid) REFERENCES uses
);
```

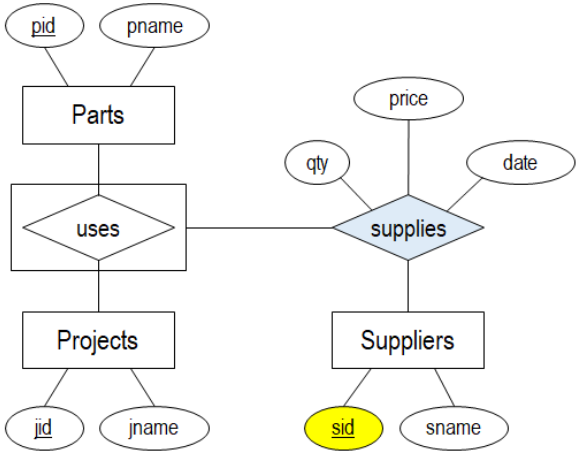
Question 1

Q1B

Design C

Notes

- Why do we need both *pid* and *jid* in *supplies*?



Supplies

```
CREATE TABLE supplies (
  pid   INTEGER,
  jid   INTEGER,
  sid   INTEGER REFERENCES Suppliers,
    :   -- qty, price, date omitted due to space constraint
  PRIMARY KEY (pid, jid, sid),
  FOREIGN KEY (pid, jid) REFERENCES uses
);
```

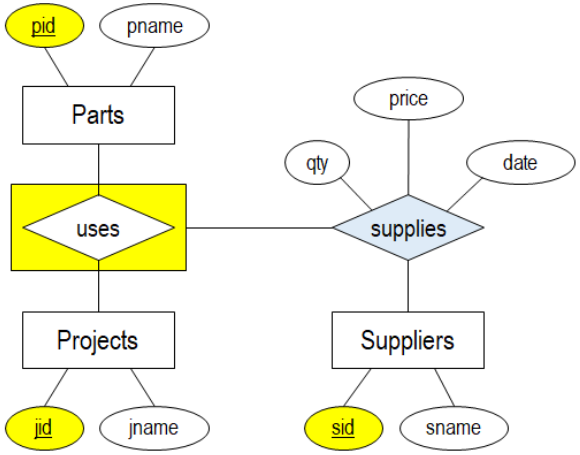
Question 1

Q1B

Design C

Notes

- Why do we need both *pid* and *jid* in *supplies*?



Supplies

```
CREATE TABLE supplies (
  pid   INTEGER,
  jid   INTEGER,
  sid   INTEGER REFERENCES Suppliers,
    :   -- qty, price, date omitted due to space constraint
  PRIMARY KEY (pid, jid, sid),
  FOREIGN KEY (pid, jid) REFERENCES uses
);
```

Question 2

Question 2

Translation

Considerations

- Enforce as many constraints captured in the ER diagram as possible
  - But if there are conflicts with project requirements, follow project requirements
  - Because some constraints may not be properly captured by the ER diagram
- Do not enforce additional constraints that are not captured in the ER diagram
  - Unless it cannot be captured by the ER diagram but still in the project requirements
- Use "*common sense*" data types
  - Money should not be `INTEGER`
  - Name should not be `NUMERIC`
  - Minimize unrestricted `TEXT` in favor of `VARCHAR` especially if sizes are known
- Add "*common sense*" constraints
  - Quantity should not be negative *(may or may not be zero)*

[#]Common sense may vary and may require some background knowledge. Should attempt on a "best effort" basis.

Question 2

Translation

Idea

- Translate from *constructs* with the fewest number of foreign key constraints
  - In most cases, there will be a *construct* with zero foreign key constraints
  - Non-zero foreign key means that there are cyclic dependencies
- Break circular dependencies
  - Similarly, break from the *constructs* with fewest number of foreign key constraints
  - If there are multiple choices, choose arbitrarily

[#]Constructs here can refer to *entity set*, *relationship set*, or *aggregate*.

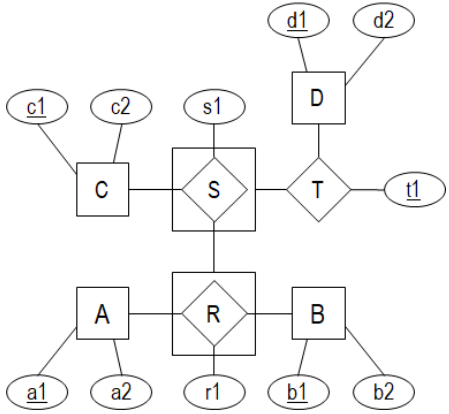
Question 2

ERD

ER 1

Question

Which *constructs* have the fewest number of foreign key constraints?



Question 2

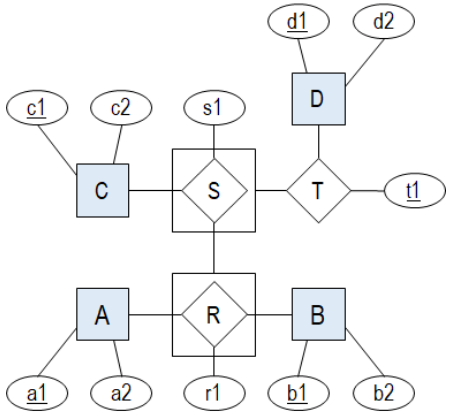
ERD

ER 1

Question

Which *constructs* have the fewest number of foreign key constraints?

Translate A, B, C, D (*3 minutes*)

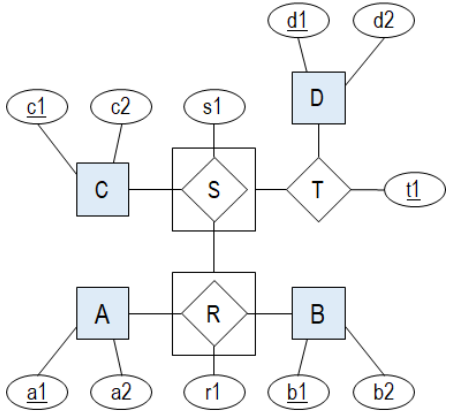


Question 2

ERD

ER 1

> **Question**
>
> Which *constructs* have the fewest number of foreign key constraints?



Translation

```sql
CREATE TABLE A (
  a1  INTEGER,
  a2  INTEGER,
  PRIMARY KEY (a1)
);
```

```sql
CREATE TABLE B (
  b1  INTEGER,
  b2  INTEGER,
  PRIMARY KEY (b1)
);
```

```sql
CREATE TABLE C (
  c1  INTEGER,
  c2  INTEGER,
  PRIMARY KEY (c1)
);
```

```sql
CREATE TABLE D (
  d1  INTEGER,
  d2  INTEGER,
  PRIMARY KEY (d1)
);
```

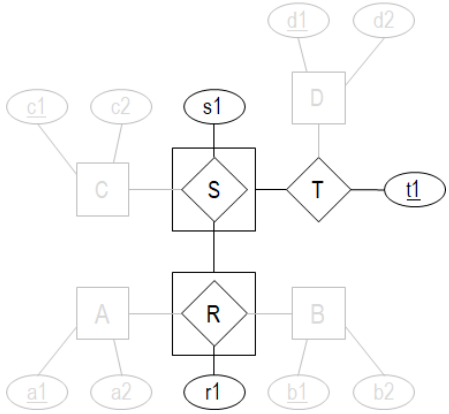
Question 2

ERD

ER 1

Question

Which *constructs* have the fewest number of foreign key constraints ***now***?

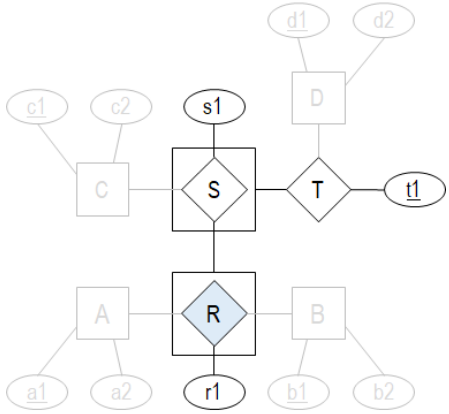


Question 2

ERD

ER 1

> **Question**
>
> Which *constructs* have the fewest number of foreign key constraints ***now***?



Translation

```
CREATE TABLE R (
  a1  INTEGER REFERENCES A,
  b1  INTEGER REFERENCES B,
  r1  INTEGER,
  PRIMARY KEY (a1, b1)
);
```

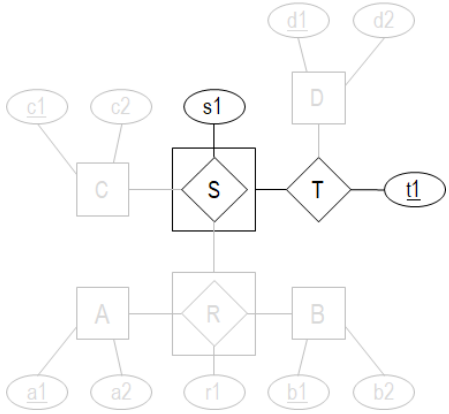
Question 2

ERD

ER 1

Question

Which *constructs* have the fewest number of foreign key constraints ***now***?

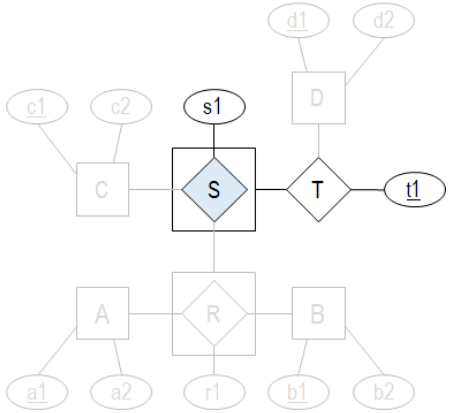


Question 2

ERD

ER 1

> **Question**
>
> Which *constructs* have the fewest number of foreign key constraints ***now***?



Translation

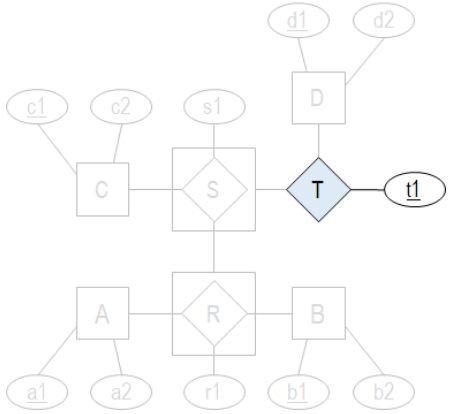
```
CREATE TABLE S (
  a1  INTEGER,
  b1  INTEGER,
  c1  INTEGER REFERENCES C
  s1  INTEGER,
  PRIMARY KEY (a1, b1, c1),
  FOREIGN KEY (a1, b1) REFERENCES R
);
```

Question 2

ERD

ER 1

> **Question**
>
> Why is *t1* part of the primary key for *T*?



Translation

```
CREATE TABLE T (
  a1  INTEGER,
  b1  INTEGER,
  c1  INTEGER,
  d1  INTEGER REFERENCES D
  t1  INTEGER,
  PRIMARY KEY (a1, b1, c1, d1, t1),
  FOREIGN KEY (a1, b1, c1) REFERENCES S
);
```

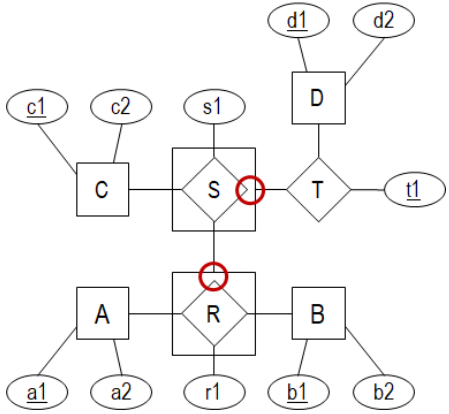
Question 2

ERD

ER 1

Important Note

> Aggregate can be thought of as both entity set and relationship set.
>
> **When an aggregate is used as an entity set, connect to the rectangle**



Question 2

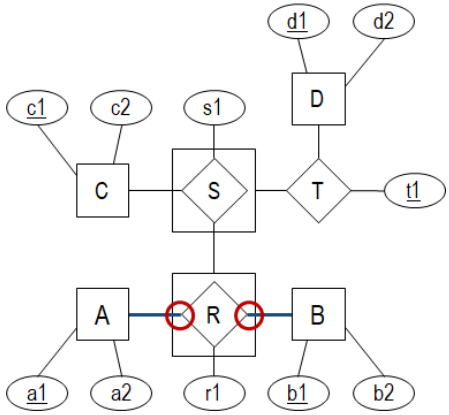
Translation
ERD
- ER 1
- ER 2
- ER 3

ERD

ER 1

Important Note

> Aggregate can be thought of as both entity set and relationship set.
>
> **When an aggregate is formed *(as relationship set)*, connect to the diamond**

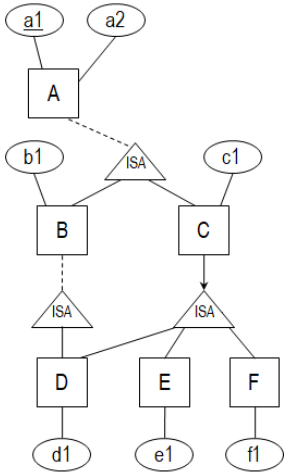


Question 2

ERD

ER 2

> Same process, look for the constructs with the fewest FK. In the case of ISA hierarchy, it will be the superclass entity sets.

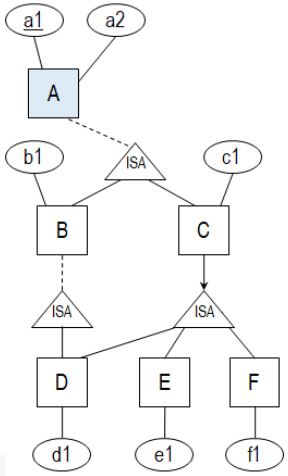


Question 2

ERD

ER 2

> Same process, look for the constructs with the fewest FK. In the case of ISA hierarchy, it will be the superclass entity sets.



Translation

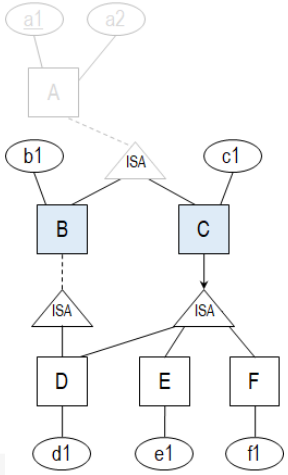
```
CREATE TABLE A (
  a1  INTEGER PRIMARY KEY,
  a2  INTEGER
);
```

Question 2

ERD

ER 2

> Same process, look for the constructs with the fewest FK. In the case of ISA hierarchy, it will be the superclass entity sets.



Translation

```
CREATE TABLE B (
  a1  INTEGER PRIMARY KEY
    REFERENCES A ON DELETE CASCADE,
  b1  INTEGER
);
```

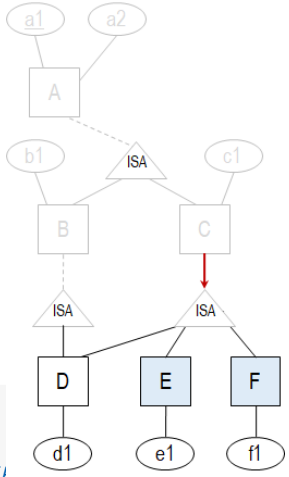
```
CREATE TABLE C (
  a1  INTEGER PRIMARY KEY
    REFERENCES A ON DELETE CASCADE,
  c1  INTEGER
);
```

Question 2

ERD

ER 2

1. Must reference *C* and not *A* *(why?)*
2. Can we enforce → on the ISA?



Translation

```
CREATE TABLE E (
  a1  INTEGER PRIMARY KEY
    REFERENCES C ON DELETE CASCADE,
  e1  INTEGER
);
```

```
CREATE TABLE F (
  a1  INTEGER PRIMARY KEY
    REFERENCES C ON DELETE CASCADE,
  f1  INTEGER
);
```

[##]1. Because otherwise there can be entries in *A* but not in *C* and finally in *E* or *F*, which is wrong.
[#]2. No, because we cannot restrict that entries in *E* are not also in *F*. Requires trigger.

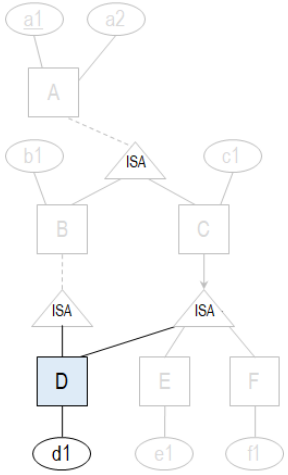
Question 2

Translation
ERD
- ER 1
- ER 2
- ER 3

ERD

ER 2

- How to do *multiple* ISA superclass?

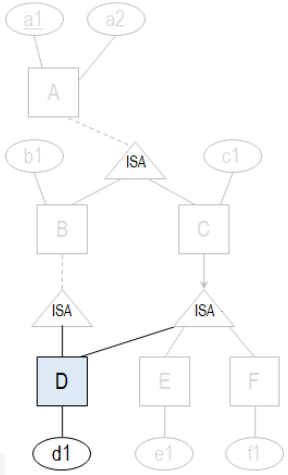


Question 2

ERD

ER 2

- How to do *multiple* ISA superclass?
- Multiple REFERENCES!



Translation

```
CREATE TABLE D (
  a1  INTEGER PRIMARY KEY
    REFERENCES B ON DELETE CASCADE
    REFERENCES C ON DELETE CASCADE,
  d1  INTEGER
);
```

Question 2

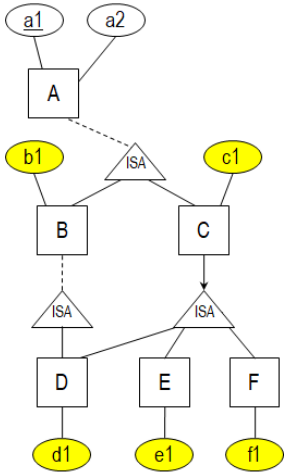
Translation
ERD
- ER 1
- **ER 2**
- ER 3

ERD

ER 2

Important Note

> Subclass inherits from superclass.
>
> **Subclass should NOT have its own key attributes**



Question 2

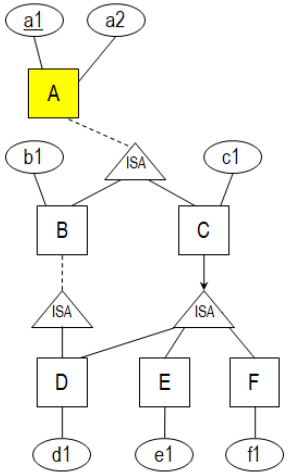
Translation
ERD
- ER 1
- ER 2
- ER 3

ERD

ER 2

Important Note

> Subclass inherits from superclass.
>
> *For clarity, superclass should be above the subclass*
>
> *(alternatively, use dashed line to connect to superclass when using a line without arrow to remove any ambiguity)*

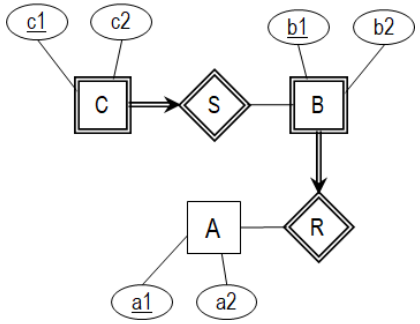


Question 2

ERD

ER 3

> Same process, look for the constructs with the fewest FK. In the case of weak entity set, it will be the owning entity set.

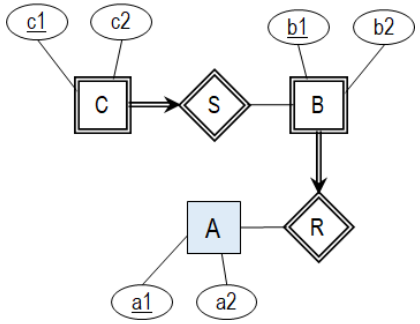


Question 2

ERD

ER 3

> Same process, look for the constructs with the fewest FK. In the case of weak entity set, it will be the owning entity set.



Translation

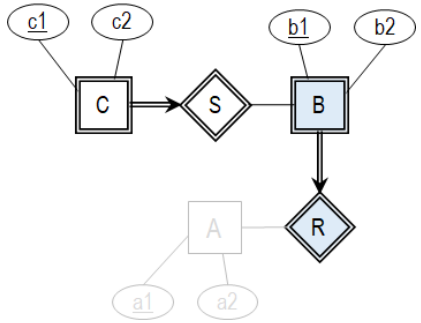
```
CREATE TABLE A (
  a1  INTEGER PRIMARY KEY,
  a2  INTEGER
);
```

Question 2

ERD

ER 3

> Same process, look for the constructs with the fewest FK. In the case of weak entity set, it will be the owning entity set.



Translation

```
CREATE TABLE B (
  a1  INTEGER REFERENCES A ON DELETE CASCADE,
  b1  INTEGER,
  b2  INTEGER,
  PRIMARY KEY (a1, b1)
);
```

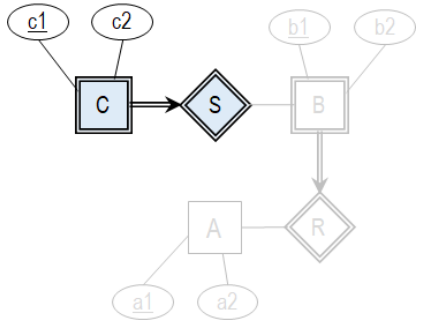
[#]Remember, weak entity set is merged with the identifying relationship set. Also, the PK must include the PK of the owning entity set.

Question 2

ERD

ER 3

> Same process, look for the constructs with the fewest FK. In the case of weak entity set, it will be the owning entity set.



Translation

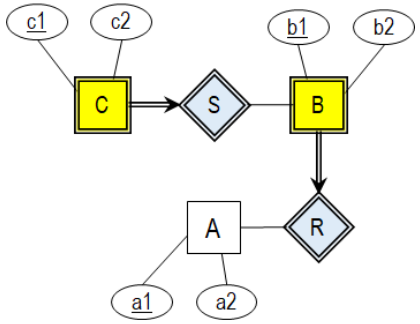
```
CREATE TABLE C (
  a1  INTEGER,
  b1  INTEGER,
  c1  INTEGER,
  c2  INTEGER,
  PRIMARY KEY (a1, b1, c1),
  FOREING KEY (a1, b1) REFERENCES B ON DELETE CASCADE
);
```

[#]Remember, weak entity set is merged with the identifying relationship set. Also, the PK must include the PK of the owning entity set.

Question 2

ERD

ER 3



Important Note

> Weak entity set is "weak" because its key attributes cannot uniquely identify the rest of the attributes.
>
> **Weak entity set must have identifying relationship set + owning entity set**

Question 2

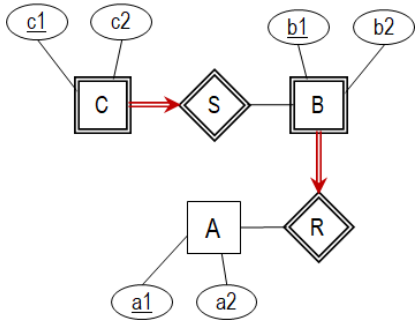
Translation
ERD
- ER 1
- ER 2
- **ER 3**

ERD

ER 3

Important Note

> Weak entity set is "weak" because its key attributes cannot uniquely identify the rest of the attributes.
>
> **Weak entity set must be connected to identifying relationship set by double-line arrow** ***(why?)***



```
postgres=# exit
Press any key to continue . . .
```