1. Post a screenshot **AND** type part of the question on top of the screenshot **so people can CTRL+F**

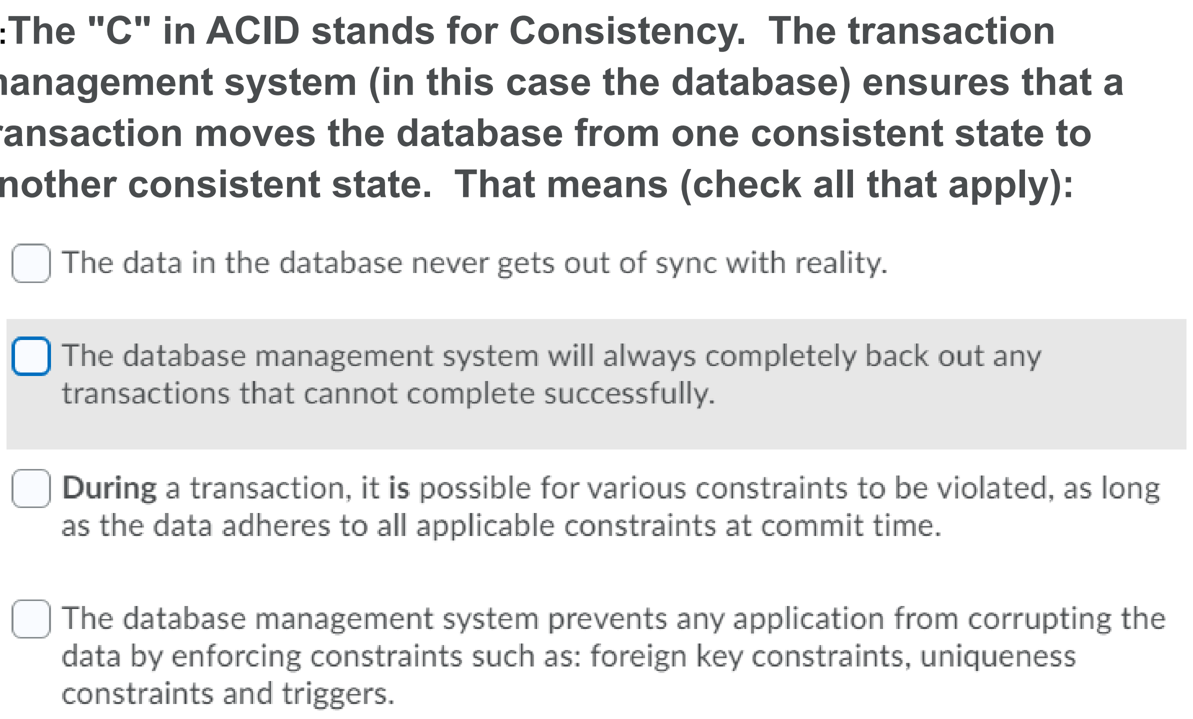
2. DO NOT REPOST THE SAME QUESTION: if it has different values, post your answer along with values for that answer

It is very important that you specify which values your answer is for

3. Once multiple people have confirmed OR if you feel very confident about your answer, highlight your answer

4. Good luck and have fun :)

5. This doc will be deleted

Q:

The "C" in ACID stands for Consistency. The transaction management system (in this case the database) ensures that a transaction moves the database from one consistent state to another consistent state. That means (check all that apply):

- ☐ The data in the database never gets out of sync with reality.
- ☐ The database management system will always completely back out any transactions that cannot complete successfully.
- ☐ **During** a transaction, it **is** possible for various constraints to be violated, as long as the data adheres to all applicable constraints at commit time.
- ☐ The database management system prevents any application from corrupting the data by enforcing constraints such as: foreign key constraints, uniqueness constraints and triggers.

A:

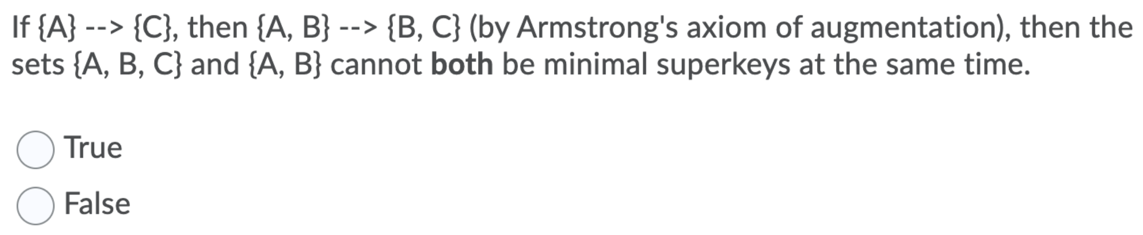
I'll say 1, 2 and 4? Tally: III
Not 3 because slide 446

Same answer

Q: If {A} --> {C}, then {A, B} --> {B, C} (by Armstrong's axiom of augmentation), then the sets {A, B, C} and {A, B} cannot both be minimal superkeys at the same time.

A: True? Tally:

Same answer

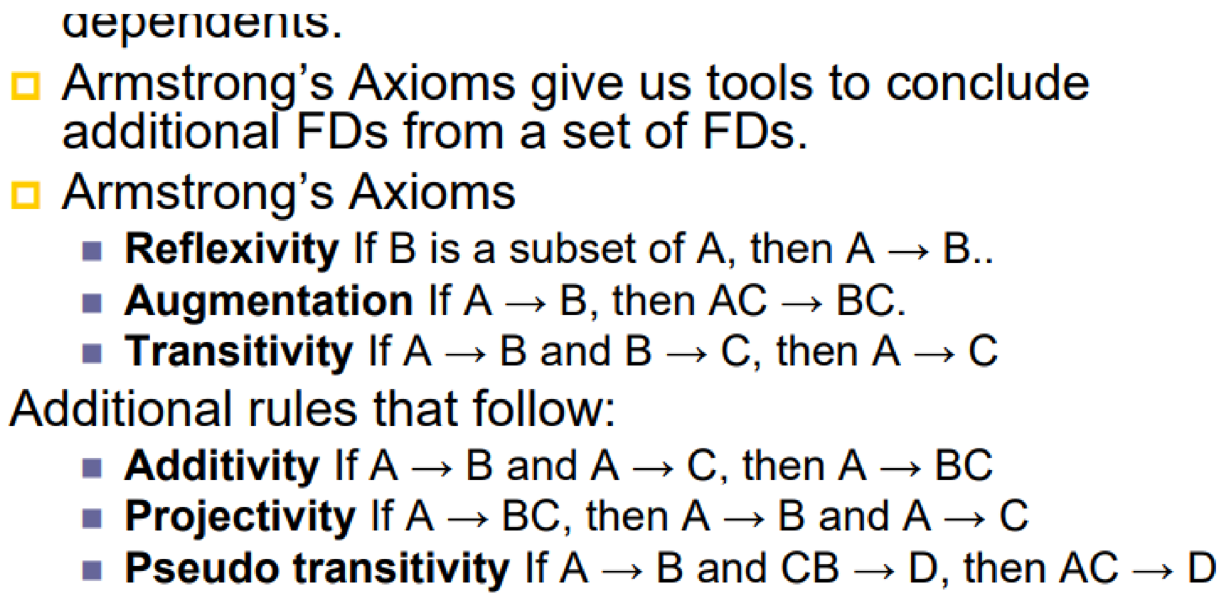


If {A} --> {C}, then {A, B} --> {B, C} (by Armstrong's axiom of augmentation), then the sets {A, B, C} and {A, B} cannot **both** be minimal superkeys at the same time.

◯ True

◯ False

Check slide 279



dependents.

- Armstrong's Axioms give us tools to conclude additional FDs from a set of FDs.
- Armstrong's Axioms
  - **Reflexivity** If B is a subset of A, then A → B..
  - **Augmentation** If A → B, then AC → BC.
  - **Transitivity** If A → B and B → C, then A → C

Additional rules that follow:

  - **Additivity** If A → B and A → C, then A → BC
  - **Projectivity** If A → BC, then A → B and A → C
  - **Pseudo transitivity** If A → B and CB → D, then AC → D

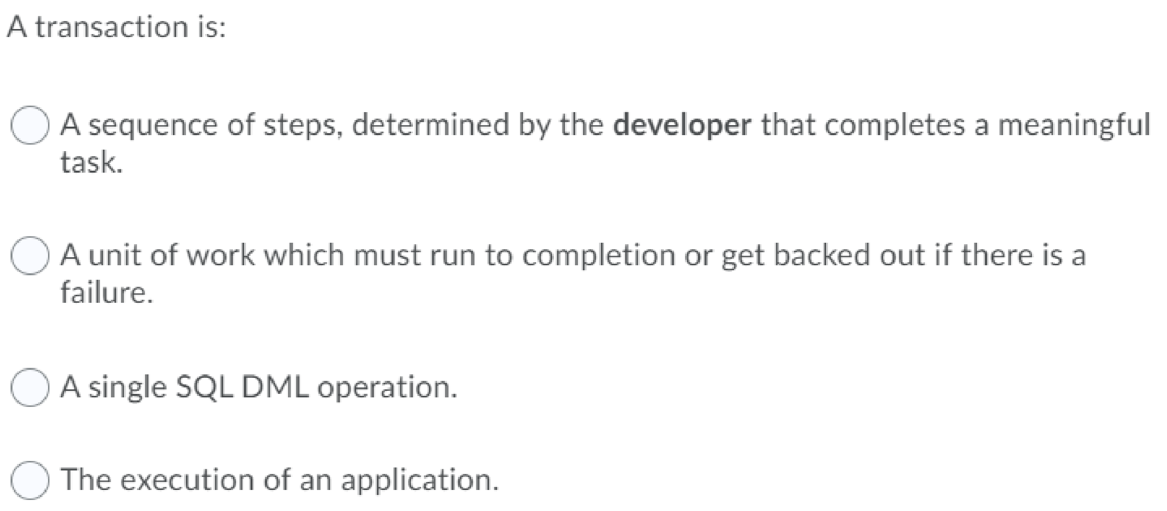
Q: In the course of normalizing a table that has a subkey, it is always possible to move each and every functional dependency from the original table into one or the other of the two tables resulting from removing the subkey.

A: true? Don't we always move a subkey to another table? Tally:

I put false, don't really know why, but i feel like that an answer Tally:

I put false

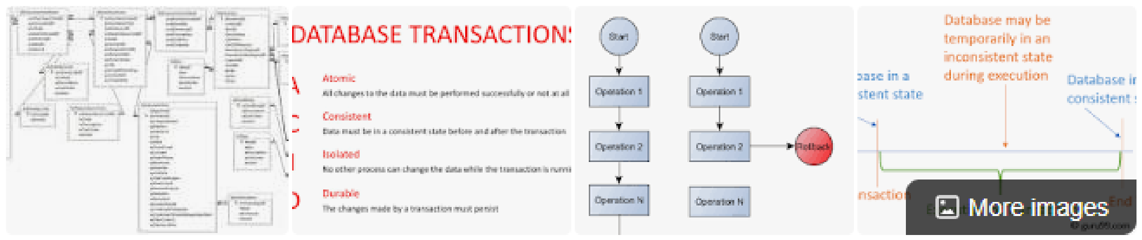
Q: a transaction is:



A transaction is:

- ○ A sequence of steps, determined by the **developer** that completes a meaningful task.
- ○ A unit of work which must run to completion or get backed out if there is a failure.
- ○ A single SQL DML operation.
- ○ The execution of an application.

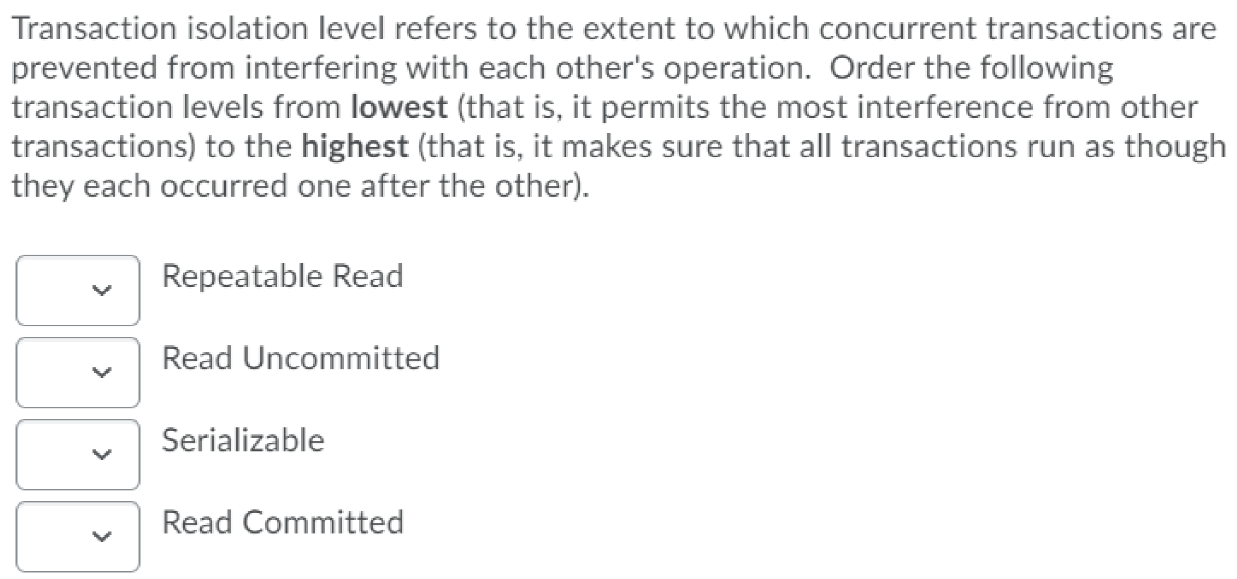
A: B Tally: +6 why not A? Slide 432 states that a transaction must either run successfully or back out thus B.
Same answer


DATABASE TRANSACTION
Atomic
All changes to the data must be performed successfully or not at all
Consistent
Data must be in a consistent state before and after the transaction
Isolated
No other process can change the data while the transaction is runni
Durable
The changes made by a transaction must persist
Start
Operation 1
Operation 2
Operation N
Rollback
Database may be temporarily in an inconsistent state during execution
More images

Transactional database

A database transaction symbolizes a unit of work performed within **a database** management system (or similar system) against **a database**, and treated in a coherent and reliable way independent of other **transactions**. A **transaction** generally represents any change in **a database**.

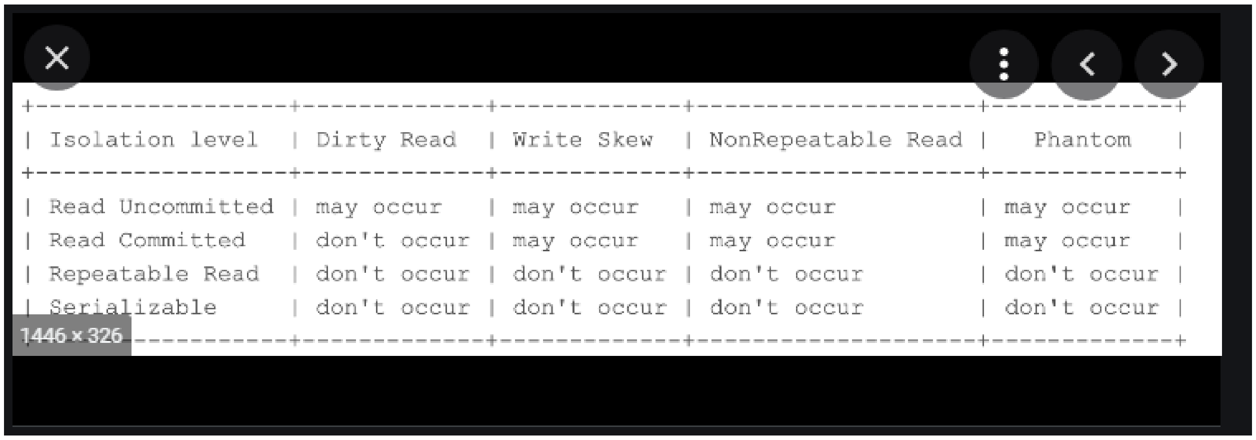
Q: transaction isolation level refers to



Transaction isolation level refers to the extent to which concurrent transactions are prevented from interfering with each other's operation. Order the following transaction levels from **lowest** (that is, it permits the most interference from other transactions) to the **highest** (that is, it makes sure that all transactions run as though they each occurred one after the other).

- [ ] Repeatable Read
- [ ] Read Uncommitted
- [ ] Serializable
- [ ] Read Committed

A:



Isolation level	Dirty Read	Write Skew	NonRepeatable Read	Phantom
Read Uncommitted	may occur	may occur	may occur	may occur
Read Committed	don't occur	may occur	may occur	may occur
Repeatable Read	don't occur	don't occur	don't occur	don't occur
Serializable	don't occur	don't occur	don't occur	don't occur

So 3,1,4,2? Tally: +3

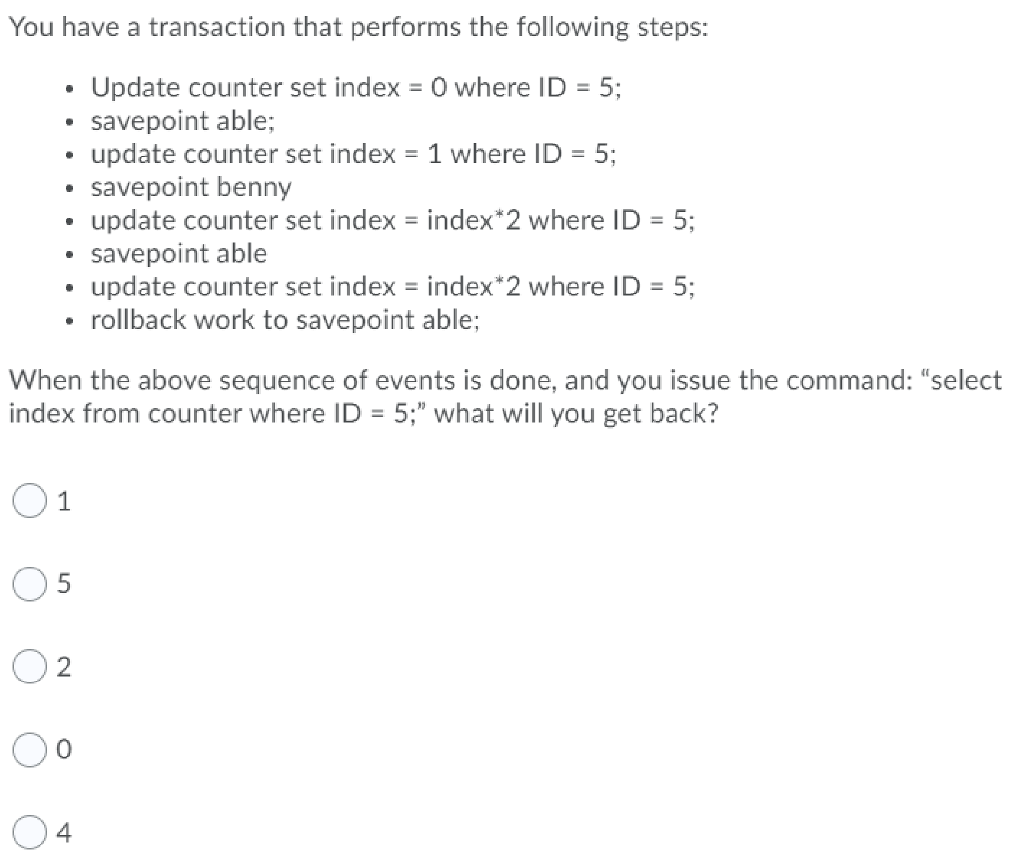
Same answer

Q: You have a transaction that performs the following steps:

- **Update counter set index = 0 where ID = 5;**
- **savepoint able;**
- **update counter set index = 1 where ID = 5;**
- **savepoint benny**
- **update counter set index = index*2 where ID = 5;**
- **savepoint able**

- **update counter set index = index*2 where ID = 5;**
- **rollback work to savepoint able;**

When the above sequence of events is done, and you issue the command: “select index from counter where ID = 5;” what will you get back?



You have a transaction that performs the following steps:

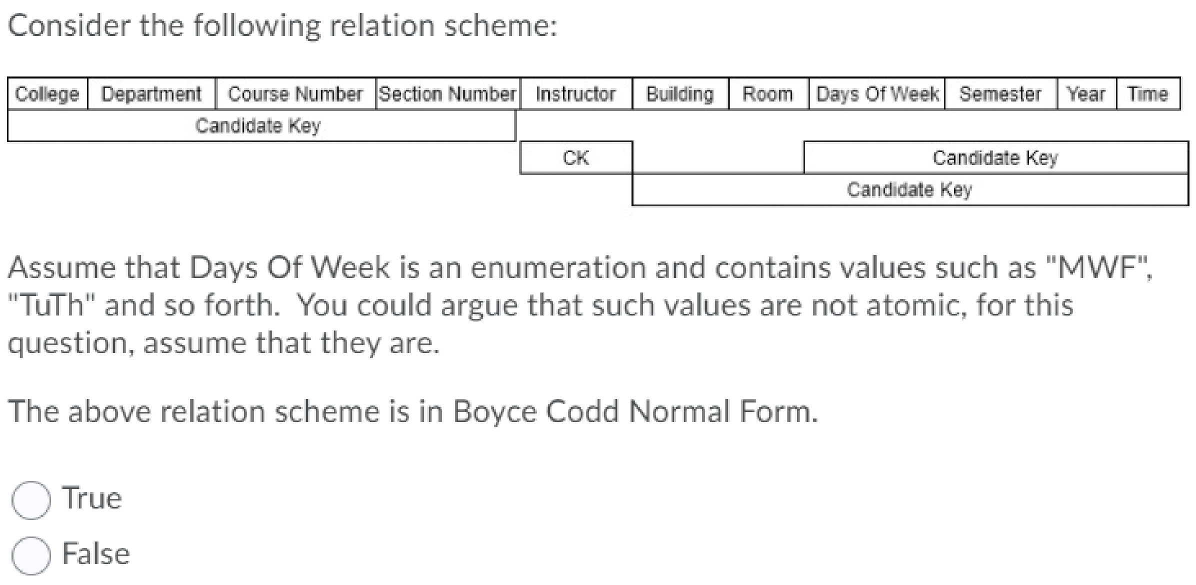
- Update counter set index = 0 where ID = 5;
- savepoint able;
- update counter set index = 1 where ID = 5;
- savepoint benny
- update counter set index = index*2 where ID = 5;
- savepoint able
- update counter set index = index*2 where ID = 5;
- rollback work to savepoint able;

When the above sequence of events is done, and you issue the command: “select index from counter where ID = 5;” what will you get back?

- ◯ 1
- ◯ 5
- ◯ 2
- ◯ 0
- ◯ 4

A: C Tally: +4
Same answer

Q: Consider the following relation scheme



Consider the following relation scheme:

College	Department	Course Number	Section Number	Instructor	Building	Room	Days Of Week	Semester	Year	Time
Candidate Key										
				CK			Candidate Key			
					Candidate Key					

Assume that Days Of Week is an enumeration and contains values such as "MWF", "TuTh" and so forth. You could argue that such values are not atomic, for this question, assume that they are.

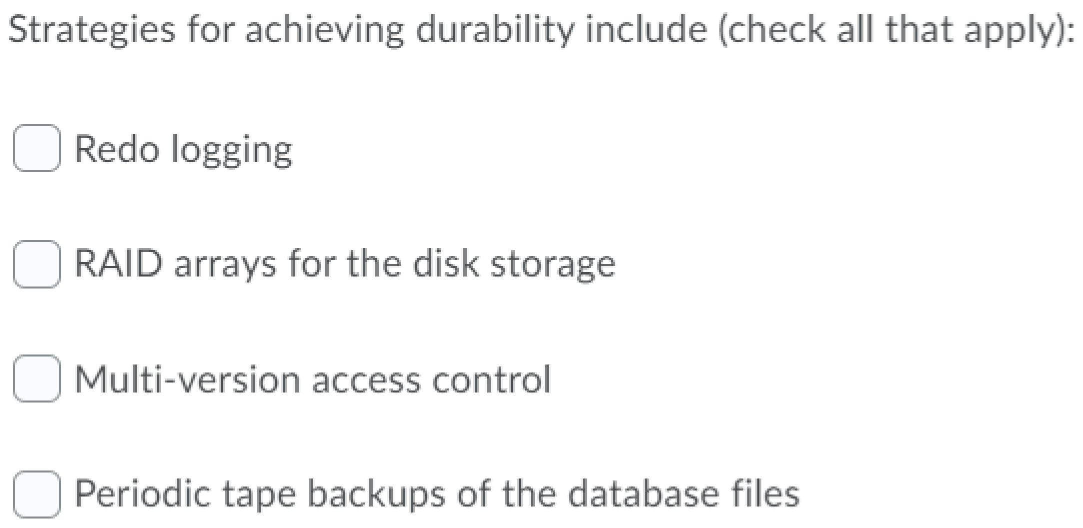
The above relation scheme is in Boyce Codd Normal Form.

◯ True

◯ False

A: T? Tally: 2
Slide 244: "Another way to say it is that a relation is in BCNF if every determinant is a candidate key."
Same answer

Q: Strategies for achieving durability include (check all that apply):



Strategies for achieving durability include (check all that apply):

☐ Redo logging

☐ RAID arrays for the disk storage

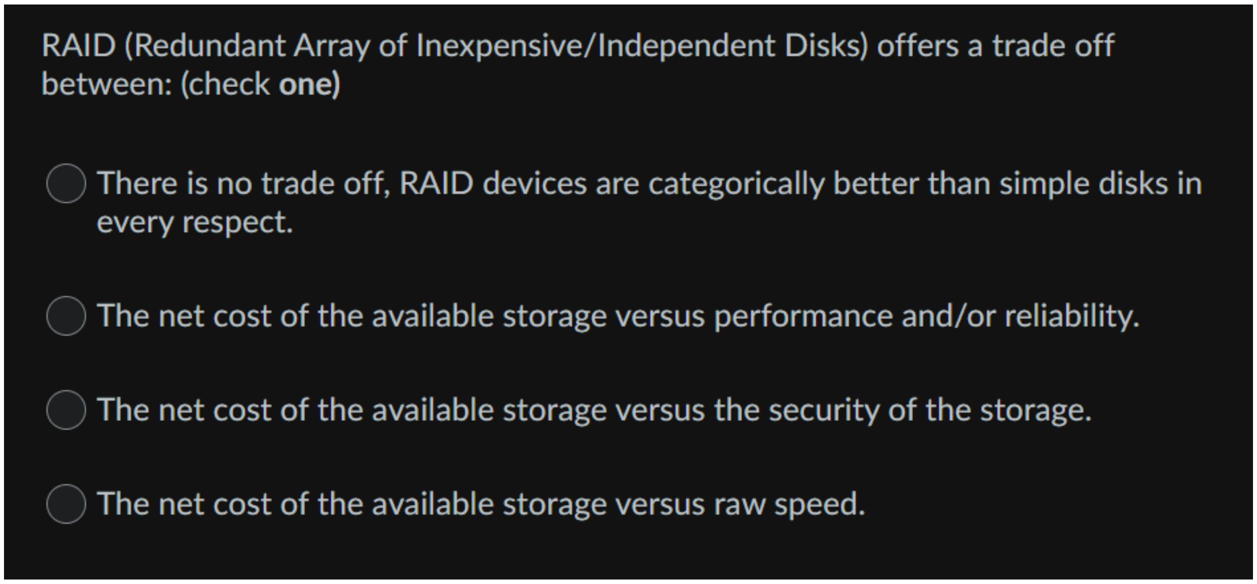
☐ Multi-version access control

☐ Periodic tape backups of the database files

A: definitely first 2 Tally:
Its not C because its multi-version concurrency control.
I think A, B, D because tape backups are also listed on slide 477. Tally: +2

Same answer

Q: RAID offers a trade off between

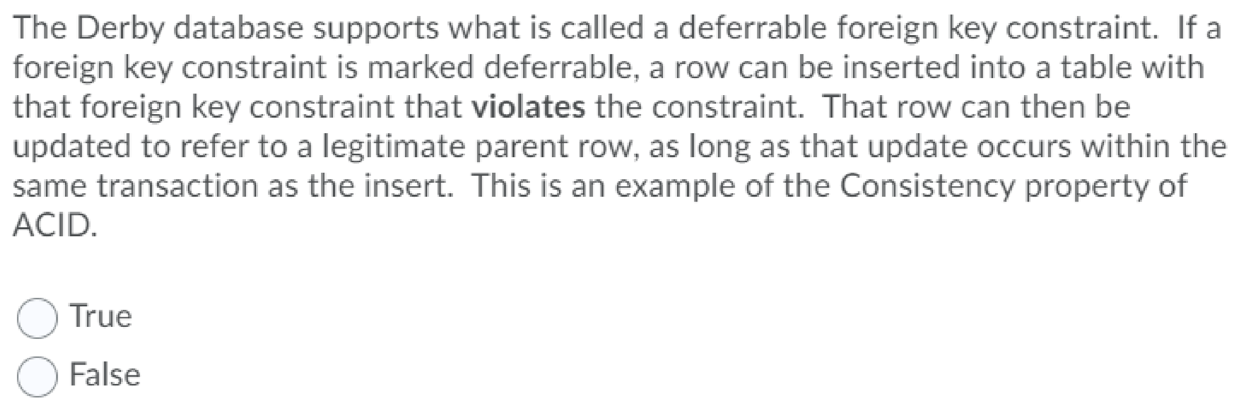


RAID (Redundant Array of Inexpensive/Independent Disks) offers a trade off between: (check **one**)

- There is no trade off, RAID devices are categorically better than simple disks in every respect.
- The net cost of the available storage versus performance and/or reliability.
- The net cost of the available storage versus the security of the storage.
- The net cost of the available storage versus raw speed.

A:
I think its B, because raid can be raid0 for more performance but less net storage and raid1 is mirroring drives
Yeah I agree to this as well ^
Tally: +1
Same answer

Q: The Derby database supports what is called

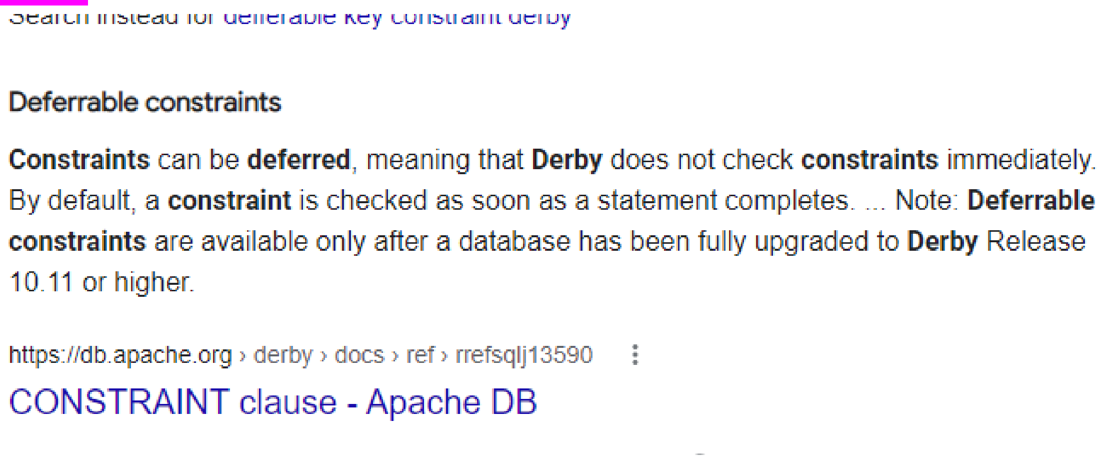


The Derby database supports what is called a deferrable foreign key constraint. If a foreign key constraint is marked deferrable, a row can be inserted into a table with that foreign key constraint that **violates** the constraint. That row can then be updated to refer to a legitimate parent row, as long as that update occurs within the same transaction as the insert. This is an example of the Consistency property of ACID.

- True
- False

A:true? Tally: +2

Same answer


Search instead for deferrable key constraint derby

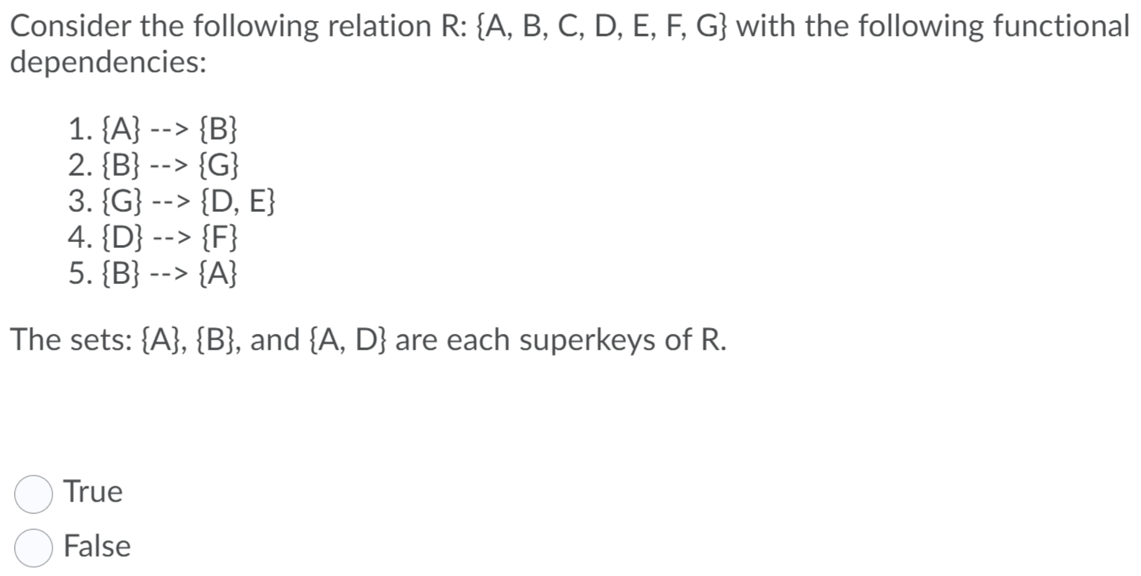
Deferrable constraints

Constraints can be **deferred**, meaning that **Derby** does not check **constraints** immediately. By default, a **constraint** is checked as soon as a statement completes. ... Note: **Deferrable constraints** are available only after a database has been fully upgraded to **Derby** Release 10.11 or higher.

https://db.apache.org › derby › docs › ref › rrefsqlj13590

CONSTRAINT clause - Apache DB

Q:Consider the following relation R: {A, B, C, D, E, F, G} with the following functional dependencies:


Consider the following relation R: {A, B, C, D, E, F, G} with the following functional dependencies:

1. {A} --> {B}
2. {B} --> {G}
3. {G} --> {D, E}
4. {D} --> {F}
5. {B} --> {A}

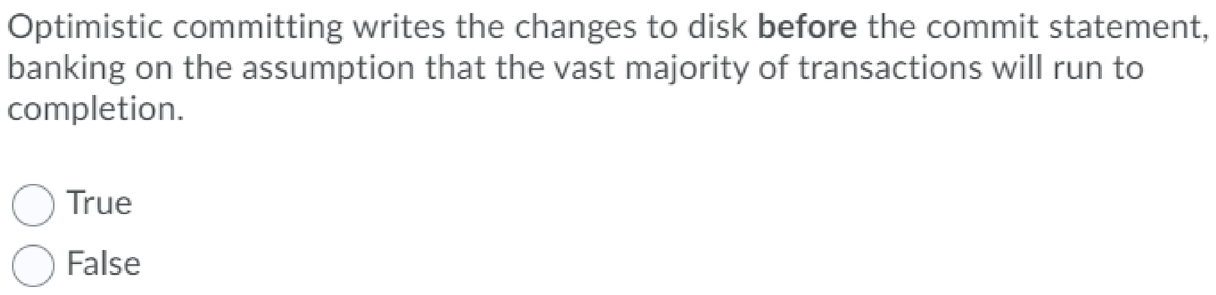
The sets: {A}, {B}, and {A, D} are each superkeys of R.

○ True
○ False

A: true? Tally:
Same answer ^

I'm thinking it's False because {c} is missing as a functional dependency?
Tally:+4
From the textbook: A super key always functionally determines all of the other attributes in a relation (as well as itself).

Q: Optimistic committing writes the changes to disk before the commit statement, banking on the assumption that the vast majority of transactions will run to completion


Optimistic committing writes the changes to disk **before** the commit statement, banking on the assumption that the vast majority of transactions will run to completion.

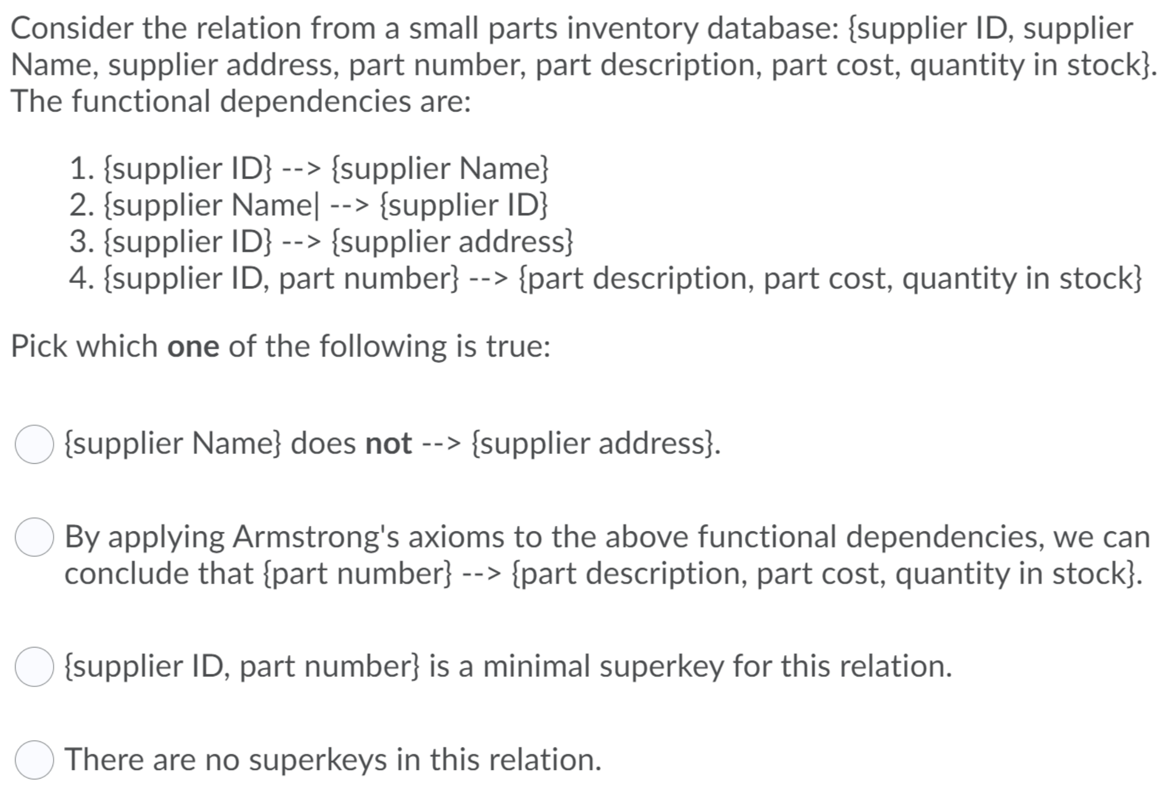
○ True

○ False

A:
True?: Tally: +3
Same answer

- **From the slides:**
- **Optimistic committing writes the changes to disk and manages the before images for transaction isolation purposes and undo.**

Q:Consider the relation from a small parts inventory database: {supplier ID, supplier Name, supplier address, part number, part description, part cost, quantity in stock}. The functional dependencies are:


Consider the relation from a small parts inventory database: {supplier ID, supplier Name, supplier address, part number, part description, part cost, quantity in stock}. The functional dependencies are:

1. {supplier ID} --> {supplier Name}
2. {supplier Name| --> {supplier ID}
3. {supplier ID} --> {supplier address}
4. {supplier ID, part number} --> {part description, part cost, quantity in stock}

Pick which **one** of the following is true:

○ {supplier Name} does **not** --> {supplier address}.

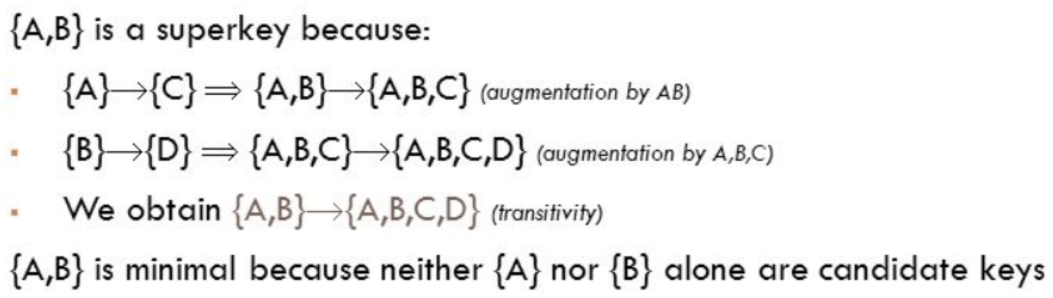
○ By applying Armstrong's axioms to the above functional dependencies, we can conclude that {part number} --> {part description, part cost, quantity in stock}.

○ {supplier ID, part number} is a minimal superkey for this relation.

○ There are no superkeys in this relation.

A: **By applying Armstrong's axioms to the above functional dependencies, we can conclude that {part number} --> {part description, part cost, quantity in stock}.**
I'm not too sure though ^ Tally: +2
Same answer ^

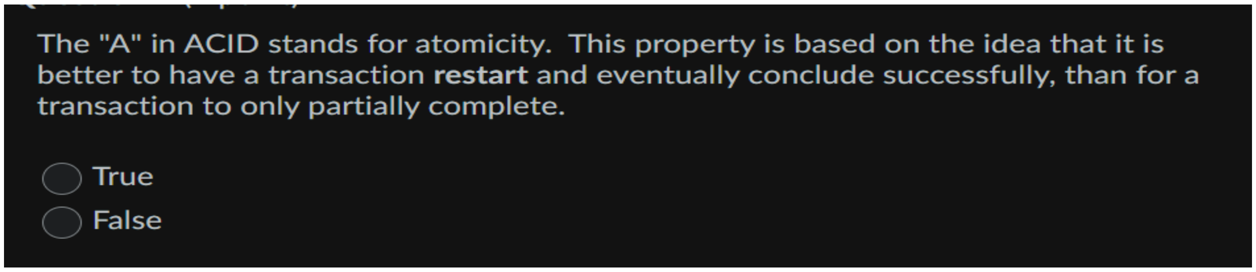
Isn't it c since supplierID and part number together are functions to everything else? Idk +3
Yeah if we would remove one of the attributes it won't be able to uniquely identify looking at the FD.


{A,B} is a superkey because:

- $\{A\}\longrightarrow\{C\} \Rightarrow \{A,B\}\longrightarrow\{A,B,C\}$ *(augmentation by AB)*
- $\{B\}\longrightarrow\{D\} \Rightarrow \{A,B,C\}\longrightarrow\{A,B,C,D\}$ *(augmentation by A,B,C)*
- We obtain $\{A,B\}\longrightarrow\{A,B,C,D\}$ *(transitivity)*

{A,B} is minimal because neither {A} nor {B} alone are candidate keys

Q: The “A” in ACID stands for atomicity.



The "A" in ACID stands for atomicity. This property is based on the idea that it is better to have a transaction **restart** and eventually conclude successfully, than for a transaction to only partially complete.

True
False

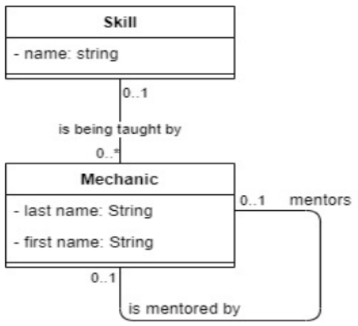
A: True Tally: + 3
Same answer

Q:You are given the following business rules:

You are given the following business rules:

- A mechanic may mentor another mechanic, but never more than one at a time.
- A mechanic may be mentored by another mechanic, but never more than one at a time.
- When a mechanic mentors another mechanic, it is for the purposes of teaching them a particular skill.
- To prevent our clerical staff from putting in invalid skill names, we have created a Skill lookup table and used a foreign key constraint to constrain the name of the skill passed on by the mentorship relationship.
- The design team has elected to make this relationship one-directional. The mentee record has the ID of their mentor, but the mentor record does not have the ID of their mentee.

One possible UML diagram for this scenario is:



Check all that apply:

Check all that apply:

☐ The name of the skill being passed and the ID the mentor are both optional.

☐ We **could** implement this as a categorization in which Mechanic may or may not be a **mentored** mechanic. In that scenario, Skill would be the parent to Mentored Mechanic, and the recursive relationship shown would go from Mechanic to Mentored Mechanic.

☐ We **could** implement this with a junction table connecting one instance of Mechanic to another instance of Mechanic. The primary key of Mentored Mechanic would be the ID of the mechanic being mentored. A second, candidate key would be the ID of the mentor.

☐ This would be better modeled as a Many to Many with**out** history with Skill coming in as a parent to the association class.

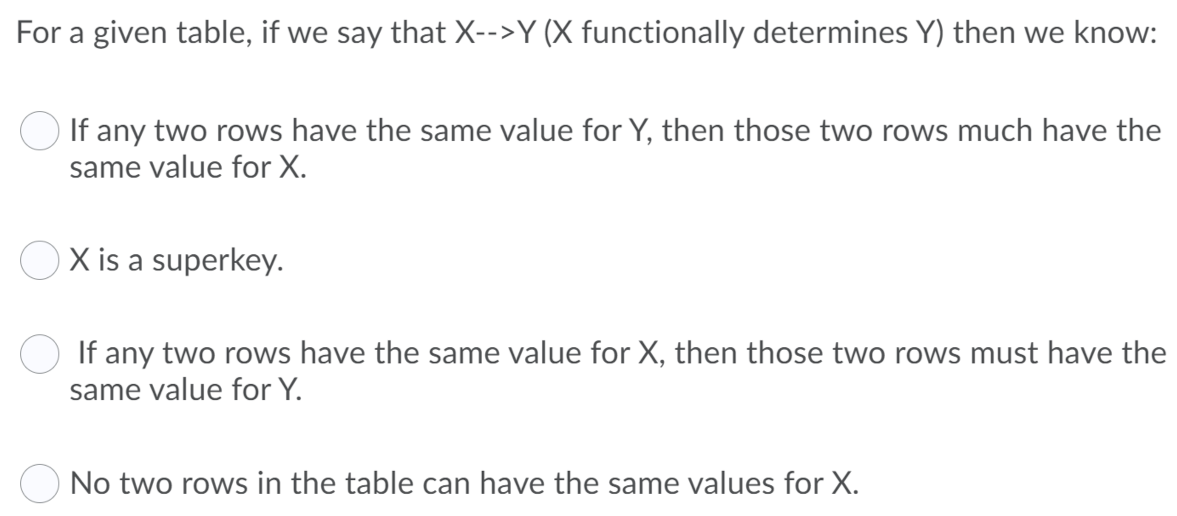
A:
3,4? But isn't 3 false since mentors can't teach more than one mechanic at a time? And same for vice versa?

I put 3 and 4

2,~~1 (categorization will make this optional)~~ i don't think it's 1? Ah yeah wouldn't make sense for the business rule, also it wouldn't be 2 since Skill would have to be applied to both types of mechanic classes

---

Q:For a given table, if we say that X-->Y (X functionally determines Y) then we know:



For a given table, if we say that X-->Y (X functionally determines Y) then we know:

○ If any two rows have the same value for Y, then those two rows much have the same value for X.

○ X is a superkey.

○ If any two rows have the same value for X, then those two rows must have the same value for Y.

○ No two rows in the table can have the same values for X.

A: A? tally:

From a page on FD: In other words, a dependency FD: *X* → *Y* means that the values of *Y* are determined by the values of *X*. Two tuples sharing the same values of *X* will necessarily have the same values of *Y*.
Wouldn't that mean A is reversed? So C? Tally: + 2
Same answer (C) ^

You guys think that after turning it in we can see the answers?
Doubt it
So far someone saying u cant see it but hopefully he puts it on
someone said that the professor will put it on, so hopefully someone turns it in and posts it here