Status	Finished
Started	Wednesday, 15 January 2025, 10:02 AM
Completed	Wednesday, 15 January 2025, 10:07 AM
Duration	4 mins 38 secs
Grade	**3.81** out of 5.00 (**76.25**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Der SQL- Standard definiert vier Ebenen der Transaktionsisolation. Am strengsten ist Serializable, das im Standard in einem Absatz definiert wird, der besagt, dass jede gleichzeitige Ausführung einer Reihe serialisierbarer Transaktionen garantiert den gleichen Effekt hat, als würden sie einzeln in einer bestimmten Reihenfolge ausgeführt.

Die anderen drei Ebenen (Read uncommitted, Read committed, Repeatable read) werden anhand von Phänomenen (Dirty Read, Nonrepeatable Read, Phantom Read) definiert, die sich aus der Interaktion zwischen gleichzeitigen Transaktionen ergeben und nicht auf jeder Ebene auftreten dürfen. Der Standard stellt fest, dass aufgrund der Definition von Serializable keines dieser Phänomene auf dieser Ebene möglich ist.

- (●) True ✓
- ( ) False

https://www.postgresql.org/docs/16/transaction-iso.html

Question **2**

Correct

Mark 1.00 out of 1.00

In PostgreSQL können Sie jede der vier Standard-Transaktionsisolationsstufen anfordern, intern sind jedoch nur drei unterschiedliche Isolationsstufen implementiert, dh der Read Uncommitted-Modus von PostgreSQL verhält sich wie Read Committed. Dies liegt daran, dass dies die einzig sinnvolle Möglichkeit ist, die Standardisolationsstufen der Multiversion-Parallelitätskontrollarchitektur von PostgreSQL zuzuordnen.

Die Tabelle zeigt auch, dass die Repeatable Read-Implementierung von PostgreSQL keine Phantom-Lesevorgänge zulässt. Dies ist im Rahmen des SQL-Standards akzeptabel, da der Standard festlegt, welche Anomalien nicht bei bestimmten Isolationsstufen auftreten dürfen; höhere Garantien sind akzeptabel.

Vervollständigen Sie die Tabelle!

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not Possible	Possible	Possible	Possible
Repeatable read	Not Possible	Not Possible	Allowed, but not in PG	Possible
Serializable	Not Possible	Not Possible	Not Possible	Not possible

Not Possible Possible

Die Antwort ist richtig.

https://www.postgresql.org/docs/16/transaction-iso.html

Question **3**

Partially correct

Mark 0.81 out of 1.00

BASE stands for basically available, soft [state] ✓ , and eventually consistent. The acronym highlights that BASE is opposite of ACID, like their chemical equivalents.

Basically available is the database's [simultaneously] ✗ accessibility by users at all times. One user doesn't need to wait for others to finish the [transaction] ✓ before updating the record. For example, during a sudden surge in traffic on an ecommerce platform, the [system] ✓ may prioritize serving product listings and accepting orders. Even if there is a slight delay in updating inventory quantities, users continue to check out items.

Soft [**state**] ✓ refers to the notion that data can have transient or temporary [state] ✓ s that may change over time, even without external triggers or inputs. It describes the record's transitional [state] ✓ when several applications update it [simultaneously] ✓ . The record's value is eventually finalized only after all [transaction] ✓ s complete. For example, if users edit a social media post, the change may not be visible to other users immediately. However, later on, the post updates by itself (reflecting the older change) even though no user triggered it.

Eventually consistent means the record will achieve [consistency] ✓ when all the [transaction] ✗ updates have been completed. At this point, applications querying the record will see the same value. For example, consider a distributed document editing [concurrent] ✗ where multiple users can [simultaneously] ✓ edit a document. If User A and User B both edit the same section of the document [simultaneously] ✓ , their local copies may temporarily differ until the changes are propagated and synchronized. However, over time, the [system] ✓ ensures eventual [consistency] ✓ by propagating and merging the changes made by different users.

[state] [concurrent] [transaction] [simultaneously] [consistency] [system]

Die Antwort ist teilweise richtig.

You have correctly selected 13.

Question **4**

Incorrect

Mark 0.00 out of 1.00

GK 10.4 Datenmanagement - Verteilte Datenbanken | Korrektheitsanforderungen

Weisen Sie die entsprechenden Anforderungen dem Text richtig zu!

Wie sind die Korrektheitsanforderungen bei der Fragmentierung von verteilten Datenbanken eingeteilt?

- Die Ursprungsrelation lässt sich aus den Fragmenten wiederherstellen und stellt somit die [`Fragmentierung`] ✗ sicher.
- Jedes Datum ist einem Fragment zugeordnet und kann somit die [`Korrektheit`] ✗ erfüllen.
- Die einzelnen Fragmente überlappen sich nicht, d.h. ein Datum ist nicht mehreren Fragmenten zugeordnet. Dies stellt somit die [`Vollständigkeit`] ✗ sicher.

[Rekonstruierbarkeit] [Einzigartigkeit] [Disjunktheit]

Die Antwort ist falsch.

Question **5**

Correct

Mark 1.00 out of 1.00

Sind diese Punkte richtig oder falsch?

- A Firebase project is like a container for all your apps and any resources and services provisioned for the project.
- A Firebase project can have one or more Firebase Apps registered to it (for example, both the iOS and Android versions of an app, or both the free and paid versions of an app).
- All Firebase Apps registered to the same Firebase project share and have access to all the same resources and services provisioned for the project.

- (●) True ✓
- ( ) False

Well done!