Klausur
BC414

Name: Völker
Vorname: Markus

Note: 100 % = 1 ,0

1. **Welche der folgenden Aussagen treffen zu:**

☒ Es ist möglich, vom aktuellen Mandanten (z.B. Mandant 805) aus, Daten anderer Mandanten zu ändern, wenn diese im gleichen R/3-System angelegt sind.

☒ Alle datenbankändernden Anweisungen (INSERT, UPDATE, DELETE, MODIFY) laden bei fehlerfreier Ausführung die Systemvariable SY-SUBRC mit dem Wert 0.

☐ Ist bei der Datenbankänderung ein Fehler aufgetreten, sollte entsprechend nachfolgendem Beispiel ein Rollback der Datenbank angestoßen werden:

```
UPDATE sbook from wa_sbook.
IF sy-subrc <> 0.
   MESSAGE e001(zthw).
ENDIF.
```

☒ Eine DB-LUW ist eine nicht teilbare Folge von Datenbankoperationen, die die Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt.

☐ Die Datenbank-LUW wird mit einen Datenbank-Commit abgeschlossen. Dieser wird im ABAP-Programm durch die Anweisung COMMIT WORK gesetzt.

☐ Die Aktualisierung der Daten erfolgt ausschließlich über den Verbucherworkprozess (Standard: asynchrone Verbuchung).

☒ Alle Teilschritte eines betriebswirtschaftlichen Vorganges werden in einer SAP-LUW zusammengefasst.

☒ Eine SAP-LUW muss so programmiert werden, dass entweder alle in ihr angewiesenen Datenbankänderungen ausgeführt werden, oder keine (Alles oder Nichts-Prinzip).

☒ Eine Datenbank-LUW wird z.B. durch das Senden eines SAP-Bildschirmbildes oder das Senden einer Dialognachricht (z.B. MESSAGE S000(YTHW).) ausgelöst.

2. **Die Anweisung**

```
UPDATE sflight
    SET   connid = '400'
    WHERE mandt  = '805' and
          carrid = 'LH' and
          connid = '401'.
```

☒ ist syntaktisch falsch.

☐ bewirkt, dass in allen Datensätzen der Tabelle SFLIGHT, die der WHERE-Bedingung entsprechen, das Feld CONNID auf 400 gesetzt wird.

☐ kann syntaktisch korrekt wie folgt ersetzt werden:

```
DATA: wa_sflight TYPE sflight.
wa_sflight-mandt = '805'.
wa_sflight-carrid = 'LH'.
wa_sflight-connid = '401'.
UPDATE sflight FROM wa_sflight.
```

3. **Welche Aussagen zum Sperrkonzept sind richtig:**

☒ Sperren werden gesetzt, um zu verhindern, dass mehrere Benutzer den gleichen Datensatz bearbeiten können.

☐ Um Sperren auch über mehrere Dialogschritte aufrechterhalten zu können, gibt es im SAP-System eine Sperrtabelle. In diese werden alle Datensätze eingetragen, die über eine SELECT-Anweisung gelesen wurden.

☒ Voraussetzung für das Sperren von Datensätzen ist ein Sperrobjekt. Dieses wird im ABAP-Dictionary angelegt. Beim Aktivieren dieses Sperrobjektes werden automatisch zwei Funktionsbausteine (ENQUEUE_<Name des Sperrobjektes> und DEQUEUE_<Name des Sperrobjektes>) erzeugt. Der ENQUEUE-Funktionsbaustein wird im ABAP-Programm aufgerufen, um einen Sperreintrag in der Sperrtabelle zu erzeugen, der DEQUEUE-Funktionsbaustein wird aufgerufen, um den Sperreintrag wieder zu löschen.

☐ Um zu verhindern, dass das ABAP-Programm möglicherweise mit veralteten Daten arbeitet, ist der zu bearbeitende Datensatz unmittelbar nach dem Lesevorgang zu sperren. Die Sperre wird erst gelöscht, wenn die Änderungen in die Datenbanktabelle(n) eingetragen, oder die Bearbeitung abgebrochen wurde.

☒ Hat Programm A eine Sperre vom Typ E auf einen Datensatz gesetzt, kann Progamm B auf diesen Datensatz keine Sperre vom Type E setzen.

☐ Hat Programm A eine Sperre vom Typ S auf einen Datensatz gesetzt, kann Progamm B auf diesen Datensatz keine Sperre vom Type S setzen.

☒ Hat Programm A eine Sperre vom Typ S auf einen Datensatz gesetzt, kann Progamm B auf diesen Datensatz keine Sperre vom Type E setzen.

4. **Welche Aussagen zur Organisation von Datenbankänderungen sind richtig?**

☐ Besteht eine Datenbank-LUW aus mehreren Dialogschritten, können in jedem Dialogschritt Datenbankänderungen in der Form UPDATE <datenbanktabelle> FROM <struktur> programmiert werden. Das Rücksetzen der Datenbank wird dadurch nicht

beeinträchtigt, d.h. treten in einem beliebigen Dialogschritt Fehler beim Aktualisieren der Datenbank auf, können alle durch einen Datenbank-Rollback zurückgesetzt werden.

- [x] Um durch einen Datenbank-Rollback alle Datenbankänderungen einer SAP-LUW zurücksetzen zu können, müssen diese in einer einzigen DB-LUW ausgeführt werden.

- [ ] Die Bündelung der Datenbankänderungen auf eine DB-LUW kann auch durch verzögert ausgeführte Unterprogramme erreicht werden. Mit der Anweisung PERFORM update IN UPDATE TASK wird erreicht, dass das Unterprogramm UPDATE nicht sofort ausgeführt, sondern vorerst nur in eine Systemtabelle eingetragen wird. Die in dieser Systemtabelle „gesammelten" Unterprogramme werden erst ausgeführt, wenn das Programm auf die Anweisung COMMIT WORK stößt.

- [ ] Wird mit der Technik „verzögert ausgeführte Unterprogramme" gearbeitet, müssen keine Sperren gesetzt werden, weil die Abarbeitung der Unterprogramme in **einer** DB-LUW erfolgt. Die physischen Sperren, die die Datenbank setzt, sind deshalb ausreichend.

- [x] Beim Einsatz der Verbuchungstechnik (CALL FUNCTION ... IN UPDATE TASK) wird die Verbuchung nicht durch das Dialogprogramm sondern immer durch ein Systemprogramm, dem Verbucherprogramm, ausgeführt.

- [x] Bei der asynchronen Verbuchung wartet das Dialogprogramm nicht auf die Ausführung der Datenbankänderungen, die durch das Verbucherprogramm vorgenommen werden.

- [ ] Standardmäßig erbt der Verbucher die im Dialogprogramm gesetzten Datenbanksperren und setzt diese nach der Verbuchung zurück. Treten bei der Verbuchung Fehler auf, wird ein Rollback der Datenbank ausgelöst. Die Datensatzsperren werden in diesem Fall nicht zurückgesetzt.

- [x] Die V2-Verbuchung wird nur dann angestoßen, wenn die V1-Verbuchung fehlerfrei ausgeführt wurde.

- [ ] Nach der erfolgreichen V2-Verbuchung werden die Sperren zurückgesetzt.

- [x] Bei jeder Aktion auf der Datenbank, die Änderungen an Tabellen bewirkt, wird der zu ändernde Satz von der Datenbank physisch gesperrt. Für die Dauer der physischen Sperre kann auf den gesperrten Datensatz nicht zugegriffen werden. Auch Lesezugriffe sind nicht möglich. Die physische Sperre wird von der Datenbank am Ende der DB-LUW (nach dem DB-COMMIT) zurückgenommen.

- [ ] Um die Zeit, in der Datensätze häufig benutzter Datenbanktabellen von der Datenbank physisch gesperrt werden, gering zu halten, sollte folgende Reihenfolge der Datenbankänderungen eingehalten werden:
  1. Änderungen an performanceunkritischen Tabellen durchführen
  2. Änderungen an performancekritischen Tabellen durchführen.
  3. Neue Einträge erzeugen (INSERT)

5. **In einem Programm sollen Datensätze in der Tabelle SBOOK geändert werden. Gegeben ist der folgende Programmausschnitt. Er soll so ergänzt werden, dass die Datenbankänderungen mittels asynchroner Verbuchungstechnik in die Datenbank eingetragen werden. Zur Verbuchung steht der Verbucher-Funktionsbaustein Z_UPDATE_SBOOK zur Verfügung. Der Exportparameter dieses Funktionsbausteins heißt ZSBOOK. Ihm ist die Struktur SBOOK zu übergeben.**

```
REPORT zupdate_sbook_scustom.
TABLES: sbook.
DATA: ok_code TYPE sy-ucomm, ok_save TYPE sy-ucom.
PARAMETERS: p_carrid type sbook-carrid,
            p_connid type sbook-connid,
            p_fldate type sbook-fldate,
            p_bookid type sbook-bookid.
START-OF-SELECTION.
 CALL SCREEN 100.
*---------------------------------------------------------------------
*Ablauflogik des Dympros 100
PROCESS BEFORE OUTPUT.
 MODULE laden_100.
PROCESS AFTER INPUT.
 MODULE user_command_100.
*---------------------------------------------------------------------
MODULE laden_100 output.
*Datensatz sperren
PERFORM enq_sperren_sbook.
SELECT SINGLE * FROM sbook WHERE carrid = p_carrid
      and connid = p_connid and fldate = p_fldate
      and bookid = p_bookid.
IF sy-subrc <> 0.
 CALL FUNCTION 'DEQUEUE_ALL'.
 LEAVE TO SCREEN 0.
ENDIF.
ENDMODULE.
*---------------------------------------------------------------------
MODULE user_command_100.
 ok_save = ok_code. clear ok_code.
 CASE ok_code.
  WHEN 'SAVE'.
* Programmieren Sie hier die Datenbankänderung mittels
* asynchroner Verbuchertechnik.
```

CALL function 'ZUPDATE_BOOK'
Exporting zsbook = sbook.
in update task.

COMMIT WORK.

6. **Die asynchrone Verarbeitung wird verwendet, wenn:**
(eine Antwort)

- [x] eine schnelle Antwortzeit wichtiger ist als die sofortige Aktualisierung der Datenbank.
- [ ] eine Transaktion in einem Hintergrund-Work-Prozeß läuft.
- [ ] die Datenbank immer auf dem aktuellsten Stand sein muß.

7. **Wodurch wird die SAP LUW im Dialogteil einer Transaktion beendet bei der asynchronen Verbuchung?**
(eine Antwort).

- [ ] CALL FUNCTION <FB> IN UPDATE TASK
- [ ] Beginn des nächsten Dialogschritts
- [x] Anweisung COMMIT WORK
- [ ] CALL SCREEN
- [ ] LEAVE TO LIST-PROCESSING

8. **Warum wird Verbuchung eingesetzt?**
(eine Antwort)

- [ ] Sammeln von Änderungen im Dialog um Sie gemeinsam ausführen zu können
- [ ] Erzeugen von Verbuchungsfunktionsbausteinen, die mehrfach verwendet werden können
- [x] Entlastung der Dialogworkprozesse
- [ ] Zeitversetztes Ausführen der Datenbankänderungen
- [ ] Trennung in V1 und V2 Verbuchung

9. **Welche Einschränkungen gibt es für die Schnittstelle eines Verbuchungsfunktiobausteins ?**

- [x] Keine Referenzübergabe
- [ ] Keine Importparameter
- [x] Keine Exportparameter
- [ ] Keine Exceptions

10. **Gegeben ist der folgende Aufruf eines Funktionsbausteins zum setzen einer Sperre:**

CALL FUNCTION 'ENQUEUE_ESTRDIR'

EXPORTING name = 'ZZESTOI'
EXCEPTIONS foreign_lock =1.

Wie reagiert das System wenn das Objekt von einem anderen Benutzer gesperrt ist?

- ☐ Transaktion wird beendet
- ☐ Das System wartet bis die Sperre gelöst wird
- ☐ Das System sendet eine Nachricht
- ☒ Die exception foreign_lock wird ausgelöst
- ☐ Der Kummulationszähler wird um 1 erhöht

11. Gegeben ist der folgende Programmausschnitt:

```
Do 5 times.
        Perform u1 on commit.
Enddo.
```

Wie oft wird das Unterprogramm u1 in die Systemtabelle eingetragen?

- ☒ 1 mal
- ☐ 5 mal
- ☐ endlos

12. **Das folgende Beispielprogramm ist gegeben:**

```
a = 1.
PERFORM F ON COMMIT.
a = 2.
PERFORM F ON COMMIT.
a = 3.
COMMIT WORK.
FORM f.
  CALL FUNCTION 'UPD_FM' IN UPDATE TASK
  EXPORTING PAR = A.
ENDFORM.
```

Wie oft wird der Funktionsbaustein 'UPD_FM' durch den Verbuchungstask ausgeführt? Was ist der Wert des Export-Parameters PAR beim Aufruf?

- ☐ 1 mal; par = 1
- ☐ 2 mal; par = 2
- ☐ endlos; par = blank
- ☒ 1 mal ; par = 3

13. **Durch welche Aktion entstehen Funktionsbausteine zum Sperren und Entsperren?**

- ☐ Anlegen der Funktionsbausteine im Functionbuilder
- ☐ Sichern des Sperrobjekts im ABAP Dictionary
- ☒ Aktivieren eines im ABAP Dictionary definierten Sperrobjekts

14. **Eine Transaktion soll einen Datensatz ändern und dabei die asynchrone Verbuchungstechnik einsetzen (Die Sperre wurde mit _scope = 2 gesetzt). Welche Schritte werden im ABAP-Programm durchgeführt? Geben Sie die Reihenfolge der Anweisungen an.**

2 ☐ Datensatz lesen

1 ☐ Datensatz sperren

3 ☐ Zu ändernde Daten an die Verbuchung übergeben

4 ☐ Datensatz entsperren (macht Verbuchung)

5 ☐ Commit Work absetzen