Probeklausur DB1

Aufgabe 1)

Richtig	Falsch	
x		Mit dem Befehl „Revoke all privileges“ werden dem Benutzer alle Rechte entzogen.
x		Eine Benutzersicht ist änderbar, wenn es Projektionen, Selektionen und nur Rechenoperationen verwendet.
	x	Mit dem Befehl “DELETE INDEX” entfernt man einen Index.
x		Bei einem Cross-Join wird das Kreuzprodukt der verbundenen Tabellen gebildet.
	x	Bei einem Left-Join wird die Schnittmenge der verbundenen Tabellen gebildet.
	x	Bei MySQL kann man mittels der Check-Option Wertebereiche für Attribute mit der Tabellendefinition festlegen.
x		In MySQL kann keine referentielle Integrität erzwungen werden.
x		Im Gegensatz zu MySQL ist ein lesender Zugriff durch einen Before-Trigger auf die den Trigger auslösende Tabelle in Oracle nicht möglich.

´

Aufgabe 2)

Kunden	(Kundennummer, Name, Vorname, Geburtsdatum, Ort)
Warenkorb	(*Kundennummer*, *Artikelnummer*, Anzahl)
Artikel	(Artikelnummer, Artikelname, Autor, Preis, Ausgabe)
Lager	(Lagernummer, Standort, *ANummer*, Lagerbestand)

Die Verwaltung des Lagerbestandes der Buchhandlung erfolgt durch ein Programm „Lagerverwaltung“, welches auf die Buchhandlungsdatenbank zugreift. Das Programm „Lagerverwaltung“ erwartet Daten in der Form

(Artikelnummer, Artikelname, Lagernummer, Standort, Lagerbestand)

a) Erläutern Sie anhand dieses Beispiels die Begriffe externe und konzeptionelle Ebene der 3-Schichten Architektur eines DBS.

- **Externe Ebene:**

Das externe Modell beschreibt die einzelnen Sichten der Benutzer. Jeder Benutzer soll nur die für ihn relevanten Daten sehen. Im Beispiel: Die durch das Programm erwartete Datenstruktur.

- **Konzeptionelle Ebene:**

Das konzeptionelle Modell beschreibt die Gesamtheit aller Daten, die innerhalb der Datenbank verwaltet werden. Im Beispiel: das Buchhandlungsschema.

b) Geben Sie einen SQL-Befehl zur Definition einer Sicht an, mit der die Anforderungen des Programms „Lagerverwaltung“ an das Datenformat erfüllt werden können.

```
Create View LagerverwaltungView

AS SELECT Artikelnummer, Artikelname, Lagernummer, Standort, Lagerbestand

FROM Artikel a join Lager l ON a.Artikelnummer=l.ANummer
```

c) Erläutern Sie anhand dieses Beispiels den Begriff der logische Datenunabhängigkeit.

Änderungen am konzeptionellen Modell (auf der konzeptionellen Ebene) können durchgeführt werden, ohne dass das externe Modell (hier das View, die Benutzersicht) geändert werden muss und somit externe Anwendungen geändert werden müssen.
Im Beispiel: Kunde wird erweitert um ein Attribut „Anrede“ ohne, dass das Programm „Lagerverwaltung“ neu kompiliert werden muss.

Aufgabe 3)

a) Benennen und erläutern Sie bitte kurz die vier Eigenschaften einer Transaktion jeweils unter Angabe eines Beispiels.

- **Atomicity(=Eine Transaktion ist eine inhaltlich zusammenhängende Menge von Datenbankoperationen, die ganz oder gar nicht ausgeführt werden.)**

  Beispiel: Überweisung von 100€

- **Consistency(=Die Daten in der Datenbank müssen valide sein.)**

  Beispiel: Das Geburtsdatum einer Person muss in der Vergangenheit liegen. Das Datum 01.01.3020 dürfte also nicht als Geburtsdatum verwendet werden, sonst wäre die Datenbank nicht mehr konsistent.

- **Isolation(=Simulation des Ein-Benutzer-Betriebs.)**

  Wahrung der Korrektheit von Daten bei gleichzeitigem ändernden Zugriff mehrerer Benutzer auf eine Dateneinheit.

  Beispiel:

  Zwei Personen wollen „ parallel“ die selbe Relation manipulieren. Damit die beiden Personen sich nicht gegenseitig stören wird ein Ein-Benutzer-Betrieb simuliert.

- **Durability(=stellt sicher, dass eine Transaktion die commited ist „gesichert“ ist (auch wenn ein Systemausfall vorliegt).)**

  Beispiel: Die Transaktion „transfer“ wird ausgeführt um 200€ von Konto A auf Konto B zu überweisen. Die Transaktion war erfolgreich, allerdings sind die Änderungen noch im Cache während das System ausfällt. Durability bedeutet hier, dass die Änderungen der Daten trotzdem gelten.

b) Erläutern Sie kurz die folgenden Codd’schen Regeln:

1. Datenintegration

Einheitliche Verwaltung aller von Anwendungen benötigten Daten durch Tabellen (relationales Modell)

2. Physische Datenunabhängigkeit

Unabhängigkeit der Anwendungsprogramme von einer Änderung der Speicherorganisationform. Nur das DBMS greift physikalisch auf die Daten zu.

3. Datenschutz

Rechte der Nutzer zur Datensicht und Datenmanipulation können verwaltet werden.

Aufgabe 4)

Buchhandlung-Schema:

Kunden	(Kundennummer, Name, Vorname, Geburtsdatum, Ort)
Warenkorb	(*Kundennummer*, *Artikelnummer*, Anzahl)
Artikel	(Artikelnummer, Artikelname, Autor, Preis, Ausgabe)
Lager	(Lagernummer, Standort, *ANummer*, Lagerbestand)

a) Es sollen alle Geschäftskunden aus Dortmund in der Form Kundennummer und Name gelistet werden. Formulieren Sie die dafür notwendigen Befehle in der relationalen Algebra **UND** in SQL.
Anm.: Geschäftskunden haben kein Geburtsdatum.

π Kundennummer, Vorname, Name(σ Geburtsdatum= NULL ∧ Ort = 'Dortmund' Kunde)

```
SELECT Kundennummer, Name FROM Kunde
WHERE Geburtsdatum IS NULL AND Ort='Dortmund'
```

b) Es soll die Anzahl der Artikel pro Ausgabeart (gebunden, broschiert, …) ermittelt werden, denen kein Lagerplatz zugeordnet ist. Es soll eine Unterabfrage genutzt werden.

```
SELECT a.Ausgabe, Count(Artikelnummer) AS Anzahl

FROM Artikel a

WHERE Not Exists
        (Select ANummer
         FROM Lager l

          WHERE ANummer=a.Artikelnummer)

GROUP BY a.Ausgabe
```

Aufgabe 5)

Lieferservice-Datenbank

Artikel (idArtikel, preis, bezeichnung, bezeichnungs_ergaenzung, inhalt, inhaltsbeschreibung)

Bestellung (idBestellung, idKunde, idGetraenkemarkt, bestelldatum, bestellstatus, wunschtermin)

Bestellposition (idBestellposition, idBestellung, idArtikel, anzahl, reduktion)

Getraenkemarkt (idGetraenkemarkt, g_name, ustID, strasse, plz, stadt)

Getraenkemarkt_has_Lieferer (idGetraenkemarkt, idLieferer)

Kunde (idKunde, anrede, vorname, nachname, strasse, wohnort, plz, tel)

Lieferer (idLieferer, passwort, anrede, vorname, nachname, geburtsdatum, strasse, wohnort, plz, tel, beschreibung)

a) Geben Sie eine Liste der Artikel aus, zu denen mindestens eine Bestellung vorliegt. In der Liste soll neben der Id des Artikels die Anzahl der Bestellungen ausgegeben werden, in der der Artikel vorkommt. Die Spaltenüberschriften sollen „Artikel“ und „Anzahl“ lauten, die Liste soll nach der Anzahl so sortiert werden, dass höchste Zahl von Bestellungen oben steht. Es soll keine Unterabfrage genutzt werden.

```
SELECT count(*) AS Anzahl, idartikel AS Artikel
FROM Bestellung
JOIN Bestellposition USING (idbestellung)
GROUP BY idartikel
ORDER BY anzahl DESC
```

b) Erstellen Sie eine Liste aller Lieferer mit Vor- und Nachnamen. Sofern der Lieferer einem Getraenkemarkt zugeordnet ist, soll der Name dieses Getränkemarktes daneben angegeben werden. Die Liste soll nach den Nachnamen der Lieferer alphabetisch sortiert sein. Es soll ein OUTER-JOIN verwendet werden. Es soll keine Unterabfrage genutzt werden.

```
SELECT l.Nachname, l.Vorname, g.Name FROM Lieferer l
LEFT JOIN Getraenkemarkt_has_Lieferer USING (idLieferer)
LEFT JOIN Getraenkemarkt g USING ( idGetraenkemarkt)
ORDER BY l.nachname;
```

c)Listen Sie alle Kunden mit Vor- und Nachnamen, die bislang noch keine Bestellung aufgegeben haben. Geben Sie zusätzlich die Postleitzahl des Wohnortes aus und sortieren Sie die Liste zunächst nach der Postleitzahl und innerhalb der Postleitzahl nach den Nachnamen der Kunden. Es sollen nur Verbundoperationen genutzt werden.

```
SELECT vorname, nachname, plz
FROM kunde
LEFT OUTER JOIN bestellung USING (idkunde)
WHERE idbestellung IS NULL
ORDER BY plz, nachname
```

Aufgabe 6)

Buchhandlung-Schema:

Kunden	(Kundennummer, Name, Vorname, Geburtsdatum, Ort)
Warenkorb	(*Kundennummer, Artikelnummer*, Anzahl)
Artikel	(Artikelnummer, Artikelname, Autor, Preis, Ausgabe)
Lager	(Lagernummer, Standort, *ANummer*, Lagerbestand)

Erstellen Sie ein Oracle-Installationsskript für eine zusätzliche Tabelle Lieferung, die Informationen für die Lieferung eines Artikels an einen Kunden beinhaltet:

Lieferung (Lieferungsnr, *Kundennummer*, *Artikelnummer*, Anzahl, *Lagernummer*)

Realisieren Sie die nötigen Einschränkungen bezüglich Primär- und Fremdschlüsseln!

```
CREATE OR REPLACE TABLE `lieferung` (
  Lieferungsnr int(11) NOT NULL,
  Kundennummer int(11) NOT NULL,
  Artikelnummer int(11) NOT NULL,
  Anzahl int(11) NOT NULL,
  Lagernummer int(11) NOT NULL,
  PRIMARY KEY (Lieferungsnr),
  FOREIGN KEY (Kundennummer) REFERENCES Kunde (Kundennummer),
  FOREIGN KEY (Artikelnummer) REFERENCES Artikel (Artikelnummer),
  FOREIGN KEY (Lagernummer) REFERENCES Lager (Lagernummer)
);
```

Aufgabe 7)

Erstellen Sie eine gespeicherte Prozedur mit den Eingabeparametern Kundennummer, alteWohnort und neuerWohnort. Anhand der Kundennummer wird überprüft, ob der eingegebene alteWohnort mit dem Datenbestand übereinstimmt. Falls nicht, dann wird dies durch eine Fehlermeldung signalisiert. Ansonsten wird der Wohnort des Kunden geändert und auf den Wert des Parameters neuerWohnort gesetzt.
Verwenden Sie die Syntax von Oracle.

```
CREATE PROCEDURE Wohnortwechsel (
          Knr         IN     INT,
          alterOrt    IN     VARCHAR2,
          neuerOrt    IN     VARCHAR2)
IS
     Wohnort              VARCHAR2(200);
     falscherWohnort    EXCEPTION;
BEGIN
   SELECT  Ort  INTO Wohnort  FROM Kunden WHERE  Kundennummer = Knr;
    IF alterOrt = RTRIM(Wohnort,' ') THEN
       UPDATE Kunden SET Ort = neuerOrt  WHERE  Kundennummer=Knr;
    ELSE
       RAISE falscherWohnort;
    END IF;
EXCEPTION
     WHEN falscherWohnort
     THEN raise_application_error (-20500,'Faktueller Wohnort fehlerh.');
END;
```

Aufgabe 8)

Lieferservice-Datenbank

Artikel (idArtikel, preis, bezeichnung, bezeichnungs_ergaenzung, inhalt, inhaltsbeschreibung)

Bestellung (idBestellung, idKunde, idGetraenkemarkt, bestelldatum, bestellstatus, wunschtermin)

Bestellposition (idBestellposition, idBestellung, idArtikel, anzahl, reduktion)

Getraenkemarkt (idGetraenkemarkt, g_name, ustID, strasse, plz, stadt)

Getraenkemarkt_has_Lieferer (idGetraenkemarkt, idLieferer)

Kunde (idKunde, anrede, vorname, nachname, strasse, wohnort, plz, tel)

Lieferer (idLieferer, passwort, anrede, vorname, nachname, geburtsdatum, strasse, wohnort, plz, tel, beschreibung)

a) Erstellen Sie eine Bestellungsübersicht in der die Namen(Vorname und Nachname), die Bestellungsnummer sowie die Gesamtzahl der pro Bestellung bestellten Artikel.
Beispiel: Die Kundin Freya Wille hat 3x Cola Light und 7x Malzbier bestellt, die Ausgabe soll wie folgt aussehen:

	IDBESTELLUNG	VORNAME	NACHNAME	GESAMTANZAHL
1	59	Freya	Wille	10

```
CREATE VIEW Bestellungen_Gesamtsicht as

SELECT idbestellung, vorname, nachname, SUM(anzahl) as Gesamtanzahl FROM Bestellung NATURAL JOIN bestellposition NATURAL JOIN artikel NATURAL JOIN kunde

group by idbestellung, vorname, nachname;
```

b) Es soll nun möglich gemacht werden auf dieser Sicht Bestellungen anhand der Bestellungsnummer zu löschen. Erstellen Sie einen Trigger der das ermöglicht.

```
CREATE OR REPLACE TRIGGER delete_Bestellung

INSTEAD OF DELETE ON Bestellungen_Gesamtsicht

BEGIN

DELETE FROM BESTELLPOSITION WHERE idbestellung = :old.idbestellung;

DELETE FROM bestellung WHERE idbestellung = :old.idbestellung;

END delete_ Bestellung;
```