Wichtige allgemeine Hinweise zum Laborpraktikum Datenbank- und Informationssysteme 1

- Die Einteilung in Teams (aka „Tandems“) und die Verteilung der „Tandems“ auf die Laborpraktikums-Termine entnehmen Sie bitte den entsprechenden Dokumenten.
- Halten Sie sich bitte genau an die Aufgabenbeschreibungen!
- Insbesondere sind alle kommunizierten Termine "Hard Deadlines", d.h. jegliche Verzögerung bei den Abgaben oder Präsentationen zieht den sofortigen Ausschluss aus dem Laborpraktikum nach sich!
- Sie können natürlich jederzeit Fragen stellen. Aber bitte: Lesen Sie zuerst einmal die Aufgabenbeschreibung genau durch – bevor sie Fragen stellen, deren Antworten bereits eindeutig aus der Aufgabenbeschreibung hervorgehen.
- Das im Laborpraktikum verwendete Datenbanksystem ist Oracle Database 19c.
- Die Vergabe von Zugangsberechtigungen erfolgt rechtzeitig im Laborpraktikum.
- Die Entwicklung Ihrer Datenbank kann unter Zuhilfenahme beliebiger Softwarewerkzeuge erfolgen. Im Labor stehen Ihnen *unter anderem* SQL Developer von Oracle, SQLcl von Oracle, Altova Database Spy, Eclipse, sowie DIA, yEd, Microsoft Visio und Microsoft Office zur Verfügung.
- **Allerdings: Ihre SQL-Skripte und -Anfragen müssen unter SQL Developer von Oracle lauffähig sein; die persönliche Vorstellung der Implementierung muss ebenfalls unter SQL Developer erfolgen.**
- Schauen Sie bitte zuerst in die zur Verfügung stehenden Unterlagen zur Lehrveranstaltung sowie die Dokumentationen zu Oracle Database 19c, SQL Developer, SQLcl und anderen verwendeten Tools, bevor Sie Fragen stellen ⇒ Motto: "RTFM"!
- Die Online-Dokumentation zu Oracle Database 19c finden Sie u. a. (im HTWG-Intranet bzw. mit HTWG-VPN) unter http://oracle19c.in.htwg-konstanz.de/.
- Weitere Informationen und Dokumentationen finden Sie auf der E-Learning Plattform Moodle im Kurs Datenbank- und Informationssysteme 1.

Viel Erfolg!

Aufgabenstellungen zum 1. Praktikumsabschnitt

Eine Agentur, die Ferienwohnungen vermittelt, möchte das Angebot von Wohnungen im Internet zugänglich machen und Buchungen durch Interessenten online ermöglichen. Erforderlich hierzu ist eine **Datenbank**, in welcher die notwendigen Daten abgelegt und verwaltet werden können.

Basis für den Datenbank-Entwurf ist folgendes **Ergebnis der Anforderungsanalyse**:

- Ferienwohnungen befinden sich in einem bestimmten Ort und Land.
- Die Information zu einer Ferienwohnung umfasst eine textuelle Beschreibung der Wohnung, die Adresse (bestehend aus Straße/Hausnummer, Ort, PLZ, Name und ISO-Code des Landes), die Anzahl der Zimmer, die Größe in qm und den Preis pro Tag.
- Manche Ferienwohnungen haben Zusatzausstattungen, z.B. eine Sauna und/oder ein Schwimmbad. Es sollen aber auch beliebige viele andere Zusatzausstattungen verwaltet werden können. Weiterhin sollen bis zu vier Bilder (inkl. Bildbeschreibung und Dateipfad) von einer Ferienwohnung verwaltet werden können.
- Touristenattraktionen (beispielsweise Freizeitparks, Badestrände, Sehenswürdigkeiten) werden für die jeweiligen Länder inklusive ihrer Beschreibung und Adresse abgelegt.
- Für jede Ferienwohnung soll die Entfernung (vom Ort) der Ferienwohnung zu (dem Ort) einer Touristenattraktionen ermittelt werden können. Hierzu soll im System die Entfernung zwischen Orten verwaltet werden können. Dabei soll gelten:
  - Die Entfernung zweier Orte soll maximal einmal in der Datenbank gespeichert werden.
  - Die Entfernung eines Ortes „von sich selbst“ soll nicht in der Datenbank gespeichert werden, sondern ist mit 0 km anzunehmen.
  - Ist die Entfernung zweier Orte nicht in der Datenbank gespeichert, so ist diese als „unbekannt“ zu interpretieren.
- Bevor Kunden Ferienwohnungen reservieren bzw. buchen können, müssen diese mit ihrem Namen und Vornamen und Geburtsdatum, sowie jeweils genau einer Adresse, einer Telefonnummer, einer Email-Adresse und einer Bankverbindung registriert sein. Kunden sollen eine eindeutige interne Kundennummer zugeordnet bekommen.
- Im System verwaltete Adressdatensätze sollen *entweder* genau zu einem Kunden *oder* zu genau einer Ferienwohnung *oder* zu genau einer Touristenattraktion *oder* genau zu einem Flughafen in Beziehung stehen können.
- Ein Bankverbindungsdatensatz besteht aus BLZ, Kontonummer, IBAN und BIC.
- Im System verwaltete Bankverbindungsdatensätze sollen zu genau einem Kunden gehören.
- Kunden können Ferienwohnungen belegen. Bei Belegungen soll durch ein Status-Flag zwischen einer (unverbindlichen) Reservierung und einer (verbindlichen) Buchung unterschieden werden. Bei einer Belegung müssen die Kunden eine Ferienwohnung auswählen und angeben, in welchem Zeitraum sie die Wohnung reservieren bzw. buchen wollen.
- Bei einer Reservierung bzw. Buchung wird das Reservierungs- bzw. Buchungsdatum festgehalten.
- Eine Reservierung kann in eine verbindliche Buchung gewandelt werden, aber nicht umgekehrt.
- Für jede Buchung erhalten Kunden im Laufe einer Woche eine Rechnung mit Betrag, Buchungsnummer, Rechnungsnummer und Rechnungsdatum.
- Nachdem die Rechnung bezahlt ist, ist die Buchung abgeschlossen und das Datum des Zahlungseingangs wird abgelegt.
- Für Ferienwohnungen sollen alle Fluggesellschaften ermittelt werden können, mit denen man die Ferienwohnung von einem Flughafen aus erreichen kann:
  - Es soll zu allen Orten in der Datenbank jeweils ein zugeordneter Flughafen verwaltet werden.
  - Ein Flughafen selbst hat eine Adresse zugeordnet.
  - Es soll in der Datenbank gespeichert werden, welche Flugstrecken von welchen Fluggesellschaften bedient werden, d.h. mit welchen Fluggesellschaften man von einem bestimmten (Start-)Flughafen einen anderen (Ziel-)Flughafen erreichen kann.
- Weiterhin soll ermittelt werden können, welche der Fluggesellschaften die beste Servicequalität bietet. Hierzu soll für jede Fluggesellschaft eine numerische Servicequalitätskennzahl (1-10) gespeichert werden.

Aufgabe A) Konzeptueller Datenbank-Entwurf

Entwickeln Sie aus dem Ergebnis der Anforderungsanalyse ein **konzeptuelles Datenbankschema** unter Verwendung des *einfachen* **Entity-Relationship-Modells**; es sind also *keine Erweiterungen* wie mehrwertige oder strukturierte Attribute, schwache Entitäts-Typen, Generalisierung/Spezialisierung etc. zulässig!

Hinweis: Ihr ER-Schema muss korrekt und vollständig (bzgl. der oben beschriebenen Anforderungen) und konsistent und redundanzfrei sein. Machen Sie Ihr ER-Schema aber nicht komplexer als in der Aufgabenstellung gefordert.

Persönliche Präsentationen (Aufgabe A): Sie müssen Ihre Lösung dem Dozenten persönlich in den Ihnen *zugewiesenen Laborpraktikums-Terminen* vorstellen. Hierbei besteht prinzipiell *Anwesenheitspflicht für alle Tandem-Mitglieder*.

Abgabe (Aufgabe A): **Vollständige Dokumentation des Entity-Relationship-Schemas** *als PDF mit Angabe von Tandemnummer, Namen und Matrikelnummern*:

- *Graphische Darstellung* des ER-Schemas als ER-Diagramm unter Angabe von Funktionalitäten *und* Kardinalitäten sowie Primärschlüsseln *und* Alternativschlüsseln.
- Zusätzliche wichtige *Daten-Integritätsbedingungen*, die nicht im ER-Diagramm darstellbar sind, sich aber aus der Anforderungsanalyse etc. ergeben: diese *müssen* als Text angegeben werden.
- Weitere notwendige *Erläuterungen*, um das ER-Schema richtig verstehen zu können.

Wichtige Hinweise:

- Nur die in der Vorlesung verwendete Notation ist erlaubt!
- Unterscheiden Sie zwischen Daten-Integritätsbedingungen und Erläuterungen.
- Bitte dokumentieren Sie das Ergebnis digital, damit Sie einfacher Korrekturen vornehmen können.
- Bei Verwendung von DIA, yEd oder Visio als Zeichentool benutzen Sie bitte die entsprechenden ER-Diagramm-Shapes.

Präsentations- und Abgabetermine und -modalitäten werden rechtzeitig bekanntgegeben!

Aufgabe B) Logischer Datenbank-Entwurf

Transformieren Sie Ihr ER-Schema in das **relationale Datenmodell**. Erstellen Sie ein möglichst *redundanzfreies, normalisiertes* **relationales Datenbankschema.** Gehen Sie dabei sehr sorgfältig vor und fassen Sie – soweit möglich und sinnvoll – Relationen zusammen.

Persönliche Präsentationen (Aufgabe B): Sie müssen Ihre Lösung dem Dozenten persönlich in den Ihnen *zugewiesenen Laborpraktikums-Terminen* vorstellen. Hierbei besteht prinzipiell *Anwesenheitspflicht für alle Tandem-Mitglieder*.

Abgabe (Aufgabe B): **Vollständige Dokumentation des relationalen Datenbankschemas** *als PDF mit Angabe von Tandemnummer, Namen und Matrikelnummern*:

- *Überarbeitetes* ER-Diagramm inkl. Integritätsbedingungen und Erläuterungen (siehe Aufgabe A).
- *Formale Beschreibung* des relationalen Datenbankschemas inklusive Primärschlüsseln, Alternativ-Schlüsseln, Fremdschlüsselbeziehungen und weiterer Integritätsbedingungen.
- Zusätzliche wichtige *Daten-Integritätsbedingungen*, die nicht mit Mitteln des relationalen Datenmodells formulierbar sind, sich aber aus der Anforderungsanalyse und dem konzeptuellen Datenbankschema etc. ergeben: diese *müssen* als Text angegeben werden.
- Weitere notwendige *Erläuterungen*, um das relationales Datenbankschema richtig verstehen zu können.

Wichtige Hinweise:

- Nur die in der Vorlesung verwendet Notation ist erlaubt!
- Unterscheiden Sie zwischen Daten-Integritätsbedingungen und Erläuterungen.
- Unterscheiden Sie zwischen intra- und inter-relationalen Integritätsbedingungen.
- Bitte dokumentieren Sie die Endfassung ("Verfeinerung") digital und geben nur diese ab!
- Ein Beispiel einer formalen Beschreibung eines relationalen Datenbankschemas finden Sie auf Moodle.

Präsentations- und Abgabetermine und -modalitäten werden rechtzeitig bekanntgegeben!

Aufgabe C) Datenbank-Implementierung

1. Erzeugen Sie eine Oracle-Datenbank für Ihr relationales Datenbankschema aus Aufgabe B. Beachten Sie bei der Implementierung *alle Integritätsbedingungen aus Aufgaben A und B* (Funktionalitäten und Kardinalitäten des ER-Schemas sowie Primärschlüssel, Alternativ-Schlüssel, Fremdschlüsselbeziehungen etc. und die von Ihnen dokumentierten zusätzlichen Daten-Integritätsbedingungen).

Erstellen Sie hierzu eine *Skript-Datei* `create_db_schema.sql` zum **Erzeugen der Tabellen etc. der Datenbank** und eine *Skript-Datei* `drop_db_schema.sql` zum **Entfernen aller Tabellen etc. der Datenbank**.

Wichtige Hinweise:

- Benennen Sie alle Constraints sinnhaft. Ein Ausnahme bilden die `NOT NULL` Constraints: diese können, müssen aber benannt werden.
- Verwenden Sie, wo möglich, bevorzugt Spalten-Integritätsbedingungen (anstatt Tabellen-Integritätsbedingungen).
- Aufgrund zirkulärer Fremdschlüssel-Abhängigkeiten muss einer der Fremdschlüssel beim Anlegen / Entfernen der Tabellen durch ein ALTER TABLE Statement an passender Stelle des jeweiligen Skriptes hinzugefügt / entfernt werden.
- Daten-Integritätsbedingungen, die Sie nicht mit Hilfe von Oracle 19c DDL-Statements ausdrücken können, müssen als Kommentare im Erzeugungs-Skript angegeben werden. Alternativ können Sie das `COMMENT ON` SQL Statement von Oracle Database 19c nutzen.
- Erläuterungen müssen als Kommentare im Erzeugungs-Skript angegeben werden. Alternativ können Sie das `COMMENT ON` SQL Statement von Oracle Database 19c nutzen.

2. Füllen Sie die Datenbank mit Beispieldaten (diese werden noch bekannt gegeben). Erstellen Sie hierzu eine *Skript-Datei* `insert_initial_data.sql` für eine **Transaktion** zum initialen **Einfügen der Datensätze** und eine *Skript-Datei* `delete_all_data.sql` für eine **Transaktion** zum **Löschen aller Datensätze aus der Datenbank**.

Wichtige Hinweise:

- Die zu verwendenden Beispieldaten werden Ihnen als Microsoft Excel-Datei zur Verfügung gestellt werden.
- Die darin enthaltenen *Beispieldaten sind nicht normalisiert und müssen* von Ihnen an Ihr Datenbankschema *entsprechend angepasst und ggfls. noch ergänzt werden* (z.B. ISO-Codes, Flugverbindungen, Entfernungen zwischen Orten).
- *Passen Sie auf keinen Fall Ihr Datenbankschema strukturell an die Beispieldaten an!* Ausgenommen hiervon sind notwendige Änderungen von Datentypen, die sich aus den Beispieldaten ergeben.
- Aufgrund zirkulärer Fremdschlüssel-Abhängigkeiten muss einer der Fremdschlüssel beim Einfügen bzw. Löschen der Datensätze durch entsprechende `ALTER TABLE` Statements *temporär* deaktiviert (d.h. an passenden Stellen im Einfüge- bzw. im Lösch-Skript abgeschaltet und später wieder angeschaltet) werden.
- Transaktionen, die den Datenbankzustand betreffen (DML-Transaktionen), müssen immer mit `COMMIT` abgeschlossen werden, damit die Änderungen am Datenbankzustand dauerhaft persistiert sind.
- Fügen Sie, wo notwendig, zu Beginn der Skripte entsprechende `ALTER SESSION` Statements zum Setzen des Datumsformats ein.

3. Entwerfen sie folgende SQL-Anfragen an Ihre Datenbank. Die Datenbankanfragen sollen dabei alle *notwendigen und sinnvollen Informationen im Ergebnis ausgeben* (nicht nur Nummern). Beachten Sie auch die weiteren Hinweise auf Seite 5.

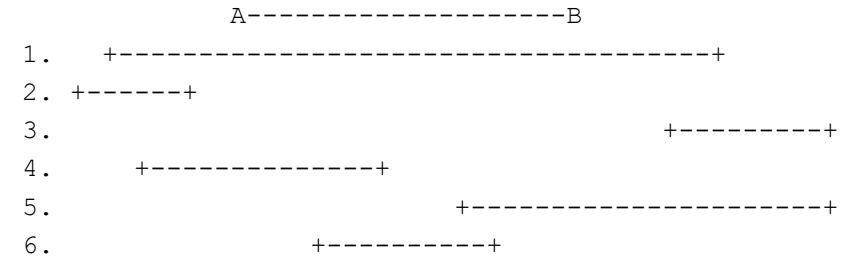
1. Welche *Belegungen* gibt es für eine *gegebene* Ferienwohnung?
2. Welche Personen haben eine *gegebene* Ferienwohnung *reserviert*? Personen, die diese Ferienwohnung mehrmals reserviert haben, sollen dabei nur einmal ausgegeben werden.
3. Wie viele *Buchungen* hat eine Person mit einer *gegebenen* Kundenummer getätigt?
4. Wie viele *Buchungen* haben Personen mit einem *gegebenen* Nachnamen getätigt? Dabei sollen nur Personen ausgegeben werden, die mindestens eine Buchung getätigt haben. Beachten Sie weiterhin, dass es mehrere Personen mit dem gleichem Nachnamen geben kann.
5. Welche Ferienwohnungen in Frankreich sind höchstens 100 km von (dem Ort) der Touristenattraktion Disneyland entfernt?

6. Welche Ferienwohnungen mit Schwimmbad in Frankreich haben mindestens eine Reservierung bzw. mindestens eine Buchung bzw. mindestens eine Belegung? (insgesamt 3 Anfragen)
7. Welche Ferienwohnungen mit Schwimmbad in Frankreich haben keine Reservierung bzw. keine Buchung bzw. keine Belegung? (insgesamt 3 Anfragen)
8. Welche Ferienwohnungen mit Schwimmbad sind in der Türkei in der Zeit vom 01.05.2016 - 21.05.2016 bereits belegt (d.h. gebucht oder reserviert)?
9. Welche Ferienwohnungen mit Schwimmbad sind in der Türkei in der Zeit vom 01.05.2016 - 21.05.2016 noch frei (d.h. nicht gebucht und nicht reserviert)?
10. Wie viele Belegungen von Ferienwohnungen gibt es für die einzelnen Länder in der Datenbank? (bitte auch Länder ausgeben, die keine Belegungen haben bzw. die keine Orte / Adressen zugeordnet haben)
11. Wie viele Reservierungen und Buchungen gibt es für die einzelnen Ferienwohnungen? (1 Anfrage; bitte auch Ferienwohnungen ausgeben, die keine Reservierungen / Buchungen haben)
12. Welche Ferienwohnungen in Frankreich haben mehr Belegungen als es Ferienwohnungen in Deutschland gibt?
13. Welches sind die "geeigneten" Fluggesellschaften für eine *gegebene* Ferienwohnung, mit denen man die Ferienwohnung *vom Ausland* aus direkt erreichen kann? ("geeignet" = Fluggesellschaften, die den Flughafen anfliegen, der dem Ort der Ferienwohnung zugeordnet ist)
14. Welche Fluggesellschaft davon (siehe Anfrage 13) hat die beste Service-Qualität?

Speichern und *dokumentieren* Sie jede einzelne *SQL-Anfrage* in einer eigenen *Skript-Datei*. Die Skript-Dateien müssen entsprechend folgendem Schema benannt sein:
`select_<number>.sql` bzw. `select_<number>_<subnumber>.sql`.

Wichtige Hinweise:

- Die `SELECT`-Statements 1, 2, 3, 4, 13 und 14 sollen mit entsprechenden Eingabeparametern *parametrisiert* sein (sog. Substitutionsvariablen in SQL Devleoper: `&<variableName>`), d.h. es soll eine interaktive Abfrage der Ferienwohnungsnummer bzw. der Kundennummer bzw. des Nachnamens erfolgen.
- Falls nötig, fügen Sie zu Beginn in die Skript-Dateien entsprechende `ALTER SESSION` Statements zum Setzen des Datumsformats ein.
- Für die Anfragen sind nur die genannten Angaben als "Eingaben" erlaubt. Beispiel: Wenn Ferienwohnungen aus einem bestimmten Land gefragt sind, dann ist nur der Name des Landes in der SQL-Anfrage als "Eingabe" erlaubt und nicht der ISO-Code des Landes direkt.
- Zur Beantwortung von Anfragen, die sich auf die Zeiträume einer Reservierung bzw. Buchungen bzw. Belegung einer Ferienwohnung beziehen, müssen Sie in "Intervall-Logik" denken – wann ist eine Ferienwohnung belegt bzw. frei, wenn eine SQL-Anfrage einen Zeitraum A–B abfragt?



- Zum umfassenden Testen Ihrer Anfragen können Sie weitere Testfälle erzeugen, in dem Sie den Datenbestand (temporär) ändern durch entsprechende `INSERT, UPDATE und DELETE` Statements. Nutzen Sie hierzu auch das Transaktionskonzept (`ROLLBACK`).

4. Legen Sie für Ihr Datenbankschema folgende SQL-Sichten (Views) an:

1. Sichten "Buchung" und "Reservierung" (mit allen Informationen von "Belegungen" *außer* dem Belegungsstatus) zur Nachbildung der Generalisierung/Spezialisierung.
2. Sicht "Familienwohnungen": beinhaltet alle Ferienwohnungen mit mehr als 100 qm Wohnfläche. Die Sicht soll genau die gleichen die Informationen wie die Tabelle für Ferienwohnungen bieten.

   Dabei sollen Änderungen der Wohnfläche *über diese Sicht* nur erlaubt sein, falls die geänderten Datensätze in der Sicht bleiben. Ebenso soll das Einfügen von Ferienwohnungen in die Datenbank *über diese Sicht* nur erlaubt sein, falls die Wohnfläche größer als 100 qm ist.

3. Sicht "UebersichtKunden": enthält die wichtigsten Angaben aller Kunden (inkl. Adresse und Bankverbindung), deren Buchungen und Reservierungen (Status, von/bis, Reservierungs- bzw. Buchungsdatum) und die wichtigsten Angaben über die zugehörigen Ferienwohnungen (Nr., Beschreibung). Des weiteren soll bei Buchungen angegeben werden, ob bereits eine Rechnung erstellt wurde.
4. Sicht "Zahlungstatus": enthält alle Buchungen inkl. der wichtigsten Angaben bzgl. Ferienwohnung (Nr., Beschreibung) und Kunde (Nr., Name), sowie die Angaben über die zugehörigen Rechnungen (Rechnungsnummer, Datum, Rechnungsbetrag) und den Zahlungsstatus ("bezahlt"/"offen") mit Datum des Zahlungseingangs (falls bezahlt).
5. Sicht "MidAgeKunden": beinhaltet alle Kunden im Alter von 30 bis 40 Jahren. Die Sicht soll alle Informationen wie die Tabelle für Kunden bieten und zusätzlich das Alter des Kunden beinhalten.

Ergänzen Sie Ihre *Create- und Drop-Skripte* `create_db_schema.sql`, `drop_db_schema.sql` um die entsprechenden SQL-Statements zum Erzeugen und Entfernen der oben geforderten Views.

Testen Sie Ihre Views (speziell die Sichten "Familienwohnungen" und "MidAgeKunden") durch sinnhafte Ergänzungen bzw. Änderungen am Datenbestand.

Persönliche Präsentationen der Implementierung (Aufgabe C)

Sie müssen Ihre Implementierung dem Dozenten persönlich in den Ihnen *zugewiesenen Laborpraktikums-Terminen* vorstellen. Hierbei besteht prinzipiell *Anwesenheitspflicht für alle Tandem-Mitglieder*.

Präsentationstermine und -modalitäten werden rechtzeitig bekanntgegeben!

***Wichtiger Hinweis*:** Die Implementierung muss dabei *vor Beginn des Laborpraktikums am Abnahmetag vollständig fertig gestellt* sein! Das bedeutet insbesondere für Teile 1 und 2, dass der Zyklus "Tabellen erzeugen - Datensätze einfügen - Datensätze löschen - Tabellen löschen" fehlerfrei abläuft und nach dem Einfügen der Datensätze der Inhalt Ihrer Datenbank mit den vorgegeben Excel-Daten inhaltlich übereinstimmend ist.

Abgabe der Implementierung (Aufgabe C)

Neben den Präsentationen der Lösungen müssen Ihre SQL-Skripte auch noch abgegeben werden.

Abgabetermine und -modalitäten werden rechtzeitig bekanntgegeben!

Optionale Aufgabenstellungen zum 1. Praktikumsabschnitt

Aufgabe Opt-I-1 (Konzeptueller Datenbankentwurf mit dem erweiterten ER-Modell)

Führen Sie den konzeptuellen Datenbankentwurf aus Aufgabe A mit Hilfe der erweiterten ER-Modells durch. Verwenden Sie dabei so viele Erweiterungen des ER-Modells wie möglich (uns sinnvoll).

Aufgabe Opt-I-2 (Relationaler Datenbankentwurf auf Basis d. erweiterten ER-Modells)

Führen Sie den relationalen Datenbankentwurf auf Basis Ihres ER-Schema (erweitertes ER-Modell) aus Aufgabe Opt-I-2 durch.

Vergleichen Sie Ihr Ergebnis mit Ihrem Ergebnis für Aufgabe B.

Aufgabe Opt-I-3 (Konzeptueller Datenbankentwurf mit UML-Klassendiagrammen)

Führen Sie den konzeptuellen Datenbankentwurf aus Aufgabe A mit Hilfe von UML-Klassendiagrammen durch.

Aufgabe Opt-I-4 (Relationaler Datenbankentwurf auf Basis UML-Klassendiagrammen)

Führen Sie den relationalen Datenbankentwurf auf Basis Ihres UML-Modells (aus Aufgabe Opt-I-4) durch.

Vergleichen Sie Ihr Ergebnis mit Ihrem Ergebnis für Aufgabe B und ihrem Ergebnis für Aufgabe Opt-I-2.

Aufgabe Opt-I-5 (Relationale Entwurfstheorie)

Überprüfen Sie Ihr relationales Datenbankschema aus Aufgabe B auf Normalformen.

1. Definieren Sie sich hierzu die funktionalen Abhängigkeiten in den einzelnen Relationen(schemata) (abgeleitet aus der Anforderungsanalyse).
2. Wenden Sie die Definitionen der Normalformen auf die einzelnen Relationen(schemata) an, um zu zeigen, in welcher Normalform sich die einzelnen Relationen(schemata) befinden.

Normalisieren Sie Ihr relationales Datenbankschema aus Aufgabe B, so dass sich alle beinhalteten Relationen(schemata) mindestens in der

1. dritten Normalform (3NF) bzw.
2. Boyce-Codd-Normal-Form (BCNF)

befinden (falls noch nicht in Ihrer Lösung für Aufgabe B erfüllt).

Aufgabe Opt-I-6 (Relationale Algebra)

Formulieren Sie sinnhafte Anfragen in der relationalen Algebra an Ihre relationale Datenbank aus Aufgabe B, z.B. für folgende SQL-Anfrage aus Aufgabe C.3:

- Anfrage 1: Welche *Belegungen* (d.h. Buchungen oder Reservierungen) gibt es für eine *gegebene* Ferienwohnung?
- Anfrage 2: Welche Personen haben eine *gegebene* Ferienwohnung *reserviert*? Personen, die diese Ferienwohnung mehrmals reserviert haben, sollen dabei nur einmal ausgegeben werden.
- Anfrage 5: Welche Ferienwohnungen in Frankreich sind höchstens 100 km von (dem Ort) der Touristenattraktion Disneyland entfernt?
- Anfrage 6: Welche Ferienwohnungen mit Schwimmbad in Frankreich haben mindestens eine Reservierung bzw. mindestens eine Buchung bzw. mindestens eine Belegung? (insgesamt 3 Anfragen)
- Anfrage 7: Welche Ferienwohnungen mit Schwimmbad in Frankreich haben keine Reservierung bzw. keine Buchung bzw. keine Belegung? (insgesamt 3 Anfragen)

Versuchen Sie die entsprechenden Anfragen in der relationalen Algebra darzustellen mittels

1. (nur) Basisoperatoren und dann
2. optimiert mittels erweiterter Operatoren ("algebraische Anfrageoptimierung")

Geben Sie diese Algebraausdrücke wieder in

1. Inline-Darstellung
2. Operatorbaum-Darstellung

Bemerkung: Nicht alle Anfragen aus C.3 lassen sich mit "unserer" einfachen relationalen Algebra darstellen, da wir hier z.B. keine Aggregation, keine Gruppierung, keine Multimengen / Duplikatseliminierung, keine Sortierung und keine geschachtelte Anfragen "unterstützen".

"Unsere" relationale Algebra kann einfach u.a. um die genannten Konzepte/Konstrukte erweitert werden, doch dies ist kein Inhalt der Lehrveranstaltung DBIS-1.