Informationssysteme - Grundlagen

Aufgabenblätter - 01.09.2020

Dipl.-Ing. Msc. Paul Panhofer BSc.[1*]

1 ZID, TU Wien, Taubstummengasse 11, 1040, Wien, Austria

Abstract:

MSC: paul.panhofer@gmail.com

Keywords:

.

.

*E-mail: paul.panhofer@tuwien.ac.at

1.) Aufgabenblatt - Gruppe A(4.Punkte)

Kompetenzen

SELECT Klauseln: Formulieren einfacher SQL Abfragen:

- **select Klausel:** Selektion von Spaltennamen, Verarbeiten von Spaltenwerten, * Operator, Spaltenalias, DISTINCT Operator.
- **where Klausel:** Formulierung logischer Terme, Logische Operatoren and, or, not, Logische Operatoren like, in, is, between Dreiwertige Logik.
- **Order By Klausel:** Sortieren von Datensätzen.
- **Case Klausel:** Bedingte Verarbeitung, Kontextbezogene Verarbeitung.

Tabellen: PRODUCT_INFORMATION

User: oe

1.Beispiel) Select Klauseln - 1.Punkt

Geben Sie alle Produkte aus, deren Listenpreis hoechstens um 100 Dollar hoeher ist als der Mindestpreis.

- Geben Sie dabei folgende Spalten aus: PRODUCT_ID, LIST_PRICE, MIN_PRICE, DELTA.

  @DELTA: Die DELTA Spalte zeigt die Differenz zwischen Listenpreis und Mindestpreis an.

- Sortieren Sie das Ergebnis der Abfrage entsprechend der Differenz des Listenpreises zum Mindestpreis!

❑

2.Beispiel) Select Klauseln - 1.Punkt

Geben Sie alle Produkte aus die in der Produktbeschreibung das Wort memory beinhalten.

- Der Listenpreis des Produkts soll dabei zwischen 100 und 400 Dollar liegen.
- Geben Sie die folgenden Spalten aus: PRODUCT_ID, LIST_PRICE, PRODUCT_DESCRIPTION

❑

3.Beispiel) Select Klauseln - 1.Punkt

Geben Sie fuer jedes Produkt die folgenden Spalten aus: PRODUCT_ID, LIST_PRICE, MIN_PRICE, INCREASE.

- Zeigen Sie nur Produkte an, fuer die sowohl ein Mindestpreis als auch ein Listenpreis gespeichert wird.
- @INCREASE: Die INCREASE Spalte beschreibt um wieviel Prozent der Listenpreis hoeher ist als der Mindestpreis.
- Sortieren Sie das Ergebnis absteigend nach den Werten der INCREASE Spalte

❑

4.Beispiel) Select Klauseln - 1.Punkt

Geben Sie fuer jedes Produkt die folgenden Spalten aus: PRODUCT_ID LIST_PRICE, CATEGORY.

- @CATEGORY: Die CATEGORY Spalte beinhaltet eine verbale Beschreibung der Kategorie des Produkts.

CATEGORY_ID	CATEGORY
11	monitor
12	printer
13	harddisk
14	memory component
15+	hardware

- Sortieren Sie das Ergebnis nach den Werten der CATEGORY Spalte, aufsteigend.

❑

1.) Aufgabenblatt - Gruppe B(5.Punkte)

Kompetenzen

SELECT Klauseln: Formulieren einfacher SQL Abfragen:

- **select Klausel:** Selektion von Spaltennamen, Verarbeiten von Spaltenwerten, * Operator, Spaltenalias, DISTINCT Operator.
- **where Klausel:** Formulierung logischer Terme, Logische Operatoren and, or, not, Logische Operatoren like, in, is, between Dreiwertige Logik.
- **Order By Klausel:** Sortieren von Datensätzen.
- **Case Klausel:** Bedingte Verarbeitung, Kontextbezogene Verarbeitung.

Tabellen: COUNTRIES, CUSTOMERS, COSTS

User: sh

1.Beispiel) Select Klauseln - 1.Punkt

Geben Sie alle Länder aus die weder in Afrika noch im Mittleren Osten liegen.

- Geben Sie dabei die folgenden Spalten aus: COUNTRY_ISO_CODE, COUNTRY_NAME, COUNTRY_REGION.
- Sortieren Sie das Ergebnis nach dem COUNTRY_NAME aufsteigend!

❑

2.Beispiel) Select Klauseln - 1.Punkt

Geben Sie jene Kunden aus, deren CUST_MARITAL_STATUS bekannt ist.

- Es sollen jedoch nur jene Kunden ausgegeben werden, deren Nachname mit einem A,B oder C beginnet.
- Geben Sie dabei die folgenden Spalten aus: CUST_FIRST_NAME, CUST_LAST_NAME, INFO, CUST_MARTIAL_STATUS.
- @INFO: Die INFO Spalte beinhaltet den vollen Namen des Kunden, getrennt durch einen Punkt

  z.B.: Abigail Ruddy -> Abigail.Ruddy@INFO
- Sortieren Sie das Ergebnis nach den Werten der CUST_LAST_NAME aufsteigend!

❑

3.Beispiel) Select Klauseln - 1.Punkt

Geben Sie fuer alle Kostendatensätze die folgenden Spalten aus: PROD_ID, UNIT_COST, UNIT_PRICE, PROFIT, STAT.

- @PROFIT: Die PROFIT Spalte berechnet sich als Differenz zwischen UNIT_PRICE und UNIT_COST.
- Berücksichtigen Sie nur Produkte für die Unitkosten anfallen.
- @STAT: Die STAT Spalte gibt die prozentuelle Gewinnspanne für jedes Produkt in Relation zum Produktpreis.
- Sortieren Sie das Ergebnis nach dem Stat und Profit absteigend!

❑

4.Beispiel) Select Klauseln - 2.Punkt

Geben Sie für alle Kostendatensätze die folgenden Spalten aus: PROD_ID, UNIT_COST, UNIT_PRICE, COMPANY_PROFIT, STAT.

- @STAT: Die STAT Spalte gibt die prozentuelle Gewinnspanne jedes Produktes in Relation zum Produktpreis an.
- @COMPANY_PROFIT: Die COMPANY_PROFIT Spalte enthält eine verbale Beschreibung des Gewinnpotentials eines Produkts.

STAT	COMPANY_PROFIT
0% - 19%	LOSS
20% - 50%	LOST_LEADER
51% - 65%	PROFIT
66%+	HIGH_PROFIT

- Berücksichtigen Sie nur Produkte für die Unitkosten anfallen.
- Sortieren Sie das Ergebnis nach den Werten der STAT Spalte.

❑

7.) Aufgabenblatt - Gruppe A(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

1.Beispiel) Krankenhaus - 8.Punkte

Für einen Krankenanstaltenverbund soll zu Verwaltungszwecken eine Datenbank entwickelt werden.

- **EMPLOYEES:** Ärzte und Pfleger gehören zum Personal.

  Mitarbeiter sind eindeutig durch eine ID **EMPLOYEE_ID** (int - not null, unique). Für Mitarbeiter wird eine Sozialversicherungsnummer **SVNR** (int - not null, unique), der Vorame **FIRST_NAME** (varchar(30) - not null), der Nachname **LAST_NAME** (varchar(30) - not null) und das Gehalt **SALARY** (int - not null) gespeichert.

  Bei Ärzten (**PHYSICANS**) wird zusatzlich noch ihr Spezialgebiet **JOB_SPECIALISATION** (varchar(30) - not null, {GENERAL_PROACTITIONER, SURGEON, DIAGNOSTICIAN}) gespeichert. Für Pfleger (**CARE_TAKERS**) wird gespeichert, welcher andere Pfleger ihm vorgesetzt ist. Es gibt allerdings auch Pfleger ohne Vorgesetzten.

- **HOSPITAL_FACILITIES:** Ein Krankenhaus wird durch eine ID **FACILITY_ID** (int - not null, unique) identifiziert. Für Krankenhäuser wird ein Name **NAME** (varchar(100) - not null, unique) und die Telefonnummer der Information **PHONE_NR** (varchar(20) - not null, unique) gespeichert.

- **WARDS:** Jedes Krankenhaus hat mindestens eine, meistens jedoch mehrere Stationen. Eine Station ist für jedes Krankenhaus zusammen mit dem Namen **NAME** (varchar(100) - not null) eindeutig. Zusätzlich wird die Anzahl der Betten **CARRYING_CAPACITY** (int - not null) gespeichert.

  Eine Station wird von genau einem Arzt geleitet, wobei ein Arzt jedoch maximal eine Station leiten kann. Zusätzlich wird gespeichert, welche Pfleger wieviele Stunden **WORKING_HOURS** (int - not null) in einer Station arbeiten. Pfleger können durchaus in mehreren Stationen arbeiten.

❑

7.) Aufgabenblatt - Gruppe B(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

1.Beispiel) Radwerkstatt - 8.Punkte

Für eine Radwerkstatt soll zur Verwaltung eine Datenbank entwickelt werden.

- **EMPLOYEES:** Mitarbeiter sind eindeutig durch eine ID **EMPLOYEE_ID** (int - not null, unique). Zusätzlich wird die Personalnummer **EMPLOYEE_CODE** (varchar(12) - not null, unique), der Nachname **LAST_NAME** (varchar(50) - not null), der Vorname **FIRST_NAME** (varchar(50) - not null) und das Gehalt **SALARY** (int - not null) gespeichet.

- **BIKES:** Fahrräder sind eindeutig durch eine ID **BIKE_ID** (int - not null, unique). Außerdem werden Marke **BRAND** (varchar(100) - not null) und Bezeichnung **NAME** (varchar(100) - not null) gespeichert. Jeder Mitarbeiter betreut mehrere Fahrräder. Jedes Fahrrad wird von einem Mitarbeiter betreut.

- **REPAIR_SERVICE:** Jedes Service wird eindeutig durch eine ID **SERVICE_ID** (int - not null, unique) identifiziert.

  Jedes Service besteht mindestens aus einem Schritt (**WORKING_PHASES**). Ein Arbeitsschritt wird immer einem einzelnen Service zugeordnet. Jeder Schritt hat eine eindeutige Nummer **PHASE_ID** (int - not null, unique) und es werden eine Beschreibung **DESCRIPTION** (varchar(255) - not null) und die Kosten **PRICE** (int - not null) für den Schritt gespeichert. Im Zuge eines Services wird ein Fahrrad wieder in Stand gesetzt.

- **SPARE_PARTS:** Es kann sein, dass für manche Arbeitschritte Ersatzteile benötigt werden. Jedes Ersatzteil ist eindeutig durch eine Produktnummer **PART_ID** (int - not null, unique). Außerdem werden der Lagerbestand **STORED_UNITS** (int - not null) und der Preis **PRICE** (int - not null) gespeichert.

❑

8.) Aufgabenblatt - Gruppe A(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

1.Beispiel) Süßigkeitenfabrik - 8.Punkte

Entwerfen Sie für die folgende Aufgabenstellung ein relationales Diagramm.

- **SWEETS:** Süßigkeiten werden eindeutig identifiziert durch eine ID **SWEET_ID** (int - not null, unique). Zusätzlich wird ein Verkaufsnamen **NAME** (varchar(100) - not null, unique) und eine Verpackungsgröße **WRAPPING_TYPE** (varchar(20) - not null, {DIN_A, DIN_A2, DIN_B2, DIN_B4}) (**E_WRAPPING_TYPES**) gespeichert. Außerdem sind die Kalorien **CALORIFIC_VALUE** (int - not null) und die Produktionskosten **COST_OF_PRODUCTION** (int - not null) bekannt.

- **PRODUCT_REALISATION_PROCESSES:** Produktionsprozesse sind eindeutig durch eine ID **PROCESS_ID** (int - not null, unique). Zusätzlich wird eine Bezeichnung **PROCESS_NAME** (varchar(50) - not null, unique) gespeichert.

  Es ist bekannt welche Süßigkeiten mit welchen Produktionsprozessen hergestellt werden können. Dabei kann ein Produktionsprozess nur einer einzigen Süßigkeit zugeordnet sein. Für eine Süßigkeit können jedoch mehrere Produktionsprozesse bekannt sein.

  Produktionsprozesse bestehen aus mehreren Teilschritten (**SUBSTEPS**). Ein Teilschritt ist eindeutig durch eine ID **STEP_ID** (int - not null, unique). Für jeden Teilschritt wird ebenfalls die Dauer **TIME_DURATION** (int - not null) gespeichert. Ein Teilschritt kann in mehreren Produnktionsprozessen enthalten sein. Für jeden Teilschritt wird ebenfalls gespeichert welcher Teilschritt **STEP_INDEX** (int - not null) er im entsprechenden Produktionsprozess ist.

- **RAW_MATERIALS:** Rohstoffe sind eindeutig durch eine ID **MATERIAL_ID** (int - not null, unique). Es ist auch eine Bezeichnung **NAME** (varchar(100) - not null, unique) und die Qualitätsstufe **QUALITY** (varchar(2) - not null, {Q1, Q2, Q3}) (**E_QUALITY_TYPES**) bekannt. Manche Rohstoffe können durch andere Rohstoffe substituiert werden. Dabei kann ein Rohstoff immer durch mehrere andere Rohstoffe ersetzt werden. Es ist auch bekannt welche Menge **AMOUNT** (int - not null) von welchem Rohstoffen in einem Teilschritt eines Produktionsprozesses verbraucht werden.

- **MACHINES:** Maschinen sind eindeutig durch eine ne ID **MACHINE_ID** (int - not null, unique). Für Maschinen wird ebenfalls der Hersteller **PRODUCER** (varchar(100) - not null) und eine Bezeichnung **NAME** (varchar(255) - not null, unique) gespeichert.

  Es ist bekannt welche Maschine für welchen Teilschritt eines Verarbeitungsprozesses benötigt wird. Zur Durchführung eines Teilschritts können dabei mehrere Maschinen eingesetzt werden. Eine Maschine kann für die Durchführung mehrerer Teilschritte zum Einsatz kommen.

- **EMPLOYEES:** Mitarbeiter sind eindeutig durch eine ID **EMPLOYEE_ID** (int - not null, unique). Für Mitarbeiter wird ebenfalls der Vorname **FIRST_NAME** (varchar(50) - not null) und Nachname **LAST_NAME** (varchar(50) - not null) sowie das Gehalt **SALARY** (int - not null) gespeichert. Arbeiter und Techniker sind Mitarbeiter. Für Mitarbeiter ist bekannt welche Maschine, welchen Teilschritts welches Produktionsprozesses sie betreuen.

  Hinweis: Verwenden Sie im Falle der Mitarbeiter Singletable Vererbung.

❑

8.) Aufgabenblatt - Gruppe B(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

2.Beispiel) Turnierverwaltung - 8.Punkte

Für eine Sportorganisation soll zur Verwaltung von Vereinen und Turnieren eine Datenbank entwickelt werden.

- **ADDRESSES:** Eine Adresse wird eindeutig identifiziert durch eine ID **ADDRESS_ID** (int - not null, unique). Für Adressen wird die Straße **STREET** (varchar(100) - not null), die Postleitzahl **POSTAL_CODE** (varchar(6) - not null) und der Staat **COUNTRY** (varchar(100) - not null, {AUSTRIA, FRANCE, GREAT_BRITAIN, RUSSIA, SPAIN, PORTUGAL}) gespeichert.

- **CLUBS:** Vereine werden eindeutig identifiziert durch einen Namen **CLUB_NAME** (varchar(100) - not null, unique). Zusatzlich wird das Gründungsjahr **FOUNDING_DATE** (date - not null) und eine Adresse an welcher der Verein seinen Sitz hat gespeichert. An einer Adresse kann maximal ein Verein seinen Sitz haben.

- **TOURNAMENTS:** Vereine veranstalten Turniere. Ein Turnier ist eindeutig durch eine ID **TOURNAMENT_ID** (int - not null, unique).

  Für ein Turnier muss es mindestens einen Verein als Veranstalter geben. Ein Verein kann mehrere Turniere veranstalten. Für Turniere wird ein Name **NAME** (varchar(100) - not null, unique), der Austragungstag **BEGIN_DATE** (date - not null) und der Name des Hauptsponsors **SPONSORED_BY** (varchar(100) - not null) gespeichert. Jedes Turnier findet an einer bestimmten Adresse statt. An einer Adresse können mehrere Turniere veranstaltet werden.

- **EVENTS:** Ein Turnier besteht aus mehreren Bewerben wobei jeder Bewerb durch einen Namen **EVENT_NAME** (varchar(50) - not null) pro Turnier eindeutig ist. Es wird ebenfalls eine Beschreibung **DESCRIPTION** (varchar(255) - not null) pro Bewerb gespeichert. In der Datenbank wird zwischen Einzelbewerben (**INDIVIDUAL_EVENTS**) und Teambewerben (**TEAM_EVENTS**) unterschieden.

- **ATTENDEES:** Teilnehmer sind eindeutig durch eine ID **ATTENDEE_ID** (int - not null). Für Teilnehmer wird eine Sozialversicherungsnummer **SVNR** (varchar(12) - not null, unique) gespeichert. Außerdem wird der Nachname **LAST_NAME** (varchar(50) - not null), der Vorname **FIRST_NAME** (varchar(50) - not null), das Geburtsdatum **DATE_OF_BIRTH** (date - not null), und eine Mailadresse **MAIL** (varchar(50) - not null, unique) gespeichert.

  Jedem Teilnehmer wird eine Adresse zugeordnet, wobei an einer Adresse mehrere Teilnehmer wohnen konnen. Manche Teilnehmer, im Folgenden Vereinsspieler (**CLUB_PLAYER**) genannt, sind Mitglieder in einem Verein und haben daher zusatzlich eine Lizenznummer **LICENCE_NR** (varchar(16) - not null, unique). Vereinsspieler sind für genau einen Verein aktiv. Teilnehmer können an Einzelbewerben teilnehmen. Einen Einzelbewerb können mehrere Teilnehmer besuchen. An Teambewerben können nur Teams teilnehmen. Ein Team kann dabei an mehreren Teambewerben teilnehmen. Ein Teambewerb wird von mehreren Teams besucht.

- **TEAMS:** Ein Team wird eindeutig identifiziert durch eine ID **TEAM_ID** (int - not null, unique). Außerdem wird der Name **NAME** (varchar(20) - nto null, unique) des Teams gespeichert. Teilnehmer können Mitglieder in einem Team sein. Ein Team besteht aus mehreren Teilnehmern, wobei Teilnehmer durchaus in mehreren Teams Mitglied sein können. Es muss allerdings bekannt sein, welche Rolle **TEAM_ROLE** (varchar(20) - not null, {ATTACK, DEFENSE, COACH }) ein Teilnehmer in einem Team einnimmt.

❑

9.) Aufgabenblatt - Gruppe A(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

1.Beispiel) Filmstudio - 8.Punkte

Entwerfen Sie für die folgende Aufgabenstellung ein relationales Diagramm.

- **EMPLOYEES:** Mitarbeiter sind eindeutig durch eine eine ID **EMPLOYEE_ID** (integer - not null, unique). Jeder Mitarbeiter hat eine eindeutige Sozialversicherungsnummer **SVNR** (varchar(10) - not null, unique), einen Vornamen **FIRST_NAME** (varchar(45) - not null) und einen Nachnamen **LAST_NAME** (varchar(45) - not null).

  Grundsätzlich wird bei Mitarbeitern zwischen (**ARTISTS**) Künstlern und (**TECHNICANS**) Technikern unterschieden, wobei fur jeden Techniker noch seine Funktion **FUNCTION** (varchar(30) - not null, {ILLUMINATOR, STATE_DESIGNER, COSMETICIAN, MAKEUP_ARTIST }) gespeichert wird. Bei Künstlern wird weiter unterschieden zwischen Drehbuchautoren (**SCREENWRITER**), Regisseuren (**DIRECTOR**) und Schauspielern (**ACTOR**). Jedem Regisseur wird ein Jahresbudget **BUDGET** (number(10,2) - not null) zugewiesen, für jeden Schauspieler wird gespeichert wo er seine Ausbildung **EDUCATION** (varchar(100) - not null) absolviert hat.

- **SCREENPLAYS:** Drehbücher werden eindeutig identifiziert durch einen Titel **TITLE** (varchar(100) - not null) und den Drehbuchautor. Außerdem wird zu jedem Drehbuch die Seitenanzahl **PAGE_NUMBER** (integer - not null), das Erstellungsdatum **CREATION_DATE** (date - not null) und eine Kurzbeschreibung **DESCRIPTION** (varchar(255) - not null) gespeichert.

  Manchmal gibt es für Drehbücher Vorlagen (**DRAFTS**). Für ein Drehbuch kann es mehrere Vorlagen geben. Eine Vorlage kann Einfluß auf mehrere Drehbücher haben. Für Vorlagen wird eine Id **DRAFT_ID** (int - not null, unique) ein Titel **TITLE** (varchar(100) - not null) und der Autor **AUTHOR** (varchar(45) - not null) der Vorlage gespeichert. Regisseure setzen Drehbücher um. Jedes Drehbuch wird von genau einem Regisseur umgesetzt.

- **MOTION_PICTURES:** Jeder Film basiert auf genau einem Drehbuch. Ein Drehbuch wird für genau einen Film verfasst. Ein Film ist eindeutig identifiziert durch einen Titel **TITLE** (varchar(100) - not null, unique). Außerdem wird noch das Veröffentlichungsjahr **DATE_OF_PUBLICATION** (date - not null), das Genre **CATEGORY** (varchar(20) - not null, {MOTION_PICTURE, DOCUMENTARY, SHORT_MOVIE, EDUCATIONAL_MOVIE, TELEVISION_COMMERCIAL}) sowie die Dauer **LENGTH** (int - not null) des Films gespeichert. Für jeden Film ist auch bekannt, welche Schausspieler weche Rolle **ROLE** (varchar(45) - not null) darin spielen. Ein Schauspieler kann pro Film nicht mehrere Rollen einnehmen. Ein Film wird an mindestens einem Filmset gedreht. An einem Filmset können mehrere Filme gedreht werden.

- **MOVIE_SETS:** Ein Filmset ist eindeutig durch seine Bezeichnung **SET_DESCRIPTION** (varchar(100) - not null) und einem Ort **LOCATION** (varchar(45) - not null) bestimmt.

- **MOVIE_AWARDS:** Filmpreise sind eindeutig durch eine Id **AWARD_ID** (int - not null, unique). Für Filmpreise wird eine Bezeichnung **NAME** (varchar(45) - not null) und das Jahr **GRANTED_AT** (date - not null) gespeichert. Außerdem wird eine Dotierung **FUNDING_AMOUNT** (int - NOT NULL) gespeichert. Es ist bekannt welche Künstler für welche Filme welche Preise bekommen haben.

9.) Aufgabenblatt - Gruppe B(8.Punkte)

Kompetenzen

Relationale Modellierung: Entwurf der Struktur einer relationalen Datenbank.

1.Beispiel) Arztpraxis - 8.Punkte

Entwerfen Sie für die folgende Aufgabenstellung ein relationales Diagramm.

- **EMPLOYEES:** Ein Mitarbeiter wird eindeutig identifiziert durch eine id **EMPLOYEE_ID** (int - not null, unique). Zusätzlich wird ein Vorname **FRIST_NAME** (varchar(45) - not null) ein Nachname **LAST_NAME** (varchar(45) - not null) und das Geburtsdatum **DATE_OF_BRITH** (date - not null) gespeichert. Außerdem wird das Gehalt **SALARY** (int - not null) (GEHALT) gespeichert.

  (**PHYSICANS**) Ärzte und (**THERAPISTS**) Therapeuten sind Mitarbeiter. Bei Therapeuten wird zusätzlich die angeschlossene Ausbildung **EDUCATION** (varchar(45) - not null, {MASSEUR, CHIROPRACTIC }) gespeichert. Bei Ärzten muss vermerkt sein welche Ärzte welche anderen Ärzte vertreten können.

- **PATIENTS:** Ein Patient wird identifiziert durch eine ID **PATIENT_ID** (int - not null, unique). Für Patienten wird eine Sozialversicherungsnummer **SV-NR** (varchar(10) - not null, unique), ein Vorname **FIRST_NAME** (varchar(45) - not null), ein Nachname **LAST_NAME** (varchar(45) - not null) und eine Telefonnummer **PHONE** (varchar(20) - not null) gespeichert.

- **MEDICAL_CHECKOUT:** Medizinische Tests werden durch eine Id **MEDICAL_CHECKOUT_ID** (int - not null, unique) identifiziert. Für Medizinsche Tests wird ein Kurzel **CODE** (varchar(20) - not null, unique) und der Durchführungstermin **DATE_OF_TEST** (date - not null) gespeichert. Es wird ebenfalls eine Bezeichnung **DESCRIPTION** (varchar(45) - not null) gespeichert.

  Ärzte erstellen Diagnoseblätter (**DIAGNOSES**). Ein Diagnoseblatt wird von einem einzigen Arzt erstellt. Ein Diagnoseblatt wird identifiziert durch den Patienten, zu dem es gehört und einem Datum **DIAGNOSED_AT** (data - not null). Es wird außerdem ein Beschreibungstext **DESCRIPTION** (varchar(255) - not null) gespeichert und eine verbale Beschreibung des Ergebnisses **DIAGNOSIS** (varchar(255) - not null). Ein Diagnoseblatt umfasst ein oder mehrere Medizinsche Tests. Ein Medizinischer Test gehört immer zu einem einzelnen Diagnoseblatt.

- **MEDICAL_ATTENDANCES:** In einem Diagnoseblatt können mehrere Maßnahmen empfohlen werden. Eine Maßnahme wird durch eine ID **ATTENDANCE_ID** (int - not null) identifiziert. Zusätzlich wird eine Bezeichnung **CODE** (varchar(30) - not null, unique), eine Beschreibung **DESCRIPTION** (varchar(45) - not null) und die Wirkung **MEDICAL_EFFECT** (varchar(255) - not null) gespeichert. Eine Maßnahme kann mehreren Diagnoseblättern zugeordnet sein.

- **TREATMENTS:** Eine Behandlung wird von einem oder mehreren Therapeuten durchgefuhrt. Die Behandlung wird eindeutig identifiziert durch den Patienten, der die Behandlung bekommt und einem Datum **TREATED_AT** (date - not null). Die Dauer **LENGTH** (int - not null) der Behandlung wird ebenso vermerkt, wie die Maßnahme, die damit abgedeckt wird. Eine medizinische Maßnahme kann mehrere Behandlungen erfordern.

- **SUBSTANCES:** Ein Verbrauchsmaterial wird identifiziert durch durch eine Artikelnummer **SUBSTANCE_ID** (int - not null, unique). Es wird auch die Bezeichnung **NAME** (varchar(45) - not null, unique) des Verbrauchsmaterials, sowie der momentane Lagerstand **STORED_UNITS** (int - not null) gespeichert. Es wird ebenfalls vermerkt wieviel von einem Verbrauchsmaterial **AMOUNT** (int - not null, unique) für einen bestimmten medizinischen Test gebraucht wird und wieviel **AMOUNT** (int - not null, unique) von einem Verbrauchsmaterial für eine bestimmte Behandlung aufgewendet wird.