UNIVERZITA PARDUBICE  
Fakulta elektrotechniky a informatiky

Školní jídelna

Seminární práce z předmětu: **Databázové systémy I**

Program: **B0688A140009 - Informační technologie**

Angelina Korotaeva

V Pardubicedne18.12.2024

Obsah

1 Analýza ............................................................................................................................ 10 1.1 Scénář ........................................................................................................................... 10

1.1.1 Strukturální a procedurální pravidla .......................................................................... 11

1.1.2 Jednoduchá analýza ................................................................................................... 12

2 Návrh relační databáze .................................................................................................... 14

2.1 Konceptuální schéma - ERD ........................................................................................ 16

2.1.1 Diskuze smyček.......................................................................................................... 16

2.1.2 ERDish věty............................................................................................................... 16

2.2 Relační model dat ........................................................................................................ 19

3 Implementace databáze................................................................................................... 21

3.1 Fyzický model dat ........................................................................................................ 22

3.1.1 Integritní omezení (IO)............................................................................................... 32

3.2 Naplnění tabulek daty.................................................................................................... 33

4 Výčet SQL dotazů............................................................................................................ 34 Závěr................................................................................................................................... 43 Přílohy ................................................................................................................................ 44

# Analýza

## Scénář

Jsem manažer školní jídelny na státní střední škole číslo 5, která byla založena v 19. století. Dnes naše škola poskytuje vzdělání studentům z různých vrstev společnosti a naše jídelna se stará o stravování jak studentů, tak zaměstnanců školy. Rozhodli jsme se vytvořit systém pro efektivní správu stravování, který nám umožní lepší organizaci každodenních operací a také poskytne větší pohodlí našim studentům a zaměstnancům.

Každý **strávník** (**student** nebo **pracovník**) má jedinečné *ID strávníka*, své *jméno*, *příjmení* a také sledovaný *zůstatek* na účtu, aby mohl objednávat jídlo. Strávníci mají *typ strávníka* a mohou být buď studenti určitého *věku* patří do určité **třídy** (*id* a *číslo* třídy), nebo pracovníci školy, u kterých sledujeme jejich *telefon* a pracovní **pozici** (*id* a *název*). Navíc máme přehled o **adresách** všech strávníků, včetně *obce*, *ulice* a *PSČ*. Někteří strávníci mohou mít zdravotní omezení, jako jsou **alergie** (jejich *id*, *název* a *produkt*) nebo **dietní omezení** (*id*, *název* a *popsaní*), které je třeba respektovat při plánování jídel. Tyto údaje o alergiích a omezeních máme přehledně zaznamenány.

Naše jídelna vytváří týdenní **menu**, které obsahují různé druhy jídel. Každé **jídlo** má své *ID*, *název*, *popis*, *kategorii* (např. polévka, hlavní chod, salát) a *cenu*. Menu je aktualizováno každý den od pondělí do pátku liší se podle **týdne** (*číslo* a *všední den*). Jednotlivá jídla jsou sestavena z ingrediencí (**složek**), které máme na skladě. Ingredience mají své *ID*, *název*, *měrnou jednotku* (např. kilogramy, litry) a také *datum platnosti*, aby bylo možné zajistit čerstvost.

Strávníci si mohou objednávat jídlo pomocí našeho systému. Každá **objednávka** má své *ID*, *datum* vytvoření, *celkovou cenu* a případné *poznámky*. Objednávka obsahuje jednotlivé **položky**, kde sledujeme konkrétní jídla, jejich *množství* a *cenu*. Pokud dojde k jakékoli změně objednávky, systém umožňuje zaznamenat její **stav** (např. čekající, připravená, doručená), kde sledujeme jejich *ID* a *název*. Strávníci mohou své objednávky hradit různými způsoby – hotově, kartou nebo přes školní účet. Pro každou **platbu** sledujeme *ID* platby, *datum*, *částku* a *metodu* platby.

Náš systém nám pomáhá udržovat přehled o všech aspektech provozu jídelny – od evidování strávníků a jejich požadavků až po správu dodávek a přípravu jídel. Věříme, že tento systém přispěje ke zlepšení kvality služeb jídelny a usnadní její každodenní provoz.

### Strukturální a procedurální pravidla

**Strukturální pravidla:**

* **SP1:** Každá objednávka musí mít datum, celkovou cenu a musí být přiřazena stavu.
* **SP2:** Jedna objednávka musí mít alespoň jednu položku.
* **SP3:** Každý strávník musí mít jméno, příjmení a typ strávníka.
* **SP4:** Každý strávník může mít přiřazeny alergie a dietní omezení.
* **SP5:** Jedna alergie nebo dietní omezení nemůže být přiřazeno stejnému strávníkovi vícekrát.
* **SP6:** Každá platba musí mít přiřazenou částku, datum a způsob platby.
* **SP7:** Platba může být propojena pouze s jedním strávníkem.
* **SP8:** Každá adresa musí obsahovat město, ulici a PSČ.
* **SP9:** Každý strávník může být propojen pouze s jednou adresou.
* **SP10:** Každá složka musí mít název a jednotku měření.
* **SP11:** Každé jídlo musí mít název, popis, cenu a kategorii.
* **SP12:** Jedno jídlo může být součástí více menu.
* **SP13:** Každé menu musí mít název, typ a datum platnosti.
* **SP14:** Každá položka musí obsahovat množství a cenu.
* **SP15:** Studenti musí mít uveden věk. Pracovníci musí mít uvedeno telefonní číslo.
* **SP16:** Každá objednávka musí mít právě jeden stav.
* **SP17:** Každé menu musí být přiřazeno konkrétnímu týdnu.

**Procedurální pravidla:**

* **PP1.** Částka platby nemůže být záporná, způsob platby může nabývat hodnot: „hotovost“, „karta“, „převod“.
* **PP2.** Zákazníci, jejichž účet je 90 dní v mínusu, nebudou moci přidat další objednávku.
* **PP3.** Zaměstnanec může pracovat pouze na 1 pozici, pokud je platnost pozice „do“ již v minulosti, nemůže zaměstnanec na této pozici pracovat.
* **PP4.** Student může být pouze v jedné třídě.
* **PP5.** Data o menu a platbách by měla být pravidelně archivována a zálohována.
* **PP6.** Pokud je strávník na nějaký produkt alergický nebo má dietní omezení, nelze do objednávky přidat jídla obsahující tento produkt nebo porušující toto omezení.
* **PP7.** Objednávky lze zadávat pouze na aktuální nebo budoucí datum. Objednávky pro minulé datum nejsou povoleny.

### Jednoduchá analýza

1. **Záznam entit**
2. **STRÁVNÍK**

# *id\_stravnik*

*\* jmeno*

*\* primeni*

*\* zůstatek*

*\* typ strávníka*

*\* ID\_adresa*

1. **STUDENT**

*# id\_stravnik*

*\* vek*

*\* ID\_trida*

1. **TRIDA**

*# id\_trida*

*\* číslo\_tridy*

1. **PRACOVNIK**

*# id\_stravnik*

*\* telefon*

*\* ID\_pozice*

1. **POZICE**

*# id\_pozice*

*\* nazev*

1. **PLATBA**

**#** *id\_platba*

*\* datum*

*\* castka*

*\* metoda*

1. **ALERGIE**

*# id\_alergie*

*\* nazev*

*\* produkt*

1. **ALERGIE\_STRAVNIK**

*\* ID\_alergie*

*\* ID\_stravnik*

1. **DIETNI\_OMEZENI**

*# id\_omezeni*

*\* nazev*

*\* popsani*

1. **OMEZENI\_STRAVNIK**

*\* ID\_omezeni*

*\* ID\_stravnik*

1. **ADRESA**

*# id\_adresa*

*\* psc*

*\* obec*

*\* ulice*

1. **OBJEDNAVKA**

*# id\_objednavka*

*\* datum*

*\* celkova\_cena*

*o poznamka*

*\* ID\_stravnik*

*\* ID\_stav*

1. **STAV**

*# id\_stav*

*\* nazev*

1. **POLOZKA**

*\* množství*

*\* cena*

*\* ID\_objednavka*

*\* ID\_jidlo*

1. **JIDLO**

*# id\_jidlo*

*\* nazev*

*\* popis*

*\* kategorie*

*\* cena\_bez\_DPH*

*\* ID\_menu*

**MENU**

*# id\_menu*

*\* nazev*

*\* typ\_menu*

*\* time\_od*

*\* time\_do*

*\* tyden\_cislo*

1. **TYDEN**

*# tyden\_cislo*

*\* vsedni\_den*

1. **SLOZKA\_JIDLO**

*\* množství*

*o poznámka*

*\* ID\_jidlo*

*\* ID\_slozka*

1. **SLOZKA**

*#id\_slozka*

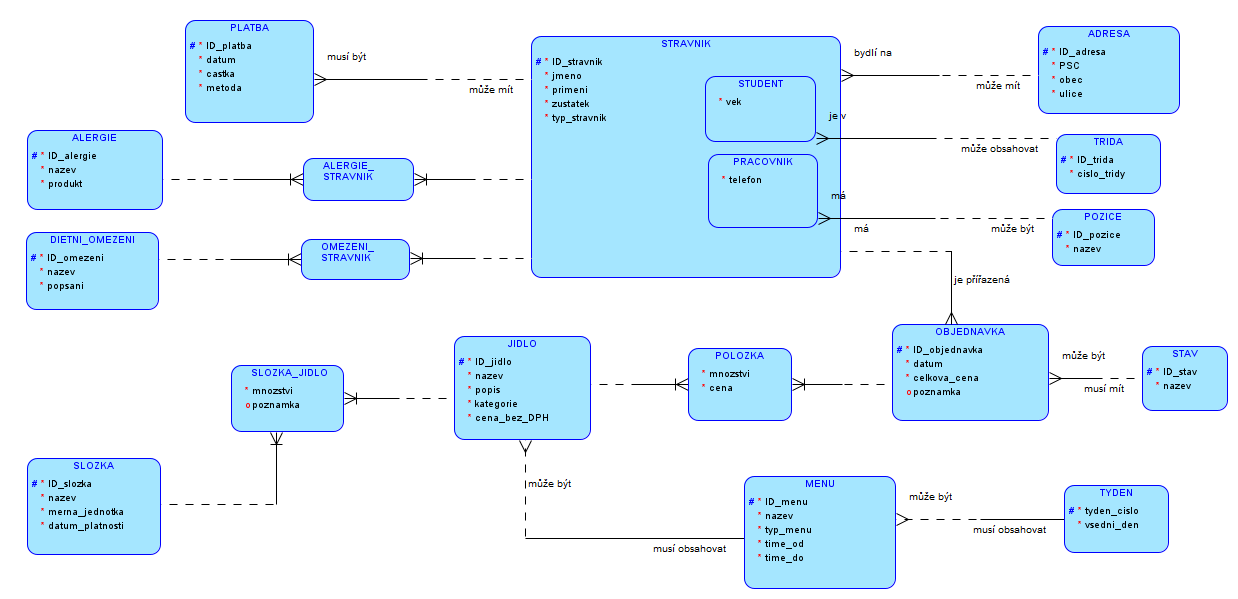
*\*nazev*

*\*merna\_jednotka*

*\*datum\_platnosti*

# Návrh relační databáze

## Konceptuální schéma – ERD



**2.1.1 Diskuze smyček**

V modelu nejsou žádné smyčky.

### ERDish věty

Strávník – Adresa:

* Každý Strávník někde bydli.
* Na určité adrese může bydlet strávník.

Jeden uživatel může mít 1 adresu, ale na dané adrese může žít více osob.

Strávník – Alergie (Typ relace: Many to many, přes Alergie\_Strávníka):

* Strávník může mít alergii.
* Alergie se může týkat strávníka.

Jeden strávník může mít více alergií a jednu alergii může sdílet více strávníků.

Strávník – Dietní omezeni (Typ relace: Many to many, přes Omezeni\_Strávníka):

* Strávník může mít dietní omezeni.
* Dietní omezeni se může objevit u strávníka.

Jeden strávník může mít více dietní omezeni a jedno dietní omezeni se může vyskytovat u více strávníků.

Strávník – Platba:

* Strávník může mít platbu.
* Platby musí být u strávníka.

Jeden strávník může mít mnoho plateb, ale každá platba může být určena pouze jednomu strávníku.

Student – Třída:

* Student musí být v určité třídě.
* Třída může mít studenta.

Jeden student může být pouze v jedné třídě, ale ve stejné třídě může být více studentů.

Pracovník – Pozice:

* Pracovník musí být na určité pozice.
* Pozice může mít pracovníka.

Jeden pracovník může zastávat pouze jednu pozici, ale na stejné pozici může být více pracovníků.

Strávník – Objednávka:

* Strávník musí mít objednávku.
* Objednávku lze zadat.

Jeden strávník může mít více objednávek, ale tuto objednávku může objednat pouze jeden strávník.

Objednávka – Stav:

* Objednávka musí mít stav.
* Stav může být přirazen k objednávce.

Jedna objednávka musí mít jeden stav, ale jeden stav může být přiřazen více objednávkám současně.

Objednávka – Jídlo (Typ relace: Many to many, přes Položka):

* Objednávka musí mít jídla.
* Jídla lze přiřadit k objednávce.

Jedna objednávka se může skládat z několika jídel, jedno jídlo může být obsaženo v několika objednávkách.

Jídlo – Složka (Typ relace: Many to many, přes Složka\_Jídla):

* Jídlo může obsahovat určité složku.
* Složka může být obsažena v jídle.

Jedno jídlo se může obsahovat několik složek, jedna složka může být obsažena ve více jídlech.

Jídlo – Menu:

* Jídlo může být v menu.
* Menu musí obsahovat jídla.

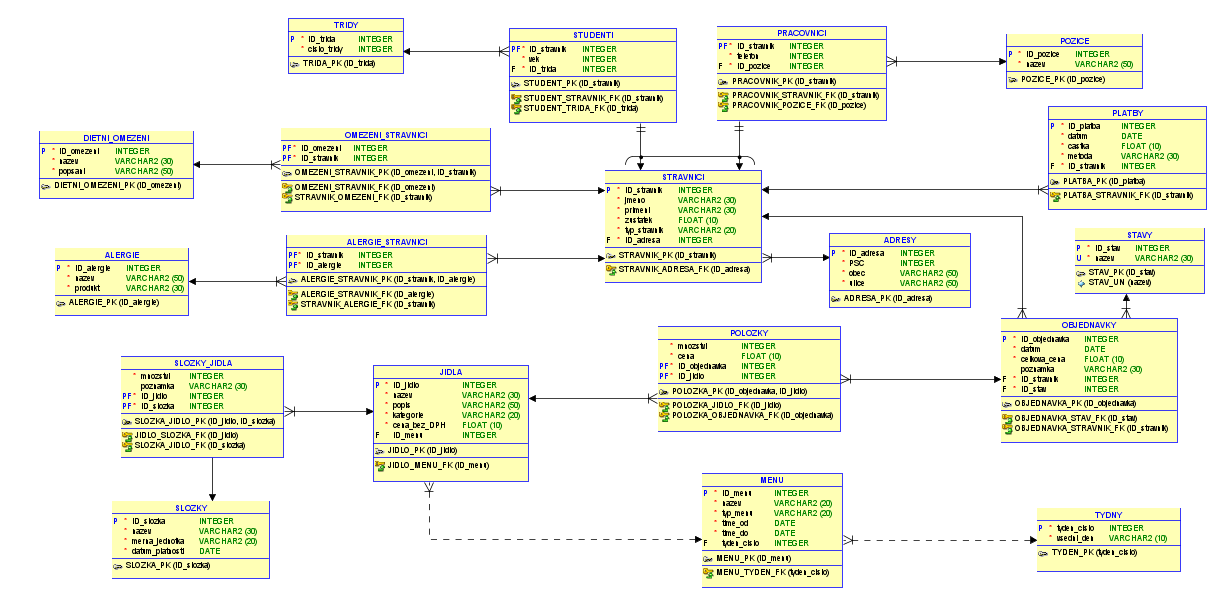
V menu může být obsaženo jedno jídlo, ale menu musí obsahovat více jídel.

Menu – Týden:

* Menu se může týkat konkrétního týdne a dne.
* Pro každý týden a den by mělo být jiné menu.

Jedno menu se může opakovat a být pro určitý týden a den, ale každý týden a den by mělo být jedno jiné menu.

## Relační model dat



Relační model dat pro databázi hudební školy byl vytvořen na základě konceptuálního modelu (ERD) a transformován do podoby relační databáze. Tento model obsahuje všechny entity, jejich atributy, primární a cizí klíče a vztahy mezi tabulkami.

Tabulky jsou pojmenovány v množném čísle, například “ADRESY”, “STRAVNICI”, “PRACOVNICI”.

**Transformace a mapovaní**

1. **Primární klíče (PK):**

Každá tabulka má svůj primární klíč, který zajišťuje jedinečnost každého záznamu.

* Tabulka **STRAVNICI** má primární klíč id\_stravnik.
* Tabulka **OBJEDNAVKY** má primární klíč id\_objednavka
* Tabulka **JIDLA** má primární klíč id\_jidlo atd.

1. **Cizí klíče (FK):**

Cizí klíče propojují entity mezi sebou, aby byly definovány vztahy.

* Tabulka **PRACOVNICI** obsahuje ID\_pozice jako cizí klíč, který odkazuje na tabulku **POZICE**.
* Tabulka **STUDENTI** obsahuje ID\_trida jako cizí klíč odkazující na tabulku **TRIDY**.
* Tabulka **JIDLA** obsahuje ID\_menu jako cizí klíč odkazující na tabulku **MENU** atd.

1. **Vazební tabulky:**

Vazby M:N byly rozloženy na samostatné tabulky s primárními a cizími klíči.

* Tabulka **ALERGIE\_STRAVNICI** propojuje tabulky **ALERGIE** a **STRAVNICI**. Obsahuje primární klíč složený z atributů is\_alergie a id\_stravnik.
* Tabulka **OMEZENI\_STRAVNICI** propojuje tabulky **DIETNI\_OMEZENI** a **STRAVNICI** atd.

**Supertyp a subtyp**

* **STRAVNICI** funguje jako supertyp pro dvě podtřídy:
  + **STUDENTI:** Mají atribut věk, který sleduje věk studenta.
  + **PRACOVNICI:** Mají atributy telefon, který sleduje telefon pracovníka.

Do tabulky **STRAVNICI** jsem přidala atribut *typ\_stravnik (ST, PR)*. Subentity sdílejí primární klíč id\_stravnik ze supertypu **STRAVNICI** a obsahují specifické atributy pro daný typ osoby.

# Implementace databáze

Sekvence a trigery jsem přidala do některých tabulek v mé práci z několika důvodů:

1. **Automatická generace primárních klíčů**  
   Sekvence a trigery zajišťují, že každá nová řádka v tabulce bude mít unikátní hodnotu v primárním klíči (např. id\_objednavka nebo id\_menu). To eliminuje riziko duplicitních hodnot a usnadňuje správu databáze.
2. **Jednodušší vkládání dat**  
   Díky triggerům není při vkládání dat nutné manuálně specifikovat hodnotu primárního klíče. Hodnota se automaticky generuje pomocí sekvence, což zjednodušuje zápisy do tabulek a snižuje riziko chyb.

Mám sekvence pro tabulky STRAVNICI, PLATBY, OBJEDNAVKY, JIDLA, MENU, ADRESY, ALERGIE, DIETNI\_OMEZENI, SLOZKY. Každá tabulka má triger pro primární klíč.

Například:

CREATE SEQUENCE s\_str START WITH 1 INCREMENT BY 1 NOCACHE;

CREATE OR REPLACE TRIGGER t\_str\_id

BEFORE INSERT ON stravnici

REFERENCING NEW AS NEW FOR EACH ROW

BEGIN

if(:new.id\_stravnik is null) then

SELECT s\_str.nextval

INTO :new.id\_stravnik

FROM dual;

end if;

END;

ALTER TRIGGER T\_STR\_ID ENABLE;

## Fyzický model dat

A close-up of a form

Description automatically generated

A close-up of a document

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

A close-up of a form

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a document

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a document

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

A close-up of a form

Description automatically generated

A close-up of a document

Description automatically generated

A close-up of a document

Description automatically generated

1. **Primární klíče (PK):**

Každá tabulka obsahuje primární klíč, který zajišťuje jednoznačnost záznamů.

1. **Cizí klíče (FK):**

Pro zajištění referenční integrity byly definovány cizí klíče.

1. **Indexy:**

Indexy byly vytvořeny automaticky pro klíče, což zajišťuje rychlé vyhledávání.

4. **Slabé entity:**

Tabulka **alergie\_stravnici** je slabou entitou, protože její záznamy závisí na existenci záznamů v tabulkách **alergie** a **stravnici**.

Tabulka **omezeni\_stravnici** je slabou entitou, protože její záznamy závisí na existenci záznamů v tabulkách **dietni\_omezeni** a **stravnici**.

Tabulka **polozky** je slabou entitou, protože její záznamy závisí na existenci záznamů v tabulkách **jidla** a **objednavky**.

Tabulka **slozky\_jidla** je slabou entitou, protože její záznamy závisí na existenci záznamů v tabulkách **slozky** a **jidla**.

5. **Supertyp/subtyp:**

Supertyp **stravnici** obsahuje obecná data všech strávníků (**studenti** i **pracovnici**). Subtypy jsou implementovány pomocí odvozených tabulek:

**studenti** s vazbou na stravnici pomocí cizího klíče;

**pracovnici** s vazbou na stravnici pomocí cizího klíče.

**6. Trigger:**

Triggery byly použity pro validaci typu osoby při vkládání nebo aktualizaci. Například:

CREATE OR REPLACE TRIGGER t\_str\_id

BEFORE INSERT ON stravnici

REFERENCING NEW AS NEW FOR EACH ROW

BEGIN

if(:new.id\_stravnik is null) then

SELECT s\_str.nextval

INTO :new.id\_stravnik

FROM dual;

end if;

END;

### Integritní omezení (IO)

* **IO1:** Částka platby nemůže být záporná, způsob platby může nabývat hodnot: „hotovost“, „karta“, „převod“.
* **IO2:** Zákazníci, jejichž účet je 90 dní v mínusu, nebudou moci přidat další objednávku.
* **IO3:** Zaměstnanec může pracovat pouze na 1 pozici, pokud je platnost pozice „do“ již v minulosti, nemůže zaměstnanec na této pozici pracovat.
* **IO4:** Student může být pouze v jedné třídě.
* **IO5:** Data o menu a platbách by měla být pravidelně archivována a zálohována.
* **IO6:** Pokud je strávník na nějaký produkt alergický nebo má dietní omezení, nelze do objednávky přidat jídla obsahující tento produkt nebo porušující toto omezení.
* **IO7:** Objednávky lze zadávat pouze na aktuální nebo budoucí datum. Objednávky pro minulé datum nejsou povoleny.
* **IO8:** Datum vytvoření objednávky musí být dřívější než datum její platby.
* **IO9:** Student musí být starší 6 let.

## Naplnění tabulek daty

Tabulky byly plněny postupně v následujícím pořadí:

1. **ALERGIE**
2. **DIETNI\_OMEZENI**
3. **STRAVNICI**
4. **ALERGIE\_STRAVNICI**
5. **OMEZENI\_STRAVNICI**
6. **ADRESA**
7. **TRIDY**
8. **POZICE**
9. **STUDENTI**
10. **PRACOVNICI**
11. **PLATBY**
12. **STAVY**
13. **OBJEDNAVKY**
14. **TYDNY**
15. **MENU**
16. **SLOZKY**
17. **JIDLA**
18. **SLOZKY\_JIDLA**
19. **POLOZKA**

# Výčet SQL dotazů

**A1:**" Seznam jídel s masem.", HK: D1

SELECT j.id\_jidlo, j.nazev AS jidlo

FROM jidla j

JOIN slozky\_jidla sj ON j.id\_jidlo = sj.id\_jidlo

JOIN slozky s ON sj.id\_slozka = s.id\_slozka

WHERE s.nazev = 'Maso';

**A2:** “Strávnici, kteří nezaplatili nebo zrušili objednávku”. HK: D2

SELECT s.id\_stravnik, s.jmeno, s.primeni

FROM stravnici s

JOIN objednavky o ON s.id\_stravnik = o.id\_stravnik

WHERE o.id\_stav IN (2, 3);

**A3:** “Studenti, kteří si objednali jídlo pouze jednou”. HK: D3

SELECT s.id\_stravnik, s.jmeno, s.primeni

FROM stravnici s

JOIN studenti st ON s.id\_stravnik = st.id\_stravnik

JOIN objednavky o ON s.id\_stravnik = o.id\_stravnik

GROUP BY s.id\_stravnik, s.jmeno, s.primeni

HAVING COUNT(o.id\_objednavka) = 1;

**A4:** “Vrací objednávku, ve které všechna jídla patří do stejné menu a celkový počet jídel v objednávce se shoduje s počtem všech jídel v tomto menu.”. HK: D4

SELECT o.id\_objednavka

FROM objednavky o

JOIN polozky p ON o.id\_objednavka = p.id\_objednavka

JOIN jidla j ON p.id\_jidlo = j.id\_jidlo

GROUP BY o.id\_objednavka

HAVING COUNT(DISTINCT j.id\_menu) = 1

AND COUNT(p.id\_jidlo) = (

SELECT COUNT(j2.id\_jidlo)

FROM jidla j2

WHERE j2.id\_menu = MAX(j.id\_menu)

);

**A5:** “Seznam pracovníků, kteří alespoň jednou zaplatili za objednávku.”. HK: D5

SELECT id\_stravnik, telefon

FROM pracovnici

JOIN platby USING (id\_stravnik);

**A6:** “Seznam objednávek se stavem „dokončeno“ a počet strávníků, kterým na účtu zbývá více než 1500 Kč.”. HK: D6

SELECT o.id\_objednavka, o.id\_stravnik, o.id\_stav, s.nazev

FROM objednavky o

JOIN stavy s ON o.id\_stav = s.id\_stav

JOIN stravnici st ON o.id\_stravnik = st.id\_stravnik

WHERE s.nazev = 'Dokončeno'

AND st.zustatek > 1500;

**A7:** “ Seznam objednávek s informacemi o stranicích z Prahy.”. HK: D7

SELECT

o.id\_objednavka,

s.jmeno AS stravnik\_jmeno,

s.primeni AS stravnik\_primeni,

a.psc AS stravnik\_psc,

a.obec AS stravnik\_obec

FROM objednavky o

NATURAL JOIN stravnici s

NATURAL JOIN adresy a

WHERE a.obec = 'Praha';

**A8:** “ Spojení všech strávníků s menu pomocí CROSS JOIN.”. HK: D8

SELECT stravnici.jmeno, menu.typ\_menu

FROM stravnici

CROSS JOIN menu;

**A9:** “ Seznam všech strávníků, jejich jména a celková cena objednávky. Pokud objednávka nebyla zadána, řádek vrátí null.”. HK: D9

SELECT stravnici.id\_stravnik,

stravnici.jmeno,

objednavky.id\_objednavka,

objednavky.celkova\_cena

FROM stravnici

LEFT OUTER JOIN objednavky ON stravnici.id\_stravnik = objednavky.id\_stravnik;

**A10:** “ Vypíše všechny záznamy z tabulky platby, i když nesouvisejí se strávnicí.”. HK: D10

SELECT platby.id\_platba, platby.castka, stravnici.jmeno

FROM stravnici

RIGHT OUTER JOIN platby ON stravnici.id\_stravnik = platby.id\_stravnik;

**A11:** “ Vypíše všechny stravnici a všechna data plateb, pokud nemají shodu, vypíše null.”. HK: D11

SELECT stravnici.id\_stravnik, stravnici.jmeno, platby.datum

FROM stravnici

FULL OUTER JOIN platby ON stravnici.id\_stravnik = platby.id\_stravnik;

**A12:** “ Vypíše jména a příjmení pouze těch strávníků, kteří měli částku z objednávky platby .”. HK: D12

SELECT jmeno, primeni

FROM stravnici

WHERE id\_stravnik IN (

SELECT id\_stravnik

FROM platby

WHERE castka > 0

);

**A13:** “ Vypočítá průměrný zůstatek na účtu všech stravníků.”. HK: D13

SELECT dotaz.stredni\_zustatek

FROM (

SELECT AVG(zustatek) AS stredni\_zustatek

FROM stravnici

) dotaz;

**A14:** “ Vypíše název a počet objednávek každého strávníků. ”. HK: D14

SELECT jmeno, (

SELECT COUNT(\*)

FROM objednavky

WHERE objednavky.id\_stravnik = stravnici.id\_stravnik) AS pocet\_objednavek

FROM stravnici;

**A15:** “ Vypíše všechna menu, která obsahují jídla.”. HK: D15

SELECT m.nazev

FROM menu m

WHERE EXISTS (

SELECT 1

FROM jidla j

WHERE j.id\_menu = m.id\_menu

);

**A16:** “ Vybírá studenty se zůstatkem na účtu nižším nebo rovným 1500 a studenty starší 10 let.”. HK: D16

SELECT s.jmeno

FROM stravnici s

JOIN studenti st ON s.id\_stravnik = st.id\_stravnik

WHERE s.zustatek <= 1500

UNION

SELECT s.jmeno

FROM stravnici s

JOIN studenti st ON s.id\_stravnik = st.id\_stravnik

WHERE st.vek > 10;

**A17:** “ Zobrazí menu, která jsou prázdná.”. HK: D17

SELECT id\_menu

FROM menu

MINUS

SELECT id\_menu

FROM jidla;

**A18:** “ Zobrazí všech strávníků, kteří mají dietní omezení.”. HK: D18

SELECT id\_stravnik

FROM stravnici

INTERSECT

SELECT id\_stravnik

FROM omezeni\_stravnici;

**A19:** “ Spojí název a popis z tabulky JIDLA a zadá je velkými písmeny. ”. HK: D19

SELECT UPPER(CONCAT(CONCAT(nazev, ': '), popis)) AS nazev\_popis\_uppercase

FROM jidla;

**A20:** “ Výstupy pro pracovníky, jejichž zůstatek na účtu je vyšší než průměrná hodnota zůstatku. ”. HK: D20

SELECT s.jmeno, s.primeni

FROM stravnici s

JOIN pracovnici st ON s.id\_stravnik = st.id\_stravnik

WHERE s.zustatek > (

SELECT AVG(zustatek)

FROM stravnici

);

**A21:** “ Vrací rozdíl mezi datem objednávky a datem platby objednávky v minutách, počítají se pouze dokončené objednávky. ”. HK: D21

SELECT

o.datum AS datum\_objednavky,

p.datum AS datum\_platby,

ROUND((p.datum - o.datum) \* 24 \* 60) AS rozdil\_v\_minutach

FROM objednavky o

JOIN platby p ON o.id\_stravnik = p.id\_stravnik

WHERE o.id\_stav = 1;

**A22:** “ průměrný věk studentů.”. HK: D22

SELECT AVG(vek) AS prumerny\_vek

FROM studenti;

**A23:** “ Výstupy id\_platba, jejichž částka je větší než průměrná částka.”. HK: D23

SELECT id\_platba, AVG(castka) AS avg\_zustatek

FROM platby

GROUP BY id\_platba

HAVING AVG(castka) > 4;

**A24:** “ Seznam složek, které mají měrnou jednotku kus.”. HK: D24

* 1. Použíti standartního filtru WHERE

SELECT nazev

FROM slozky

WHERE merna\_jednotka = 'kus';

* 1. Použití vnořeného dotazu ve WHERE

SELECT nazev

FROM slozky

WHERE merna\_jednotka IN (SELECT merna\_jednotka FROM slozky

WHERE merna\_jednotka = 'kus');

* 1. Použíti GROUP BY

SELECT nazev

FROM slozky

GROUP BY nazev, merna\_jednotka

HAVING merna\_jednotka = 'kus';

**A25:** “ Seznam strávníků, kteří zadali více než jednu objednávku, seřazený podle počtu jejich objednávek. ”. HK: D25

SELECT id\_stravnik, COUNT(\*) AS pocet\_objednavek

FROM objednavky

WHERE id\_stravnik IS NOT NULL

GROUP BY id\_stravnik

HAVING COUNT(\*) > 1

ORDER BY pocet\_objednavek DESC;

**A26:** “ Vytvoření pohledu zobrazujícího seznam menu podle typu menu.”. HK: D26

CREATE VIEW menu\_obed AS

SELECT id\_menu, nazev, typ\_menu

FROM menu

WHERE typ\_menu = 'OBED';

**A27:** “ Seznam studentů, kteří si objednali a nemají žádné alergie nebo dietní omezení. ”. HK: D27

SELECT DISTINCT s.id\_stravnik, str.jmeno, str.primeni

FROM studenti s

JOIN stravnici str ON s.id\_stravnik = str.id\_stravnik

JOIN objednavky o ON s.id\_stravnik = o.id\_stravnik

WHERE NOT EXISTS (

SELECT 1

FROM omezeni\_stravnici os

WHERE os.id\_stravnik = s.id\_stravnik

)

AND NOT EXISTS (

SELECT 1

FROM alergie\_stravnici als

WHERE als.id\_stravnik = s.id\_stravnik

);

**A28:** “ Vytvoří objednávku pro pracovníka, jehož telefonní číslo začíná číslicí 2.”. HK: D28

INSERT INTO objednavky (id\_objednavka, datum, celkova\_cena, id\_stravnik, id\_stav)

SELECT s\_obj.NEXTVAL, SYSDATE, 100, p.id\_stravnik, 4

FROM pracovnici p

JOIN stravnici s ON p.id\_stravnik = s.id\_stravnik

WHERE p.telefon LIKE '2%';

**A29:** “Zvýší částku o 100,2, pokud je způsob platby hotovostní a stará částka byla vyšší než hodnota průměrné částky.”. HK: D29

UPDATE platby

SET castka = castka + 100.2

WHERE metoda = 'hotově'

AND castka > (SELECT AVG(castka) FROM platby);

**A30:** “ Odstraní duplicity z tabulky složky vyhledáváním podle jmen.”. HK: D30

DELETE FROM slozky

WHERE id\_slozka IN (

SELECT id\_slozka

FROM (

SELECT MIN(id\_slozka) AS id\_slozka

FROM slozky

GROUP BY nazev

HAVING COUNT(\*) > 1

)

);

Závěr

Moje databáze slouží ke sledování objednávek ve školní jídelně a obsahuje informace o návštěvnících, jejich objednávkách, jídlech a platbách. Každé objednávce odpovídá pouze jeden způsob platby (hotovost, karta nebo kupony). Databáze dodržuje politiku standardizace, která minimalizuje nadbytečné údaje a umožňuje efektivní správu dat pomocí dotazů SQL.

Při práci na tomto projektu jsem se naučila, jak vytvořit databázi a navrhnout strukturu databáze, která dodržuje politiku standardizace, což znamená, že tabulky jsou dobře uspořádané a minimalizují nadbytečná data. Naučila jsem se také vytvářet vztahy mezi tabulkami a spravovat data pomocí dotazů SQL pro výběr, vkládání, úpravu a mazání. Používala jsem nástroje Oracle SQL Developer Data Modeler a Oracle SQL Developer, které mi pomohly lépe porozumět návrhu databáze. Tyto dovednosti se mi budou v budoucnu hodit při práci na dalších databázových projektech. Databáze je flexibilní a připravená na rozšíření, ale vidím možnosti, jak zlepšit zabezpečení a optimalizovat výkon při práci s velkými daty. Tento projekt mi poskytl cenné zkušenosti a motivoval mě k dalšímu rozvoji v oblasti návrhu databází.

Přílohy

1. DDL skript pro vytvoření tabulek, omezení, sekvencí, indexů, triggerů, apod.
2. DML skript pro naplnění tabulek daty.
3. Vygenerovaná dokumentace z Oracle SQL Developer Data Modeler (kapitola 3.1 Fyzický model dat)
4. Soubor .dmd včetně složky generované Oracle SQL Developer Data Modeler(Datový model musí souhlasit s DDL skriptem)
5. Obrázek konceptuálního modelu (ERD) vygenerovaný z Oracle SQL Developer Data Modeler.
6. Obrázek datového modelu vygenerovaný z Oracle SQL Developer Data Modeler.
7. Skript se všemi SQL dotazy (kapitola 4. Výčet SQL dotazů).