3 Databáze (3 body)

1. Uvažujte transakční rozvrh $T_{12} = X_1(A), W_1(A), U_1(A), X_2(A), R_2(A), U_2(A), S_1(B), R_1(B), U_1(B)$. V zápisu rozvrhu X_i, resp. S_i znamená požadavek na exkluzivní, resp. sdílený zámek i-té transakce, U_i je odemčení. R_i a W_i odpovídá čtení a zápisu proměnné v i-té transakci. Odpovídá rozvrh T_{12} požadavkům na dvoufázový (2PL) uzamykací protokol? Pokud ne, opravte jej tak, aby požadavkům odpovídal a pokud možno nedošlo ke změně vzájemného pořadí všech uvedených čtení a zápisů.
2. Nad tabulkou $Teplota(\underline{Rok, Mesic, Den}, Stupne)$, obsahující data za roky 2010 až 2019, napište dotaz, který zjistí, ve kterých měsících roku 2019 teplota klesla pod nulu alespoň dva dny za sebou. (Situace, kdy teplota klesla pod nulu poslední den jednoho a první den následujícího měsíce neřešte.)

11 Relační databáze (otázka studijního zaměření – 3 body)

Mějme online obchod, který vede evidenci prodejů v relační databázi v tabulce *sales*. Tabulka *sales* má následující strukturu:

- **sale_id** (int): primární klíč
- **date** (date): datum uskutečnění prodeje
- **price** (int): celková cena prodeje

- **customer** (varchar): login uživatele, který nákup provedl
- **customer_full_name** (varchar): plné jméno uživatele ve tvaru „jméno příjmení“, který nákup provedl
- **address** (varchar): adresa pro doručení ve formátu „Město;Ulice;Číslo popisné;Poštovní Směrovací číslo“.

Jedná se o jedinou tabulku v systému.

1. Upravte schéma databáze (tabulka *sales*) tak, aby splňovalo 3. normální formu.
2. Pro upravené schéma z předchozího bodu napište SQL dotaz, který pro každou ulici v databázi vrátí součet všech prodejů (sloupeček **price**). V dotazu je třeba ošetřit situaci, kdy je v databázi uložena ulice, ale není k ní uveden žádný prodej. V takovém případě bude pro danou ulici vrácena hodnota 0.
3. Při používání upraveného schématu z 1. bodu se ukázalo, že je třeba optimalizovat vyhledávání prodejů pro daný den. Jak je možné tuto optimalizaci realizovat s využitím SQL serveru?

Při řešení můžete použít SQL dialekty běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

Mějme cestovní kancelář, která uchovává svou nabídku zájezdů v relační databázi v tabulce *tours*. Tabulka *tours* má následující strukturu:

- **tour_id** (int): unikátní identifikátor zájezdu
- **title** (varchar): název zájezdu
- **country_code** (char[3]): identifikátor cílové země (dle ISO 3166-1)
- **description** (varchar): textový popis zájezdu
- **date_from** (date): termín odjezdu
- **date_to** (date): termín příjezdu
- **price** (int): cena zájezdu
- **price_description** (varchar): textový popis co cena zájezdu (ne)zahrnuje

1. V průběhu práce s databází se zjistilo, že tabulka *tours* neodpovídá dostatečně realitě: Stejné zájezdy (stejný název, popis a cílové místo) jsou obvykle nabízeny ve více termínech. Zároveň každý termín zájezdu může mít přiřazeno několik různých cen (např. 1-lůžkový / 2-lůžkový pokoj). Upravte schéma tak, aby odpovídalo popsaným skutečnostem.
2. Napište dotaz, který bude mít na výstupu stejnou strukturu jako původní tabulka *tours* (stejný seznam sloupců, tj. *tour_id*, *title*,...,*price_description*) a vrátí pro každý zájezd a každý jeho termín nejnižší cenu zájezdu (*price*) a jí odpovídající popisek ceny (*price_description*).
3. Vlivem chyby v navázaném informačním systému se do databáze zanesly nekonzistence, které je třeba odstranit. Napište dotaz, který vrátí všechny *id* zájezdu (*tour_id* ve vámi definovaném schématu), pro které platí, že buď nemají přiřazený žádný termín, nebo pro žádný z přiřazených termínů neexistuje záznam s odpovídající cenou.

Při řešení tolerujeme použití SQL dialektů běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

10 Relační databáze (otázka studijního zaměření – 3 body)

Uvažujme následující tabulky v relační databázi: *Person* a *Task*. Tabulka *Person* má následující strukturu :

- **ID** (int): primary key
- **Name** (varchar): jméno osoby
- **Age** (int): věk osoby

V tabulce Person jsou následující záznamy:

ID	Name	Age
1	Tom	25
2	Jane	30
3	Adam	29
4	Max	36

Tabulka *Task* má následující strukturu:

- **ID** (int): primary key
- **Description** (varchar): popis tasku
- **Deadline** (date): deadline pro splnění tasku
- **Done** (bool): informace, zda byl task splněn
- **PersonID** (int): foreign key vedoucí na ID v tabulce Person

V tabulce Tasks jsou následující záznamy:

ID	Description	Deadline	Done	PersonID
1	watch movie	2022-05-30	false	1
2	cook dinner	2022-05-31	false	1
3	plan holiday	2022-06-30	false	1
4	submit paper	2022-04-20	true	2
5	watch movie	2022-05-30	false	2
6	write thesis	2022-07-30	true	4

1. Napište SQL dotaz, který bude mít na výstupu stejnou strukturu jako tabulka *Person*. Dotaz vrátí tabulku, kde bude osoba uvedena právě jednou, pokud platí, že má přiřazený nesplněný úkol po uplynutí deadlinu (k datu zkoušky).
2. Napište, jaký výstup budou mít následující dotazy. V případě chyby v SQL dotazu vyznačte místo s chybou a chybu vysvětlete.
   - `SELECT Name FROM Person WHERE ID Not In (SELECT PersonID From Task)`
   - `SELECT Name, Description FROM Person JOIN Task ON (ID = PersonID) WHERE Age < 30`
3. V průběhu práce s databází bylo zjištěno, že relace mezi tabulkou Person a Task neodpovídá skutečnosti. Konkrétně k práci na jednomu tasku může být přiřazeno vícero (neomezeně) osob, z nichž jedna může být za provedení tasku zodpovědná. Upravte schéma relační databáze tak, aby odpovídalo popsaným skutečnostem a zároveň splňovalo první normální formu (1NF).

Při řešení můžete použít SQL dialekt běžně známých databází (Oracle, MySQL, PostgreSQL, MSSQL, ...).

10 Relační databáze a JSON (otázka studijního zaměření – 3 body)

Mějme běžný relační databázový systém, ve kterém je databáze a z ní nás zajímá jedna tabulka:

`users (id: int, name: varchar, settings: text)`

Sloupec `id` je primární klíč, `name` celé jméno uživatele a `settings` obsahuje JSON se strukturou obsahující konfiguraci uživatele (této struktuře rozumí jen webová aplikace používající databázi). Uveďme příklad takové konfigurace:

```
{
  "window": {
    "x": 42,
    "y": 54,
  },
  "theme": "dark",
  "notifications": true,
}
```


Konfigurace je tedy serializovaná struktura s parametry, jejichž hodnoty jsou řetězce, celá čísla, booly, nebo vnořené struktury (rekurzivní vnoření není dovoleno, položky uvnitř struktury `window` musí být již pouze skaláry). Dále je garantováno, že všechny názvy položek obsahují pouze takové znaky, aby bylo možné je přímo zapsat v JavaScriptu jako identifikátory, délky názvů nepřesáhnou 64 znaků a délky řetězcových hodnot 255 znaků.

1. Kterou normální formu výše uvedené schéma porušuje a proč? Jaké praktické výhody a nevýhody má toto porušení normální formy? (Uveďte stručně jednu výhodu a jednu nevýhodu.)
2. Navrhněte úpravu schématu (případně dat celkově) tak, aby databáze danou normální formu neporušovala.
3. Napište SQL dotaz, pomocí kterého může aplikace pracující s databází získat kompletní nastavení uživatele ve vámi navrženém upraveném schématu. Technické detaily formátování případně spojování řetězců neřešte (beztak se často liší dle použitého dialektu SQL), ale zaměřte se na to, aby váš dotaz vrátil z databáze všechna potřebná data, ze kterých si může aplikace původní JSON sestavit.

3 ER diagramy a dotazy v SQL (3 body)

Navrhněte (velmi) zjednodušený systém evidence letadel na letištích. Pro každé letiště chceme evidovat jeho jméno a jedinečnou zkratku. Dále chceme evidovat data a časy přistání letadel, jejich výrobce, typ a imatrikulaci (jedinečný kód letadla). Neřešte mezní situace, jako například změnu imatrikulace letadla, změnu zkratky či jména letiště, či slučování výrobců letadel.

1. Pro daný příklad navrhněte ER diagram.
2. ER diagram převeďte na relační schéma (UML diagram tříd či SQL DDL) splňující 3NF. Vyznačte klíče a cizí klíče.
3. Napište dotaz v SQL, který pro dané letiště a den vypíše žebříček výrobců letadel dle počtu přistání jejich strojů.