Část I (60 bodů)

1. **Vytvořte diagram ER konceptuálního schématu** pro databázi uměleckých děl galerie: [**10**]

   Každé dílo má přiřazeno jednoznačné inventární číslo, rok pořízení a název. Ve sbírce jsou dva druhy děl, konkrétně obrazy a sochy. Každý obraz obsahuje strukturovaný údaj o rozměrech (šířka a výška), naopak každá socha má váhu v kilogramech a informaci o materiálu, ze které je vyrobena (např. bronz, žula atp.). Zaměstnanec galerie je jednoznačně určen rodným číslem, dále u něj uchováváme adresu trvalého bydliště, nepovinnou emailovou adresu a libovolný počet telefonních čísel. Kromě toho má každý zaměstnanec osobní číslo, které je jednoznačné jen v rámci oddělení, ve kterém daný zaměstnanec pracuje. Oddělení má název a unikátní kód. V rámci galerie jsou pořádány výstavy. Každá výstava má svůj název a rok realizace, dohromady tyto údaje každou výstavu identifikují. V rámci výstavy je vystaveno libovolné množství děl, za každé z nich pak odpovídá (v rámci dané výstavy) konkrétní zaměstnanec. Některé výstavy mohou navazovat na jiné, v systému tyto návaznosti také potřebujeme uchovávat.

   Nepřidávejte uměle generované identifikátory, alespoň jednou použijte ternární vztahový typ a specifikujte kritéria pokrytí a překrytí u ISA hierarchie. Namodelujte jen to, co je požadováno.

Užitečné odkazy: http://erdia.stejspet.cz/, https://creately.com/ER-diagram-software, https://www.draw.io/

2. **Transformujte ER schéma** z předcházejícího příkladu **do schématu v relačním modelu**. Schémata jednotlivých tabulek popište textovou notací. Uveďte všechny klíče a cizí klíče, nepřidávejte uměle generované identifikátory. Předpokládejte relační model bez `NULL` hodnot. [**10**]

3. Předpokládejme následující relační schéma databáze hotelu:

   Zákazník (<u>rodné číslo</u>, jméno, příjmení, město)
   Pokoj (<u>číslo</u>, typ, velikost, popis)
   Ubytování (<u>id</u>, <u>pokoj, datum začátku</u>, datum konce, rodné číslo)
   (pokoj) $\subseteq$ Pokoj (číslo)
   (rodné číslo) $\subseteq$ Zákazník (rodné číslo)

   Vytvořte SQL výrazy pro následující dotazy: [**15**]

   a) Najděte unikátní křestní jména zákazníků z *Brna*, kteří nikdy neměli žádné ubytování v pokoji typu *jednolůžkový* ani *dvoulůžkový*.

   b) Najděte rodná čísla a příjmení zákazníků, kteří alespoň v jednom okamžiku měli alespoň dvě ubytování najednou.

   c) Pro každé město zjistěte celkový počet ubytování v pokojích typu *dvoulůžkový* realizovaných zákazníky žijícími v daném městě. Omezte se však jen na zákazníky s křestním jménem jiným než *Aneta*. Ve výsledku uveďte jen ta města, kde takový počet je roven alespoň *5*.

4. Předpokládejme relační schéma z předcházejícího příkladu. **Vytvořte výraz pro SQL/XML dotaz**, kde pro každé ubytování (na samostatném řádku výsledné tabulky) uvedete v prvním sloupci jeho id a ve druhém zkonstruujte XML hodnotu podle následujícího vzoru: [**5**]

```
<ubytovani od="datum začátku" do="datum konce">
   <pokoj>typ pokoje</pokoj>
   <zakaznik>rodné číslo zákazníka</zakaznik>
</ubytovani>
```

5. Opět předpokládejme stejné relační schéma. **Formulujte výrazy v relační algebře** pro tyto dotazy: [**10**]

Zákazník (<u>rodné číslo</u>, jméno, příjmení, město)
Pokoj (<u>číslo</u>, typ, velikost, popis)
Ubytování (<u>id</u>, <u>pokoj, datum začátku</u>, datum konce, rodné číslo)
(pokoj) $\subseteq$ Pokoj (číslo)
(rodné číslo) $\subseteq$ Zákazník (rodné číslo)

Užitečné symboly: × ∪ ∩ \ ÷

a) Čísla a velikosti pokojů typu *jednolůžkový*, ve kterých nikdy neproběhlo ani jedno ubytování zákazníka s příjmením *Veselý* žijícím v *Děčíně*.

b) Jména a příjmení zákazníků s příjmením *Smutný*, kteří byli alespoň jednou ubytovaní v každém pokoji typu *dvoulůžkový*.

6. **Najděte všechny klíče** v relačním schématu s množinou atributů $\{\mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$ a množinou funkčních závislostí $\{\mathtt{D} \to \mathtt{G}, \mathtt{CDE} \to \mathtt{F}, \mathtt{G} \to \mathtt{E}, \mathtt{F} \to \mathtt{CE}, \mathtt{FG} \to \mathtt{D}\}$. Postup komentujte, použijte algoritmus Lucchesi-Osborn. Následně **určete normální formu** každé jednotlivé závislosti a také celého uvedeného schématu. [**10**]

Část II (40 bodů)

8. Databázové **pohledy** [5]

a) Pohled je definovaný `SELECT` dotazem a obsahuje data, která odpovídají výsledku tohoto dotazu v okamžiku vytvoření daného pohledu pomocí `CREATE VIEW` — a) true false

b) Existující pohledy je možné používat ve `FROM` klauzulích stejným způsobem jako obyčejné tabulky — b) true false

c) Pohled může být závislý na jiném pohledu, dokonce i tranzitivně, ale nikdy nemůže vzniknout jako spojení (join) více tabulek a/nebo pohledů — c) true false

d) Je-li uvedena klauzule `WITH LOCAL CHECK OPTION`, provedení aktualizační operace bude umožněno jen tehdy, pokud změna bude v pohledu viditelná (např. u operace `INSERT` bude vkládaný řádek splňovat případné vyhledávací podmínky v definici pohledu) — d) true false

e) Klauzule `WITH CHECK OPTION` je nepovinná; je-li však uvedena a není-li vybrána její varianta, za výchozí je považována varianta `CASCADED` — e) true false

9. **Uspořádaný soubor** (sorted file) [5]

a) Záznamy v uspořádaném souboru jsou seřazeny podle vybraného vyhledávacího klíče; ten však musí být jednoduchý, nesmí se tedy skládat z více atributů (sloupců) — a) true false

b) Časová složitost (cena) sekvenčního průchodu uspořádaným souborem je rovna n, kde n je počet záznamů (řádků) v dané tabulce — b) true false

c) Časová složitost (cena) vyhledání prvního záznamu podle libovolné vyhledávací podmínky je při použití uspořádaného souboru vždy logaritmická — c) true false

d) Časová složitost (cena) vložení nového záznamu je konstantní — d) true false

e) V praxi se v uspořádaném souboru používají nejrůznější oblasti přetečení, soubor jako celek je čas od času reorganizován (přeuspořádán) — e) true false

10. **Konfliktní operace** v transakčním zpracování [5]

a) Existují čtyři varianty konfliktních operací: `read-read`, `write-read`, `read-write` a `write-write` — a) true false

b) Libovolně vybraná dvojice databázových operací v daném rozvrhu (historii) tvoří konfliktní pár právě tehdy, když první časově předchází druhou, alespoň jedna z nich je `write` a obě pracují se stejnou entitou — b) true false

c) Konfliktní operace typu `write-read` mohou způsobit problém označovaný jako neopakovatelné čtení (unrepeatable read) — c) true false

d) Dva rozvrhy (historie) jsou konfliktově ekvivalentní právě tehdy, když mají zcela stejné konfliktní operace — d) true false

e) Rozvrh (historie) je konfliktově uspořádatelný (serializovatelný) právě tehdy, když je konfliktově ekvivalentní nějakému sériovému rozvrhu — e) true false

11. Striktní dvoufázový zamykací protokol [5]

a) Pokud chce transakce provést operaci `write`(A), musí nejprve úspěšně získat (nebo už vlastnit) exkluzivní nebo alespoň sdílený zámek pro entitu A — a) true false

b) Všechny požadované zámky musí být u striktního dvoufázového zamykacího protokolu získány najednou hned na úplném začátku transakce — b) true false

c) Všechny získané zámky jsou uvolněny až na úplném konci transakce, tedy jako inherentní součást provádění operací `COMMIT` nebo `ABORT` — c) true false

d) Striktní dvoufázový zamykací protokol garantuje, že precendenční graf je acyklický, a tedy že rozvrh (historie) je konfliktově uspořádatelný (serializovatelný) — d) true false

e) Striktní dvoufázový zamykací protokol je schopen garantovat zotavitelnost rozvrhu (historie), ale už není schopen předcházet kaskádovému rušení transakcí — e) true false

12. Pojmenujte a definujte jednotlivé **normální formy**. Jaký je vzájemný vztah těchto normálních forem? Z jakého důvodu se v relačních databázích obvykle snažíme dosáhnout co nejvyšší normální formy? [5]

13. Vyjmenujte a vysvětlete jednotlivé **Coffmanovy podmínky**. S jakou problematikou souvisí a jakým způsobem je můžeme použít? [5]

14. Vysvětlete použití **prioritní fronty** v rámci algoritmu vnějšího třídění (N-way External Merge Sort), využití stránek systémové paměti a dopad na velikost běhů. [5]