1. **Vytvořte diagram ER konceptuálního schématu** pro databázi na správu rezervací ubytování: [**10**]

   Každý hotel je identifikován dvojicí obsahující název hotelu a název obce, ve které se vyskytuje. Hotel může obsahovat libovolné množství nabízených pokojů. Každý pokoj má své číslo, to je jednoznačné jen v rámci daného hotelu. Kromě toho má pokoj ještě informaci o jeho typu (např. jednolůžkový, dvoulůžkový s manželskou postelí atp.) a celkovou kapacitu. Registrovaní uživatelé mohou vytvářet rezervace ubytování, v rámci jedné rezervace vždy jen jednoho konkrétního pokoje. Uživatel je jednoznačně určen svým přihlašovacím jménem, dále u něj uchováváme přístupové heslo a alespoň jednu emailovou adresu. Rezervace ubytování má vygenerován unikátní kód, dále obsahuje informaci o datu začátku a datu konce ubytování a celkový počet nocí. Rozlišujeme přitom dva druhy rezervací, a to předběžné a potvrzené. Potvrzená rezervace obsahuje uhrazenou částku a datum provedení platby, naopak předběžná rezervace obsahuje datum, do kdy je nutné ji potvrdit. V systému dále potřebujeme uchovávat případná partnerství mezi dvojicemi hotelů.

   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.

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 seriálů:

   Herec (jméno, rok narození, národnost, věk)
   Seriál (název, žánr, jazyk)
   Hraje (id, seriál, číslo série, číslo epizody, jméno, rok narození, hodnocení)
   (seriál) ⊆ Seriál (název)
   (jméno, rok narození) ⊆ Herec (jméno, rok narození)

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

   a) Najděte unikátní jména herců mladších *20* let, kteří nikdy nehráli v žádném seriálu zařazeném do žánru *romantika* nebo *drama*.

   b) Najděte jména, roky narození a národnosti herců, kteří alespoň jednou hráli v nějakém seriálu v alespoň dvou různých bezprostředně následujících epizodách stejné série.

   c) Pro každého herce zjistěte celkový počet odehraných epizod v libovolných seriálech patřících do žánru *drama*. Omezte se však jen na herce s národností jinou než *česká*. Ve výsledku uveďte jen ty herce, kde takový počet je roven alespoň *50*.

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

```
<hraje serial="název seriálu" serie="číslo série" epizoda="číslo epizody">
  <jmeno>jméno herce</jmeno>
</hraje>
```

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

   Herec (jméno, rok narození, národnost, věk)
   Seriál (název, žánr, jazyk)
   Hraje (id, seriál, číslo série, číslo epizody, jméno, rok narození, hodnocení)
   (seriál) ⊆ Seriál (název)
   (jméno, rok narození) ⊆ Herec (jméno, rok narození)

   a) Názvy a žánry seriálů natočených v *angličtině*, ve kterých ani jednou nehrál herec *české* národnosti starší než *30* let.

   b) Jména, roky narození a národnosti herců jiné než *české* národnosti, kteří si alespoň jednou zahráli v každém seriálu natočeném v *češtině*.

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

Užitečné symboly: $\varnothing \neq \subseteq \cup \cap$

7. **Logické a NULL hodnoty** v jazyku SQL

   a) Jazyk SQL pracuje s tříhodnotovou logikou, kde kromě tradičních hodnot `TRUE` a `FALSE` máme ještě třetí logickou hodnotu `NULL` reprezentující neznámou logickou hodnotu

   b) Je-li hodnota výrazu p neznámá a současně výraz q je vyhodnocen na logickou pravdu, výraz p `OR` q bude bez ohledu na p vždy vyhodnocen na `TRUE`

   c) Chceme-li otestovat, jestli výraz e nabývá `NULL` hodoty, můžeme použít tradiční porovnávání $e =$ `NULL` stejně jako predikát e `IS NULL`

   d) Řádek se ve výsledku `SELECT` dotazu (bez agregací) objeví právě tehdy, když se podmínka ve `WHERE` klauzuli (je-li přítomna) vyhodnotí na `TRUE`

   e) Podmínka integritního omezení `CHECK` na neprázdné tabulce není splněna jen v tom případě, když je vyhodnocena na `FALSE`

8. **Databázové kurzory**

   a) Kurzor je řídicí strukturou, která nám umožňuje zpracování jednotlivých řádků výsledku `SELECT` dotazu

   b) Kurzor umožňuje procházení řádků jen v dopředném směru, ledaže je už při jeho vytváření použit modifikátor `SCROLL`

   c) K přístupu k jednotlivým řádkům se používá konstrukt `FETCH`, např. `FETCH ABSOLUTE 4 FROM` …, chceme-li přistoupit ke 4. řádku od aktuální pozice

   d) Pokud není nalezen požadovaný řádek, jsou navráceny `NULL` hodnoty

   e) Obousměrný kurzor sice lze používat k dopřednému i zpětnému přístupu k řádkům, vracet se však vždy lze nejvýše jen o počet řádků uvedený v parametru `WINDOW`

9. Vnitřní **organizace primárního souboru**

   a) Každý blok (stránka) primárního souboru je rozdělen na sloty, v nich se ukládají jednotlivé záznamy (řádky tabulky); není přípustné, aby jeden záznam byl uložen ve více blocích najednou (např. v jednom začíná, ve druhém končí)

   b) Každý záznam (řádek tabulky) je jednoznačně určen vnitřním identifikátorem *rid* (record id), který se skládá z čísla bloku (stránky) a čísla slotu

   c) Pokud schéma tabulky obsahuje sloupec typu `CHAR(10)`, je nezbytně nutné zvolit sloty proměnlivé velikosti

   d) Pokud se pro evidenci zaplněnosti slotů pro záznamy (řádky) pevné velikosti zvolí bitmapa, je nutné jednotlivé sloty vždy zaplňovat souvisle počínaje prvním

   e) Základem organizace bloku (stránky) pro záznamy proměnlivé velikosti je adresář, který pro každý slot obsahuje informaci o jeho offsetu a velikosti

10. **Uspořádatelnost** (serializovatelnost) v transakčním zpracování

    a) Sériový rozvrh (serial schedule) z definice nikdy nemůže obsahovat žádné konfliktní dvojice databázových operací

    b) Precedenční graf (precedence graph) je graf, jehož vrcholy tvoří úspěšně skončené transakce a orientované hrany reprezentují konfliktní dvojice operací (WR, RW, WW) mezi příslušnými transakcemi

    c) Rozvrh (historie) není konfliktově uspořádatelný (conflict serializable) právě tehdy, když obsahuje alespoň jeden orientovaný cyklus nad 2 či více transakcemi

    d) Konfliktová uspořádatelnost sama o sobě nemůže v dynamické databázi vyloučit vznik fantomů (phantoms)

    e) Existuje rozvrh (historie), který je uspořádatelný (serializovatelný), přestože není konfliktově uspořádatelný

11. **Problematika deadlocků** (uváznutí) v transakčním zpracování

    a) Deadlock je situace, kdy dvě nebo více transakcí každá vzájemně čeká jedna na druhou kvůli uvolnění požadovaných zámků

    b) Dvoufázový zamykací protokol vzniku deadlocků zabránit nedokáže, avšak striktní dvoufázový zamykací protokol už ano

    c) Vznik deadlocku je možné detekovat hledáním cyklů v grafu čekání (wait-for graph); vrcholy tohoto grafu tvoří všechny aktivní a úspěšně ukončené (commitované, tedy nikoli abortované) transakce

    d) V případě detekovaného deadlocku je nutné nějakou z dotčených transakcí zrušit (restartovat), např. tu, která odvedla nejméně práce, drží nejméně zámků atp.

    e) Předpokládejme strategii wound-wait při prevenci vzniku deadlocků a konkrétně situaci, kdy transakce T_1 požaduje zámek držený T_2: pokud má T_1 vyšší prioritu, dojde ke zrušení (restartu) T_2, v opačném případě T_1 počká

12. Vyjmenujte a stručně vysvětlete alespoň 6 vlastností a principů typických pro **NoSQL databáze**.

13. Vyjmenujte jednotlivé **ACID vlastnosti** a podrobně vysvětlete jejich význam.

14. Vysvětlete pojmy blokovacího a redukčního faktoru. Jak se stanoví počet záznamů, počet bloků a blokovací faktor pro výsledek operace kartézského součinu (**spojení křížem**) pro dvě tabulky? [**5**]