3. Předpokládejme následující relační schéma databáze letů:

Pilot (číslo licence, jméno, příjmení, věk, oddělení)
Letadlo (registrace, značka, typ, kapacita)
Let (id, pilot, registrace, datum a čas odletu, délka letu)
(registrace) ⊆ Letadlo (registrace)
(pilot) ⊆ Pilot (číslo licence)

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

a) Najděte unikátní příjmení a věk pilotů starších *50* let, kteří nikdy neletěli s letadlem typu *A380-800* ani *A380-900* značky *Airbus*.

b) Najděte čísla licencí a věk pilotů, kteří mají ve svém letovém plánu konflikt, tedy že v alespoň jednom okamžiku mají být na dvou či více letech najednou.

c) Pro každé oddělení zjistěte celkový počet letů realizovaných piloty daného oddělení pomocí letadel značky *Boeing*. Omezte se však jen na piloty s příjmením *Veselý*. Ve výsledku uveďte jen ta oddělení, kde je takový počet roven alespoň *1000*.

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

```
<let odlet="datum a čas odletu" delka="délka letu">
  <pilot>číslo licence pilota</pilot>
  <letadlo>typ letadla</letadlo>
</let>
```

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

Pilot (číslo licence, jméno, příjmení, věk, oddělení)
Letadlo (registrace, značka, typ, kapacita)
Let (id, pilot, registrace, datum a čas odletu, délka letu)
(registrace) ⊆ Letadlo (registrace)
(pilot) ⊆ Pilot (číslo licence)

a) Registrace a typy letadel značky *Airbus*, které nikdy nebyly využity na žádném letu s pilotem mladším *30* let a příjmením *Smutný*.

b) Jména a příjmení pilotů z oddělení *CA3*, kteří alespoň jednou letěli s každým typem letadla značky *Boeing*.

6. **Najděte všechny klíče** v relačním schématu s množinou atributů $\{X, Y, Z, U, W\}$ a množinou funkčních závislostí $\{XYZ \to U, Y \to W, W \to Z, U \to XZ, UW \to Y\}$. Postup komentujte, použijte algoritmus Lucchesi-Osborn. Následně **určete normální formu** každé jednotlivé závislosti a také celého uvedeného schématu. [**10**]

7. Dotazovací **predikáty** v jazyku SQL [5]

a) Podmínka `a BETWEEN x AND y` je ekvivalentní podmínce `(x < a) AND (a < y)`

b) Symbol `%` lze v šabloně predikátu `LIKE` použít jako zástupný symbol pro libovolný počet opakování jednoho libovolného znaku

c) Množinové porovnávání `value = ALL (query)`, kde `query` je poddotaz, lze ekvivalentně přepsat na `value IN (query)`

d) Množinové porovnávání `ANY` vrátí hodnotu `UNKNOWN`, pokud je příslušný poddotaz vyhodnocen na tabulku ubsahující jen řádky s `NULL` hodnotami

e) Predikát `EXISTS` je splněn (vyhodnocen na `TRUE`) i v případě, že příslušný poddotaz je vyhodnocen na prázdnou tabulku

8. Databázové kurzory [5]

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. B^+-stromový index [5]

a) B^+-strom je výškově vyvážený strom, časová složitost základních operací je logaritmická

b) Vnitřní uzly (včetně kořenového) mají jinou strukturu než uzly listové, ty totiž obsahují `rid` ukazatele na záznamy do primárního souboru

c) Vnitřní uzly obsahují řádově i stovky dělících hodnot, každý uzel je fyzicky uložen jako jeden blok (stránka) indexového souboru

d) B^+-stromový index může být konstruován jen pro vyhledávací klíče, které jsou unikátní

e) Uzly v listové úrovni B^+-stromu jsou navzájem provázány pomocí ukazatelů

10. Dvoufázový zamykací protokol [5]

a) Dvoufázový zamykací protokol pracuje ve dvou fázích, zamykací a odemykací, tyto fáze se mohou v případě potřeby překrývat

b) Pokud chce transakce provést operaci `read`(A), musí nejprve úspěšně získat (nebo už vlastnit) exkluzivní nebo alespoň sdílený zámek pro entitu A

c) Všechny požadované zámky musí být u dvoufázového zamykacího protokolu získány najednou hned na úplném začátku transakce

d) Dvoufázový zamykací protokol garantuje, že graf čekání (wait-for graph) je acyklický, a tedy že rozvrh (historie) je konfliktově uspořádatelný (serializovatelný)

e) Dvoufázový zamykací protokol je schopen garantovat zotavitelnost rozvrhu (historie), ale už není schopen předcházet kaskádovému rušení transakcí

11. **Problematika fantomů** (phantoms) v transakčním zpracování [5]

a) Fantom nemůže nastat ve statických databázích, tedy takových, kde se připouští jen operace `SELECT` a `UPDATE`, nikoli však `INSERT` nebo `DELETE`

b) Možnost vzniku fantomu nastává v situaci, kdy jedna transakce pracuje s množinou entit, zatímco druhá transakce tuto množinu mění

c) Zabránit vzniku fantomů je možné použitím vhodného indexového zamykání, např. v případě B^+-stromů je možné zamykat celé podstromy, a tedy intervaly

d) Nejsou-li k dispozici vhodné indexy, běžně se používají snadno implementovatelné predikátové zámky

e) Fantom způsobí, že precedenční graf pro daný rozvrh (historii) nebude acyklický

12. Co jsou **Armstrongovy axiomy** a k čemu se používají? Definujte tyto axiomy, uveďte jejich názvy a určete, které z nich jsou nezávislé a co to znamená. [5]

13. Jaké podmínky musí splňovat libovolně vybraná dvojice databázových operací vyskytujících se v daném rozvrhu (historii), abyste je mohli označit za **konfiktní operace**? Pojmenujte jednotlivé varianty takových konfliktních operací. [5]

14. Na jakých aspektech, informacích nebo strukturách závisí výběr vhodných algoritmů pro jednotlivé operace v rámci **vytváření plánu vyhodnocení** výrazu SQL dotazu? [5]