CS2102 Final Examination Cheat-sheet

Part 1 Relational Algebra

Relational algebra operators:

1. Unary operators: selection σ, projection ρ, renaming π;

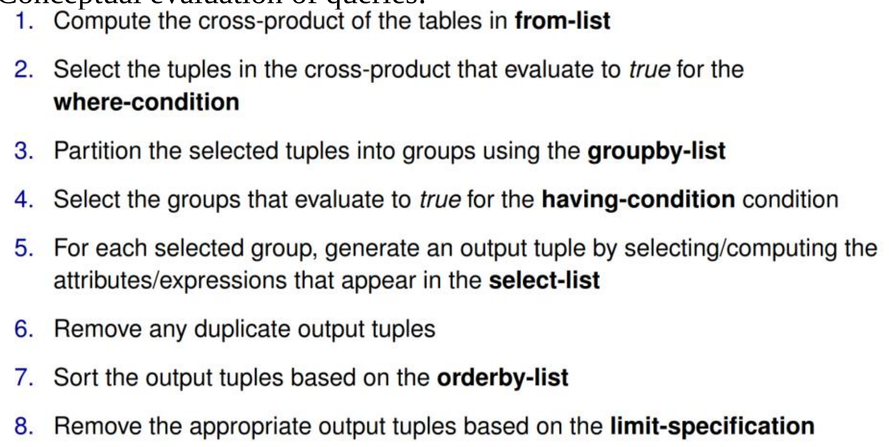
2) Binary operators: union ∪, intersect ∩, difference —, cross-product ×, natural join $\bowtie$, inner join $\bowtie_c$, left outer join $\rightarrow_c$, left semi-join $\ltimes_c$, left anti-join $\rhd_c$.

Part 2 SQL Language

1. SQL built-in data types: boolean, integer, numeric (3, 1), real, char (50), varchar (60), text, date, timestamp.
2. PostgreSQL specific data types: serial, with/without time-zone.
3. The domain of each data type includes a special value `NULL`.
4. SQL uses a three-valued logic system: *true*, *false*, *unknown*.
5. A <u>foreign key constraint</u> is: `A` references `B`, where `A` must be one set of `B`’s values or `NULL`, and `B` must be primary key or unique.
6. Schema constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`, `DEFAULT`.
7. Set operations: `UNION` [`ALL`], `INTERSECT` [`ALL`], `EXCEPT` [`ALL`].
8. Join operations: `NATURAL`, `INNER`, `LEFT/RIGHT/FULL OUTER`.
9. Comparison predicate: `IS (NOT) NULL`, `IS (NOT) DISTINCT FROM`.
10. Subquery expressions: `EXISTS`, `IN`, `ANY/SOME`, `ALL`, `UNIQUE`.
11. Aggregate functions: `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`.
12. Common table expression (CTE): `WITH XXX AS (…) SELECT …;`
13. Conditional expression: `CASE ??? WHEN … THEN … ELSE … END`
14. Like operator: `WHERE name LIKE ‘___%e’`, underscore matches any single character, % matches any sequence of 0 or more characters.
15. Others: `SELECT ‘price is ’ || round(price/1.3) …`
16. Order-by & limit: `ORDER BY … desc LIMIT 3` (ASC by default).
17. View: `CREATE VIEW … AS`.
18. `GROUP BY` can be used with `HAVING`. However, the `SELECT` / `HAVING` clause for `GROUP BY` can includes those: 1) appear in the `GROUP BY` clause; 2) an aggregate function; 3) `GROUP BY` includes a key.
19. To use aggregate functions without `GROUP BY` clause, then the select clause cannot contain anything except for aggregate functions.

Part 3 SQL Evaluation, Constraint & Transaction

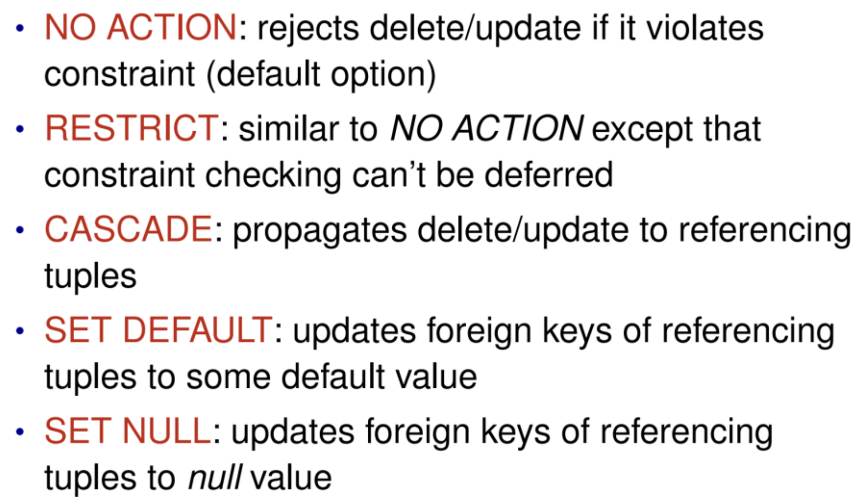
1. Conceptual evaluation of queries:



1. Compute the cross-product of the tables in **from-list**
2. Select the tuples in the cross-product that evaluate to *true* for the **where-condition**
3. Partition the selected tuples into groups using the **groupby-list**
4. Select the groups that evaluate to *true* for the **having-condition** condition
5. For each selected group, generate an output tuple by selecting/computing the attributes/expressions that appear in the **select-list**
6. Remove any duplicate output tuples
7. Sort the output tuples based on the **orderby-list**
8. Remove the appropriate output tuples based on the **limit-specification**

2. Constraints are by default checked immediately (at the end of statement level), but you can optionally defer it to the end of transaction level.

3. To handle foreign key constraint violations: `ON DELETE/UPDATE` can be appended with



- NO ACTION: rejects delete/update if it violates constraint (default option)
- RESTRICT: similar to *NO ACTION* except that constraint checking can’t be deferred
- CASCADE: propagates delete/update to referencing tuples
- SET DEFAULT: updates foreign keys of referencing tuples to some default value
- SET NULL: updates foreign keys of referencing tuples to *null* value

4. To achieve *<u>atomicity</u>*, each transaction begins with `BEGIN` and ends with `COMMIT/ROLLBACK`. All statements will be treated as one unit.

If one statement within the block fails, the whole transaction will be aborted. However, if we do not use the concept of transaction, let’s say one of the statements fail, the else will still be committed.

5. Serializable executions: we say an execution of S is serializable if it is equivalent (final database state is the same & every value read is the same) to some serial executions of S.
6. Four isolation levels: read uncommitted (*weakest*), read committed (*default mostly*), repeatable read, serializable (*strongest*).
7. Three undesired results: dirty read, non-repeatable read, phantom read.

Part 3 Functional Dependencies & Normalization

1. Armstrong's axioms
1) Reflexivity: $If\ Y \subseteq X, then\ X \to Y$;
2) Augmentation: $If\ X \to Y, then\ X \cup Z \to Y \cup Z$;
3) Transitivity: $If\ X \to Y \cap Y \to Z, then\ X \to Z$.
2. For two sets of functional dependencies F and G, 1 $F^+ = G^+ \Leftrightarrow F \equiv G$.
3. A desired decomposition needs to satisfy two standards:
1) Lossless: $(R_1 \cap R_2 \to R_1) \in F^+$ or $(R_1 \cap R_2 \to R_2) \in F^+$
2) Dependency preserving: A decomposition preserves all dependencies if and only if $F_1 \cup F_2 \cup \ldots \cup F_n \equiv F$ where $F_i = \{X \to Y \in F^+ | X, Y \subset R_i\}$ defined as the projected functional dependency on fragment R_i.
4. Three normal forms:

BCNF:

Trivial, or

X is a superkey for R

3NF:

Trivial, or

X is a superkey for R, or

A is part of some candidate key for R

2NF:

Trivial, or

X is not a proper subset of a candidate key for R, or

A is part of some candidate key for R

5. Two normalization strategies
1) BCNF decomposition algorithm: For each dependency $X \to Y$ that violates BCNF, recursively decompose $(S - R_i) \cup \{X^+, (R_i - X^+) \cup X^+\}$. The result is guaranteed to be lossless (*but may not preserve dependencies*).
2) 3NF synthesis algorithm: Let $S = \emptyset$, for each $X \to Y$ in the minimum cover of F, add a schema containing $X \cup Y$ to S if S does not have it. Add a schema containing any candidate key of F to S if S does not have it. The result is guaranteed to be lossless and dependency preserving.
6. To get the minimal cover of a set of functional dependencies:

1. Every right-hand side is a single attribute
2. For no functional dependency X→A in F and proper subset Z of X is F – {X→A} ∪ {Z→A} equivalent to F
3. For no functional dependency X →A in F is the set F – {X→A} equivalent to F

7. To get the closure of a set of attributes:

```
▪X^(0) := X
▪Repeat
    ▪X^(i+1) := X^(i) ∪ A , where A is the union of
    the sets Z of attributes such that there
    exist Y → Z in F, and Y ⊂ X^(i)
▪Until X^(i+1) := X^(i)
▪Return X^(i+1)
```

Part 4 Relational Calculus

1. There are two types of relational calculus: domain relational calculus (DRC) and tuple relational calculus (TRC). We will focus on latter now.
2. A TRC is in the form of $\{T|P\}$, where T is the only free variable.
3. The following two forms are commonly used:
1) Exists & and: $\exists X\ (p \wedge q)$;
2) Any & imply: $\forall X\ (p \Rightarrow q)$.
4. $A \Rightarrow B \equiv \neg A \vee B \equiv \neg(A \wedge \neg B)$
