Review for Final

Chapters 6 and 7

Chapter 6. SQL: Data Manipulation

- SQL is a nonprocedural language consisting of standard English words such as SELECT, INSERT, UPDATE, and DELETE that can be used by professionals and non-professionals alike.
- It is both the formal and de facto standard language for defining and manipulating relational databases.

Chapter 6. SELECT Statement

- **The SELECT statement is the most important statement in the language and is used to express a query.**
- **It combines the three fundamental relational algebra operations of *Selection*, *Projection*, and *Join*.**
- **Every SELECT statement produces a query result table consisting of one or more columns and zero or more rows.**

Chapter 6. SELECT Statement

- **SELECT Statement Format:**

```
SELECT [DISTINCT | ALL]
    {* | [columnExpression [AS newName]] [,...] }
FROM        TableName [alias] [, ...]
[WHERE      condition]
[GROUP BY   columnList]  [HAVING condition]
[ORDER BY   columnList]
```

Chapter 6. SELECT Statement

- **The SELECT clause identifies the columns and/or calculated data to appear in the result table.**
- **All column names that appear in the SELECT clause must have their corresponding tables or views listed in the FROM clause.**
- **The WHERE clause selects rows to be included in the result table by applying a search condition to the rows of the named table(s).**

Chapter 6. SELECT Statement

- **The ORDER BY clause allows the result table to be sorted on the values in one or more columns. Each column can be sorted in ascending or descending order. If specified, the ORDER BY clause must be the last clause in the SELECT statement.**
- **SQL supports five aggregate functions (COUNT, SUM, AVG, MIN, and MAX) that take an entire column as an argument and compute a single value as the result. It is illegal to mix aggregate functions with column names in a SELECT clause, unless the GROUP BY clause is used.**

Chapter 6. SELECT Statement

- **The GROUP BY clause allows summary information to be included in the result table. Rows that have the same value for one or more columns can be grouped together and treated as a unit for using the aggregate functions. In this case, the aggregate functions take each group as an argument and compute a single value for each group as the result.**
- **The HAVING clause acts as a WHERE clause for groups, restricting the groups that appear in the final result table. However, unlike the WHERE clause, the HAVING clause can include aggregate functions.**

Chapter 6. SUBSELECT

- **A *subselect* is a complete SELECT statement embedded in another query. A subselect may appear within the WHERE or HAVING clauses of an outer SELECT statement, where it is called a *subquery* or *nested query*.**
- **Conceptually, a subquery produces a temporary table whose contents can be accessed by the outer query. A subquery can be embedded in another subquery.**

Chapter 6. Subquery

- There are 3 types of subquery: *scalar*, *row*, and *table*.
- A *scalar subquery* returns a single column and a single row, i.e, a single value. In principle, a scalar subquery can be used whenever a single value is needed.
- A *row subquery* returns multiple columns, but only a single row. A row subquery can be used whenever a row value constructor is needed, typically in predicates.
- A *table subquery* returns one or more columns and multiple rows. A table subquery can be used whenever a table is needed; for example, as an operand for the IN predicate.

Chapter 6. Join

- **If the columns of the result table come from more than one table, a *join* must be used, by specifying more than one table in the FROM clause and typically including a WHERE clause to specify the join column(s).**
- **The ISO standard allows *Outer joins* to be defined. It also allows the set operations of *Union*, *Intersection*, and *Difference* to be used with the UNION, INTERSECT, and EXCEPT commands.**

Chapter 6. Other SQL DML Statements

- **Besides SELECT, the SQL DML includes:**
- **the INSERT statement to insert a single row of data into a named table or to insert an arbitrary number of rows from one or more other tables using a subselect;**
- **the UPDATE statement to update one or more values in a specified column or columns of a named table;**
- **the DELETE statement to delete one or more rows from a named table.**

Chapter 7. SQL: Data Definition

- **The ISO standard provides eight base data types: boolean, character, bit, exact numeric, approximate numeric, datetime, interval, and character/binary large objects.**
- **The SQL DDL statements allow database objects to be defined. The CREATE and DROP SCHEMA statements allow schemas to be created and destroyed; the CREATE, ALTER, and DROP TABLE statements allow tables to be created, modified, and destroyed; the CREATE and DROP INDEX statements allow indexes to be created and destroyed.**

Chapter 7. CREATE TABLE

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] [...,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)]
  [MATCH {PARTIAL | FULL}
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,...]}
 {[CHECK (searchCondition)] [,...] })
```

Chapter 7. SQL: Data Definition

- **The ISO SQL standard provides clauses in the CREATE and ALTER TABLE statements to define *integrity constraints* that handle required data, domain constraints, entity integrity, referential integrity, and general constraints.**
- ***Required data* can be specified using NOT NULL.**
- ***Domain constraints* can be specified using the CHECK clause or by defining domains using the CREATE DOMAIN statement.**

Chapter 7. SQL: Data Definition

- ***Primary keys*** **should be defined using the PRIMARY KEY clause and** ***alternate keys*** **using the combination of NOT NULL and UNIQUE.**
- ***Foreign keys*** **should be defined using the FOREIGN KEY clause and update and delete rules using the subclauses ON UPDATE and ON DELETE.**
- ***General constraints*** **can be defined using the CHECK and UNIQUE clauses. General constraints can also be created using the CREATE ASSERTION statement.**

Chapter 7. View

- **A *view* is a virtual table representing a subset of columns and/or rows and/or column expressions from one or more base tables or views.**
- **A view is created using the CREATE VIEW statement by specifying a *defining query*.**
- **CREATE VIEW Statement:**

```
CREATE VIEW ViewName [ (newColumnName [,...]) ]
AS subselect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Chapter 7. View

- **Views may not necessarily be a physically stored table but may be recreated each time it is referenced.**
- **Views can be used to simplify the structure of the database and make queries easier to write.**
- **They can also be used to protect certain columns and/or rows from unauthorized access. Not all views are updatable.**

Chapter 7. Two View Approaches

- ***View resolution*** **merges the query on a view with the definition of the view producing a query on the underlying base table(s). This process is performed each time the DBMS has to process a query on a view.**
- **An alternative approach, called *view materialization*, stores the view as a temporary table in the database when the view is first queried. So, queries based on the materialized view can be much faster than recomputing the view each time. One disadvantage with materialized views is maintaining the currency of the temporary table.**