Relational Database Model

- The relational model is a type of logical database model that was conceived by E.F. Codd in 1969
- The relational model is based on set theory and predicate logic
    - It is well formalized, so its behavior is predictable
- A relational database consists of tables (relations) that are linked together via the use of primary and foreign keys
    - A FOREIGN KEY in a table is a primary key from a different table that has been posted into the table to create a link between the two tables
- Relational database tables are made up of rows and columns
    - Rows are called the table extension or tuples
        - The ordering of rows in a table does not matter
    - Columns are called the table intension or schema
        - The ordering of columns in a table does not matter
        - All values in a column must conform to the same data format (e.g. date, text, currency, etc.)
    - Each cell in a database table (a row-column intersection) can contain only one value
        - no repeating groups are allowed
- Some principles of the relational model
    - Entity Integrity
        - A primary key in a table must not contain a null value
            - Guarantees uniqueness of entities and enables proper referencing of primary key values by foreign key values
    - Referential Integrity
        - A value for a foreign key in a table must either
            - Be null (blank)
            - Match exactly a value for the primary key in the table from which it was posted
    - One Fact, One Place
        - Fact = a pairing of a candidate key attribute value with another attribute value
            - Facts are found in the extensional data

Integrity

The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency.

The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is especially important in transaction-oriented database systems.

Conformity

The DBMS transforms entered data to conform to required data structures.

That is, the DBMS formats the physically retrieved data to make it conform to the user's logical expectations.

For example, imagine an enterprise database used by a multinational company. An end user in England would expect to enter data such as July 11, 2010, as "11/07/2010." In contrast, the same date would be entered in the United States as "07/11/2010." Regardless of the data presentation format, the DBMS must manage the date in the proper format for each country.

Advanced SQL programming

Different SQL JOINs

Before we continue with examples, we will list the types of JOIN you can use, and the differences between them.

JOIN:

Return rows when there is at least one match in both tables

SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name

LEFT JOIN:

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name

RIGHT JOIN:

The RIGHT JOIN keyword Return all rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name

FULL JOIN:

The FULL JOIN keyword return rows when there is a match in one of the tables.

SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name

Get data from 3 tables using where keyword

Get data from 3 tables using join keyword

Subqueries

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

- a subquery must be enclosed in the parenthesis.
- a subquery must be put in the right hand of the comparison operator, and
- a subquery cannot contain a ORDER-BY clause.
- a query can contain more than one sub-queries.

In general, there are three types subqueries:

- single-row subquery, where the subquery returns only one row.
- multiple-row subquery, where the subquery returns multiple rows,.and
- multiple column subquery, where the subquery returns multiple columns.

Single-row subqueries can only be used with single-row comparison operators, and multiple-row subqueries can be used only with multiple-row operators.

SELECT * FROM employee WHERE sal = (SELECT MIN(sal) FROM employee);

SELECT name, sal FROM employee WHERE sal > ANY (SELECT MIN(sal) FROM employee GROUP BY dept_no);

SELECT id, name, dept, sal FROM employee WHERE (dept, sal) IN (SELECT dept, MIN(sal) FROM employee GROUP BY dept)

Views

A view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view. In some NoSQL databases, views are the only way to query data.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table. Consequently, a view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.
- Views can join and simplify multiple tables into a single virtual table.
- Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc.) and presents the calculated results as part of the data.
- Views can hide the complexity of data. For example, a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table.
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.
- Depending on the SQL engine used, views can provide extra security.

Query Optimization

For any given query, there may be a number of different ways to execute it.

Each operation in the query (SELECT, JOIN, etc.) can be implemented using one or more different Access Routines.

For example, an access routine that employs an index to retrieve some rows would be more efficient that an access routine that performs a full table scan.

The goal of the query optimizer is to find a reasonably efficient strategy for executing the query (not quite what the name implies) using the access routines.

Optimization typically takes one of two forms: Heuristic Optimization or Cost Based Optimization

In Heuristic Optimization, the query execution is refined based on heuristic rules for reordering the individual operations.

With Cost Based Optimization, the overall cost of executing the query is systematically reduced by estimating the costs of executing several different execution plans.

Concurrency control and Transaction management

When multiple transactions are trying to access the same sharable resource, there could arise many problems if the access control is not done properly. There are some important mechanisms to which access control can be maintained. Earlier we talked about theoretical concepts like serializability, but the practical concept of this can be implemented by using Locks and Timestamps. Here we shall discuss some protocols where Locks and Timestamps can be used to provide an environment in which concurrent transactions can preserve their Consistency and Isolation properties.

Lock Based Protocol:

A lock is nothing but a mechanism that tells the DBMS whether a particular data item is being used by any transaction for read/write purpose. Since there are two types of operations, i.e. read and write, whose basic nature are different, the locks for read and write operation may behave differently.

The Two Phase Locking Protocol defines the rules of how to acquire the locks on a data item and how to release the locks.

The Two Phase Locking Protocol assumes that a transaction can only be in one of two phases. In growing phase a transaction can acquire all locks on all data items it needs. Once it starts releasing locks (shrinking phase), it cannot again lock any new data items.

Timestamp Ordering Protocol:

The timestamp ordering protocol ensures that any pair of conflicting read/write operations will be executed in their respective timestamp order. This is an alternative solution to using locks.

Database performance tuning

Database tuning is comprised of a group of activities used to optimize and regulate the performance of a database. It refers to configuration of the database files, the database management system (DBMS), as well as the hardware and operating system on which the database is hosted. The goal of database tuning is to maximize the application of system resources in an attempt to execute transactions as efficiently and quickly as possible. The large majority of DBMS are designed with efficiency in mind; however, it is possible to enhance a database's performance via custom settings and configurations.

- I/O tuning
- DBMS tuning
- Database maintenance

Distributed relational systems and Data Replication

Replication, or the copying of data in databases to multiple locations to support distributed applications, is an important new tool for businesses in building competitive service advantages. New replicator facilities from several vendors are making this technology much more useful and practical than it's been in the past.

Replication provides users with their own local copies of data. These local, updatable data copies can support increased localized processing, reduced network traffic, easy scalability and cheaper approaches for distributed, non-stop processing.

Security considerations

- broad range of information security controls to protect databases
- against compromises of their confidentiality, integrity and availability
- involves various types or categories of controls, such as technical, procedural/administrative and physical
- traditional measures include firewalls and network-based intrusion detection systems
- more critical as networks are increasingly opened to wider access, in particular access from the Internet
- system, program, function and data access controls, along with the associated user identification, authentication and rights management functions

Privileges:

The database administrator controls who has privileges to access or update DBMS objects. This person also controls who can create objects, and creators of the objects control who can access the objects.

Triggers

Triggers enforce security authorizations or business-specific security considerations. Triggers can be executed before an SQL statement is executed, after an SQL statement is executed, or for each row of an SQL statement. Also, triggers can be defined for DELETE, INSERT, and UPDATE statement execution.