Review questions

Saturday, February 10, 2024 1:06 PM

QUESTION 1
The main construct for representing data in the relational model is a relation. A relation consists of a relation schema and a relation instance. The relation instance is a table, and the relation schema describes the column heads for the table.
The schema specifies the relation's name, the name of each field (or column, or attribute), and the domain of each field. A domain is referred to in a relation schema by the domain name and has a set of associated values
An instance of a relation is a set of tuples, also called records, in which each tuple has the same number of fields as the relation schema. A relation instance can be thought of as a table in which each tuple is a row, and all rows have the same number of fields.
A relation schema specifies the domain of each field or column in the relation instance. These domain constraints in the schema specify an important condition that we want each instance of the relation to satisfy: The values that appear in a column must be drawn from the domain associated with that column. Thus, the domain of a field is essentially the type of that field
The degree, also called arity, of a relation is the number of fields. The car□dinality of a relation instance is the number of tuples in it

QUESTION 3
An integrity con□straint (IC) is a condition specified on a database schema and restricts the data that can be stored in an instance of the database. If a database instance satisfies all the integrity constraints specified on the database schema, it is a legal instance. A DBMS enforces integrity constraints, in that it permits only legal instances to be stored in the database.

A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple. A set of fields that uniquely identifies a tuple according to a key constraint is called a candidate key for the relation; we often abbreviate this to just key. In the case of the Students relation, the (set of fields containing just the) sid field is a candidate key
Two distinct tuples in a legal instance (an instance that satisfies all ICs, including the key constraint) cannot have identical values in all the fields of a key. 2. No subset of the set of fields in a key is a unique identifier for a tuple.
the key must identify tuples uniquely in all possible legal instances of the relation. By stating that {login, age} is a key, the user is declaring that two students may have the same login or age, but not both. Out of all the available candidate keys, a database designer can identify a primary key. Intuitively, a tuple can be referred to from elsewhere in the database by storing the values of its primary key fields.
In SQL, we can declare that a subset of the columns of a table constitute a key by using the UNIQUE constraint. At most one of these candidate keys can be declared to be a primary key, using the PRIMARY KEY constraint. (SQL does not require that such constraints be declared for a table.)

To ensure that only bona fide students can enroll in courses, any value that appears in the studid field of an instance of the Enrolled relation should also appear in the sid field of some tuple in the Students relation. The studid field of Enrolled is called a foreign key and refers to Students. The foreign key in the referencing relation (Enrolled, in our example) must match the primary key of the referenced relation (Students); that is, it must have the same number of columns and compatible data types, although the column names can be different. Finally, we note that a foreign key could refer to the same relation. For example, we could extend the Students relation with a column called partner and declare this column to be a foreign key referring to Students. Intuitively, every student could then have a partner, and the partner field contains the partner's sid. The observant reader will no doubt ask, "What if a student does not (yet) have a partner?" This situation is handled in SQL by using a special value called null. The use of null in a field of a tuple means that value in that field is either unknown or not applicable (e.g., we do not know the partner yet or there is no partner). The appearance of null in a foreign key field does not violate the foreign key constraint. However, null values are not allowed to appear in a primary key field (because the primary key fields are used to identify a tuple uniquely).The foreign key constraint states that every studid value in Enrolled must also appear in Students, that is, studid in Enrolled is a foreign key referencing Stu□dents. Specifically, every studid value in Enrolled must appear as the value in the primary key field, sid, of Students. Incidentally, the primary key constraint for Enrolled states that a student has exactly one grade for each course he or she is enrolled in. If we want to record more than one grade per student per course, we should change the primary key constraint.

Every potential IC violation is generally checked at the end of each SQL statement execution, although it can be deferred until the end of the transaction executing the statement,

QUESTION 4
What should we do if an Enrolled row is inserted, with a studid column value that does not appear in any row of the Students table? In this case, the INSERT command is simply rejected. 2. What should we do if a Students row is deleted? The options are: Delete all Enrolled rows that refer to the deleted Students row. Disallow the deletion of the Students row if an Enrolled row refers to it. Set the studid column to the sid of some (existing) 'default' student, for every Enrolled row that refers to the deleted Students row.
For every Enrolled row that refers to it, set the studid column to null. In our example, this option conflicts with the fact that studid is part of the primary key of Enrolled and therefore cannot be set to null. Therefore, we are limited to the first three options in our example, although this fourth option (setting the foreign key to null) is available in general. 3. What should we do if the primary key value of a Students row is updated? The options here are similar to the previous case. SQL allows us to choose any of the four options on DELETE and UPDATE. For example, we can specify that when a Students row is

deleted, all Enrolled rows that refer to it are to be deleted as well, but that when the sid column of a Students row is modified, this update is to be rejected if an Enrolled row refers to the modified Students row
The options are specified as part of the foreign key declaration. The default option is NO ACTION, which means that the action (DELETE or UPDATE) is to be rejected. Thus, the ON UPDATE clause in our example could be omitted, with the same effect. The CASCADE keyword says that, if a Students row is deleted, all Enrolled rows that refer to it are to be deleted as well. If the UPDATE clause specified CASCADE, and the sid column of a Students row is updated, this update is also carried out in each Enrolled row that refers to the updated Students row. If a Students row is deleted, we can switch the enrollment to a 'default' student by using ON DELETE SET DEFAULT. The default student is specified as part of the definition of the sid field in Enrolled; for example, sid CHAR(20) DEFAULT '53666'.

QUESTION 5
a program that runs against a database is called a transaction, and it can contain several statements (queries, inserts, updates, etc.) that access the database. If (the execution of) a statement in a transac□tion violates an integrity constraint, should the DBMS detect this right away or should all constraints be checked together just before the transaction com□pletes? By default, a constraint is checked at the end of every SQL statement that could lead to a violation, and if there is a violation, the statement is rejected.
Whenever a Students tuple is inserted, a check is made to see if the honors course is in the Courses relation, and whenever a Courses tuple is inserted, a check is made to see that the grader is in the Students relation. How are we to insert the very first course or student tuple? One cannot be inserted without the other. The only way to accomplish this insertion is to defer the constraint checking that would normally be carried out at the end of an INSERT statement. SQL allows a constraint to be in DEFERRED or IMMEDIATE mode.
A constraint in deferred mode is checked at commit time. In our example, the foreign key constraints on Boats and Sailors can both be declared to be in deferred mode. We can then insert a boat with a nonexistent sailor as the cap□tain (temporarily making the database inconsistent), insert the sailor (restoring consistency), then commit and check that both constraints are satisfied.

QUESTION 6
A relational database query (query, for short) is a question about the data, and the answer consists of a new relation containing the result.

QUESTION 7-whole section 2.5 for details(also can look up in answers pdf)

QUESTION 8
A view is a table whose rows are not explicitly stored in the database but are computed as needed from a view definition.The view B-Students has three fields called name, sid, and course with the same domains as the fields sname and sid in Students and cid in Enrolled. (If the optional arguments name, sid, and course are omitted from the CREATE VIEW statement, the column names sname, sid, and cid are inherited.) This view can be used just like a base table, or explicitly stored table, in defining new queries or views. Given the instances of Enrolled and Students shown in Figure 3.4, B-Students contains the tuples shown in Figure 3.18. Conceptually, whenever B-Students is used in a query, the view definition is first evaluated to obtain the corresponding instance of B-Students, then the rest of the query is evaluated treating B-Students like any other relation referred to in the query.

The physical schema for a relational database describes how the relations in the conceptual schema are stored, in terms of the file organizations and indexes used. The conceptual schema is the collection of schemas of the relations stored in the database. While some relations in the conceptual schema can also be exposed to applications, that is, be part of the external schema of the database, additional relations in the external schema can be defined using the view mechanism. The view mechanism thus provides the support for logical data independence in the relational model. That is, it can be used to define relations in the external schema that mask changes in the conceptual schema of the database from applications. For example, if the schema of a stored relation is changed, we can define a view with the old schema and applications that expect to see the old schema can now use this view. Views are also valuable in the context of security: We can define views that give a group of users access to just the information they are allowed to see.

The SQL-92 standard allows updates to be specified only on views that are defined on a single base table using just selection and projection, with no use of aggregate operations.3 Such views are called updatable views. This definition is oversimplified, but it captures the spirit of the restrictions. An update on such a restricted view can always be implemented by updating the underlying base table in an unambiguous way.
(In general, if the view did not include a key for the underlying table, several rows in the table could 'correspond' to a single row in the view. This would be the case, for example, if we used S.sname instead of S.sid in the definition of GoodStudents. A com□mand that affects a row in the view then affects all corresponding rows in the underlying table.) We can insert a GoodStudents row by inserting a row into Students, using null values in columns of Students that do not appear in GoodStudents (e.g., sname, login). Note that primary key columns are not allowed to contain null values. Therefore, if we attempt to insert rows through a view that does not contain the primary key of the underlying table, the insertions will be rejected. For example, if GoodStudents contained sname but not sid, we could not insert rows into Students through insertions to GoodStudents.
Updatable Views in SQL:1999 The new SQL standard has expanded the class of view definitions that are updatable, taking primary key constraints into account. In contrast to SQL-92, a view definition that contains more than one table in the FROM clause may be updatable under the new definition. Intuitively, we can update a field of a view if it is obtained from exactly one of the underlying tables, and the primary key of that table is included in the fields of the view. SQL:1999 distinguishes between views whose rows can be modified (updat□able views) and views into which new rows can be inserted (insertable□into views): Views defined using the SQL constructs UNION, INTERSECT, and EXCEPT (which

we discuss in Chapter 5) cannot be inserted into, even if they are updatable. Intuitively, updatability ensures that an updated tuple in the view can be traced to exactly one tuple in one of the tables used to define the view. The updatability property, however, may still not enable us to decide into which table to insert a new tuple.

An important observation is that an INSERT or UPDATE may change the underlying base table so that the resulting (i.e., inserted or modified) row is not in the view! For example, if we try to insert a row 51234, 2.8 into the view, this row can be (padded with null values in the other fields of Students and then) added to the underlying Students table, but it will not appear in the GoodStudents view because it does not satisfy the view condition gpa > 3.0. The SQL default action is to allow this insertion, but we can disallow it by adding the clause WITH CHECK OPTION to the definition of the view. In this case, only rows that will actually appear in the view are permissible insertions.

Views involving more than one base table can, in principle, be safely updated. The B-Students view we introduced at the beginning of this section is an example of such a view. Consider the instance of B-Students shown in Figure 3.18 (with, of course, the corresponding instances of Students and Enrolled as in Figure 3.4). To insert a tuple, say Dave, 50000, Reggae203 B-Students, we can simply insert a tuple Reggae203, B, 50000 into Enrolled since there is already a tuple for sid 50000 in Students. To insert John, 55000, Reggae203 , on the other hand, we have to insert Reggae203, B, 55000 into Enrolled and also insert 55000, John, null, null, null into Students. Observe how null values are used in fields of the inserted tuple whose value is not available. Fortunately, the view schema contains the primary key fields of both underlying base tables; otherwise, we would not be able to support insertions into this view. To delete a tuple from the view B-Students, we can simply delete the corresponding tuple from Enrolled

QUESTION 9

If we decide that we no longer need a base table and want to destroy it (i.e., delete all the rows and remove the table definition information), we can use the DROP TABLE command. For example, DROP TABLE Students RESTRICT destroys the Students table unless some view or integrity constraint refers to Students; if so, the command fails. If the keyword RESTRICT is replaced by CASCADE, Students is dropped and any referencing views or integrity constraints are (recursively) dropped as well; one of these two keywords must always be specified. A view can be dropped using the DROP VIEW command, which is just like DROP TABLE. ALTER TABLE modifies the structure of an existing table.