CS 1555

Lecture 7

**SQL – Data Definition Language (continued)**

Creating domains

- Domain is a schema component for defining datatype macros

- Basic datatype

- DEFAULT value

- CHECK (validity conditions)

- Examples:

- CREATE DOMAIN sectno\_dom AS SMALLINT;

- CREATE DOMAIN gpa\_dom DECIMAL (3,2) DEFAULT 0.00;

- CREATE DOMAIN ssn\_dom CHAR(11) CONSTRAINT ssn\_dom\_value CHECK(VALUE BETWEEN ‘000-00-0000’ AND ‘999-99-9999’);

Removing a domain

- DROP DOMAIN <dname> [RESTRICT | CASCADE];

- Restrict removes domain if it is not used

- Cascade removes domain and replaces all its uses to its underlying datatype

Example – Creating a constraint for a domain

- CREATE DOMAIN M\_Code AS CHAR (10) CHECK (value IN (‘CS’, ‘Film’, ‘History’));

- CREATE TABLE STUDENT (

<normal database stuff>

Major M\_Code,

Minor M\_Code,

CONSTRAINT STUDENT\_Major\_Minor CHECK (Major != Minor));

- Should specify multiple constraints separately

Example – CHECK Major in-line, instead of naming it as a constraint

- CREAT TABLE STUDENT (

<normal database stuff>

Major CHAR(10) CHECK (Major IN(‘CS’, ‘Film’, ‘History’)),

Minor CHAR(10) CHECK ((Minor IN (‘CS’, ‘Film’, ‘History’) AND (Major != Minor));

Constraint management

- To modify a constraint, drop it first then add a new one using ALTER TABLE

Table schema evolution

- ALTER command allows you to alter the domain of an attribute, or add/drop an attribute or constraint

- ALTER TABLE <table-name> ALTER [COLUMN]

- Domain change of an attribute

- ALTER TABLE Student ALTER QPA DECIMAL(4, 2);

- Warning: type narrowing is possible as in C/C++

- Set or drop the default value of an attribute

- ALTER TABLE Section ALTER COLUMN Head DROP DEFAULT;

Modifying a table schema

- ALTER TABLE <table-name> ADD [COLUMN];

- ALTER TABLE <table-name> DROP [COLUMN] –Option;

- Cascade or Restrict (default)

**Formal Query Languages: Relational Algebra**

Steps in processing a query

- SQL statement 🡪 Scan, parse validate 🡪 Intermediate form of query

- Intermediate form of query 🡪 Query optimizer 🡪 Execution plan

- Execution plan 🡪 Query code generator 🡪 Code to execute query

- Code to execute query 🡪 Runtime DM Processor 🡪 Results of running query

Relational algebra

- Operands on entire relations

- Operands and result are relation

- Set theory operations: union, intersection, difference, Cartesian product

- Specific relational operations: selection, projection, join, division

- Complete set of relational algebra operations: select, project, product, union, difference

Selection

- Denoted by Greek letter sigma

- Unary operator: takes in one table and outputs one table

- Selection condition: any logical expression on attributes of r involving any comparison operator

- Will return a relation with the exact same schema (attributes)

- How can I get a copy of relation S? – Select all that are true from S, returns full table

- How can I get an empty copy of relation S? – Select all that are false from S, just returns table schema (attributes)

Projection

- Denoted by Greek letter pi

- Unary operator

- Takes in an attribute list of a table and filters

- Project (SID, Major) of S with attributes(SID, Name, Major): will return a relation with a table schema of only SID and Major

- If another row is added to the original relation, a projected tuple will be added to the result of the projection

- Notably, set theory requires tuples to be unique. If a projected tuple is a duplicate (because it doesn’t have all the attributes of the original relation), it will not be added to the projected relation

Relational Algebra Expressions

- Query: List the QPA of all students (SID) in CSD whose QPA is greater than 2.5

- STUDENT(SID, Fname, Sname, Dept, Major, QPA)

Three notations for previous example

- Nesting the operations: Projection (SID, QPA) of ((Selection (Dept = ‘CSD’ and QPA > 3.5) STUDENT)

- Sequence of operations: set result of selection operator to temporary table, then do projection on that temporary table

- Query tree: leaf nodes are relations and internal nodes are operations

- Can rename attributes of the result

Properties of sigma and pi

- Sigmacond1(sigmacond2(R)) = sigmacond2(sigmacond1(R))

- Sigmacond1(sigmacond2(R)) = sigmacond1 and cond2(R)

- pilist1(pilist2(R)) = pilist1(R) if list 1 is a subset of list 2

- All this helps with performance – “short-circuiting”

Efficient/optimized queries

- Reduce cost of computing (time complexity): short-circuit, execute faster comparisons first

- Reduce memory needs (space complexity): execute selections with high selectivity first to reduce size of intermediate tables, execute projections as early as possible to reduce tuple size

Selectivity

- Ratio of number of records that satisfy condition to total number of records