CSC343 **Summer 2003** **August 5, 2003**

Tutorial 10 Course Review

[1] Introduction

. To study the DBMS, we divide topics into the following three categories: database design, database programming and database system implementation.

. Database design: how to construct a useful database based on application requirements.

. Database programming: how to write queries to access data in the database.

. Database system implementation: how to store data in the second/tertiary storage, how to access data efficiently, how to execute queries, how to control transaction processing, etc.

. In this course, we studied the first two topics: database design and database programming, and will review in detail below. Database system implementation is the content of the following course CSC443.

[2] Database Design

[2.1] General Process

Application --> E/R Design --> Relational Schemas --> Normalized Relational Schemas
--> Relational Database

[2.2] Application Description

. We start the design from application description, which is provided by users.

. Understand and analyze the requirements of the application.

[2.3] E/R Design

. Recognize each entity set and its attributes

-- An entity set is a collection of entities and is represented by oval.

. Recognize relationships between entity sets and the multiplicity (many-one, many-many, one-one)

-- A collection of relationships is called a relationship set which is visualized

as a table.
-- A relationship is represented by a diamond.
-- A relationship can have attributes and multiple roles.

. Determine whether weak entity sets, Is-A relationship

-- A weak entity set has some or all of keys from other entity sets, and is represented by double rectangle.
-- A Is-A relationship is a special one-one relationship and is represented by a triangle.

. Identify keys and other constraints

-- A key is one or more attributes uniquely identifying an entity.
-- Other constraints may include referential integrity, domain constraints, etc.

. Draw diagram

[2.4] From E/R to Relational Models

. Entity sets --> relations
. E/R relationships --> relations
. If R is many-one from E to F, combine R and E.
. Deal with weak entity sets and Is-a relationship.
. Discover keys for each relation.

[2.5] Relational Schemas

. Decide data types for each attribute.

. Determine functional dependencies for each relation (possibly recheck the application description).

. Related concepts

-- Terminologies in the relational model

relation, tuple, attribute, domain, primary key, foreign key, superkey, cardinality, degree, instance, relation schema, relational database schema.

-- Functional dependency

X->Y means whenever two tuples agree on X, they also agree on Y.

-- Rules of FD's (splitting and combining, Armstrong's axioms, others).

-- Given FD's, find all implied FD's by calculating the closure of attributes.

-- Closure and its calculation

-- Find keys

[2.6] Normalization

. Decompose each relation into 3NF or higher

. Related concepts

-- Lossless join and its testing

Let {R1,R1,...,Rn} be a decomposition of R and F be a set of FD's over R, the decomposition has a lossless join w.r.t F, iff for *every* instance r of R satisfying F,

r = PI_R1 r Join ... Join PI_Rn r.

-- Project FD's into decomposed relations

-- Dependency preserving decompositions and its testing

If R is a schema, F is a set of FD's of R, {R1,...,Rk} is a decomposition of R, then {R1,...,Rk} preserves F iff

F+ = (PI_R1 F union ... union PI_Rk F)+

-- A decomposition with a lossless join doesn't necessarily preserve dependencies.

-- A decomposition that preserves dependencies doesn't necessarily have a lossless join.

-- Our goal is to find a decomposition which has a lossless join, dependency preserving, and no redundancy. But we cannot achieve all of them.

-- BCNF can have a lossless join, no redundancy, but not necessary dependency preserving.

-- 3NF can have a lossless join, preserves dependencies, but may have redundancy.

-- BCNF and decomposition algorithm

A relation R is in BCNF iff whenever there is a nontrivial FD X->Y for R, it is the case that X is a superkey.

-- Minimal cover of FD's and its calculation

-- 3NF and decomposition algorithms

A relation R is in 3NF iff whenever there is a nontrivial FD X->Y for R,
it is the case that X is a superkey or Y is contained in some key.

Up to now, we have finished the database design based on the application description. But to use the database, we should choose a DBMS, create relations and import data into the database.

[2.7] Relational Database Creation

. In this course, we choose DB2 as our DBMS. We talked about how to use DB2 in CDF in system-level mode and interactive mode.

. Use DDL to manipulate relations (DBA does)

-- Create, list and delete relations

. Use DML to manipulate data (users, programmers do)

-- Insert, list, update and delete records.
-- Import/export data from/to ASCII files.

. Run script

% db2 -f imput.file output.file -l history.file

[3] Database Programming

[3.1] General Process

User/Programmers --> SQL Queries --> Relational Algebra --> .....
--> Relational Database

. In order to access the database, users/programmers send SQL queries to the DBMS.

. Expressions in SQL queries are then converted to the relational algebra.

. In this course, we focus on how to write SQL and relational algebra queries. How to convert from SQL to algebra will be discussed in CSC443.

. We talked two ways of the SQL programming. One is stand-alone. The other is to embed

the SQL into other conventional languages, such as C, so called embedded SQL programming.

[3.2] Relational Algebra

. Relation algebra is a procedural language which specifies how to get query results.

. A basic expression of the relation algebra consists of either a relation in the database or a constant.

. Fundamental operations include selection, projection, union, set difference, cartesian product and rename.

. Additional operations are defined in terms of the fundamental operations. They don't add the power of the algebra, but are useful to simplify common queries. They include set intersection, theta-join, natural join, division.

. We only consider relational operations on sets, i.e., no duplicated tuples.

[3.3] SQL

. SQL Clause General Form

(1) SELECT [DISTINCT] fields
(2) FROM tables
(3) [WHERE predicate]
(4) [GROUP BY fields]
(5) [HAVING predicate]
(6) [ORDER BY fields] ASC/DESC

-- Clauses must be in the order of (1),(2),(3),(4),(5),(6).
-- Only the first two are required.
-- Usually, we cannot use HAVING clause without a GROUP BY clause.

. Select

-- Select specific list of columns from a table.
-- Use an asterisk (*) to select all the columns from the table.
-- Select keeps duplicates. Use the DISTINCT option to eliminate duplicated rows.
-- Use the options AS clause to assign a meaningful output attribute name.

. From

-- List tables to join
-- Use range variables to refer to a table name

-- Use nested table expression (subqueries)

. Where

-- Select specific rows from a table.
-- Specify a condition consisting of one or more predicates.
-- Use a subquery or correlated subquery. Usually, a correlated subquery will be evaluated many times, once for each assignment of a value to some term in the subquery that comes from a tuple variable outside the subquery.
-- Comparison operators include: =, <>, <, >, <=, >=
-- Use boolean operators: AND, OR, NOT.
-- Use IS NULL and IS NOT NULL to check null values.
-- Use s LIKE P and s NOT LIKE P to check whether a string is in a pattern.
-- Use s in R, s > ALL R, s > ANY R to compare a scalar value with a one-column relation.

. Group BY

-- Group rows according to the group.
-- The column names in the SELECT clause must be either grouping columns or aggregate operations (AVG, COUNT, MAX, MIN, SUM).

. Aggregation Operations

-- SUM, AVG, MAX, MIN, COUNT
-- Always produce a scalar value

. Set Operators

-- UNION, INTERSECT, EXCEPT
-- No duplicates in default.
-- Use UNION ALL, INTERSECT ALL, EXCEPT ALL to keep duplicates.

. HAVING

-- Apply a qualifying condition to groups.
-- An aggregation in a HAVING clause applies only to the tuples of the group being tested.
-- Any attribute of relations in the FROM clause may be aggregated in the HAVING clause. But only those attributes that are in the GROUP BY list may appear unaggregated in the HAVING clause.

. ORDER BY

-- Sort the rows by the values in one or more columns.
-- The column names in the ORDER BY clause do not have to be specified in the select

list.
-- Use ASC (default) or DESC to order rows in ascending or descending order.

[3.4] Embedded SQL Programming

. Any embedded SQL statements are preceded by the keywords EXEC SQL.
. All program variables used in SQL statements must be enclosed in an SQL declaration section and preceded by a colon.
. To retrieve multiple tuples, we need to declare a cursor, open it, use it to fetch one tuple at a time, and close it.
. Use SQLCODE to check error or warning messages for each important operation.
. To embed SQL in C, we create .sqc file, precompile to get .c file, and compile/link to get executable file.