Data and Its Applications (CS4.301)

Monsoon 2021, IIIT Hyderabad
20 September, Monday (Lecture 9)

Taught by Prof. Kamal Karlapalem

Relational Data Model

Relational Algebra (contd.)

The divide operation is another binary operation on relations. If $R(Z) \div S(X) = T(Y)$, then $Y = Z - X$, and T contains all those rows of R which are connected (by equality of some attribute) to *all* rows of S.
The relational algebra expression for this is
$T_1 = \pi_Y(R)$
$T_2 = \pi_Y((S \times T_1) - R)$ [$S \times T_1$ gives the relation covering all tuples of S relating to all tuples of T_1; subtracting T_1 from this gives all the rows which are *not* supposed to be in the answer]
$T = T_1 - T_2$.

The set of operations $\{\pi, \sigma, \cup, -, \times\}$ is called a complete set of relational algebra operations. Any query language equivalent to this set is called relationally complete.

In addition to these, database applications include aggregate functions like `SUM`, `COUNT`, `AVERAGE`, `MIN`, `MAX`. If $T = {}_X R_Y$, then the tuples of R are grouped according to the attribute X and then the function(s) Y is/are carried out on each group.

The outer join operation is similar to the equijoin, but it include *all* rows of one of the relations. In the case of the left outer join, all rows of R are included, and NULLs are included wherever needed. Correspondingly, we have a right outer join and a full outer join.

Relational Database Design

Relational theory helps us formally compare different schemata. It deals with the design of relational databases.

Some basic guidelines are:

- clear semantics for attributes (schemata's meanings should be easy to explain)
- reduce redundant values in tuples
- reduce null values in tuples (nulls should be reserved for exceptions)
- disallow spurious tuples (relations should have the lossless join property)

Functional Dependencies

The key idea in relational database design is that of functional dependencies. Consider a relational schema $R(A_1, \dots, A_n)$, and let $X, Y \subseteq \{A_1, \dots, A_n\}$. Then we say that there is a dependency $X \to Y$ if, for all tuples t_1, t_2, $t_1[X] = t_2[X] \implies t_1[Y] \implies t_2[Y]$. We say that X determines Y.
Note that if X is unique for all tuples, then it trivially determines all other attributes of R.

FDs ae used to specify constraints, test relation states to see if they are legal, and to improve the schema by removing undesirable dependencies.

Let F be the set of FDs for R. The set of all FDs that hold on all instances satisfying F is called the closure $F * $ of F.
We use $F \vDash X \to Y$ to denote that the dependency $X \to Y$ is inferred from F.

Some inference rules for functional dependencies are:

- reflexivity: if $Y \subseteq X$, then $X \to Y$
- augmentation: if $X \to Y$, then $XZ \to YZ$.
- transitivity: if $X \to Y$ and $Y \to Z$, then $X \to Z$.

These three inference rules are sound (anything we infer from F using them will hold when F holds) and complete (anything that holds when F holds can be inferred from F using them).