Note of Lecture 3

Yinda Zhang

January 2021

1 Preliminaries

In this section, we show some preliminary definitions and notations.

Domain (dom): A set of all possible values.

Relation (R): Some subset of values from the domain of all possible values. We use $R[A_1, ...A_k]$ to denote a relation over attributes A_1 through A_k. R links all of these different attributes together.

$$R \subseteq dom(A_1) \times ... \times dom(A_k)$$

Table (r): One particular subset in R. We use $r[A_1, ...A_k]$ to denote a table. Sometimes it is called an instance in database theory.

2 Relational Algebra

In this section, we show some key operations in relational algebra.

Selection (σ): It takes a subset of rows from a table. Specifically, selection applies some predicate P to each row of a table and takes the rows where P is true. For example, $\sigma(Grade = C+)$ represents that we select the rows where the grade is equal to $C+$.

Projection (Π): It takes a subset of columns from a table. For example, $\Pi(Name, Course)$ represents that we project this table to name and course.

Cartesian product ($\times$): Cartesian product combines multiple tables together. Let R and S be two tables. In $R \times S$, the Cartesian product combines all pairs of rows in R and S.

Join ($\bowtie$): In fact, for the Cartesian product, typically we don't combine arbitrary unrelated tables together. Suppose that two tables R and S share a particular column (course). We may want a new table T, which is a combination of R and S on the course column, which we call the join operation. We can regard this particular operation of combining R and S together and merging them on the course column as a combination of two operations. Specifically, we should first take the product of R and S and then filter out those pairs where there is a matching between courses ($\sigma(R \times S)_{R.course=S.course}$). In the course, we may often focus on the equality joins, which means that the combination of two tables is on some equality conditions.

3 Normal Forms

3.1 Update Anomaly

As shown in Table 1, suppose that there is a table with five column, orderId, item, quantity, user name, and email address. We can find that there is a dependency between user name and email address, *i.e.*, the same users always have the same email addresses. However, the same users might buy multiple products. If we need to change one user's email address. We have to iterate through this entire table and change their email address, which we call an update anomaly.

| orderId | item | quantity | user name | email address |

Table 1: Example of Update Anomaly

An update anomaly is a situation where you want to make one change (*e.g.*, the link between user and email address), but you end up having to change more rows. It leads to what we call normal forms, which are different levels of decomposition of a table to remove different types of update anomalies.

3.2 1NF

The first normal form (1NF) is the basic requirement on a lack of redundancy in your table. The first normal form stipulates that

- All rows are unique.

It indicates that there's a lack of redundancy at a row level.

In terms of information theory, we can represent it by

- $H(R) = \log |R|$.

Remind that $\log |R|$ is the maximum possible entropy, so 1NF indicates that at a row level, the table has the maximum possible entropy. Another way of saying 1NF is saying that no tuple level compression is possible.

3.3 2NF

The second normal form (2NF) requires that

- The table is already in 1NF.
- Each row is uniquely referenced by a single primary key column.

It means that in each table there is a column like student ID. A student ID uniquely references each student in a database.

In terms of information theory, we can represent it by

- $\forall$ table, $\exists\ X_i\ s.t.\ H(X_i) = \log |R|$.

3.4 3NF

The third normal form (3NF) stipulates that

- The table is already in 2NF.
- There are no transitive functional dependencies.

Transitive functional dependencies: First, we have known that the functional dependencies is that one column uniquely determines the other column, which means that knowing this column tells you everything you need to know about the other column. Table 1 is an example of transitive functional dependency. The orderId is the primary key, and it can determine every other column. We can find that user name can also determine the email address. It is the transitive functional dependency, because the user name is not a primary key.

In terms of information theory, we can represent it by

- For all pairs of attributes A and B, $H(B|A) = 0 \rightarrow H(A) = \log |R|$.

BCNF: It turns out that certain types of still anomalies can arise when there are multiple keys. BCNF (Boyce-Codd Normal Form) is an extension of 3NF to avoid such cases.

3.5 4NF

We explain 4NF by an example shown in Table 2.

Course	Book	Lecturer
AHA	Silberschatz	John D
AHA	Nederpelt	John D
AHA	Silberschatz	William M
AHA	Nederpelt	William M
AHA	Silberschatz	Christian G
AHA	Nederpelt	Christian G
OSO	Silberschatz	John D
OSO	Silberschatz	William M

Table 2: Example of 4NF

This is a table of university courses, the books recommended for the course, and the lecturers who will be teaching the course. Since there are multiple recommended books for some courses there is redundancy. However, this redundancy is not a functional dependency.

This particular structure is called a multivalued dependency. We can find that, in terms of information theory, $H(Book|Course, Lecture) = H(Book)$. Basically we're saying that book is independent of the knowledge of both course and lecturer. If you know both the course and lecture, there's no new information about the distribution of books.

3.6 Summary

1NF	no duplicates	no row level reduction
2NF	unique keys	no key level reduction
3NF/BCNF	no FDS	no functional dependencies reduction
4+ NF	no MVD	all variables that are linked have some dependencies

Table 3: Summary of normal forms

4 Armstrong's Axioms

In this section, we show the Armstrong's Axioms.

Transitivity: $x \to y, y \to z$, then $x \to z$

It means that if x determines y and y determines z, then x determines z.

Augmentation: $x \to y$, then $x, A \to y, A$

It means that if x determines y, then x, A determines y, A.

Reflexibility: $x \to$ any subset of x,

Prove transitivity using conditional entropy: Based on the $x \to y$ and $y \to z$, we can get that $H(y|x) = H(x,y) - H(x) = 0$ and $H(z|y) = H(y,z) - H(y) = 0$. Therefore, $H(z|x) = H(x,z) - H(x) = H(x,z) - H(x,y) = H(x,z|x,y) = H(z|y) = 0$.