Functional dependencies and normalization I. SEXY AF

1 Two approaches to database design

- Top down approach: Entity-Relationship (ER) model
  - A graphical description of the DB
  - Start with a set of requirements
  - Identify the entities that you need to represent data about
  - Identify the attributes of those entities
- At the next step we construct the Relational data model
- Bottom up approach
  - Start with initial tables and attributes
  - analyse the relationships among the attributes
  - Re-design the tables and attributes in a ”better” way
  - This becomes tricky for large databases
  - We need a formalization of this approach

2 Well designed relational databases

- No redundancy: every data item is stored once
  - e.g. keep the address of the customer only in one place
  - exception of foreign keys (they act as pointers)

  1. Minimise the amount of space required
  2. Simplify maintenance of the database
- If an item is stored twice (or more), then:
  - every time we update it, we need to change it in many places
  - we may have inconsistencies (two different values for the same item)
- Purpose of normalisation
  - every relation represents a ”real world” entity
  - single valued columns
  - avoid redundancy
  - data is easy to update correctly

3 Redundancy

- **Set valued** attributes in the ER diagram: result in multiple rows in the corresponding table
- **Dependencies** between attributes cause redundancy

4 Data anomalies: terminology

Redundancy: repeating data in multiple different locations
Modification anomaly: failure to maintain all existing instances of a specific value
Deletion anomaly: losing other values as a side effect when you delete data
Insertion anomaly: when new data items are inserted, we need to add must more irrelevant data

5 Decomposition

- **Schema refinement (decomposition)**: uses two (or more) relations to store the original relation
- No update anomalies

6 Normalization theory

- The result of ER modelling: needs further refinement to reduce data redundancy
- **Decomposition** of the relations (tables)
  - this can be done manually for a small DB
  - For a larger DB we need a formalization of the approach
- Functional dependencies among data items:
  - Strongly affect the data anomalies
  - The fundamentals of the underlying **normalization theory**
  - Specify which are the **candidate/primary/foreign keys** (**entity integrity** and **referential integrity**)
  - Specify which attributes combine in the new tables

7 Relational keys

- **Candidate key**: a minimal (not minimum) set of attributes whose values uniquely identify the tuples
- **Primary key**: The candidate key selected to identify rows uniquely with the table
- **Alternate key**: Those candidate key(s) not selected as primary key
- **Simple key**: The key consists of one attribute
- **Composite key**: The key consists of several attributes

8 Functional data dependencies

- Functional data dependency
  - Describes the relationship among attributes in the same relation
  - Let A and B be two sets of attributes; we say that "B is **functionally dependent** on A" (denotes $A \rightarrow B$) if each value of A is associated with exactly one value of B
- Informally: if we know the attribute values of the set A then we know the (unique) values for the set B
- The attribute values of B can be determined by
  1. Calculation
  2. Lookup
- In a functional data dependency ($A \rightarrow B$)
  - **determinant**: the set of all attributes on the left hand side (i.e. A)
  - **dependant**: the set of all attributes on the right hand side (i.e. B)
- **Full** functional dependency $A \rightarrow B$:
  - B is functionally dependent on A
  - B is not functionally dependent on any proper subset of A
- **Partial** functional dependency $A \rightarrow B$:
  - B is functionally dependent on A
  - B remains functionally dependent on at least one proper subset of A

- **Transitive** functional dependency:
  - If there exist functional dependencies $A \to B$ and $B \to C$
  - Then the functional dependency $A \to C$ also exists and is called **transitive**
- By the definition of relational keys
  - A candidate key is a minimal set of attributes which functionally determine all attributes in a relation
- How can we determine all functional dependencies?
  - Some of them are obvious from the semantics
  - some others follow from specification/discussions with customers
- Let F be a set of functional dependencies
- The closure of F (denotes F^+) is the set of all functional dependencies that are implied by the dependencies in F

9 The closure of a set F of dependencies

- To compute the closure F^+ of F we need a set of inference rules
- Armstrong's axioms
  1. Reflexivity: if $B \subseteq A$, then $A \to B$
  2. Augmentation: if $A \to B$ then $A, C \to B, C$
  3. Transitivity: if $A \to B$ and $B \to C$ then $A \to C$
- These inference rules are complete
  - Given a set X of functional dependencies, all dependencies implied by X can be derived from X using these rules
- And sound
  - no additional functional dependencies (which are not implied by X) can be derived using these rules
- These properties can be proved by definition of a functional dependency
- Further rules can be derived from Armstrong's axioms:
  - Decomposition: if $A \to B, C$ then A, b and A, C
  - Union: if $A \to B$ and $A \to C$ then $A \to B, C$
  - Composition: if $A \to B$ and $C \to D$ then $A, c \to B, D$
- For example, proof of the Union rule using the axioms:
  - $A \to B$, augmentation with $C \Rightarrow A, C \to B, C$
  - $A \to C$ augmentation with $A \Rightarrow A, A \to A, C$ (i.e. $A \to A, C$)
  - Transitivity from the last two dependencies: $A \to A, C \to B, C$
- To compute the closure F^+ of F
  - Apply repeatedly Armstrong's axioms (or the above 3 extra rules) to get the closure of F

10 Functional data dependencies

- Let F be a set of transitive dependencies
- Let A be a set of attributes in a relation
- The closure of A under F (denoted A^+):
  - The set of all attributes that can be implied by the attributes of A using functional dependencies from F
- We can compute the closure A^+ (under F)
  - Similarly to the closure F^+
  - Start with the functional dependencies of F, which have attributes A on the left hand side
  - Repeatedly apply Armstrong's axioms
- By the definition of relational keys:
  - A candidate key is a minimal set of attributes such that A^+ (under F^+) includes all attributes in a relation