Contents

Databse Design

- Need of a formal method for analysing how the relations and attributes are grouped
- A measure of appropriateness or foodness other than the intuition of the design
  - To asses the quality of the design
- Measures
  - Design guidelines
  - Functional Dependencies
  - Normalisation

Design Guidelines

- A set of informal guidelines
  - Can be used as measures to determine the quality of a relation schema design
    * Attribute Semantics
    * Reduction of Redundancy
    * Reduction of NULLs
    * Generation of Spurious Tuples
- These measures are not always independent of one another

Attributes Semantics

- Attributes belonging to a relation have certain real-world meaning

- Semantics of a relation
  - Refers to its meaning resulting from the interpretation of attribute values in a tuple
- Careful entity relationship modeling and accurate mapping to logical dsign help to ensure that a relational schema design has clear meaning

Guideline 1

- Design a relation schema so that is it easy to explain its meaning
- Give relations and attributes meaningful names
- Do not combine attributes from multiple entity types and relationship types into a single relation
  - Straightforward to interpret
  - Easy to explain its meaning

Reduction of Redundancy

- One goal of database schema design is to minimise the storage space used
- Grouping attributes into relation schemas has a significant effect on storage space
- Storing merged entities in singple relations leads to another problem, *update anomalies*
- Update anomalies can be classified into
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Insertion Anomalies

- To insert a new employee into EMP_DEPT, it is necessary to include either
  - All attributes values for the department that the employee works for
  - NULLs, if the employee is not yet assigned
- Consistency becomes an issue
- Inserting a new department is difficult

Deletion Anomalies

- Deletion of Employees and Departments inextricably linked
  - If we delete the last employee currently assigned to a particular department, the information related to department is lost from the database
- This problem does not occur is using separate relations

Modification Anomalies

- Modification akes consistency an issue
- If the manager of a department is changed
  - It is necessary to update the typles of every employee who works for that department
  - Records can easly get out of sync
- This problem does not occur is using separate relations

Guideline 2

- Design the relation schemas so that no insertion, deletion of modification anomalies are present
  - If anomalies are present, note them clearly and ensure all application programs operate correctly
- This second guideline is consistent with guideline 1

Reduction of NULLs

- If many attributes do not apply to all the typles of a relation, you end up with many NULL values in those tuples
  - Waste storage space
  - Can make understanding attribute meanings more difficult
  - Leads to difficultly with joins
  - Difficulty with aggregate functions
    - COUNT and SUM
- A NULL value may typically have two interpretations
  - Missing but inapplicable
    - Post Code for Irish Addresses (outside of Dublin)

  - Missing by applicable
    * An employees date of birth is empty
      · Uknown
      · Known but absent

Guideline 3

- Avoiding placing attributes in a relation schema whose values may frequencly be NULL
  - If NULLs are unavoidable, ensure they apply in exceptional cases and not the majority of typles
- Using space efficiently and avoiding joins on NULL values are the main criteria for deciding upon attribute inclusion or exclusion
  - If excluded, create a separate relation for that attribute

Violating

- If only 15% of employees have an office, the including an Office Number attribute in the EMPLOYEE relation would violate guideline 3
  - Instead, create an EMPLOYEE_OFFICE relation
  - This could contain the attributes
    * Ssn, Office Number
  - A typle is entered in the relation for all employees with an office

Generation of Spurious Tuples

- Joins across relations should only be performed on *Primary Key - Foreign Key* pairs of attributes
- If joins are performed on attributes which are not a Primary Key - Foreign Key pairing, spurious tuples are generated as a result
  - These tuples represent information which is not valid

Guideline 4

- Design relation schemas so that they can be joined using equality conditions on primary key, foreign key pairs
  - This guarantees that no spurious tuples are generated by the join
- Avoid relations that contain matching attributes that are not foreign key, primary key combination

Design Guidelines

- Informal measures used to determine the quality of a relational schema design
  - Ensure that attribute semantics are easily understood
  - Reduce the redundant information in tuples
  - Reduce the number of NULL values in tuples
  - Ensure that spurious typles are not generated by enforcing primary key, foreign key matching

Database Optimisation

- Need of a formal method for analysing how the relations and attributes are grouped
- A measure of appropriateness or goodness other than the intuition of the designer
  - To access the quality of the design
- Measures
  - Design guidelines
  - Functional dependencies
  - Normalisation

Functional Dependencies

- Formal tool for analysis of relational schemas
  - Enables the designer to detect and describe design problems in more precise terms
- One of the most important concepts in relational schema design theory
- Main tool for measuring the appropriateness of grouping of attributes into relations

Definition

- A *functional dependency* is a constraint between two sets of attributes
- Suppose our relational database schema has n attributes
  - A_1, A_2, ..., A_n
- Think of the whole database as being described by a single universal relation

  - $R = \{A_1, A_2, ..., A_n\}$
- A functional dependency denoted by $X \rightarrow Y$, specifies a constraint on the possible tuples that can form a relation state r of R
  - Between two sets of attributes X and Y
  - X and Y are subsets of the relation R
- The constraint is
  - For any two tuples t_1 and t_2 in r(R) that have $t_1[X] = t_2[X]$
- The values of the attributes set X from a tuple in r, uniquely (of *functionally*) determine the values of the attribute set Y
  - We can say that:
    * There is a *function dependency* from X to Y
  - or
    * Y is *functionally dependent* on X
- The abbreviation for functional dependency if FD
  - The set of attributes X is called the left-hand side of the FD, Y is called the right-hand side
- Thus
  - X functionally determines Y in a relation schema R if, and only if, whenever two tuples agree on their X values, they must necessarily agree on their Y values

Things to Note

- If X is a candidate key of R, then
  - $X \rightarrow Y$ for any subset of attributes Y or F
  - This, $X \rightarrow R$
  - In other words, if X has to be unique for every instance of R, then X uniquely determines all the other attribute values of R
- If $X \rightarrow Y$ in R, this does not necessarily imply that $Y \rightarrow X$ in R
  - Not commutative

Identification of FDs

- A *functional dependency* is a property of the semantics or meaning of the attributes

  - A database designer will use their understanding of the semantics of the attributes of R to specify the functional dependencies that must hold on all instances of R
    * Entity relationship modelling supports the development of this understanding

Example

- Consider

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

- Using the semantics of the attributes and relation, the following FDs should hold:
  - Ssn → Ename
  - Pnumber → {Pname, Plocation}
  - {Ssn, Pnumber} → Hours

Disproving a FD

- You cannot use a single set of data - r(R) - to prove a FD, but you can use it to disprove a FB

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Moffman
Brown	Data Structures	Martram

- Canot prove Text → Course
- Can disprove Teacher → Course

Constraints

- Whenever the semantics of two sets of attributes of R indicate that a FD should hold, the dependency is specified as a *constraint*
- Hence, FDs are used to further enhance a relation schema R, by specifying constraints that must hold *at all times*

Normalisation

- The normalisation process takes a relation schema through a series of tests to certify whether is satisfies a certain *normal form*
- There are a number of normal forms:
  - First Normal Form
  - Second Normal Form
  - Third Normal Form
  - Boyce-Codd Normal Form
- Evaluates each relation against the criteria for normal forms
  - Decompose relations where necessary
- Can be considered *relational design by analysis*
  - ER Modelling
  - Mapping to Relational Schema
  - Functional Dependencies
  - Normalisation
- The process of analysing relation schemas based upon their primary keys and functional dependencies in order to:
  - Minimise redundancy
  - Minimise insertion, deletion, and modification anomalies
- Relations which do not pass the normal form tests are decomposed into smaller relation schemas
- Normalisation through decomposition must confirm two properties in the resulting database design
  - Non-Additive or Lossless Join Property
    - This guarantees that spurious tuple generation does not occur
  - Dependency Preservation Property
    - This ensures that each functional dependency is represented in an individual relation
- Provides database designers with:
  - A formal framework for analysing relations based upon their primary keys and functional dependencies

- A set of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalised to the desired degree

First Normal Form

- In 1NF all attribute values must be atomic
  - The word atom comes from the Latin atomis, meaning indivisible (or literally, "not to cut")
- 1NF dictates that at every row-column intersection, there exists only one value, not a list of values
- The benefits from this rule should be fairly obvious
  - If lists of values are stored in a single column, there is no simple way to manipulate those values

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Achieving 1FN DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATION

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire

Dnumber	Dlocation
5	Sugarland
5	Houston

- 1NF also disallows multi-valued attributes that are themselves composite
- Remove the attributes that violates 1NF and put it into a separate, new relation
- Add the primary key of the original relation to the new relation
  - This will serve as a foreign key
- The primary key of the new relation is a composite primary key
- This *decomposes* a non-1NF relation into two 1NF relations

Second Normal Form

- A table is said to be in Second Normal Form if:
  - It is 1NF compliant
  - Every non-key column if *fully functionally dependent* ont he entire primary key
- In other works
  - Tables should only store data relating to one "thing" (or entity)
  - That entity should eb described by its primary key

Full Functional Dependency

- A FD X $\rightarrow$ Y is said to be a *full functional dependency* if the removal of any single attribute from the set of attributes X means that the dependency no longer holds
  - Think of X as a composite primary key
  - All the other attributes must be dependent upon the full key, not just part of it
- For an attribute A ε X
  - (X - {A}) does not functionally determine Y

Partial Functional Dependency

- A FD X $\rightarrow$ Y is said to be a *partial functional dependency* is a single attribute can be removed from the set of attributes X yet the dependency still holds
- For any attribute A ε X
  - (X - {A}) $\rightarrow$ Y

Achieving 2NF

- Limit the FDs to only the parts of the key that they are dependent upon
- Decompose the relation into separate relations using these FDs
- The primary key of the new relations is the left-hand side of the FD
- This *decomposes* a non-2NF relation into a set of new 2NF relations

Third Normal Form

- A table is said to be in Third Normal Form if:
  - It is 2NF compliant
  - No non-key attributes are *transitively dependent* upon the primary key
- A function dependency X $\rightarrow$ Y in the relation R, is said to be a *transitive dependency* if:
  - There exists a set of attributes Z in R which is neither a candidate key or a subset of any key of R
  - And both X $\rightarrow$ Z and Z $\rightarrow$ Y hold true

Transitive Dependency

- Ssn $\rightarrow$ Dmgr_ssn is a transitive dependency through Dnumber
  - Ssn $\rightarrow$ Dnumber and Dnumber $\rightarrow$ Dmgr_ssn hold
  - Dnumber is not a key itself or a subset of any key of EMP_DEPT

Achieving 3NF

- Identify any transitive dependencies in the relation
- Decompose the relation into two separate relations using these transitive dependencies
- The primary key of the new relation is the middle attribute of the transitive dependency
- This *decomposes* a non-3NF relation into two new 3NF relations

2NF and 3NF

- Any functional dependency in which the left hand side is…
  - a non-key attribute or
  - a component attribute of a coposite primary key

…is a problematic FD

- 2NF and 3NF remove these problem FDs by decomposing them into new relations

Superkey

- A *superkey* SK is any set of attributes in the relation R, whose combined values will be unique for every tuple
  - $t_1[SK] \neq t_2[SK]$
- Every relation has one default superkey - the key of all its attributes
  - As, by definition, every instance of a relation must be unique

Boyne-Codd Normal Form

- BCNF was created to be a simpler form of 3NF
  - Sometimes called 3.5NF
- However, it was found to be stricter than 3NF
  - Every relation in BCNF is also in 3NF
  - Every relation in 3NF is not necessarily in BCNF
- A table is said to be in Boyce-Codd Normal Form if:
  - Whenevr a functional dependency $X \rightarrow Y$ holds in the relation R, X is a superkey of R

Achieving BCNF

- Identify all functional dependencies in the relations
  - Identify any FDs where the left-hand side is not a superkey
- Decompose the relation into separate relations, creating a new relation for the offending FD
- The primary key of the new relation is the left hand attribute of the offending functional dependency
- This *decomposes* a non-BCNF relation into two new BCNF relations

Normalisation

- Normalisation tests a relation schema to certify whether it satisfies a *normal form*
  - 1NF
  - 2NF
  - 3NF
  - BCNF
- Evaluate each relation against the criteria for each normal form in turn
  - Decompose relations where necessary

Modelling a Database

- Identify and model
  - The required entities and attributes
  - The relationships between those entities
- Map from the conceptual model to a relational schema
- Identify the functional dependencies in the relation schemas
- Normaise the relation schemas