Functional Dependencies – Part 1

Introduction

Database Design Quality

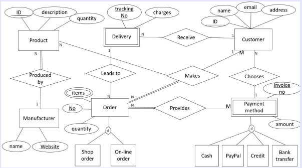
- A fundamental question in database design:

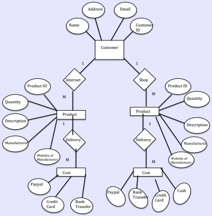
  What constitutes a "well-designed" database schema?

- We have learnt that:
  - A database design often starts with building an EER model.
  - An EER model can then be translated to a relational database schema.

- However, such an EER model may not be "*perfect*". Instead, it is common to have many different EER models for the same application.

Database Design Quality - Examples [1]





[1] Previous COMP2400/6240 students' solutions for an EER modelling question

Database Design Quality

- Some desirable properties of a “well-designed” database schema
  - **Completeness**
    Has all relevant information been captured?
  - **Redundancy freeness**
    Has the doubling of relevant information been avoided (if possible)?
  - **Consistent understanding**
    Is the meaning of all relevant information consistent?
    Is the meaning of NULL clear?
    - Does not apply
    - Unknown
    - Known but absent
  - **Performance**
    Can the database schema lead to the good performance for given tasks?

Motivating Example

- Suppose that we want to store the enrolment information (i.e., *course no*, *semester* and *unit*) of students (i.e., *name*, *student id* and *date of birth*) in a relational database.

- **Is the design of the relation ENROLMENT good?**

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1989	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123456	11/09/1987	COMP2400	2009 S2	8

Motivating Example – Data Inconsistency

- **Any inconsistency problems with these tuples?**

Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1989	COMP8740	2011 S2	12

  The same student has different DoBs. *This seems unreasonable.*

Michael	123458	21/04/1985	COMP2400	2009 S2	6
Fran	123456	11/09/1987	COMP2400	2009 S2	8

  There are different units for the same course in the same semester. *That should not happen.*

Tom	123456	25/01/1989	COMP8740	2011 S2	12
Fran	123456	11/09/1987	COMP2400	2009 S2	8

  The different students have the same ID. *This is unacceptable.*

Motivating Example – Data Redundancy

- **Any redundancy problems with these tuples?**

Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12

    There exists redundant information about students.

Tom	123456	25/01/1989	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP8740	2011 S2	12

    There exists redundant information about courses.

Motivating Example – Update Anomalies

- **What could happen to update operations (e.g., insert, delete and update)?**

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1988	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123456	11/09/1987	COMP2400	2009 S2	6

- **Modification anomalies**: If changing the DoB of Michael, then ...
- **Insertion anomalies**: If inserting a new course COMP3000, then ...
- **Deletion anomalies**: If deleting the enrolled course COMP2400 of Fran, then ...

Database Design Issues

- We have see the following database design issues so far:
  - Data inconsistency
  - Data redundancy
  - Update anomalies

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1989	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123456	11/09/1987	COMP2400	2009 S2	8

- **Can we avoid these issues when designing a database?**

Database Design Issues - Motivating Example

- We may fix those database design issues through breaking a relation into smaller relations.

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1988	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123457	11/09/1987	COMP2400	2009 S2	6

- For example, each tuple in ENROLMENT represents **three** different facts:
  1. Information about students
  2. Information about courses
  3. Course enrolment of students

Database Design Issues - Motivating Example

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1988	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123457	11/09/1987	COMP2400	2009 S2	6

⇓

STUDENT		
Name	StudentID	DoB
Tom	123456	25/01/1988
Michael	123458	21/04/1985
Fran	123457	11/09/1987

COURSE	
CourseNo	Unit
COMP2400	6
COMP8740	12

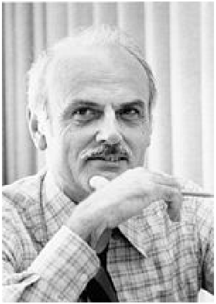
ENROL		
StudentID	CourseNo	Semester
123456	COMP2400	2010 S2
123456	COMP8740	2011 S2
123458	COMP2400	2009 S2
123458	COMP8740	2011 S2
123457	COMP2400	2009 S2

Functional Dependencies – Part 2

Definition and Identification

Codd and Functional Dependencies

- **Functional dependencies** (FDs) were introduced by Codd in 1971 [1]
- Edgar F. Codd of IBM Research (1923-2003) invented the **relational data model** for data management in 1970.
- He received the ACM Turing Award in 1981 for his contributions on the theoretical foundations of relational databases:
  - **Functional dependencies**
  - **Normalization**
    – Boyce–Codd Normal Form (BCNF)
  - **Query languages**
    – Relational Calculus
    – Relational Algebra



[1] *Further Normalization of the Data Base Relational Model*. E. F. Codd, IBM Research Report, San Jose, California, 1971.

Why Functional Dependencies?

- We need some **formal way** of analysing whether a database schema is well-designed, or why one is better than another.

- FDs are developed to define the **goodness** and **badness** of (relational) database design in a formal way.

  - **Top down**: start with a relation schema and FDs, and produce smaller relation schemas in certain normal form (called *normalisation*).

  - **Bottom up**: start with attributes and FDs, and produce relation schemas (*not popular in practice*).

FDs tell us "relationship between and among attributes"!

Functional Dependencies – Informal Description

- We have two FDs on ENROLMENT:

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1988	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123457	11/09/1987	COMP2400	2009 S2	6

  - StudentID **functionally determines** Name and DoB, i.e.,

    $\{\text{StudentID}\} \rightarrow \{\text{Name, DoB}\}$

  - CourseNo **functionally determines** Unit, i.e.,

    $\{\text{CourseNo}\} \rightarrow \{\text{Unit}\}$

Functional Dependencies – Informal Description

- A **FD** says that, within a relation, the values of some attributes determine the values of other attributes.

Animal	$\rightarrow$	Legs
Ostrich		2
Wombat		4

- If attributes A, B, C determine attributes D, E, then we write

$$\{A, B, C\} \rightarrow \{D, E\}$$

  - This means, if two tuples have the same values for A, B and C, then they must also have the same values for D and E.
  - A, B and C are the **determinant**, while D and E are the **dependent**.

Formal Definition

- Let R be a relation schema.
  - A **FD** on R is an expression $X \to Y$ with attribute sets $X, Y \subseteq R$.
  - A relation $r(R)$ **satisfies** $X \to Y$ **on** R if, for any two tuples $t_1, t_2 \in r(R)$, whenever the tuples t_1 and t_2 coincide on values of X, they also coincide on values of Y.

$$\begin{array}{c} t_1[X] = t_2[X] \\ \Downarrow \\ t_1[Y] = t_2[Y] \end{array}$$

- A FD is **trivial** if it can *always* be satisfied, e.g.,
  - $\{A, B, C\} \to \{C\}$
  - $\{A, B, C\} \to \{A, B\}$
- **Syntactical convention**: (1) Instead of $\{A, B, C\}$, we may use ABC. (2) $A, B, \ldots$ for individual attributes and $X, Y, \ldots$ for sets of attributes.

Exercise - Functional Dependencies on Relations

- Consider the following relations with attributes {A,B,C,D,E}. Do they satisfy: (1) $AB \to E$; (2) $C \to DE$;

$r_1(R)$				
A	**B**	**C**	**D**	**E**
1	4	1	9	4
1	4	2	8	9
1	4	3	8	9

$r_2(R)$				
A	**B**	**C**	**D**	**E**
1	3	1	3	8
1	3	2	4	8
1	2	2	4	9

- **Check:**

	$r_1(R)$	$r_2(R)$
(1) $AB \to E$	no	yes
(2) $C \to DE$	yes	no

How to Identify FDs in General?

- A functional dependency specifies a constraint on the relation schema that must hold **at all times**.
- In real-life applications, we often use the following approaches:

  (1) **Analyse data requirements**
  Can be provided in the form of discussion with application users and/or data requirement specifications.

  (2) **Analyse sample data**
  Useful when application users are unavailable for consultation and/or the document is incomplete.

(1) Identifying FDs - Analyse Data Requirements

- Consider the following relation schema:

$$\text{RENTAL}=\{\text{CustID},\text{CustName},\text{PropertyNo},\text{DateStart},\text{Owner}\}.$$

- **Data requirements:**
  1. Each customer can be uniquely identified by his or her customer ID.

     $\{\text{CustID}\} \rightarrow \{\text{CustName}\}$
  2. A customer cannot rent two or more properties from the same date.

     $\{\text{CustID, DateStart}\} \rightarrow \{\text{PropertyNo}\}$
  3. A customer cannot rent the same property more than once.

     $\{\text{PropertyNo, CustID}\} \rightarrow \{\text{DateStart}\}$
  4. Each property can be uniquely identified by its owner.

     $\{\text{Owner}\} \rightarrow \{\text{PropertyNo}\}$

(2) Identifying FDs - Analyse Sample Data

- Can you find some FDs on ENROLMENT based on the sample data?

ENROLMENT					
Name	StudentID	DoB	CourseNo	Semester	Unit
Tom	123456	25/01/1988	COMP2400	2010 S2	6
Tom	123456	25/01/1988	COMP8740	2011 S2	12
Michael	123458	21/04/1985	COMP2400	2009 S2	6
Michael	123458	21/04/1985	COMP8740	2011 S2	12
Fran	123457	11/09/1987	COMP2400	2009 S2	6

- We may have:
  - {StudentID} $\rightarrow$ {Name, DoB};
  - {CourseNo} $\rightarrow$ {Unit};
  - {StudentID, CourseNo, Semester} $\rightarrow$ {Name, DoB, Unit};
  - {Name} $\rightarrow$ {StudentID} $\times$;
  - {DoB} $\rightarrow$ {StudentID} $\times$;
  - ......

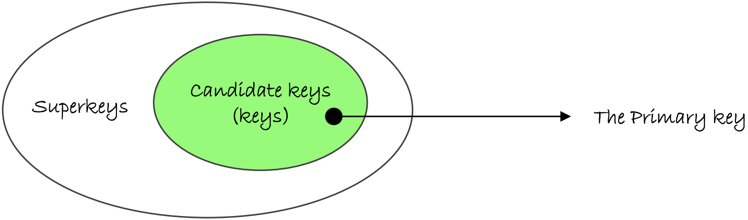
Limitations: Sample data needs to be a true representation of **all possible values** that the database may hold.

Functional Dependencies – Part 3

Finding Keys

A Bunch of Keys

- We will need keys for defining the normal forms later on.
  - A *subset of the attributes* of a relation schema R is a **superkey** if it uniquely determines all attributes of R.
  - A superkey K is called a **candidate key** if no proper subset of K is a superkey.
    - That is, if you take any of the attributes out of K, then there is not enough to uniquely identify tuples.
  - **Candidate keys** are also called **keys**, and the **primary key** is chosen from them.



Finding Keys

- Given a set Σ of FDs on a relation R, the question is:

How can we find all the (candidate) keys of R?

Australian National University

Implied Functional Dependencies

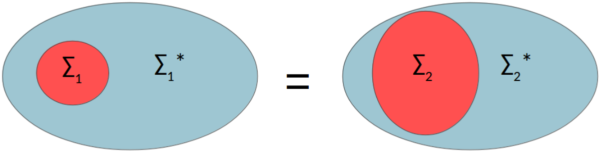
- To design a good database, we need to consider **all possible FDs**.
- If each student works on one project and each project has one supervisor, does each student have one project supervisor?

$$\begin{array}{c}\{\{\text{StudentID}\}\rightarrow\{\text{ProjectNo}\},\\ \{\text{ProjectNo}\}\rightarrow\{\text{Supervisor}\}\}\end{array} \quad\models\quad \{\text{StudentID}\}\rightarrow\{\text{Supervisor}\}$$

- We use the notation $\Sigma \models X \rightarrow Y$ to denote that $X \rightarrow Y$ is **implied** by the set Σ of FDs.
- We write Σ^* for all possible FDs **implied** by Σ.

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.

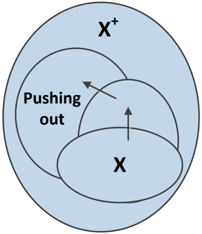


- **Example:** Let $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z\}$ and $\Sigma_2 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$.
  We have $\Sigma_1 \neq \Sigma_2$ but $\Sigma_1^* = \Sigma_2^* = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$.
  Hence, Σ_1 and Σ_2 are equivalent.

- **Questions:**
  1. Is it possible that $\Sigma_1^* = \Sigma_2^*$ but $\Sigma_1 \neq \Sigma_2$? **Yes**
  2. Is it possible that $\Sigma_1^* \neq \Sigma_2^*$ but $\Sigma_1 = \Sigma_2$? **No**

Implied Functional Dependencies

- Let Σ be a set of FDs. Check whether or not $\Sigma \models X \rightarrow W$ holds? We need to
  1. Compute **the set of all attributes** that are dependent on X, which is called the **closure** of X under Σ and is denoted by X^+.
  2. $\Sigma \models X \rightarrow W$ holds iff $W \subseteq X^+$.
- **Algorithm**[1]
  - $X^+ := X$;
  - repeat until no more change on X^+
    - for each $Y \rightarrow Z \in \Sigma$ with $Y \subseteq X^+$, add all the attributes in Z to X^+, i.e., replace X^+ by $X^+ \cup Z$.



[1] See Algorithm 15.1 on Page 538 in [Elmasri & Navathe, 7th edition] or Algorithm 1 on Page 555 in [Elmasri & Navathe, 6th edition]

Implied Functional Dependencies – Example

- Consider a relation schema $R = \{A, B, C, D, E, F\}$, a set of FDs $\Sigma = \{AC \rightarrow B, B \rightarrow CD, C \rightarrow E, AF \rightarrow B\}$ on R.
- Decide whether or not $\Sigma \models AC \rightarrow ED$ holds.

  1. We first build the closure of AC:

$$\begin{array}{lll} (AC)^+ & \supseteq AC & \text{initialisation} \\ & \supseteq ACB & \text{using } AC \rightarrow B \\ & \supseteq ACBD & \text{using } B \rightarrow CD \\ & \supseteq ACBDE & \text{using } C \rightarrow E \\ & = ACBDE & \end{array}$$

  2. Then we check that $ED \subseteq (AC)^+$. Hence $\Sigma \models AC \rightarrow ED$.

- **Can you quickly tell whether or not $\Sigma \models AC \rightarrow EF$ holds?**

Finding Keys

- **Fact**: A key K of R always defines a FD $K \rightarrow R$.
- **Algorithm**[2]:

  Input: a set Σ of FDs on R.

  Output: the set of all keys of R.

  - for every subset X of the relation R, compute its closure X^+
  - if $X^+ = R$, then X is a superkey.
  - if no proper subset Y of X with $Y^+ = R$, then X is a key.
- A **prime attribute** is an attribute occurring in a key, and a **non-prime attribute** is an attribute that is not a prime attribute.

[2] It extends Algorithm 15.2(a) in [Elmasri & Navathe, 7th edition, pp. 542], or Algorithm 2(a) or in Algorithm 2(a) in [Elmasri & Navathe, 6th edition pp. 558] to finding all keys of R

Exercise – Finding Keys

- Consider a relation schema $R = \{A, B, C, D\}$ and a set of functional dependencies $\Sigma = \{AB \rightarrow C, AC \rightarrow D\}$.
  1. List all the keys and superkeys of R.
  2. Find all the prime attributes of R.
- **Solution:**
  1. We compute the closures for all possible combinations of the attributes in R:
     - $(A)^+ = A$, $(B)^+ = B$, $(C)^+ = C$, $(D)^+ = D$;
     - $(AB)^+ = ABCD$, $(AC)^+ = ACD$, $(AD)^+ = AD$, $(BC)^+ = BC$, $(BD)^+ = BD$, $(CD)^+ = CD$
     - $(ABC)^+ = ABCD$, $(ABD)^+ = ABCD$, $(ACD)^+ = ACD$, $(BCD)^+ = BCD$
  2. Hence, we have
     - AB is the only key of R.
     - AB, ABC, ABD and $ABCD$ are the superkeys of R.
     - A and B are the prime attributes of R.

Exercise – Finding Keys

- Checking all possible combinations of the attributes is too tedious!

  Example: Still consider a relation schema $R = \{A, B, C, D\}$ and $\Sigma = \{AB \rightarrow C, AC \rightarrow D\}$. List all the keys of R.

- **Some tricks:**
  - If an attribute *never* appears in the dependent of any FD, this attribute must **be part of each key**.
  - If an attribute *never* appears in the determinant of any FD but appears in the dependent of any FD, this attribute must **not be part of each key**.
  - If a proper subset of X is a key, then X must **not be a key**.

Finding Keys - Example

- Consider ENROLMENT and the following FDs:
  - {StudentID} $\rightarrow$ {Name};
  - {StudentID, CourseNo, Semester} $\rightarrow$ {ConfirmedBy, Office};
  - {ConfirmedBy} $\rightarrow$ {Office}.

ENROLMENT					
Name	StudentID	CourseNo	Semester	ConfirmedBy	Office
Tom	123456	COMP2400	2010 S2	Jane	R301
Mike	123458	COMP2400	2008 S2	Linda	R203
Mike	123458	COMP2600	2008 S2	Linda	R203

- What are the keys, superkeys and prime attributes of ENROLMENT?
  - {StudentID, CourseNo, Semester} is the only key.
  - Every set that has {StudentID, CourseNo, Semester} as its subset is a superkey.
  - StudentID, CourseNo and Semester are the prime attributes.