Week 7
Normalization (Database Design)

Questions for me?

Chapter Objectives

- What is normalization and what role does it play in database design
- About the normal forms 1NF, 2NF, 3NF, BCNF, and 4NF
- How normal forms can be transformed from lower normal forms to higher normal forms
- That normalization and E-R modeling are used concurrently to produce a good database design
- That some situations require denormalization to generate information efficiently

What is Normalization ?

- A technique for producing a set of relations with desirable properties (minimum data redundancy), given the data requirements of an enterprise.

- First developed by E.F. Codd (1972)

Normal Forms

There are several normal forms:

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF
- 5NF

As normalization proceeds, relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

Why Normalize ?

- An ER Model is not always available as a starting point for design
- To reduce redundant data in existing design
- To increase integrity of data and stability of design
- To identify missing tables, columns and constraints

Example of a bad design:

Example : FIRST (S#, P#, STATUS, CITY, QTY)

- Suppliers supply quantities of parts and live in some city.
- Each city has a status.

Fig : Relation FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	50	Guelph	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

(Example adapted from Textbook by C.J.Date)

Drawbacks of a bad design:

- A poorly designed relation contains redundant data, which leads to update anomalies :

- Can I add a supplier ‘S6’ who lives in Guelph?
  - No, unless S6 supplies a part - because P# is a part of the key and cannot be left NULL.
  - Not a valid insert

    insert into FIRST values (‘S6’, NULL, ‘GUELPH’, NULL, NULL);
- What if I have to update ‘S1’ city to Guelph?
  - Causes high risk of inconsistency and is inefficient as well.
- Now try deleting supplier ‘S3’ from the relation.
  - It has a side- effect! It unintentionally removes other relevant information that Guelph has a status of 50.

Drawbacks of a bad design:

- A poorly designed relation contains redundant data.

- Data redundancy causes update anomalies :
- **Insertion Anomalies** -An *independent* piece of information cannot be recorded in a relation unless an *irrelevant* information must be inserted together at the same time.
- **Modification Anomalies** - The update of a piece of information must occur at multiple locations (if not, it leads to inconsistency).
- **Deletion Anomalies** - The deletion of a piece of information *unintentionally* removes other information.

- Normalization is used to remove update anomalies by decomposing the relations into smaller relations based on the constraints.

What is normalization?

- We should be able to convert bad relations into good relations.
  - How do we know if we have a good or bad relation?
  - How can we convert bad relations into good relations?
- Relations are classified into classes called **normal forms**. Relations in a class satisfy some conditions – conditions of the normal form.
- The process of converting relations into relations in good normal forms is **normalization**.

First Normal Form

- All key attributes defined
- No repeating groups in table (only atomic values)
- All attributes dependent on

  primary key

Example : 1NF

Example : FIRST(S#,STATUS,CITY,P#,QTY)

- Suppliers supply certain quantities of parts and are located in some city.
- Every city has a status

Fig : Relation FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P3	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Example of a table not in 1NF

- A table that contains one or more repeating groups is not a relation by definition
- And it is not in the 1st normal form.

Examples :

- ✓ Relation DUMMYF given below is not in 1NF

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
			P2	200
			P3	400
			P4	200
			P5	100
			P6	100
S2	10	Paris	P1	300
			P2	400
S3	10	Paris	P3	200
S4	20	London	P1	200
			P4	300
			P5	400

Query: Find city of those suppliers who supply part P2.

```
SELECT DISTINCT CITY
FROM    FIRST
WHERE  pnum = 'P2';
```

Ans: NULL

UnNormalized to 1NF

Remove repeating group by entering appropriate data into the empty columns of rows containing repeating data ('flattening' the table).

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P3	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Query: Find city of those suppliers who supply part P2.

```
SELECT DISTINCT CITY
FROM    FIRST
WHERE  pnum = 'P2';
```

Ans:

```
CITY
--------
London
Paris
```

FIRST: 1NF – Repeat slide

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P3	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

- ✓ Is FIRST in 1NF ? YES
- ✓ Does FIRST have update anomalies ? YES – see slide 8 for definition of anomaly.

1NF but with anomalies

Process: To overcome the above anomalies in relation FIRST , FIRST has to be decomposed into smaller relations (projections).

Process:

1. Identify the key
2. Detect anomalies
3. Decompose into smaller relations
4. Prove that the decomposition is good!

Functional Dependency

Given two attributes X and Y of a relation R, Y is **functionally dependent** on X if whenever two tuples agree on their X-value, they also agree on their Y-value (each X-value in R has associated with it exactly one Y-value in R).

- **X** $\rightarrow$ **Y** means "X functionally determines Y" or "Y is functionally dependent on X".
- X is the **determinant** of the functional dependency X $\rightarrow$ Y, and Y is the **dependant**.

Functional Dependency

List all FDs in FIRST:

(S#, P#) -> CITY

(S#, P#) -> STATUS

(S#, P#) -> Qty

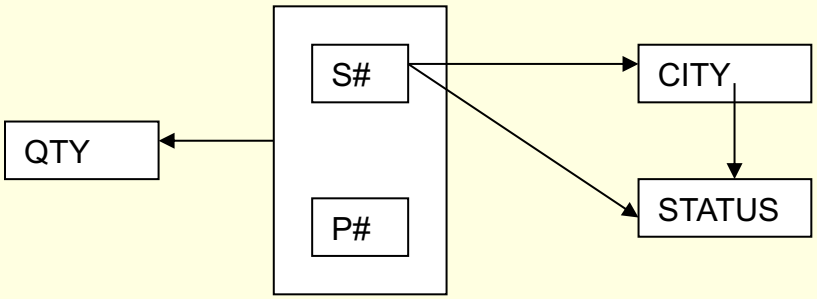
S# -> CITY

S# -> STATUS

CITY -> STATUS

You may either list all FDs or draw an FD diagram

FD Diagram of FIRST:



Properties of FDs

- X may or may not be the key attribute of R
- X and Y may be composite
- X and Y could be mutually dependent on each other

  Husband $\rightarrow$ Wife Wife $\rightarrow$ Husband

- The same Y-value may occur in multiple tuples

Properties of FDs

An FD could be *trivial* or *non-trivial*

An FD is *trivial*

- ✓ if it is impossible for it not to be satisfied
- ✓ if the RHS is a subset of the LHS
  - ✓ E.g. (city, status) -> status

In Normalization, the FDs of interest are :

- ✓ non-trivial
- ✓ hold for all times

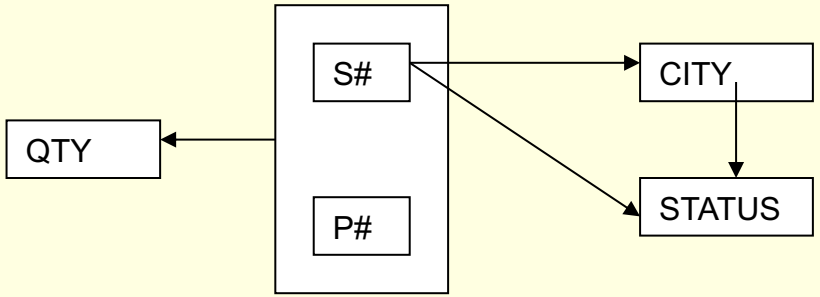
FDs

- To find the set of all FDs, we can either use the requirements given (method commonly used in real-world scenarios) OR
- use an algorithm that finds the closure of X under F, where X is a set of attributes and F is a set of FDs.
- The set of all attributes functionally determined by X under F is called the closure of X under F and denoted by X+.

Example of a bad design:

FIRST – key: S#, P#

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	50	Guelph	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400



Full Functional Dependency :

Given attributes X and Y of a relation R, Y is fully functionally dependent on X if Y is functionally dependent on X but not on any proper subset of X.

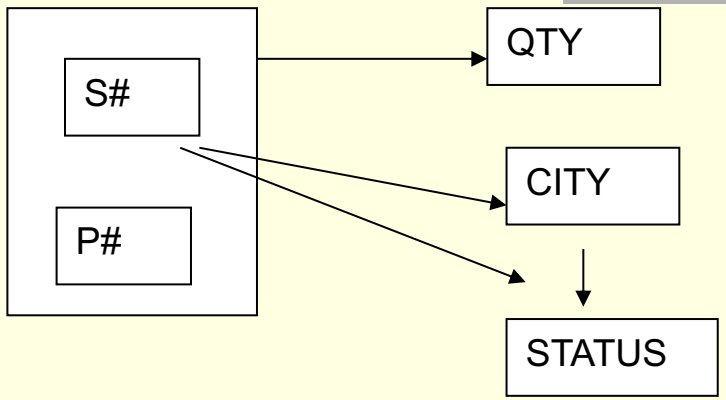
Transitive Functional Dependency

Given attributes X, Y and Z of a relation R, Z is transitively dependent on X iff

$X \rightarrow Y$, $Y \rightarrow Z$ and $X \rightarrow Z$

Non-full FDs in FIRST

FD diagram of FIRST:



There are 2 FDs that are non-fully FD on the key:

(S#, P#) -> CITY

- because CITY depends on the key and CITY also depends on S# alone, which is a part of the key

(S#, P#) -> STATUS

- because STATUS depends on the key and STATUS also depends on S# alone, which is a part of the key)

SECOND NORMAL FORM (2NF)

A relation R is in 2NF iff it is in 1NF and every non-prime-key attribute is fully functionally dependent on the primary key.

- A non-prime-key attribute is one that is not a part of the primary key.

- Is FIRST in 2NF ?
- How to normalize FIRST into 2NF ?
  - Remove all partial (or non-full) dependencies from FIRST by decomposing FIRST into two smaller relations SP and SECOND

SECOND NORMAL FORM (2NF)

Decompose FIRST into 2 smaller relations such that the non-prime attributes that are non-fully functionally dependent on the key are separated from those that are fully FD :

SP(S#,P#,QTY)
SECOND(S#,CITY,STATUS)

Non-loss Decomposition – to be continued

- To overcome the update anomalies in a relation R, R is decomposed into smaller relations (projections) r_1, r_2…….
- A bad decomposition loses information.
- In a good decomposition
  - The join of decomposed relations restores the original relation. R= $r_1 \infty r_2$ …..
  - Decomposed relations can be maintained independently.

Non-loss Decomposition

- Example:
- Consider the schema U(A,B,C) and the decomposition R1(A,B), R2(A,C).

U

A	B	C
5 7	10 10	6 4

R1

A	B
5 7	10 10

R2

A	C
5 7	6 4

R1 ∞ R2 = U

A	B	C
5 7	10 10	6 4

This example adapted from Dr. Fangju Wang's notes

Non-loss Decomposition

- Consider the schema U(A,B,C) and the decomposition R1(A,B), R2(B,C)

U

A	B	C
5 7	10 10	6 4

R1

A	B
5 7	10 10

R2

B	C
10 10	6 4

R1 ∞ R2 NOT= U

A	B	C
5 5	10 10	6 4
7 7	10 10	6 4

This is a lossy decomposition –has extra or spurious tuples

This example adapted from Dr. Fangju Wang's notes

Non-loss Decomposition

- It is impossible to examine all the instances of a relation schema for testing if a decomposition is lossless. A more formal method is given next.
- Rissanen's Rules: Decomposition of R into R1 and R2 is good if
  - The common attributes of R1 and R2 form a candidate key for at least one of the pair AND
  - Every FD in R can be logically derived from those in R1 and R2

SECOND NORMAL FORM (2NF)

- ✓ Is the following decomposition of

First (S#,P#, CITY, STATUS, QTY) into

SP and SECOND good ?

SP(S#,P#,QTY) SECOND(S#,CITY,STATUS)

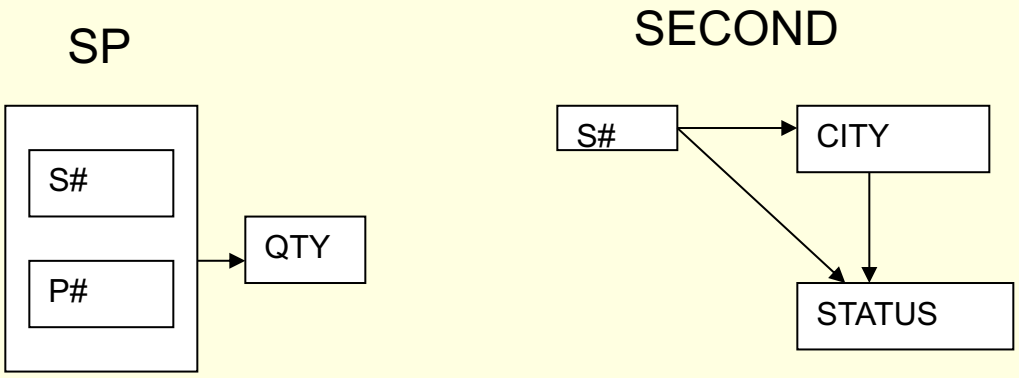
1. SP ∩ SECOND = S# (Non-loss)

   Is S# a key of either of the 2 relations ✔

2. Dependencies preserved? ✔

SECOND NORMAL FORM (2NF)

- FD Diagram of SP and SECOND



- ✓ SP in 2NF? SECOND in 2NF?
- ✓ Lets check for update anomalies in SECOND ?
  - ✓ SECOND has anomalies because it has transitive dependencies (next slide)

Transitive Functional Dependency

Given attributes X, Y and Z of a relation R, Z is transitively dependent on X iff

$X \rightarrow Y$, $Y \rightarrow Z$ and $X \rightarrow Z$

In our example,

S# -> CITY

CITY -> STATUS

S# -> STATUS

Therefore, STATUS is transitively dependent on S#

THIRD NORMAL FORM(3NF)

- A relation is in 3NF iff it is in 2NF and in which every non-prime-key attribute is non-transitively dependent on the primary key.

- Is SECOND(S#,CITY,STATUS) in 3NF ?

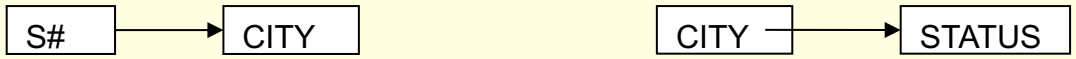
THIRD NORMAL FORM(3NF)

- How to normalize SECOND into 3NF ?
  - Remove all transitive dependencies from SECOND by decomposing SECOND into two smaller relations SC and CS.

    SC(S#,CITY) CS(CITY,STATUS)
  - Make sure that the decomposition is good !
  - Try other possibilities:
    - (S#, STATUS) and (S#, CITY) ??
    - (S#, STATUS) and (CITY, STATUS) ??

THIRD NORMAL FORM(3NF)

- FD Diagram of SC and CS



Help me Codd !

The key(1NF), the whole key(2NF), and nothing but the key (3NF) - so help me Codd !!

(Taken from Thomas Connnolly)