Individuell uppgift 3, Sammanfattning

Yasir Riyadh Jabbar (TIDAA KTH)

Kurslitteratur: Kapitel 6 (hela), 9.1-9.2

CHAPTER 6 Normalization and Denormalization

6.1 Objectives of Normalization

In relational model the objective is to represent data, relations and constraints in good manner. But due to there is many options to be considered, the choosing of relations becomes hard. So, to improve the design we use normalization. The aim of normalization is:

- eliminate repetitive data.
- create a stable set of relationships.
- make model more flexible and can be extended for adding new attributes.
- It will be free of anomalies (inconsistent state of database).

6.2 Anomalies

The effect of anomalies prevents some data from being represented or lose it while data is being updated.

1. Update anomaly: It occurs if value in one row is changed, resulting it inconsistent with the same data in another row.
2. Insertion anomaly: It occurs when adding info about object that is not exist (ex. insert info about student who is not registered yet).
3. Deletion anomaly: It occurs when deleting row that may contain attributes that shouldn't be deleted

These anomalies are caused by functional, multivalued, and join dependencies. To avoid it, there are 3 normal forms (NF): 1NF, 2NF, and 3NF. Normal form is step of normalization that meets the requirements of that form.

6.3 Functional Dependency

It means that only one value of attribute B in all rows can be found from another attribute A if A functionally determines B. But there is different values of A for given B value (many-to-1 relationship).

Student Table

SID	LName	Major	Credits	Status	SSN
S101	Sam	Control	91	Senior	1000
S102	Oliver	Math	92	Senior	1010
S103	James	Control	10	Freshman	1020
S104	John	Math	60	Junior	1030
S105	Oliver	IT	27	Freshman	1040

Ex. from Student table:

1) all attributes are functionally dependent (FD) on SID

   {SID}→ {LName, Major, Credits, Status, SSN}

2) Also, all attributes are FD on SSN

   {SSN}→ {SID, LName, Major, Credits, Status}

3) Status is FD on Credits

   {Credits} → {Status}

6.4 Super, Candidate, and Primary Keys

1) Super key: It is attribute (or attributes) that identifies any row. Ex. {SID} and {SSN}.
2) Composite key: It contains more than one attribute.
3) Candidate key: It is minimal identifier that can not removed attribute from it without losing its uniqueness.
4) Primary key: it is candidate key used to identify rows. SID or SSN can be used in student table (privacy rules determine what key we choose).

Note: only 1 primary key can be used in each relation.

6.5 Normalization Using Primary Keys

First Normal Form (1NF):

Condition for 1NF: each attribute for each row must be single-valued

If relation is not in 1NF, there is 3 alternatives to convert it to 1NF.

- 1st solution (**good**): creating new table contains the primary key with the multivalued attribute.
- 2nd solution (not proper): adding new column for each multivalued attribute.
- 3rd solution (not proper): making multivalued attribute part of primary key.

Example for 1st solution (converting non-1NF to 1NF):

Student Table

SID	LName	Major	Credits	Status	SSN
S101	Sam	Control	91	Senior	1000
S102	Oliver	Math, IT	92	Senior	1010
S103	James	Control, Power	10	Freshman	1020
S104	John	Math	60	Junior	1030
S105	Oliver	IT	27	Freshman	1040

We create new table (Major) contains the primary key (SID) with each major. The new primary key now is {SID, Major}.

NewStudent Table

SID	LName	Credits	Status	SSN
S101	Sam	91	Senior	1000
S102	Oliver	92	Senior	1010
S103	James	10	Freshman	1020
S104	John	60	Junior	1030
S105	Oliver	27	Freshman	1040

Major Table

SID	Major
S101	Control
S102	Math
S102	IT
S103	Control
S103	Power
S104	Math
S105	IT

Second Normal Form (2NF) and Full FD:

In 2NF all non-key attributes must be fully FD on the key

Conditions for 2NF

1) **If it is in 1NF and primary key is single attribute then its in 2NF.**
2) **If it is in 1NF and primary key is composed key then we perform projections on the relation.**

Projection Steps

1) Identify each of non-full FD attributes,
2) Remove attributes that depend on determinants that found in step1,
3) Place determinants in separate tables with their dependent attributes,
4) Old table will contain only the remainder attributes.

Example for using projection steps to obtain 2NF:

Class Table (not in 2NF)

ClassNo	SID	LName	FID	Schedule	Room	Grade
MT10A	S101	Sam	F102	Tu11	A20	B
MT10A	S102	Oliver	F102	Tu11	A20	
CT30A	S103	James	F103	W9	B24	A
CT30A	S104	John	F103	W9	B24	C
MT20B	S101	Sam	F104	F11	A20	

The FD is:

{classNo, SID} → {LName, FID, schedule, room, grade}

And the not-fully FD are:

classNo → {FID, schedule, room}

SID → LName

The Class relation is not in 2NF because LName is not fully FD on the key {classNo, SID}. Using projection, we obtain new relations:

1) Grade is fully FD on the key {classNo, SID}

New relation: **StudGrade (classNo, SID, grade)**

2) LName is not fully FD on the key {classNo, SID} but only on SID

New relation: **StudLName (SID, LName)**

3) The old table will contain the remainder attributes

New relation: **NewClass (classNo, FID, schedule, room)**

NewClass Table

ClassNo	FID	Schedule	Room
MT10A	F102	Tu11	A20
CT30A	F103	W9	B24
MT20B	F104	F11	A20

StudGrade Table

ClassNo	SID	Grade
MT10A	S101	B
MT10A	S102	
CT30A	S103	A
CT30A	S104	C
MT20B	S101	

StudLName Table

SID	LName
S101	Sam
S102	Oliver
S103	James
S104	John

Advantages after being 2NF in the previous example:

1) schedule can be updated in a single row in NewClass.
2) class info can be added to NewClass without student registration.
3) student info can be inserted to StudLName without its class information.
4) grade row can be deleted from StudGrade without losing class and student info.

Third Normal Form (3NF) and Transitive Dependency:

Some 2NF may still have anomalies. For example:

Student (SID, LName, major, credits, status)

For the following functional dependency:

SID → {credits, status} we have also: **credits → status**

So, we can get 'status' from SID in two ways: directly and transitively.

Condition for 3NF

Removing transitive dependencies (cause anomalies) by making each non-key attribute functional dependent on entry key.

Example for obtaining 3NF:

transitive dependency (**SID → credits → status**)
1) New relation: **NewStudent (SID, LName, major, credits)**
2) New relation: **NewStatus (credits, status)**

Student Table (not in 3NF)

SID	LName	Major	Credits	Status
S101	Sam	Control	91	Senior
S102	Oliver	Math	92	Senior
S103	James	IT	10	Freshman
S104	John	Power	60	Junior
S105	Oliver	IT	27	Freshman

NewStudent Table

SID	LName	Major	Credits
S101	Sam	Control	91
S102	Oliver	Math	92
S103	James	IT	10
S104	John	Power	60
S105	Oliver	IT	27

NewStatus Table

Credits	Status
10	Freshman
27	Freshman
60	Junior
91	Senior
92	Senior

Boyce-Codd Normal Form (BCNF):

BCNF is more stringent than 3NF. It states that if X is super key and functional dependency X → A exists, then the relation is in BCNF.

Checking for BCNF

1) Identifying and verifying all determinants that they are super keys
2) Breaking up the relation by projection if they are not super keys
3) Creating new relation for each determinant with all attributes it determines.

Example:

Student Table (not BCNF)

SID	LName	Major	Credits	Status
S101	Sam	Control	91	Senior
S102	Oliver	Math	92	Senior
S103	James	IT	10	Freshman
S104	John	Power	60	Junior
S105	Oliver	IT	27	Freshman

In the table, SID and credits are the determinants. But credits is not super key, so this relation is not BCNF.

Practical Example for using (1NF, 2NF and 3NF)

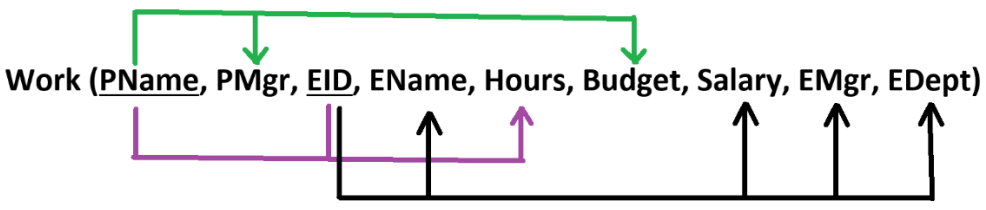
For Info about business projects (P for project, E for employee):

Work Table

PName	PMgr	EID	EName	Hours	Budget	Salary	EMgr	EDep
proj1	Sam	E01	James	20	200 000	50 000	Jack	1
proj1	Sam	E02	Robert	35	200 000	40 000	James	2
Proj2	John	E01	James	10	300 000	50 000	Jack	1
Proj2	John	E03	Smith	25	300 000	55 000	James	2

Work (PName, PMgr, EID, EName, Hours, Budget, Salary, EMgr, EDept)

1) Functional dependencies:



PName → {PMgr, Budget}
EID → {EName, Salary, EMgr, EDept}
{PName, EID} → Hours
EDept → EMgr … (each department has only one manager)

The primary key is the composed {PName, EID}

2) Checking for BCNF

Each of determinants is not super key (ex. EDept), so relation is not BCNF.

3) Checking for 1NF

For composite key {PName, EID} each value in the table is single-valued, so the relation in 1NF.

4) Checking for 2NF

There is non-fully dependencies in the relation:

PName → PMgr, Budget
EID → EName, Salary, EMgr, EDept

Using projection steps, we obtain new relations:

New relation: **Project (PName, PMgr, Budget)**
New relation: **Employee (EID, EName, Salary, EMgr, EDept)**
New relation: **NewWork (PName, EID, Hours)**

5) Checking for 3NF

Relation	Has transitive dependency?	In 3NF?
Project	No	Yes
NewWork	No	Yes
Employee	Yes EDept → EMgr Reason: EDept is not super key, and EMgr is not part of candidate key	No

So, Employee relation needs rewrite to:

New relation: **NewEmployee (EID, EName, Salary, EDept)**

New relation: **Department (EDept, EMgr)**

The new 3NF relations for Work table are:

Project (PName, PMgr, Budget)

NewEmployee (EID, EName, Salary, EDept)

Department (EDept, EMgr)

NewWork (PName, EID, Hours)

Project Table

PName	PMgr	Budget
proj1	Sam	200 000
Proj2	John	300 000

Department Table

EMgr	EDep
Jack	1
James	2

NewEmployee Table

EID	EName	Salary	EDep
E01	James	50 000	1
E02	Robert	40 000	2
E03	Smith	55 000	2

NewWork Table

PName	EID	Hours
proj1	E01	20
proj1	E02	35
Proj2	E01	10
Proj2	E03	25

6.6 Properties of Relational Decompositions

The decomposition process is technique to place all attributes in one relation (called universal relation) then decomposed it into smaller relations (each relation is in 3NF) to get some qualities and other improvements properties.

Attribute Preservation:

In some relations, an attribute may appear more than once (reptation) to represent relationships. With decomposition process, there is no attribute reptation and no data is lost because each attribute exists in one of the relations at least once.

Dependency Preservation:

Every functional dependency must be satisfied in at least one decomposed relation. As example, if we apply decomposition process to the universal relation A and decomposed it to new smaller relations A1 and A2 then we will find the dependencies are exists in A1, A2 or combination of them.

Lossless Decomposition:

If universal relation A is decomposed to new smaller relations A1 and A2, then the natural joining of A1 and A2 reconstructs the original universal relation A. No data loses at all.

6.7 The Normalization Process

In addition to starting from entity-relationship diagram, the normalized relations can also be developed using two processes: analysis and synthesis.

Analysis:

1) listing of all attributes that exists in the universal relation for the database.
2) Identifying functional dependencies between attributes then convert the universal relation (using projection) to smaller normalized relations.

Synthesis:

It is just bottom-up process, doing the opposite of the analysis process. It combines attributes based on functional dependencies to related sets to develop normalized relations.

6.8 When to Stop Normalizing

The object of normalization process is to reach BCNF. However, performing these steps of process depends on the needs of the applications and the database requirements. Also, it is important to take into account what the final form will be. As example, if dependencies is not preserved at BCNF checking step, then it's wise to stay in 3NF and no farther advanced in normalization process.

6.9 Non-normalized Databases

It is preferable to use normalized databases when the source data is relatively simple, such as employee data, sales, students, inventory. However, there is some situations in which non-normalized databases are favored, such as having frequent data reads with infrequent updates, storing original documents, or requiring multiple versions of data.

CHAPTER 9 Transaction Management

9.1 ACID Properties of Transactions

The block of one or more SQL statements is called transaction. Without suitable concurrency controls (prevents interference among processes) and recovery (restoring database to the correct state) methods, database system may be easily damaged.

As example for simple transaction, is to update the credits value for specific student SID in university database. The steps in this transaction as follows:

1. locating the SID for this student.
2. Fetching its page from the file to the buffer.
3. Updating SID in the buffer.
4. Storing the update from buffer to the file.

The transaction can ne ended or terminated by two ways:

1. If successfully executed, the database becomes in new consistent state (committed).
2. Otherwise, the transaction is aborted. In this case, the database must be restored (rolled back) to the consistent state it was in.

ACID properties

The characteristics of all transactions refers to ACID properties:

1. Atomicity: transaction must be executed by all steps or none (partial transaction cause inconsistent state).
2. Consistency: database must be in consistent state.
3. Isolation: any change made by one transaction process must be isolated from another one until the first one finished (commit).
4. Durability: the log file is needed to record all operations for recovery.

9.2 Need for Concurrency Control

Concurrency control system prevents interference among processes and allows simultaneous processing to be done. We have three cases:

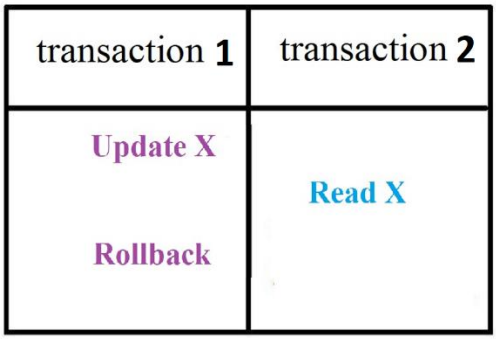
1. If the process is only reading data, there is no interference.
2. If the process is accessing different locations of database, there is also no interference.
3. But if two transactions making updates to same data at same time (or one updates and other reads), interference will occur.

Problems caused by concurrency:

1. **Lost update problem**: an update done to value by transaction is lost as it is overwritten by the update done by another transaction. Example

transaction **1**	transaction **2**
read_item(X) X = X + N	X = X + 10 write_item(X)

2. **Uncommitted Update Problem (dirty read)**: It occurs when the first transaction is modifying value which is then read by second transaction, and the first transaction subsequently rolls back its update. Example:



3. **Problem of Inconsistent Analysis**: It occurs when transaction reads many values but a second transaction updates some of them during the execution of the first.

4. **Phantom data problem**: It occurs when first transaction reads some rows, another transaction inserts row, and then the first transaction reads the rows again and sees the new row.