6.0 Relational Database Design

6.1 Pitfall of relational model

- The main goal of relational database is to create / find good collection of relational schemas. Such that database should allow us to store data / information without unnecessary redundancy and should also allow to retrieve information easily. That is the goal of relational database design should concentrated
    - o To avoid redundant data from database.
    - o To ensure that relationships among attributes are represented.
    - o To facilitate the checking of updates for the violation of database integrity constraints.

- A bad relational database design may lead to:
    - o Repetition of information.
        - That is, it leads data redundancy in database, so obviously it requires much space.
    - o Inability to represent certain information.

Example 1: Consider the relational schema

Branch_loan = (beranch_name, branch_city, assets, customer_name, loan_no, amount)

branch_name	branch city	assets	customer name	loan no	amount
kathmandu	baneshwor	25000	rohan	L - 15	3000
Lalitpur	patan	19000	mohan	L - 17	5000
Kathmandu	baneshwor	25000	raju	L - 19	10000
Pokhara	Prithibi nagar	17000	manoj	L - 10	7000
Lalitpur	patan	19000	swikar	L - 30	9000

Redundancy:

- Data for branch_name, branch_city, assets are repeated for each loan that provides by bank.
- Storing information several times leads waste of storage space / time.
- Data redundancy leads problem in Insertion, deletion and update.

Insertion problem

- We cannot store information about a branch without loan information, we can use null values for loan information, but they are difficult to handle.

Deletion problem

- In this example, if we delete the information of Manoj (i.e. DELETE FROM branch_loan WHERE customer_name = 'Manoj';), we cannot obtain the information of pokhara branch (i.e. we don't know branch city of pokhara, total asset of pokhara branch etc.

Update problem

- Since data are duplicated so multiple copies of same fact need to update while updating one. It increases the possibility of data inconsistency. When we made update in one copy there is possibility that only some of the multiple copies are update but not all, which lead data/database in inconsistent state..

6.2 Decomposition

The idea of decomposition is break down large and complicated r elation in no. of simple and small relations which minimized data redundancy. It can be consider principle to solve the relational model problem.

Definition

The decomposition of relation schema R= (A_1, A_2, . ,An) is a set of relation schema { R_1, R_2, ----- R_m}, such that $R_i \subseteq R \ \forall\ 1 \leq i \leq m$ and $R_1 \cup R_2 \cup . . \cup R_m = R$.

That is all attributes of an original schema (R) must appear in the decomposition (R_1, R_2).
That is, $R= R_1 \cup R_2$. if $R \neq R_1 \cup R_2$ then such decomposition called lossey join decomposition. That is, $R \neq \Pi_{R1}(R) \bowtie \Pi_{R2}(R)$. Decomposition should ***lossless join decomposition***.

A decomposition of relation schema R into R_1 and R_2 is lossless join iff at least one of the following dependencies is in F^+.

$R_1 \cap R_2 \rightarrow R_1$

$R_1 \cap R_2 \rightarrow R_2$

Example 1: The problems in the relational schema branch_loan (illustrated in above example) can be resolved if we replace it with the following relation schemas.

Branch (# branch_name, branch_city, assets)
Loan (customer_name, loan_number, branch_name, amoun)

Example 2: Consider the relation schema to store the information a student maintain by the university.

Student_info (#name, course, phone_no, major, prof, grade)

name	course	phone_no	major	prof	grade
John	353	374537	Computer Science	Smith	A
Scott	329	427993	Mathematics	James	S
John	328	374537	Computer Science	Adams	A
Allen	432	729312	Physics	Blake	C
Turner	523	252731	Chemistry	Miller	B
John	320	374537	Computer Science	Martin	A
Scott	328	727993	Mathematics	Ford	B

Problems:

Redundancy:

- Data for major and phone_no of student are stored several times in the database once for each course that is taken by a student.

Complicates updating:

- Multiple copies of some facts may lead to update which leads possibility of inconsistency.
- Here, change of phone_no of John is required we need to update three records (tuples) corresponding to the student John. If one of the three tuples is not changed there will be inconsistency in the date.

Complicate insertion

- If this is only the relation in the database showing the association between a faculty member and the course he / she teaches, then the information that a given professor is teaching a given course cannot be entered in the database unless a student is registered in the course.

Deletion Problem:

- If the only one student is registered in a given course then the information as to which professor is of fering the course will be lost if this is only the relation in the database showing the association between the faculty member and the course he / she teaches. If database have another relation that establishes the relationship between a course, the deletion of 4th & 5th tuple in this relation will not cause the information about the councils teach to be lost.

Solution?

⇒ Decomposition

The problems in this relation schema student_info can be resolved if we it with the following relation schemas.

Students (#name, phone_no, major)
Transcript (#name, course, grade)
Teacher (course, prof)

- Here, first relat ion schema gives the phone number and major subject of each student such information will be stored only once for each student. Thus, any changes in the phone number will thus require changes in only one tuple of this relation
- The second relation schema 'TRANSCRIP' stores the grade of each student in each course. So, to insert information about student phone number, major and the course teaches by which the professor.
- Third relation schema records the teacher of each course.

What is the problem of such [redacted] decomposition?

- One of the disadvantages of orignal relation [redacted] schema 'student_info' with these true relational schemas is that retrieval of certain information required to performed natural join operation.
- We know that to resolve the problem of bad relat ional database design we need to decompose relation but the problem is that how to decompose relation. That is, we require to process decomposition of relation that may give good relational database. Normalization gives the approach for designing the best relational database under the five normal forms.

6.3 Normalization:

Normalization is the process of reducing redundant data / information in the database by decomposing the long relation. It is an approach for designing reliable database system. Normalization theory is built under the concept of the normal form. There are several normal form called

First Normal Form. (1NF)
Second Normal Form (2NF)
Third Normal Form (3NF)
Fourth Normal Form (4NF)
Fifth Normal Form (5NF)

A relation is said to in particular normal form if it satisfy the set of this particular normal form if it satisfy the set of this particular normal form's constraints. In practical, third normal form is sufficient to design reliable database system.

Normalization theory is based on:

- Functional dependencies
- Multi valued dependencies.

6.3.1 Needs of normalization

In database, there is considerable amount of data redundancy and also serious insertion, deletion and update. Thus to reduce the data redundancy as far as possible and for easy insert, delete and update operation in relational database system we require normalization. It does not allow any inconsistency in database system.

6.3.2 Objectives of Normalization

Dr. codd suggested five normal form for relational database design without data redundancy and serious insertion, update and deletion. According to him, the objectives of normalization are:

1. To free a collection of relation from undesirable insertion, update and deletion.
2. To move the relational model more informative to use.
3. To increase the lifetime of application programs.

6.3.3 Properties of relation after normalization

1. No data value should be duplicate in different row or tuple.
2. A value must be specified for every attribute in a tuple.
3. Important information should not be accidentally lost.
4. When new data / record are inserted to relation, other relation in database should not be affected

6.3.4 Non -Normalized Relation

A relation schema R is said to be non-normalized relation if any domain of attributes of R are non atomic.

Example:

faculty	prof	course	
		course_name	**department**
computer	X	DBMS	Computer
		Software Engineering	Computer
		Microprocessor	Electronics
	Y	Mathematical Analysis	Math
		Probability Theory	Stat
Physics	Z	Modern Physic	Physics
	Z	Dynamics	Physics
	P	Vector Analysis	Math

Figure: non-normalized relation

- Here, the attribute of relation schema, that is course is divided into course_name and department. That is, domain of attribute course is non atomic.
- Here, each row contains multiple set of values.

6.3.5 Key and non key attribute (prime and non prime attribute)

An attribute A in relation R is said to be key attribute if A is any part of candidate key; otherwise A is called non key attribute.

Example: Consider a relational schema 'student_course_info'
Student_course_info=(#name,course,grade,phone,major,department)
With the following FDs
F={name→phone_no,major, course Mathdepartment, name,course Mathgrade}

- Here, name and course are key attributes and phone_no, major and department are non key attribute.

Example: Let R=(A,B,C,D,E) and FDs F={AB →C,B →D,C→E}
- Here AB is composite primary key (composite candidate key) so attribute A and B are key attributes and attributes C, D and E are non key attributes.

6.4 Normal Forms

6.4.1 First Normal Form

- A relation schema R is said to be in first normal form if domain of all attributes of R are atomic. That is, for any relational schema R to be in first normal form, each attribute of relation not divisible and each row must contain single set of values for all attributes.
- The previous relational schema 'course_info' can be covert into first normal form as below

prof	course_name	faculty	department
X	DBMS	computer	Computer
X	Software Engineering	computer	Computer
X	Microprocessor	computer	Electronics
Y	Mathematical Analysis	computer	Math
Y	Probability Theory	computer	Stat
Z	Modern Physic	Physics	Physics
Z	Dynamics	Physics	Physics
P	Vector Analysis	Physics	Math

Figure: course_info relation in first normal form

- First normal form enforce relation in tabular form.
  Here, FDs F={prof → faculty,course_name→department}
- Here key attributes are prof and course and key of relation is {prof,course_name}
- The representation of relation 'course_info' in first normal form has several drawbacks. Major problem is data redundancy.

Data redundancy

- Given professor assigned to corresponding faculty is repeated a number of times.
- Given course offered by corresponding department is repeated number of times.
- Data redundancy leads problem in insertion, deletion and update.

Update problem

- If professor change faculty to teach then we must change all rows where that professor appears, it may leads inconsistency in database.

Deletion problem

- If we have to delete the professor who teach only one subject (course_name) in such case deletion cause the loss of information about the department to which the course belongs.

Insertion problem

- Suppose department introduce new course (i.e. course_name) such information can not insert without professor name and assigned faculty. This is possible inserting only null values instead of professor name and his/her assigned faculty.

6.4.2 Second Normal Form

- A relation schema R is said to be in second normal form (2NF) if it is first normal form and all non key attributes are fully functionally dependent on relation key (s).
- A second normal from does not allow partial dependency between non key attribute and relation key (s). But it does nor enforce that non key attribute may not functionally dependent on another non key attribute. (In fact, 2NF that does not allow such dependency called relation in 3NF).

Example: Let us consider the relation 'student_info'
Student-info=(<u>#name,course</u>,phone_no,major,prof,grade)
with the functional dependencies
F={name→phone_no,name→major,course→prof,name,course→grade}

- Here, in this relation grade is fully functionally dependent on key (name,course). But phone_no and major are partially dependent on name and prof and prof is partially dependent on course.
- Since, second normal form does not allow partial dependency so the given relation 'student_info' is only in 1NF but not in 2NF.

Remarks: The relation can be converted into 2NF as below:

Student-general-info=(name,phone_mno,major)
Transcript=(#name,course,grade)
Teacher=(#course,prof)

- Here the functional dependencies on relations are as below:

Student-general-info

F={name→phone_no,name→major}

Phone_no and major are fully functional dependent on key 'name' of relation schema 'student-general-info'. So the is in 2NF. Moreover, here is no functional dependency between non key attributes (phone_no & major). So this relation is also in 3NF (discuss later)

Transcript

F={name,course→grade}

Here grade is fully functionally dependent on relation key (name, course). So this relation is in 2NF. Moreover, here is no fu nctional dependency between non key attributes because here is only one non key attribute grade.

Teacher

F={course→prof}

Here. Prof is fully functionally dependent on relation key course. So this relation is in 2NF. Moreover, here is no functional dependency between non key attributes. Here is only one non key attributes prof.

6.4.3 Third Normal Form

- A relation R is said to be in third normal form if it is in 2NF, every non key attributes in non transitively and fully functionally dependent on the every candidate key.
- That is, relational schema in 3NF does not allow partial dependency, transitive dependency. For any relation that is in 3NF, it must in 2NF, every non key attribute must fully functionally dependent on relation key (s) and relation must not exist partial dependency, transitive dependency and no functional dependency between non key attribute.
- The problems with a relation schema that is not in 3NF (problems with a relational schema in 2NF) are
  - If a relation schema R contains a transitive dependency Z →X→A , we can not insert an values for x in relation along with X value. That is, we can not independently record the fact that for each value of X there is one value of A (insertion problem). Similarly, deletion of a Z→A association also required the deletion of X →A association (deletion problem).
  - If a relation schema R contains a partial dependency (That is, an attribute A dependent on subset X of the key K of R.. i.e.K is key of R, X⊂K and X→A) then association between X and A (i.e. X→A) can not express (insert) unless remaining part of K is present in a tuple, remaining part of k must also be express (insert) since K is a key of relation, these part can not be null.
  - In the above relation 'course_detail'

    Course_detail=(#course,prof,#room,enroll_limit)

There is a transitive dependency, course→room→enroll_limit). We can eliminate this transitive dependency by decomposing the relation 'course_detail' info into the relations

Course_prof_info=(#course,prof,enroll_limit)
Fds={course→prof,course→enroll_limit}

Course_room_info=(course,room)
Fds=no functional dependency exist because course and room both are here foreign key.

That is, all relations in 3NF are as below:

Given relation in 1NF

Course_detail=(#course,prof,room,room_capacity,enroll_limit)
Fds={course→(prof,room,room_capacity,enroll_limit),room→room_capacity}

Relations in 2NF

Course_detail=(#course,prof,room,emroll_limit)
FDs={course→prof,course→room,course→enroll_limit,room→enroll_limit}

Room_detail=(#room,room_capacity)
Fds={room→room_capacity}

Relations in 3NF

Course_prof_info=(#course.prof,enroll_limit)
FDs={course→prof, course→enroll_limit}

Course_room_info=(course,room)
No FDs

Room_detail=(#room,room_capacity)
FDs={room→eoom_capacity}

6.5 Boyce Codd Normal Form (BCNF)

A relation schema R is said to be in Boyce Codd normal form if it is in 3NF and every FDs in F^+ should be in the form X→A where X ⊆S and A ∈S and at least one of the following condition hold:

(a) X→A is a trival FD (i.e. A∈X) or
(b) X→R (i.e. X is a superkey of R, here X called determinant of functional dependency X→R)

-
- The BCNF imposes stronger constraints on the type of FDs allow in a relation.
  - It allows only those non trivial functional dependencies whose determinants are candidate sperkeys of relation. But in case of 3NF, it allow non trivial FDs whose determinant is not a candidate superkey if right-hand side of FDs contained a candidate key.
  - That is BCNF enforce more stronger constraints than 3NF.

Example: Let us consider the relation schema "grade" which is in 3NF.
Grade=(#name,student_id,course,grade)

- Assume that , each student has unique name and unique student_id then set of FD
  FDs={name,course →grade, student_id,course→grade,name→student_id,student_id→name}

- Here this rela tion has two candidate keys {name,course} and {student_id,course}. Each of those composite key has common attribute course.
- Here the relation grade is not in BCNF because the dependencies {student_id→name}and {name→student_id} are non trivial and there determinants are not superkey of relation grade.

Drawback of this relation (which is not 3NF)

- Data redundancy: The association between name and corresponding student_id are repeated.
- Update problem: Any changes in one of these attribute value (name or student_id) has to reflected all tuples; otherwise there will be inconsistency in the database.
- Insertion problem: The student_id can not be associated with the student name unless the student has registered in a course.
- Deletion problem: The association between student_id and student name is lost if the student deletes all courses he/she is reistered in.

- Solution ?
  student_info=(#student_id,name)
  grade(<u>#student_id,course</u>,grade)

These problems of relation schema i n 3NF occur since relation may have overlapping candidate keys. BCNF removes this problem. So it is stronger than 3NF.

Example: Let us consider the relational schema
Student_info=(student_id,name,phone_no,major)

where student_id, name and phone_no assumed to be unique in this relation. The following FDs satisfy this relation.
FDs={student_id→name,student_id→phone_no,student_id→major, name→student_id, name→phone_no, name→major, phone_no→student_id,phone_no→name,phone_no→major}

Here each non trivial FD involves candidate key as determinant. Hence the relation 'student_info' is in BCNF.