Unit III

Relational DataBase Design

Unit III Contents

- Relational Model: Basic concepts, Attributes and Domains, CODD's Rules.
- Relational Integrity: Domain, Referential Integrities, Enterprise Constraints.
- Database Design: Features of Good Relational Designs, Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF,BCNF.

CO Mapped : CO1, CO3

Dr. S.R.Khonde

Relational Model

- The Relational model uses a collection of tables to represent both data and the relationships among those data.
- Tables are also known as relations.
- Relation: made up of 2 parts:

➢ Instance: a table, with rows and columns.

#rows =cardinality , #fields = degree / arity

➢ Schema: specifies name of relation, plus name and type of each column

E.g.: Students(*sid:* string, *name:* string, *login:* string, *age*: integer, *gpa:* real)

Relational Model Contd…

attributes

column

SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

tuple

table (relation)

Relational Model Contd…

- **Advantage:**
  - Structural Independence
  - Its simple to navigate
  - Greater Flexibility
  - Better Security

- **Disadvantages**
  - Performance
  - Data Complexity
  - Hardware and Software overhead
  - Physical Storage Consumption

Components

- The relational model consists of three major components
- The set of relations and set of domains that defines the way data can be represented (data structure)
- Integrity rules that define the procedure to protect the data (data integrity)
- The operations that can be performed on data (data manipulation)

Codd's Rule

- Dr. Edgar Frank Codd was a computer scientist while working for IBM he invented the relational model for database management.

- Codd proposed thirteen rules (numbered zero to twelve) and said that if a Database Management System meets these rules, it can be called as a Relational Database Management System.

- These rules are called as Codd's12 rules.

- Hardly any commercial product follows all.

Codd's Rule Cont..

- Rule 0 : Foundation Rule
- Rule 1: Information Rule
- Rule 2: Guaranteed Access Rule
- Rule 3: Systematic Treatment of NULL Values
- Rule 4: Active Online Catalog
- Rule 5: Powerful and Well-Structured Language
- Rule 6: View Updating Rule

Codd's Rule Cont..

- Rule 7: High-Level Insert, Update, and Delete Rule
- Rule 8: Physical Data Independence
- Rule 9: Logical Data Independence
- Rule 10: Integrity Independence
- Rule 11: Distribution Independence
- Rule 12: Non-Subversion Rule

Relational Integrity

- Integrity Constraint is a mechanism to prevent invalid data entry into table to maintain the data consistency.

- Mainly used to provide security and consistency to the database in various operations.

- Types of constraints
  - Domain Integrity Constraint
  - Entity Integrity Constraint
  - Referential Integrity Constraint
  - Enterprise Constraint

Domain Integrity Constraint

- The domain constraint are considered as the most basic form of integrity constraints.

- Domain integrity means it is the collection of valid set of values for an attribute.

- Constraints -
  - Not Null
  - Unique
  - Default
  - Check

Entity Integrity Constraint

Primary Key Constraint –

- It uniquely identify each record in a table
- It does not allow NULL and duplicate values
- Combination of Not Null and Unique

SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	45
8002	Tony	5th	23

Not allowed as Primary Key Values must be unique

- A relation/table can have only one primary key, which may consist of single or multiple fields.

Dr. S.R.Khonde

Referential Integrity Constraint

Foreign Key

- A foreign key is an identifier in a table that matches the primary key of a different table.

- The foreign key creates the relationship with a different table, and referential integrity refers to the relationship between these tables.

- It ensures the relationships between tables in a database remain accurate by applying constraints to prevent users or applications from entering inaccurate data or pointing to data that doesn't exist.

Referential Integrity Constraint

For referential integrity to hold in a relational database, any column in a base table that is declared a foreign key can contain either a null value, or only values from a parent table's primary key.

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

Dr. S.R.Khonde

Enterprise Constraint

- It is also referred as Semantic Constraints.
- They are additional rules specified by users or database administrators.
- These rules are depending upon the requirements and constraints of the business for which the database system is being maintained.

- eg. A class can have maximum 30 students
- eg. A teacher can teach maximum 2 subject a semester
- eg. A employee can work on max 5 projects at a time

Relational Database Design

Basic elements of design process:

- Defining the problem or objective
- Researching the current database
- Designing the data structures
- Constructing database relationships
- Implementing rules and constraints
- Creating database views and reports
- Implementing the design

Features of Good Relational Design

- Reduce redundancy
- Easy access to data
- More accuracy and integrity of information
- Data entry, updates and deletions should be efficient.

Bad Database design may lead to:

- Repetition of information
- Inability to represent certain information
- Consist of anomalies – Insertion, Deletion , Updation /Modification

Anomalies

Emp_no	Name	Salary	Branch_no	Branch_add
105	Mohan	15,000	B001	Calcutta
108	Sohan	21,000	B001	Calcutta
109	Ruchika	29,000	B002	Delhi
115	Sourabh	18,000	B001	Calcutta
116	Mitalee	35,000	B002	Delhi
117	Ganesh	40,000	B003	Mumbai

Normalization

- Normalization is a database design technique which is used to organize the tables in such a manner that it should reduce redundancy and dependency of data.
- It divides larger tables to smaller tables and links these smaller tables using their relationships.

- Types of Normalization-
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)

Dr. S.R.Khonde

Decomposition

Definition -

The decomposition of a relation schema

$$R = \{A1, A2, \ldots, An\}$$

is its replacement by a set of relation schemes

$$\{R1, R2, \ldots, Rm\}$$

such that

$$Ri \subseteq R \qquad \text{for } 1 \leq i \leq m$$

and $R1 \cup R2 \cup \ldots \cup Rm = R$

Decomposition Example

R = { Emp_no, name, salary, branch_no, branch_add}

Decompose into

R1 = {Emp_no, name , salary}

R2 = {branch_no, branch_add}

Where R1 $\subseteq$ R and R2 $\subseteq$ R and R1 $\cup$ R2 = R

Decomposition Example

STDINF ={Name,Course,Ph_No,Major,Prof,Grade}

Decompose

Student = {Name, Ph_No, Major}

Teacher = {Course, Prof }

Course = {Name, Course, Grade}

Functional Dependency

- The attributes of a relation is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table. This is called functional dependency.

- If attribute A of a relation uniquely identifies the attribute B of same relation then it can represented as

$$A \rightarrow B$$

  which means attribute B is functionally dependent on attribute A.

Functional Dependency

R = { <u>Emp_no</u>, name, salary, branch_no, branch_add}

Functional Dependencies – {emp_no → name, emp_no → salary,
emp_no → branch_no,
branch_no → branch_add}

R = {<u>Name</u>, Course, Ph_No, Major, Prof, Grade}

Functional Dependencies – {name → ph_no,
Name → major,
Course → prof,
Name,course → grade }

Dr. S.R.Khonde

Dependencies and Logical Implications

Consider

relation schema - R

Set of FDs – F

then any functional dependency

$$x \rightarrow y$$

is said to be logically implied from F if that FD can be logically derived from FDs, satisfied on relation schema R

$$F \models x \rightarrow y$$

Inference or Armstrong's Axioms

F1 : Reflexivity : $\mathbf{x \rightarrow x}$

F2 : Augmentation :

$x \rightarrow y \models \mathbf{xz \rightarrow yz}$

F3 : Transitivity :

$x \rightarrow y$ and $y \rightarrow z \models \mathbf{x \rightarrow z}$

F4 : Additivity :

$x \rightarrow y$ and $x \rightarrow z \models \mathbf{x \rightarrow yz}$

F5 : Projectivity :

$x \rightarrow yz \models$ **$x \rightarrow y$ and $x \rightarrow z$**

F6 : Pseudotransitivity :

$x \rightarrow y$ and $yz \rightarrow w \models \mathbf{xz \rightarrow w}$

Example

Eg – R=(A,B,C,D) and F = {A $\rightarrow$ B, A $\rightarrow$ C, BC $\rightarrow$ D}

Using additivity rule A $\rightarrow$ B and A $\rightarrow$ C will be

$$F \models A \rightarrow BC$$

Using transitivity rule A $\rightarrow$ BC and BC $\rightarrow$ D will be

$$F \models A \rightarrow D$$

Closure of Functional Dependency

The set of functional dependencies and all logically implied functional dependencies form a closure of F.

Denoted by F+

Eg – R=(A,B,C,D) and $F = \{A \rightarrow B,\ A \rightarrow C,\ BC \rightarrow D\}$

$$F \models \{ A \rightarrow BC ,\ A \rightarrow D\}$$

$F+ = \{ A \rightarrow B,\ A \rightarrow C,\ BC \rightarrow D ,\ A \rightarrow BC ,\ A \rightarrow D \}$

Closure of Attribute Set

It is a set of all attributes that are dependent on X and derived using the FDs in F.

Denoted by X+

Algorithm to compute X+

X+ = X (where X is candidate key)

while (changes to X+) do

for each FD w → z in F do

begin

if w ⊆ X+ then

X+ = X+ ∪ z

end

Dependencies

Full Functional Dependency

Given a relation schema R and an FD $x \rightarrow y$,

y is fully functionally dependent on x

if there is no z,

where z is proper subset of x

such that $z \rightarrow y$

Eg – F = { $ab \rightarrow c$, $b \rightarrow c$ } where ab is CK

As c is depend on subset b

So c is not fully functionally dependent on ab

Dependencies

Partial Dependency

Given an relation R with the functional dependencies F defines on the attributes of R and K as a candidate key, if X is a proper subset of K and if $F \models X \rightarrow A$, then A is said to be partially dependent on K.

Eg – $F = \{ ab \rightarrow c , b \rightarrow c \}$

If ab is candidate key

Then as c is depend on subset b

So c is partially dependent on b

Dr. S.R.Khonde

Dependencies

Transitive Dependency

Given a relation schema R with Fds F defines on the attributes X,Y, and A.

If the set of Fds contains $X \rightarrow Y$ and $Y \rightarrow A$ then we can say that $X \rightarrow Y \rightarrow A$

so attribute A is transitively dependent on X.

Example

R={ name, course, grade, ph_no, major, course_dept}

F = { course → course_dept

name → ph_no,

name → major,

name, course → grade }

Candidate Key – name,course

Find Full functional dependency and partial functional dependency?

First Normal Form (1NF)

A relation is in First Normal Form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

OR

- An attribute (column) of a table cannot hold multiple values.
- It should hold only atomic values.

First Normal Form (1NF)

Example: Suppose a company wants to store the names and contact details of its employees.

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.

First Normal Form (1NF)

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Note: Using the **First Normal Form**, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Dr. S.R.Khonde

Second Normal Form (2NF)

- A table is said to be in 2NF if the following conditions hold:
  - Table is in 1NF (First normal form)
  - No Partial Dependency
  - Every non-prime attribute should be functionally dependent on prime attribute.

- An attribute that is not part of any candidate key is known as non-prime attribute.

Second Normal Form (2NF) Cont..

Consider relation

R = {stu_name, course, ph_no, dept, grade }

F = { stu_name,course → grade,

stu_name → ph_no,

stu_name → dept }

Primary Key – stu_name,course

Is above relation in 2NF?

Second Normal Form (2NF) Cont..

R = {stu_name, course, ph_no, dept, grade }

Decompose using functional dependencies

such that all functional dependencies preserve.

R1 = {stu_name, ph_no, dept }

R2 = {stu_name, course, grade }

Second Normal Form (2NF) Cont..

R={manufacturer,Model,Model_name, Manu_country}

F = {manufacturer,model → model_name,

manufacturer → manu_country }

Key = {manufacturer,model}

Is in 2NF?

Decompose -

R1 = {manufacturer, model, model_name}

R2 = {manufacturer, manu_country}

Third Normal Form (3NF)

- For a relation to be in Third Normal Form, it must satisfy following conditions :

  - It should be in Second Normal form
  - No non-prime attribute is transitively dependent on prime key attribute (no transitive dependency)

Third Normal Form (3NF) Cont..

R = {emp_id,emp_name,emp_zip,emp_city}

Key = {emp_id}

F = {emp_id → emp_name ,

emp_id → emp_zip,

emp_zip → emp_city }

Is the relation in 3NF ?

No, because of transitive dependency

emp_id → emp_zip → emp_city

Third Normal Form (3NF) Cont..

R = {emp_id,emp_name,emp_zip,emp_city}

Decompose using functional dependency

R1 = {emp_id,emp_name,emp_zip}

R2 = {emp_zip, emp_city}

Third Normal Form (3NF) Cont..

Example -

R = {course,prof,room,room_cap,enroll_limit}

Key = {course}

F = { course $\rightarrow$ prof,

course $\rightarrow$ room,

course $\rightarrow$ enroll_limit,

room $\rightarrow$ room_cap,

room $\rightarrow$ enroll_limit }

Is above relation in 3NF?

Boyce Code Normal Form (BCNF)

BCNF is an extension of Third Normal Form on strict terms.

BCNF states that :

- The relation should be in 3NF
- For any functional dependency, $X \rightarrow A$, X must be a super-key.

Boyce Code Normal Form (BCNF)

R = {emp_id,emp_dept, nationality, dept_type, dept_no }

Key = {emp_id}

F = { emp_id $\rightarrow$ emp_dept,

emp_id $\rightarrow$ nationality,

emp_dept $\rightarrow$ dept_type,

emp_dept $\rightarrow$ dept_no }

Is the relation in BCNF?

R1 = {emp_id, emp_dept,nationality}

R2 = {emp_dept, dept_type, dept_no}

Boyce Code Normal Form (BCNF)

Example

R = { author, nationality, book_title, category, no_of_pages}

Key = {author}

F = {author $\rightarrow$ nationality,

Author $\rightarrow$ book_title,

book_title $\rightarrow$ category,

book_title $\rightarrow$ no_of_pages }

Is the above relation in BCNF?

Decomposition Algorithm

These are two important properties associated with decomposition.

- Lossless Join
- Dependency Preservation

Decomposition Algorithm

Lossless Join -

A decomposition of a relation R into schemes Ri ($1 \leq i \leq n$) is said to be a lossless join decomposition or simply lossless if for every relation (R) the natural join of the projections of R gives the original relation R; i.e.,

$$R = R1 \bowtie R2 \bowtie \ldots \bowtie Rn$$

If $R \subseteq R1 \bowtie R2 \bowtie \ldots \bowtie Rn$ then the decomposition is called lossy.

Decomposition Algorithm

Example: R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

Model Name	Category
a11	Canon
s20	Nikon
a70	Canon

Price	Category
100	Canon
200	Nikon
150	Canon

Decomposition Algorithm

R1 $\bowtie$ R2

Model Name	Price	Category
a11	100	Canon
a11	150	Canon
s20	200	Nikon
a70	100	Canon
a70	150	Canon

R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

Decomposition Algorithm

Dependency Preserving

Given a relation R where F is a set of functional dependencies, R is decompose into the relations R1,R2,...,Rn

with the functional dependencies

F1,F2,...,Fn

Then this decomposition of R is dependency preserving if the closure of $F1 \cup F2 \cup \dots \cup Fn$ is identical to F+

Decomposition Theorem

A decomposition of relation R<(x,y,z),F> into R1<(x,y),F1> and R2 <(x,z),F2> is:

a) dependency preserving if every functional dependency in R can be logically derives from the functional dependencies of R1 and R2 i.e. $(F1 \cup F2)^+ = F^+$

b) is lossless if the common attribute x of R1 and R2 form a key of at least one of these i.e. $x \rightarrow y$ or $x \rightarrow z$

Dr. S.R.Khonde

Decomposition Theorem

Example -

Let R(a,b,c) and F = {$a \rightarrow b$ }

Check weather above relation is lossless and dependency preserving if decompose as

R1 (a,b) and R2 (a,c)

END OF UNIT III