Cursors

Cursors are used to read the data from the table row by row, and process it Step by step procedure to use cursor

1. Declare cursor.

2. declare continue handler to stop the loop

3. open the cursor.

4. fetch the row from the cursor.

5. check whether reached to last row leave the loop

6. process the row.

7. goto step 4

8. once come out of the loop then close the cursor.

delimiter //

create procedure displayallemp()

begin

declare vset,vempno int default 0;

declare vname varchar(20);

declare empcur cursor for select empno,ename from emp; declare continue handler for NOT FOUND set vset=1;

open empcur;

lable1: loop

fetch empcur into vempno,vname;

if vset=1 then

leave lable1;

end if;

select vempno,vname;

end loop;

close empcur;

end//

delimiter ;

2. write a procedure to display all the employees whose sal < avg sal of its own department

delimiter //

create procedure displayempbyavg()

begin

declare vset,vempno,vdeptno int default 0;

declare vename,vjob varchar(20);

declare vsal,vavgsal float(9,2);

declare empcur cursor for select empno,ename,job,sal,deptno from emp; declare continue handler for NOT FOUND set vset=1;

open empcur;

label1:loop

fetch empcur into vempno,vename,vjob,vsal,vdeptno;

if vset=1 then

leave label1;

end if;

select avg(sal) into vavgsal

from emp

where deptno=vdeptno;

if vsal<vavgsal then

select vempno,vename,vjob,vsal,vdeptno,vavgsal;

end if;

end loop;

close empcur;

end//

delimiter ;

to write functions in mysql, we need to set a global variable

set GLOBAL log\_bin\_trust\_function\_creators=1;

write a function to calculate experience of employee

create function calculateexp(ehiredate date) returns int

-> begin

-> declare vexp int;

-> set vexp=floor(datediff(curdate(),ehiredate)/365);

-> return vexp;

-> end//

to call the function

select empno,ename,hiredate,calculateexp(hiredate)

-> from emp;

write a function to generateemail

email should be 3 rd to 6 th character from ename, followed by . and 1 st 3 characters of job, followed by @muycompany.com

delimiter //

create function generateemail(enm varchar(20),ejob varchar(20)) returns varchar(50) begin

declare vemail varchar(50);

set vemail=concat(substr(enm,3,4),'.',left(ejob,3),'@mycompany.com'); return vemail;

end//

delimiter ;

To write trigger

Triggers are used for data analysis purpose or for security purpose. 1. create table to store trigger data

create table dept\_audit(

-> did int,

-> dname varchar(20),

-> old\_dname varchar(20),

-> newloc varchar(20),

-> oldloc varchar(20),

-> username varchar(20),

-> chang\_date date,

action varchar(20));

delimiter //

create trigger updatedepttr before update

on dept

for each row

insert into dept\_audit values(old.deptno,new.dname, old.dname,new.loc,old.loc,user(),curdate(),’update’);

end//

delimiter ;

create trigger insertdepttr before insert

on dept

for each row

insert into dept\_audit values(new.deptno,new.dname, null,new.loc,null,user(),curdate(),'insert');

create trigger deletedepttr after delete

on dept

for each row

insert into dept\_audit values(old.deptno,null,

old.dname,null,old.loc,user(),curdate(),'delete');

In trigger we get 2 special variables old and new

|  | insert | delete | update |
| --- | --- | --- | --- |
| old | null | data | existing data in the table |
| new | data | null | the record after changes are done |

Exception handling

declare <exception-action> handler <exception> <statements>

exception action can be either continue/ exit

exception are of 3 types

• SQLEXCEPTION

• error code

• NOT FOUND

delimiter //

create procedure inserdept(did int, edname varchar(20),edloc varchar(20)) begin

declare exit handler for SQLEXCEPTION select 'error occured';

insert into dept values(did,edname,edloc);

select did,edname,edloc,'duplicate entry';

end//

delimieter ;

Internally mysql stores data in tablespace,

tablespace contains many files

these files are divided into 3 types

1. control file--→table and database metadata is stored is stored in control file 2. data file-→ data is stored in data files.

3. redolog files--→ redo log files are use for roll back and commit

Normalization

1. 1NF, 2NF, 3NF, BCNF

| acid | custid | cname | balance | mobile | email | gender | type | date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 | 100 | Kishori | 33333 | 454645 | aa@gmail | F | Saving | 1 jan  20 |
| 1001 | 100 | Kishori | 66666 | 5555 | aa@gmail | F | current | 1 jan  21 |
| 1003 | 100 | Kishori | 33333 | 5555 | aa@gmail | F | demat | 1 jan  20 |
| 1004 | 200 | Rajan | 55555 | 45454 | r@gmail | M | saving | 1 jan  19 |
| null | 201 | Revati |  | 5666 | ww@gmail | F |  |  |

Revati came to bank for enquiry, but she did not open the account, so no acid is there , hence we will not be able to add her entry in the table to retain customer information, this is called insertion anamoly

If kishori changes her phone number for a/c 1000, it will not get reflected in other accounts, this is called as updation anamoly

if rajan closes the account, then bank will loose customer information along with account information, it is called as deleteion anamoly.’

| acid | custid | balance | type | date |
| --- | --- | --- | --- | --- |
| 1000 | 100 | 33333 | Saving | 1 jan  20 |
| 1001 | 100 | 66666 | current | 1 jan  21 |
| 1003 | 100 | 33333 | demat | 1 jan  20 |
| 1004 | 200 | 55555 | saving | 1 jan  19 |
| null | 201 |  |  |  |

| custid | cname | mobile | email | gender |
| --- | --- | --- | --- | --- |
| 100 | Kishori | 6666 | aa@gmail | F |
| 200 | Rajan | 45454 | r@gmail | M |
| 201 | Revati | 5666 | ww@gmail | F |

1NF

According to the E.F. Codd, a relation will be in 1NF, if each cell of a relation contains only an atomic value. This normal form states that an attribute of a relation cannot hold multiple values. It should hold only single-valued attributes. Values stored in an attribute should be of the same domain.

| Stud  id | Sna  me | cid | Cname | Fid | Fname | Email | mar  ks |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Djh | 100 | Database | 1 | Kishori | abc@gmail.com,wert@rediff.c om | 99 |
| 1 | Djh | 101 | Java | 2 | Madhura | abc@gmail.com,wert@rediff.c om | 99 |
| 2 | ettty | 100 | Database | 1 | Kishori | eee@gmail.com,wwww@yah oo.com,rrr@rediff.com | 98 |
| 2 | ettty | 102 | Data  structure | 2 | Ganesh | Eee11@gmail.com,wwww123 @yahoo.com | 98 |

| 1 | Djh | 100 | Database | 1 | Kishori | abc@gmail.com, | 99 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Djh | 100 | Database | 1 | Kishori | wert@rediff.com | 99 |
| 1 | Djh | 101 | Java | 2 | Madhura | wert@rediff.com |  |
| 1 | Djh | 101 | Java | 2 | Madhura | abc@gmail.com | 99 |
| 2 | ettty | 100 | Database | 1 | Kishori | eee@gmail.com,wwww@yah oo.com,rrr@rediff.com | 98 |
| 2 | ettty | 102 | Data  structure | 2 | Ganesh | Eee11@gmail.com,wwww123 @yahoo.com | 98 |

2NF

According to the E.F. Codd, a relation is in **2NF,** if it satisfies the following conditions: • A relation must be in 1NF.

• And the candidate key in a relation should determine all non-prime attributes or no partial dependency should exist in the relation.

**Prime attributes:** The attributes which are used to form a candidate key are called prime attributes.

**Non-Prime attributes:** The attributes which do not form a candidate key are called non-prime attributes.

**Partial Dependency:** If a non-prime attribute can be determined by the part of the candidate key in a relation, it is known as a partial dependency. Or we can say that, if L.H.S is the proper subset of a candidate key and R.H.S is the non-prime attribute, then it shows a partial dependency.

**Example of partial Dependency:** Suppose there is a relation **R** with attributes **A, B,** and **C.**

prime attributes

sid, cid

non prime attributes

sname, cname, fid, fname, email,marks

sid--→ sname,email,

cid -→ cname

sid+cid→ marks, fid, fname, marks

student\_course

| Stud  id | cid | Fid | Fname | mar  ks |
| --- | --- | --- | --- | --- |
| 1 | 100 | 1 | Kishori | 99 |
| 1 | 101 | 2 | Madhura | 99 |
| 2 | 100 | 1 | Kishori | 98 |
| 2 | 102 | 2 | Ganesh | 98 |

student

| Stud  id | Sna  me | Email |
| --- | --- | --- |
| 1 | Djh | abc@gmail.com |
| 1 | Djh | wert@rediff.com |
| 2 | ettty | eee@gmail.com, , |
| 2 | ettty | wwww@yahoo.com |
| 2 | ettty | rrr@rediff.com |

Course

| cid | Cname |
| --- | --- |
| 100 | Database |
| 101 | Java |
| 102 | Data  structure |

3NF

**Third Normal Form (3NF)**

According to the E.F. Codd, a relation is in **third normal form (3NF)** if it satisfies the following conditions:

• A relation must be in second normal form (2NF).

• And there should be no transitive functional dependency exists for non prime attributes in a relation.

Third Normal Form is used to achieve data integrity and reduce the duplication of data.

A relation is in 3NF if and only if any one of the following conditions will satisfy for each non-trivial functional dependency **X→ Y:**

1. X is a super key or candidate key

2. And, Y is a prime attribute, i.e., Y is a part of candidate key.

**Transitive Dependency:** If **X → Y and Y→ Z** are two functional dependencies, **X → Z** is called as a transitive functional dependency.

prime attribute-→non prime --→ nonprime

the following table is not in 3NF

student\_course

| Stud  id | cid | Fid | mar  ks |
| --- | --- | --- | --- |
| 1 | 100 | 1 | 99 |
| 1 | 101 | 2 | 99 |
| 2 | 100 | 1 | 98 |
| 2 | 102 | 3 | 98 |

| Fid | Fname |
| --- | --- |
| 1 | Kishori |
| 2 | Madhura |
| 3 | Ganesh |

| Stud  id | Sna  me |
| --- | --- |
| 1 | Djh |
| 2 | ettty |

| Stud  id | Email |
| --- | --- |
| 1 | abc@gmail.com |
| 1 | wert@rediff.com |
| 2 | eee@gmail.com, , |
| 2 | wwww@yahoo.com |
| 2 | rrr@rediff.com |

**Boyce-Codd Normal Form (BCNF)**

Boyce-Codd Normal Form (BCNF) is the advance version of the third normal form (3NF) that’s why it is also known as a **3.5NF**

According to the E.F. Codd, a relation is in **Boyce-Codd normal form (3NF)** if it satisfies the following conditions:

• A relation is in 3NF.

• And, for every functional dependency, **X → Y,** L.H.S of the functional dependency (X) be the super key of the table.

In this example, we have a relation R with three columns: Id, Subject, and Professor. We have to find the highest normalization form, and also, if it is not in BCNF, we have to decompose it to satisfy the conditions of BCNF.

**Id Subject Professor**

101 Java Mayank

101 C++ Kartik

102 Java Sarthak

103 C# Lakshay

104 Java Mayank

**Interpreting the table:**

• One student can enroll in more than one subject.

o Example: student with **Id 101** has enrolled in **Java and C++**.

• Professor is assigned to the student for a specified subject, and there is always a possibility that there can be multiple professors teaching a particular subject.

**Finding the solution:**

• Using Id and Subject together, we can find all unique records and also the other columns of the table. Hence, the Id and Subject together form the primary key. • The table is in 1NF because all the values inside a column are atomic and of the same domain.

• We can't uniquely identify a record solely with the help of either the Id or the Subject name. As there is no partial dependency, the table is also in 2NF.

• There is no transitive dependency because the non-prime attribute i.e., Professor, is not deriving any other non-prime attribute column in the table. Hence, the table is also in 3NF.

• There is a point to be noted that the table is not in **BCNF (Boyce-Codd Normal Form)**. **Why is the table not in BCNF?**

As we know that each professor teaches only one subject, but one subject may be taught by multiple professors. This shows that there is a dependency between the subject & the professor, and the subject is always dependent on the professor **(professor -> subject).** As we know that the professor column is a non-prime attribute, while the subject is a prime attribute. This is not allowed in BCNF in DBMS. **For BCNF, the deriving attribute (professor here) must be a prime attribute.**

How to satisfy BCNF?

In Example 3, we will decompose the table into two tables: the Student table and the Professor table to satisfy the conditions of BCNF.

Student Table

**P\_Id S\_Id Professor**

1 101 Mayank

2 101 Kartik

3 102 Sarthak

4 103 Lakshay

5 104 Mayank

Professor Table

**Professor Subject**

Mayank Java

Kartik C++

Sarthak Java

Lakshay C#

Mayank Java

Professor is now the primary key and the prime attribute column, deriving the subject column. **Hence, it is in BCNF.**

Proj Proj Proj Empno Ename Grade Sal Proj Alloc Code Type Desc scale Join Date Time 001 APP LNG 46 JONES A1 5 12/1/1998 24 001 APP LNG 92 SMITH A2 4 2/1/1999 24 001 APP LNG 96 BLACK B1 9 2/1/1999 18 004 MAI SHO 72 JACK A2 4 2/4/1999 6 004 MAI SHO 92 SMITH A2 4 5/5/1999 6

It is in 1NF

is it in 2NF

to check partial dependency

proj code--→ proj type, project description empno-→ename, grade,sal

projcode+empno-→ joining date, allocation time

project

Proj Proj Proj

Code Type Desc

001 APP LNG

001 APP LNG

001 APP LNG

004 MAI SHO

004 MAI SHO

employee

Empno Ename Grade Sal

scale

46 JONES A1 5

92 SMITH A2 4

96 BLACK B1 9

72 JACK A2 4

92 SMITH A2 4

proj emp

Proj Empno Proj Alloc Code Join Date Time 001 46 12/1/1998 24 001 92 2/1/1999 24 001 96 2/1/1999 18 004 72 2/4/1999 6 004 92 5/5/1999 6 following table is not in 3 NF

employee

Empno Ename Grade Sal

scale

46 JONES A1 5

92 SMITH A2 4

96 BLACK B1 9

72 JACK A2 4

92 SMITH A2 4

empno-→grade-→ salary scale

Empno Ename Grade

46 JONES A1

92 SMITH A2

96 BLACK B1

72 JACK A2

92 SMITH A2

grade

Grade Sal

scale

A1 5

A2 4

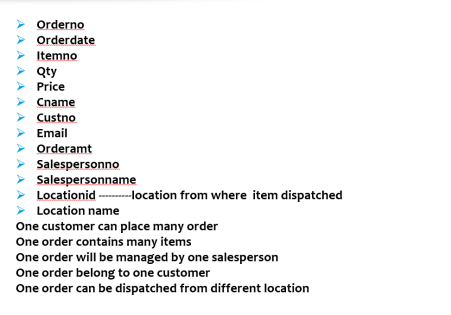
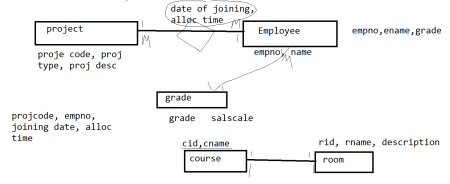
B1 9

A2 4

A2 4

| one-one | any one side key can be added into another side |
| --- | --- |
| one- many, many  one | add key of one side into many side table as a foreign key |
| many-many | create new table and add primary key of both sides |

ER diagram(Entity Relation diagram)



| ORDE  R ID | DAT  E | ITEMN  O | QT  Y | PRIC  E | CI  D | CNAM  E | EMAIL | AM  T | SI  D | SNAM  E | LI  D | L  NAME |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 10  APR | 100 | 3 | 300 | 10  0 | Kishor  i | aa@d  kj | 300  0 | S1 | xxx | 1 | delhi |
| 1 | 10  APR | 200 | 4 |  | 10  0 | kishor  i | aa@d  kj | 300  0 | S1 | xxx | 2 | Mumb  ai |
| 2 | 11a  pr | 100 | 4 | 200 | 10  1 | Revati | r@wej | 500  0 | S2 | yyy | 1 | Delhi |
| 2 | 11  apr | 200 |  |  | 10  1 | Revati |  |  |  |  |  |  |

Is it in 1NF------yes

Is it in 2NF

1. It should be in 1 NF ----yes

2. Check for partial dependency

Prime attribute ----- orderno, item no

Orderno---→order date,cname,cno,email,orderamt,salespersonid,salespersonname, Itemno

orderno, item no---→qty,price,locationid, location name

order

(Orderno,order date,cname,cno,email,orderamt,salespersonid,salespersonname

Cno--→cname,email

Salesperson id-→sname

Order\_item

(orderno, item no,qty,price,locationid, location name

Check for 3NF

(Orderno,order date,cname,cno,email,orderamt,salespersonid,salespersonname Is it in 3 NF ---no

Cno--→cname,email

Salesperson id-→sname

Customer

(cno,cname,email)

Salesman

(Salesperson id,sname)

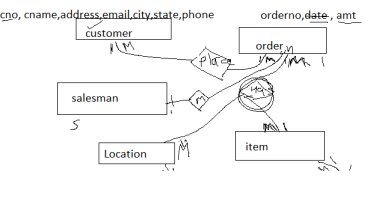
order

(Orderno,order date, cno, ,orderamt,salespersonid)

Location(location id,lname)

Order\_item

(orderno, item no,qty,price,locationid)



Types of models

Conceptual model

If you draw ER diagram with entity name and relation

Logical model

In conceptual model if you add list of attribute then it is logical model

Physical model

In logical model if you define data types of each attribute, primary key,foreign key