

5/8/24

* Super-Key:

(combinations)

Candidate Key (^{subset of}
super key)

Attribute or set of attributes which identifies each row uniquely (CNIC & Date of Birth) (Combination)

* Primary Key:

subset of Candidate Key and super key.

Persons

Date of Birth	CNIC	Fname	Lname	Passport	City	Religion
14-5-1980	315-7896	Ali	Ahmed	R 35147	Lahore	Muslim
14-5-1990	178-9167	George	Brown	R 17876	Lahore	Christian
13-7-1995	189-7645	Bilal	Shareef	R 13843	Karachi	Muslim
17-7-1998	478-9715	Amna	Ashraf	R 39561	Karachi	Muslim
:	:	:	:	:	:	:
:	:	:	:	:	:	:

* X (Date of birth, fname, lname)

(Not a candidate key)

✓ (CNIC, religion)

: Candidate key
(Identifies uniquely)

✓ (CNIC, fname, lname)

X (fname, lname, city)

✓: candidate key

✓ (CNIC, city)

X: Not candidate key

✓ (CNIC)

✓ (CNIC, date of birth)

* PK would be CNIC (Primary Key)

④ Foreign Key: Primary Key used in other table to make data consistent is called foreign key.

PK

Courses

Course_code	Course_name	Course description	Course credit hours	Course quality
CS-PF	Programming Fundamentals	PF with C#	3	A
IT-PF	Programming Fundamentals	PF with python	3	B
CS-DB	Database System	Database concept	3	C
IT-DB	Database System	Database core	3	D

FK

Class_code	Course_code	Room #
CS-PF-M	CS-PF	3
IT-DF-M	IT-PF	4
CS-DB-M	CS-DB	5
IT-DB-M	IT-DB	4

X IT-DBS (It not allows to add)

* COURSE code present in Courses

table and classes table to make relation. PK of courses will be FK in classes.

* If any other course is added in classes as course code IT-DBS,

it will not be added in it
and shows error.

* Super-Key must uniquely
identifies (concept Revised)

* Constraints of Database:

(i) Primary Key constraints (Mutually unique)

(ii) Foreign Key constraints (not null)

* Secondary Key: used for retrieval

of data from the table.

* Integrity of Data achieving
means correctness of data:

(i) Entity Integrity Rule (Entries unique)

(ii) Referential Integrity Rule
(May be null or any reference table value)

* SQL → DDL

(Data definition language)

→ CREATE table

→ DROP table

→ ALTER table

: Structure
defines

: Change in
schema

* Add column, change column/ Data type
drop column, constraint, create/update/delete

④ Data manipulation (DML):

→ Insert, update 'delete'

⑤ Common SQL Datatypes:

pg: 254
slide: 280

① Numeric:

→ Number(L,D) or Numeric(L,D)

→ Integer

→ SmallInt : L: Length

→ Decimal(L,D) : D: Decimal

② Character:

→ Char(L)

→ VarChar(L) or varchar(L)

③ Date:

→ DATE

⑥ CREATE Table:

CREATE TABLE tablename (

column 1 data type constraint,

column 2 datatype constraint

PRIMARY KEY

FOREIGN KEY

CONSTRAINT);

→ Constraints :

- { ① Primary Key (Unique or not)
 null
- ② Foreign Key
- { ③ UNIQUE
- ④ NOT NULL

* CREATE TABLE Courses (

course_code varchar(5) PRIMARY KEY

course_description varchar(100),

course_name varchar(50),

course_credits_hours SMALLINT

course_guideline varchar(500)

);

→ CREATE TABLE Classes (

class_code varchar(5) PK,

course_code varchar(5),

Room_no INTEGER

FOREIGN KEY (course_code)

References courses

);

: It's not necessary that every table contains foreign key but primary key is necessary.

① INSERT INTO TABLE:

→ INSERT INTO COURSES

(course_code, course_name,
course description, course_credit
hours, course guidelines)

(define
our order
of insert)

values ("CS-PF", "Programming Fundamentals",
"This is PF courses", 3, "ABC")
(opt)

→ INSERT into classes values ("CS-PF-M",
"CS-PF", 4);

(without programming
and following
order)

② UPDATE:

→ Update classes set Roomno=4

where class_code = "CS-PF-M"

: where is
important
otherwise applied
in whole

③ DELETE:

→ Delete from classes

where class_code = "CS-PF-M"

: table
exists
but rows
deleted

: delete
specific

① Check:

Delete from courses

where course-code = "IT-DB"

(Not work because course-code
is Primary Key)(we have to
update in classes also to delete).

12/8/24

① DDL:

- Data Definition Language
- CREATE Table / constraint
 - DROP Table / constraint
 - ALTER Table / constraint

② Integrity Rules:

PRIMARY KEY

DDL

FOREIGN KEY

DML

③ Constraints:

{ ① NOT NULL

④ FOREIGN KEY

② UNIQUE

⑤ Check (Defined condition)

③ PRIMARY KEY

⑥ Default (Value assigned by system)

Name
Assign

* ALTER (change one or more columns) (structure change)
(Column data modify, Insert new one, drop one column).

—0_n (Maybe zero or multiple)
(Zero or many)
—0₁ (zero or one).

† Customer:

```
CREATE TABLE customer(  
    cus_code Integer PRIMARY KEY,  
    cus_lname Varchar(50),  
    cus_fname Varchar(50),  
    cus_initial Varchar(50),  
    cus_areacode Integer,  
    cus_phone Integer,  
    cus_balance Integer  
)
```

```
→ CREATE TABLE INVOICE/  
    inv_number Integer PRIMARY KEY,  
    inv_date Date,  
    cus_code Integer
```

: CHECK (column IN ())

FOREIGN KEY (cus_code)

references customer

);

: (constraints name assignment UNIQUE())

(CONSTRAINTS CUS_U11 UNIQUE(^{by me}_{from}))

(at last of CREATE DDL statement)

→ CREATE TABLE INVOICE_1(

INV_NUMBER NUMBER PRIMARY KEY,

CUS_CODE NUMBER,

INV_DATE DATE DEFAULT SYSDATE,

FOREIGN KEY (CUS_CODE)

REFERENCES CUSTOMER

);

→ CREATE TABLE LINES(

INV_NUMBER NUMBER PRIMARY KEY,

LINE_NUMBER NUMBER,

P_code NUMBER

);

→ CREATE TABLE XXZ(

FNAME VARCHAR (15),

LNAME VARCHAR (15),

PRIMARY KEY (FNAME,LNAME)).

* ① DB → Semester Project:

- Deadline 15-08-24
- 1 - Choose System
 - 2 - Enlist its requirements
 - 3 - Enlist its features
 - 4 - Design Database Model
 - 5 - Design Python App

→ CREATE TABLE LINES (

INV-NUMBER PK,

);

* ② On-Update Cascade (Deleting courses is possible when updated in classes) (Search it) *

→ Create table lines(

inv-number varchar(50),

line-number varchar(50),

p-code Varchar(50),

Line-units smallint,

Line-price smallint,

PRIMARY KEY(inv-number, line number)

FOREIGN KEY (p_code) References Product

FOREIGN KEY (inv_number) References Invire

) ;

19/8/24

④ Lucid Charts (for ERD)

* JOIN:

TO relate columns of
two tables (INNER, OUTER, LEFT,
RIGHT and FULL)

Types

1- INNER JOIN (only similar rows)

2- OUTER JOIN

Left Right

Orders		
Order-id	Customer-id	Order Date
1	4	13-1-22
2	9	14-2-22
3	8	8-1-23
7	NULL	1-2-24
18	1	5-8-23

Inner Join Eg:

→ Select order_id, customer_id, customer_name, order_date
from orders INNER JOIN customers on orders.customer_id =
(customers.customer_id -
(This incorrect format))

(Not specified about
customer_id of which table)

PK		
Customer-id	Customer-name	Customer-city
1	Ali	Lah
2	Ahmad	Lah
4	Abdullah	Lah
8	Fahad	Kar
3	Amir	ISL

④ Left Join (First table record and also the common record of other table).

: select order_id, customer_id, customer_name from orders, customers

where orders.customer_id = customers.customer_id
(without using join syntax)
(join one or more table by above syntax)

⑤ select order_id, order_date, customer.fname, employees.lname

from orders, customer, employee

where orders.employee_id = employees.employee_id
AND orders.customer_id = customer.id

: (Three tables joined)

id, date,
fname of
customer,
lname of
employee

customer_id

⑥ JOINS increases execution time

(NOSQL solves this issue).

⑦ Self-join (joins records of itself).

(<>) (Not equal operator).

⑧ Group By (Aggregate function)

→ Select max(salary) from employees
group by deptno.

* Group By or Aggregate functions, if criteria required then we wing HAVING statement.

* Query ① (Display order details (id, product name, quantity, price))

Query ② (Total amount of each order).

(i) Select order-id, product-name, product-quantity, unit^{price} from order details, products where order^{details}.product-id = products.product-id.

(ii) Select discount, orders.order-id, order-id from order-details, order
where order-details.order-id = orders.order-id

Next class Quiz

21/8/24

* Chapter # 8 pg: 349

* JOINS and subquery:

→ Select → From

→ Where → Having

→ Select * from table
where (subquery).

① Subquery can return:

(i) One single value

(ii) A list of values

(iii) A virtual table

② Where clause in Subquery

③ Having clause in Subquery.

④ Any and all (Greater than all)
(Greater than atleast one)

⑤ Correlated Subquery : QOT
will work as nested loop.

⑥ Virtual table.

28/8/24

→ Query to display department wise
count of employees.

emp-num	dept-num	emp-name	sal	hire-date
---------	----------	----------	-----	-----------

: select deptno, count(*) as
Count from emp group by deptno

* **View** → Virtual table : to repeat specific query.

: Create view ^{DeptCountEmp} AS

(Select dept-num, count(emp-no)
as EmployCount from emp
group by dept-num)

: Select * from DepartCountEmp

→ View will be updated as table
updated later.

→ If primary key is identified
then DML operations can
be applied, otherwise not.

→ Security purpose or access
limit to user can be provided
by view. (Base table columns
can be added into view, so
limited access can be provided to user).

→ How to use DB with
high-level language:

→ DB } Oracle / MySQL
DBMS }

→ API / Package / Driver Program.
(source to communicate).

→ PyMySQL (API) (Package : pymysql)
(in MySQL)

① Steps:

- ① Import the package that communicate with dbms.
- ② Create database connection
- ③ Create cursor
- ④ Execute Query
- ⑤ Close connection & cursor

* import pymysql
function start ← dbcon = None : None
dbcursor = None (Null)

try :

dbcon = pymysql.connect

(host="localhost", user="root",
password="123", database="test")

dbcursor = dbcon.cursor()

query = "Select email, password
from users"

dbcursor.execute(query)

row = dbcursor.fetchone()

pycharm
(IDE)
MySQL Command Line

row2 = dbcursor.fetchone() : 1st method

rows = dbcursor.fetchall() : (for rows:
print(r[0], r[1]))

except Exception as e:

print(str(e))

finally :

if dbcursor!=None:
dbcursor.close() →
if dbcon!=None:
dbcon.close() →

④ For App:

→ Login (Email, Password)

→ def signup(email, password):

" Same as getUserData

query = "Insert into user values (%s, %s)"

args = (email, password) : "tuple"

dbcursor.execute(query, args).

dbcon.commit() : to save

signup = True

:

finally :

:

return signup

email = input("Enter your email to register")

password = input("Enter your password to register")

signup = signup(email, password)

* def signin(email, password):

dbcon = None

dbcursor = None

signin = False

try:

dbcon

dbcursor = dbcon.cursor()

qquery = "Select email, password
from users"

for r in dbcursor.execute(qquery)

signin = True if r[0] == email &

except Exception as e:

str(e)

r[1] == password

finally:

*

qquery = "Select email, password from
users where email=%s
and password=%s"

args = (email, password)

29/8/24

* File Format and Permanent Data Storage . Updating Poorly Designed Databases.

* Composite Primary Key
(PROJ_NUM,

* 1NF (Normalization) (ReDesign table with Proper Primary Key without any repeating group)

(1st. Normal Form) (Non-repeating group)

* Functional dependence (one key is determined by other)

* Partial dependency (one key has not relational with other) (emp-name has direct relation emp-num not with complete primary key) (One key has relational with some keys of composite key).

→ To solve Partial dependency issue , we make other table:

Proj-Num	Proj-Name
----------	-----------

→ Project table, Employee table,
Assignment table (It is 2NF).

④ Transitive dependency

(Make non-key attributes into
separate table and choose
it from other table).

⑤ SQL:

Complete chapter.

⑥ Syllabus:

→ Chapter: 1 complete

→ Chapter: 2 complete

→ Chapter: 3 complete

→ Chapter: 4 complete

→ Chapter: 6 till 3NF

→ Chapter: 7 complete

→ Chapter: 8 Till Views.

→ (conceptual paper) (NF, ERD,
python etc.....)