Sanjivani College of Engineering, Kopargaon
Department of Electronics & Computer Engineering
(An Autonomous Institute)
Affiliated to Savitribai Phule Pune University
Accredited 'A' Grade by NAAC

Subject: Database Management Systems(DBMS) & SQL

by

Mr. Nitin Bhopale

Unit-III: Database modification and Relational Database

Database Modification using SQL Insert, Update and Delete Queries. UTC,TIMESTAMPS. PL/SQL: concept of Stored Procedures & Functions. Cursors, Triggers, Assertions, roles and privileges, Embedded SQL, Dynamic SQL. Relational Model: Basic concepts, Attributes and Domains, Relational Integrity: Domain, Referential Integrities, Database Design: Normalization, Atomic Domains and Normal Form.

Data-Manipulation Language for Relational Database

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	building	budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

select *instructor.name*
from *instructor*
where *instructor.dept name* = 'History';

El Said Califieri

Relational Model:

Relation:

Example:

Rollno	Name	Age
1	A	11
2	B	33
3	C	22

Attributes:

Tuples/Records:

Domain:

Relation name:

Informal	Formal
Table	**Relation**
Column name/header/field	**Attributes**
Rows	**Tuples**
Values /Data Type	**Domain**
Table name	**Relation name**
Table definition	**Schema of relation**

Degree of relation: No of attributes in relation

Rollno	Name	Age
1	A	11
2	B	33
3	C	22

Cardinality of relation: No of tuples in relation

Rollno	Name	Age
1	A	11
2	B	33
3	C	22

Properties of relation

- Distinct relation name
- Each cell of relation have exact one value
- Distinct attribute name
- Distinct tuple name
- No order significance for attribute
- No order significance for tuple

rollno	name	address	marks
1	Abhishek	kopargaon	40
2	Akshay	shirdi	50
3	Abhi	kopargaon	70
4	Akash	rahuri	30
5	Aman	Aurangabad	80
6	Dany	Sangamner	75

Referential Integrity

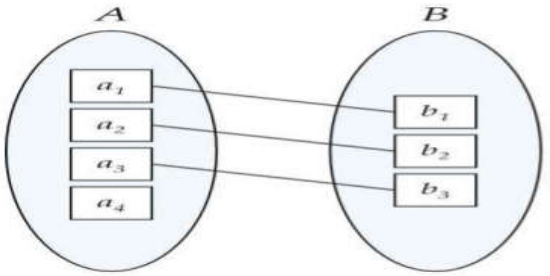
Candidate key

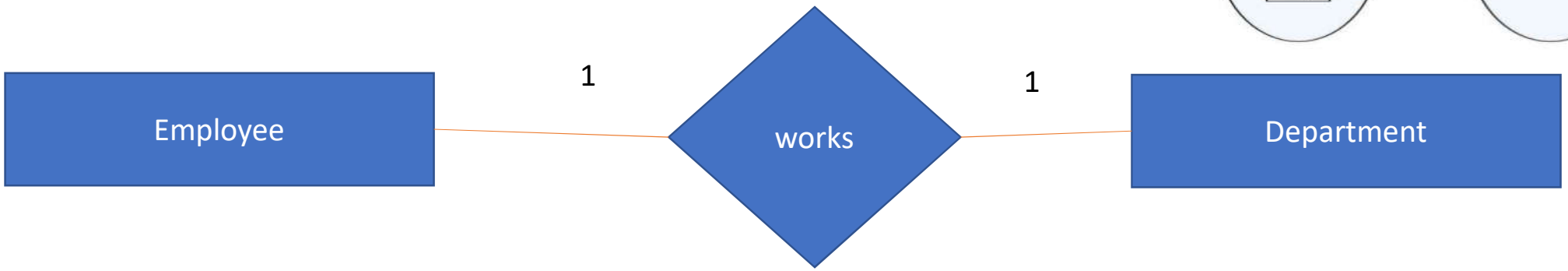
Primary key

Foreign key

rollno	name	address	marks
1	Abhishek	kopargaon	40
2	Akshay	shirdi	50
3	Abhi	kopargaon	70
4	Akash	rahuri	30
5	Aman	Aurangabad	80
6	Dany	Sangamner	75

Relationship(Cardinality) (1:1)





eid	Ename	Eage
E1	A	22
E2	B	21
E3	C	24

Referenced (base table)

eid	did
E1	D1
E2	D3
E3	D2

Referencing table

did	Dname	dloc
D1	IT	Delhi
D2	ETC	Mumbai
D3	MECH	Shirdi

Referenced (base table)

Primary key ?

Merging of table ?

Reduced Table

eid	Ename	Eage	did
E1	A	22	D1
E2	B	21	D3
E3	C	24	D2

Relational Integrity

Self referential Integrity

eid	Ename	Eage	Manager
E1	A	22	D1
E2	B	21	E1
E3	C	24	E2

Referential Integrity

Activity(referencing=derived) → **Student(referenced=base)**

Act_name	Act_date	Winner
Drawing	22/11/21	1
Chess	21/10/22	1
Badminton	24/11/22	2

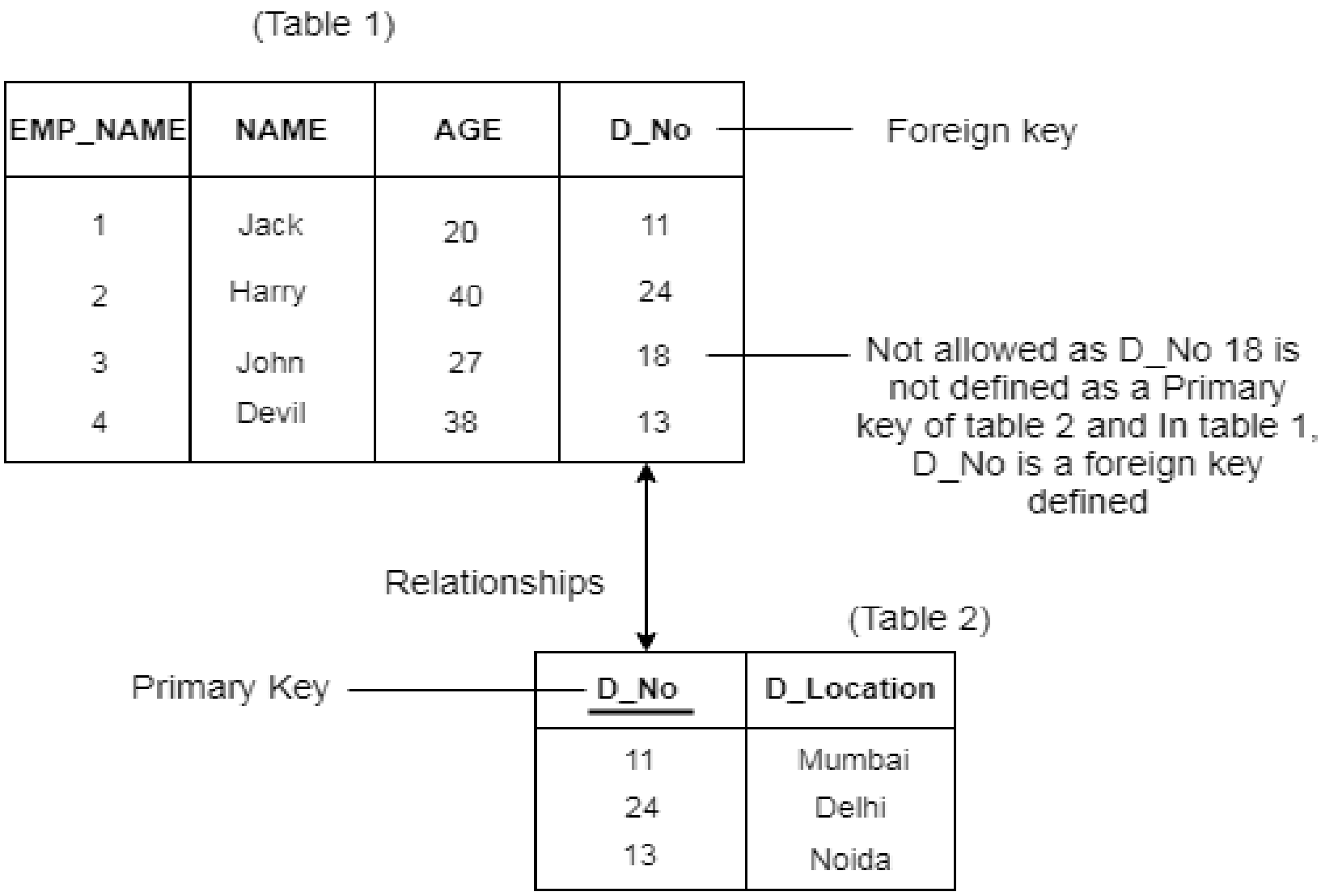
Stud_rn	Sname	Saddress
1	A	shirdi
2	B	kopargaon
3	C	yeola

- **Foreign key used to reduce data redundancy**
- **Referencing attribute must be subset of referenced /referred attribute**

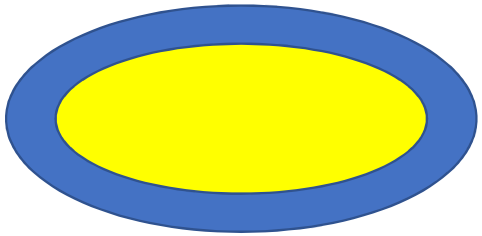
Referential Integrity at same table ➔

Emp_id	Emp_name	Emp_Manager
1	A	---
2	B	1
3	C	2

Referential Integrity Constraints



Types of Attributes



- Simple (EMP_ID)
- Composite (EMP_NAME)
- Single valued (EMP_DOB)
- Multivalued (EMP_EMAILID)
- Derived (EMP_POSERVICE) OR (EMP_AGE)

Functional Dependancy

* It is method that describes relationship between attributes

$\therefore \quad X \longrightarrow Y$ [X determines Y / Y determined by X]

where X → Determinant attribute &

Y → Determined attribute / dependant attribute.

Ex. Srn → Sname

Case ① ∴

Srn	Sname
1	Nilesh
2	Nilesh

✓ Valid case

Case ②

Srn	Sname
1	Nilesh
1	Nilesh

✓ Valid case

Case ③

Srn	Sname
1	Nilesh
2	Amar

✓ Valid case

Case ④

Srn	Sname
1	Nilesh
1	Amar

✗ Invalid case.

FD: (X -> Y)
if (t1.x = t2.x) then
t1.y = t2.y

FD for record fetch ??????

rn	name	marks	Dept	Course
1	A	40	IT	C1
2	B	60	EC	C1
3	A	40	IT	C2
4	B	60	EC	C3
5	C	80	IT	C3
6	D	80	EC	C2

→ rn-> name
→ rn->marks
name->rn
Dept->course
Course->dept
Marks->dept
→ {rn,marks}->marks
{marks,dept}->course
→ {dept,course}->marks

FD: (X -> Y)
if (t1.x = t2.x) then
t1.y = t2.y

Super Key(SK): attribute or set of attribute that uniquely describes the record/tuple/row.
{rn},{rn,name},{rn,marks},….
max no of SK =(2^n)-1
n: no of attributes

CK ??? ???

For R(A,B,C) find out SK,CK ?

A	B	C
1	1	1
2	1	2
3	2	1
4	2	2

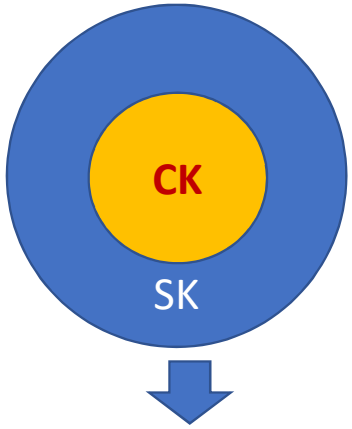
SK: attribute or set of attributes that uniquely describes the record/ tuple/ row.

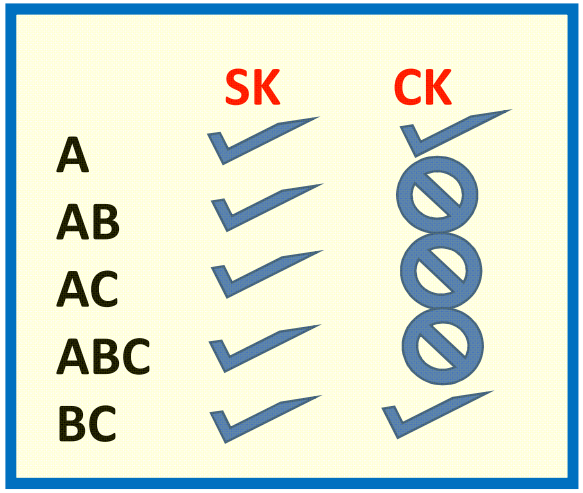
SK➔ {A, AB,AC,ABC,BC}

CK: It is SK where proper subset is not a SK

⇒ *Minimal SK (as min as possible)*

CK➔A,BC





Every CK is SK but every SK will not be CK.

If one proper subset of {B,C,D} is SK then it CAN NOT be CK.

S1={1,2,3} S2={1,2}

S2 C S1 ➔ S2 IS SUBSET OF S1

S2 C S1 ➔ S2 IS PROPER SUBSET OF S1

S2 C S1 ➔ S2 IS SUBSET OF S1 &&&

S1 C S2 ➔ S1 IS NOT SUBSET OF S2

PK ??? ???

For R(A,B,C) find out SK,CK ?

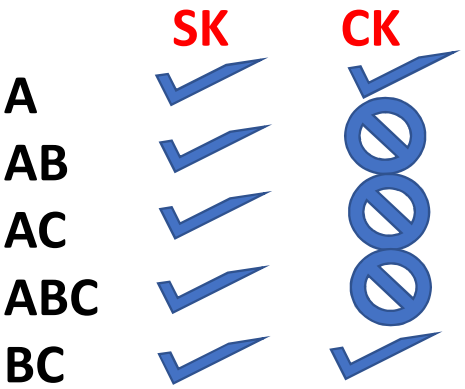
A	B	C
1	1	1
2	1	2
3	2	1
4	2	2

SK: attribute or set of attributes that uniquely describes the record/ tuple/ row.

SK➔ {A, AB,AC,ABC,BC}

CK: It is SK where proper subset is not a SK
⇒ Minimal SK

CK➔A,BC



Every CK is SK but every SK will not be CK.

If one proper subset of {B,C,D} is SK then it CAN NOT BE CK.

PK: One CK with no null value and remaining will be either alternate/secondary key

Therefore A or BC will be PK.
PK SHOULD BE ONLY 1

Types of FD

① Trivial – It is always reflexive & valid

i.e If $X \rightarrow Y$ then **X->Y OR X-> X**

Y is subset of X

$\therefore$ LHS $\cap$ RHS $\neq$ empty (ϕ)

eg. Srn Sname $\rightarrow$ Srn $\neq \phi$.

Note :– Four cases doesnot involve in trivial.

② Non trivial – If $X \rightarrow Y$ then $X \cap Y = \phi$

i.e Srn $\rightarrow$ Sname

Srn $\rightarrow$ Saddress

Srn $\rightarrow$ Scontactno.

Properties of FD (Armstrong's Axioms/ Inference Rules)

* Reflexivity - If Y is subset of X then

THIS IS ALWAYS VALID $X \rightarrow Y$ & $X \rightarrow X$ (eg. Srn → Srn)

* Augmentation — If $X \rightarrow Y$ then $XZ \rightarrow YZ$

(eg. Srn → Sname
Srn Sadd → Sname Sadd)

* Transitive - If $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$

(eg. Srn → Sname Sname → Scity

Then Srn -> Scity

* Union - If $X \rightarrow Y$ & $Y \rightarrow Z$ then

$X \rightarrow YZ$

* Decomposition - If $X \rightarrow YZ$ then

$X \rightarrow Y$ & $X \rightarrow Z$

[If $XY \rightarrow Z$ then $X \rightarrow Z$ $Y \rightarrow Z$ Invalid case]

* Left side can't be decomposed.

* Pseudotransitivity - If $X \rightarrow Y$ & $WY \rightarrow Z$ then

$WX \rightarrow Z$

* Composition - If $X \rightarrow Y$ & $Z \rightarrow W$ then

$XZ \rightarrow YW$.

For R(A,B,C) (a) A is CK find no. of SK ?

$\therefore$ {A, AB, AC, ABC} Possible SK.

(b) if AC is CK. find no. of SK ?

{ACB, AC} => Possible SK.

* Closure Method (Attribute Closure)

It is used to find all the candidate key from table

If for general table attributes are A, B, C, D then

Eg. R(A, B, C, D) can have FD as.

FD { A → B, B → C, C → D }

then Closure of A means (A^+) what A can determine.

Topic ..... Unit No. .....

Attribute Closure / Closure set.

R(A, B, C, D, E)

FD {A → B, B → C, C → D, D → E}

⇒ . A → ABCDE

B → CDEB

C → DEC

E → E

(Attribute Closure) . A^+ ⇒ ~~contains~~ Contains set of attributes determined by A

$A^+ = \{A, B, C, D, E\}$ ⇒ (SK.) [set of attr. whose closure contains all the attributes of given relation]

$AD^+ = \{A, D, B, C, E\}$ ⇒ (SK)

$B^+ = \{B, C, D, E\}$ ⇒ Not SK.

$CD^+ = \{C, D, E\}$ ⇒ Not SK

∴ $AB^+ = \{A, B, C, D, E\}$ (SK) ∵ A → A, B, C, D, E ∴ AB → ——

∴ AC, AD, ABC, ABCDE

∵ $BCDE^+ \Rightarrow \{B, C, D, E\}$ ∴ Total possible SK = (16)

CK ⇒ A AD ⇒ proper subset ⇒ {A} {D}

✓SK ✗CK ✗CK

Find all possible CK for given Relation

$R(A, B, C, D, E)$

$FD = \{A \rightarrow B,\ D \rightarrow E\}$

$A^+ = \{A, B\}$

$BC^+ = \{B, C\}$

$ABCDE^+ = \{A, B, C, D, E\}$

(SK)

$ABDE^+ = \{A, B, D, E\}$ (✗ SK)

$ACDE^+ = \{A, C, D, E, B\}$ (✓ SK)

$ACD^+ = \{A, C, D, B, E\}$ (✓ SK) $\Rightarrow$ CK ***

$AC^+ = \{A, C, B\}$ (✗ SK)

$CD^+ = \{C, D, E\}$ (✗ SK)

$AD^+ = \{A, D, E\}$ (✗ SK)

∵ None of proper subset of ACD is (SK)

*** ∴ ACD is CK

Find all possible CK for given Relation

$R(A, B, C, D, E)$

FD $\{A \to B, D \to E\}$

$(AB)^+ = \{A, B, C, D, E\}$ SK

SK

$(ACD)^+ = \{A, C, D, E, B\}$ SK $\because A \to B$

$ACD^+ = \{A, C, D, B, E\}$ SK $\because D \to E$

$AC^+ = \{$

$CD^+ =$

AD^+

A^+

C^+

D^+

} X SK $\therefore ACD^+ \Rightarrow$ CK & SK.

(Attribute part of CK) $\therefore$ Prime attribute $\{A, C, D\}$

Non prime attribute $\Rightarrow \{B, E\}$.

$\because$ A, C, D not available on RHS of any FD. $\therefore$ Only one CK.

Find all possible CK for given Relation

$R(A, B, C, D)$

FD $\{ A \to B,\ B \to C,\ C \to A \}$

$\Rightarrow$ $(ABCD)^+ = \{A, B, C, D\}$ $\Rightarrow$ SK

$(AD)^+ = \{A, C, D, B\}$ $\Rightarrow$ SK

$(AD)^+ = \{A, D, B, C\}$ $\Rightarrow$ SK $\because A \to B$ & $B \to C$

$A^+ = \{A, B, C\}$ $\Rightarrow$ × SK

$D^+ = \{D\}$ $\Rightarrow$ × SK

$\because$ Proper subsets are not SK $\therefore$ AD^+ is CK

To check for other CK ?

$\because$ A is on RHS of $C \to A$ (from FD.)

$\therefore$ AD^+ can be CD^+ $\Rightarrow$ SK

$\therefore$ $C^+ = \{C, A, B\}$ $\Rightarrow$ × SK

$D^+ = \{D\}$ $\Rightarrow$ × SK.

$\therefore$ CD^+ is also CK

$\therefore$ Prime attributes are $\{A, D, C\}$

$\therefore BD^+ \Rightarrow SK$

$B^+ = \{B, C, A\} \times SK$

$D^+ = \{D\} \times SK$

$\therefore$ BD is CK

Prime attributes $\{A, D, C, B\}$

$\therefore$ $CK = \{AD, CD, BD\}$

Find all possible CK for given Relation

Find all CK, in a reln

$R\{A, B, C, D, E, F\}$

$F = \{ AB \to C,\ C \to DE,\ E \to F,\ D \to A,\ C \to B \}$

$\Rightarrow (AB)^+ = \{A, B, C, D, E, F\}$ SK

$(ABC)^+ = \{A, B, C, D, E, F\}$ SK

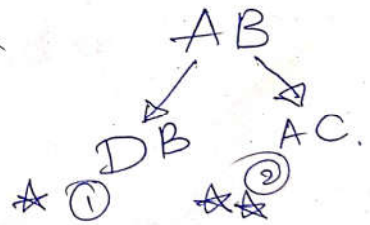
$(ABDE)^+ = \{A, B, C, D, E, F\}$ SK

$(ABF)^+ = \{A, B, C, D, E\}$ SK

$AB^+ = \{A, B, C, D, E, F\}$ SK & CK.

$A^+ = \{A\}$ ✗ SK.

$B^+ = \{B\}$ ✗ SK



∴ AB is ck.

Prime attri: $\{A, B\}$.

$DB^+ = \{D, B, A, C, E\}$.

$D^+ = \{A, D\}$ ✗ SK

$A^+ = \{A\}$ ✗ SK.

∴ DB is CK

∴ Prime attribute is $\{A, B, D\}$

∴ $CB^+ =$ SK.

$C^+ = \{C, B\}$ ✗ SK

$B^+ = \{B\}$ ✗ SK.

∴ CB is CK

∴ Prime attribute is $\{A, B, C, D\}$

(2) ∴ AC → SK but not CK

$A^+ = \{A\}$

$C^+ = \{C, D, E, F, A, B\}$ → SK.

∴ C is CK.

∵ No proper subset of C is SK.

∴ CK = {AB, BD, C}

Non prime attr. = {E, F}

Find all possible CK for given Relation

Start could be

~~A~~B ~~C~~DE ~~F~~ $^+ = \{A, B, C, D, E, F\}$

★ $BDE^+ = \{A, B, C, D, E, F\}$

$BD^+ = \{BDACEF\}$ SK

$B^+ = \{$ ∝ SK?

$PA = \{B, D\}$

BD

CD BC

$CK = \{BD,$

Find all possible CK for given Relation

SOLVE

Eg. $R(A, B, C, D)$ can have FD as.

$FD\{A \to B, B \to C, C \to D\}$

then Closure of A means (A^+) what A can determine

$\therefore A^+ = BCDA$ $\quad [\because A \to B, B \to C \Rightarrow A \to C]$

$\therefore$ 'A' is candidate key (CK) by which all other attributes can be determined

$\therefore B^+ = CDB$.

$\therefore C^+ = CD$

$\therefore D^+ = D$.

Prime attribute is attribute used to make candidate key (CK)

$\therefore$ 'A' is prime attribute

& B, C, D are non prime attribute.

$\therefore$ BC,D cannot be the candidate key.

AB+=(ABCD)
-> is not candidate key BUT IS SK.

Find all possible CK for given Relation

SOLVE

Eg. R(A, B, C, D)

FD $\{ A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D,\ D \rightarrow A \}$

$A^+ = BCDA$

$B^+ = BCDA$

$C^+ = CDAB$

$D^+ = ABCD$

$\therefore$ CK $= \{ A, B, C, D \}$

Prime att. A, B, C, D

& Non prime att. 'Null' or ϕ

Find all possible CK for given Relation

R(A, B, C, D) if FD = { A → B, BC → D, E → C, D → A } Find CK

SOLVE

AE^+ = BDCAE ✓

CK = { AE, } ⇒ { AE, DE, BE }

(D) E^+ ⇒ ADEBC.

(B) E^+ ⇒ BECDA

CE^+ ⇒ CE ∝ No CK.

Normalization

It is technique to reduce or remove redundancy from table.

Eg.

	Srn	Sname	Sage
*	1	Amit	21
	2	Nilesh	20
*	1	Amit	21

∴ This is row level duplicacy.

This can be avoided by introduction of Primary key.

Eg.

(PK) Srn	Sname	Cid	Cname	Fid	Fname
1	Nitesh	30	C	100	Arun
2	Amar	31	C++	101	Chandrakant
3	Nilesh	30	C	100	Arun
4	Mahesh	32	Java	102	Samir
5	Nil	30	C	100	Arun.

This is column level duplicacy Because of which 3 problems

CreatedHere. Insertion anomaly → . i.e If only new course is introduced

Deletion anomaly → i.e deletion of entire information from row.

SQL → delete from tablename where Srn = 3.

Updation anomaly → i.e Need to update at all possible location

update student set Sname = 'XYZ' where Srn = 1.

- Solution is Normalization.

Srn	Sname

Cid	Cname

Fid	Fname

i.e form 3 tables for the redn of anomalies.

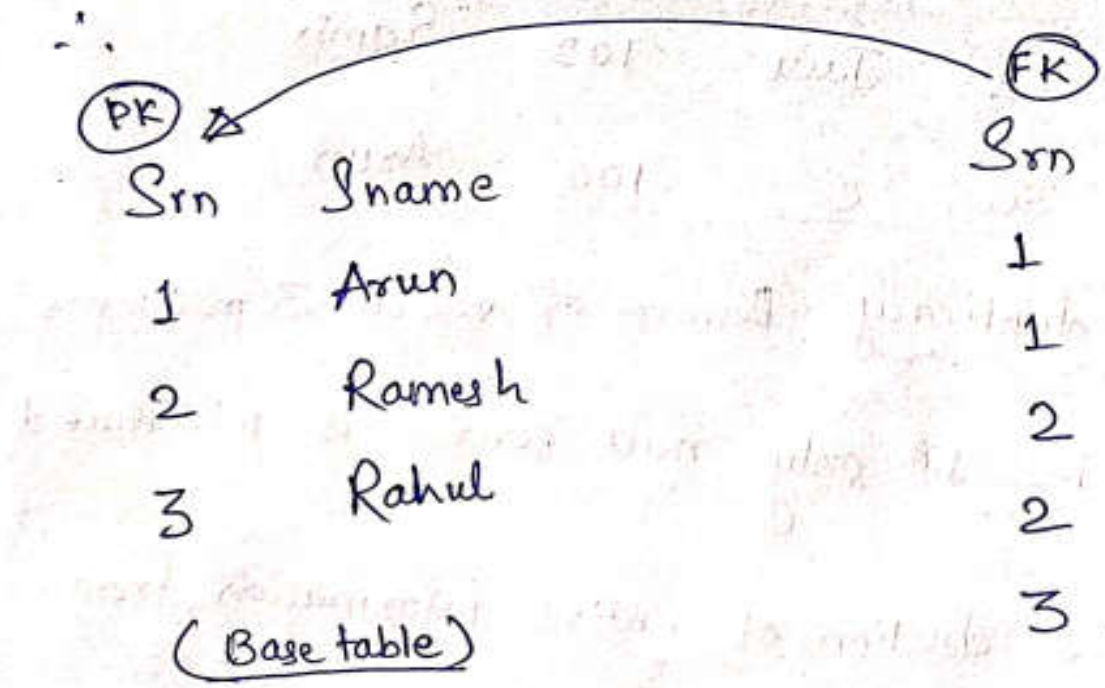
1st NF

Srn	Sname	Scourse
1	Arun	C/C++
2	Ramesh	Java, .NET
3	Rahul	SQL

Unique Single values not present
∴ 1st NF invalid

Srn	Sname	Scourse1	Scourse 2
1	Arun	C	C++
2	Ramesh	Java	.NET
3	Rahul	SQL	Null

→ This is not applicable if more entries.



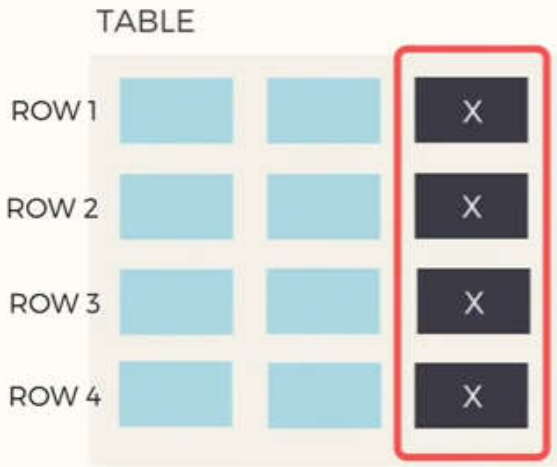
PK ⇒ Srn

PK ⇒ Srn + Scource

Composite key.

normalization is

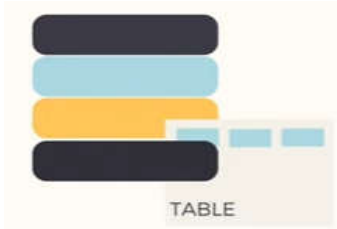
a technique of organizing the data into multiple related tables, to minimize DATA REDUNDANCY.



- Repetition of data increases the size of database.
- Other issues like:
  - Insertion Problems
  - Deletion Problems
  - Updation Problems

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337



issues due to redundancy.

1. Insertion Anomaly
2. Deletion Anomaly
3. Updation Anomaly

Deletion Anomaly

Loss of a related dataset when some other dataset is deleted.

Insertion Anomaly

To insert redundant data for every new row (of Student data in our case) is a data insertion problem or anomaly.

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Mr. X leaves, and Mr. Y joins as the new HOD for CSE

STUDENTS TABLE

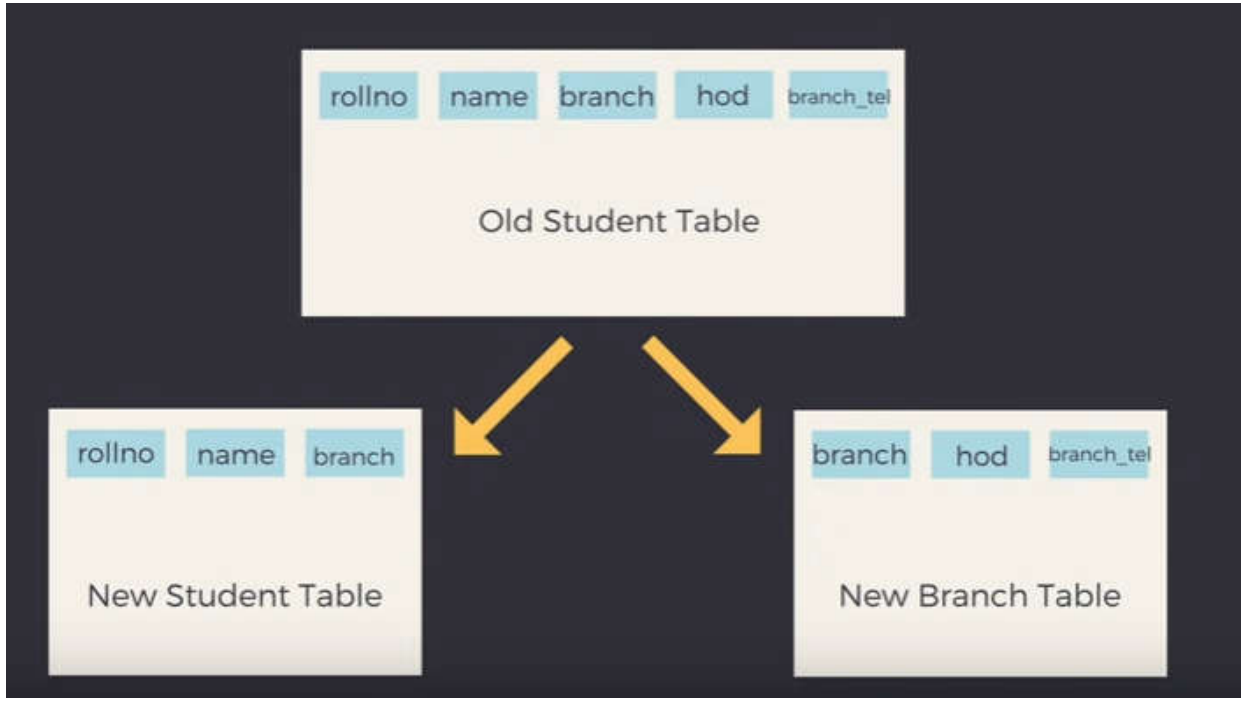
rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	~~Mr. X~~ Mr. Y	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

STUDENTS TABLE

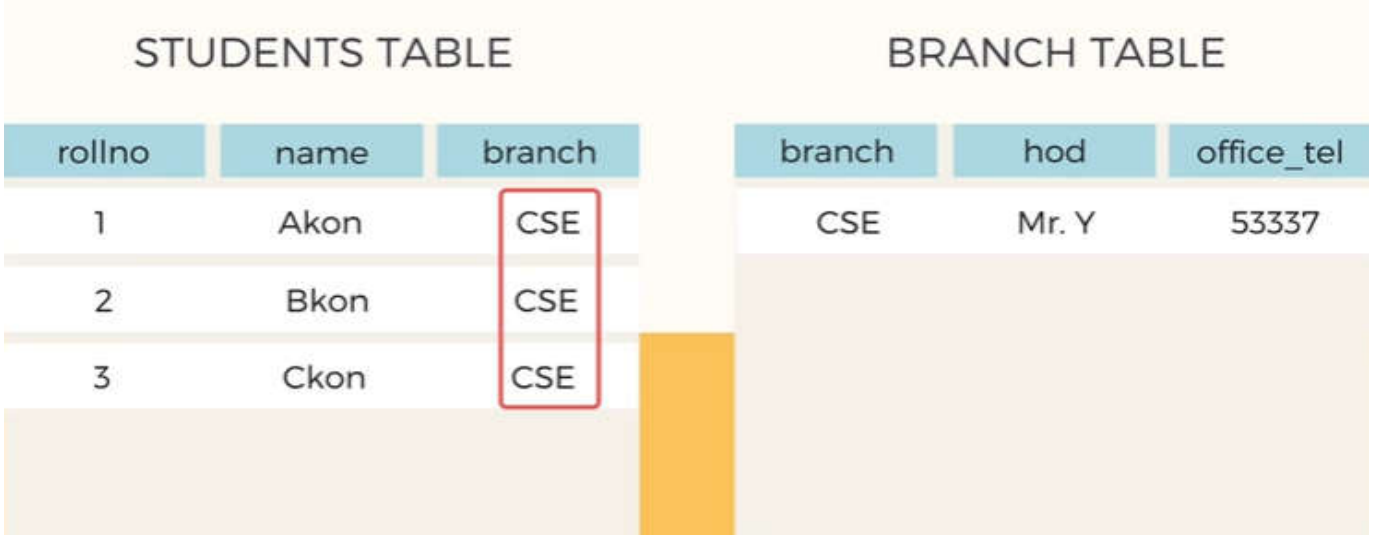
rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

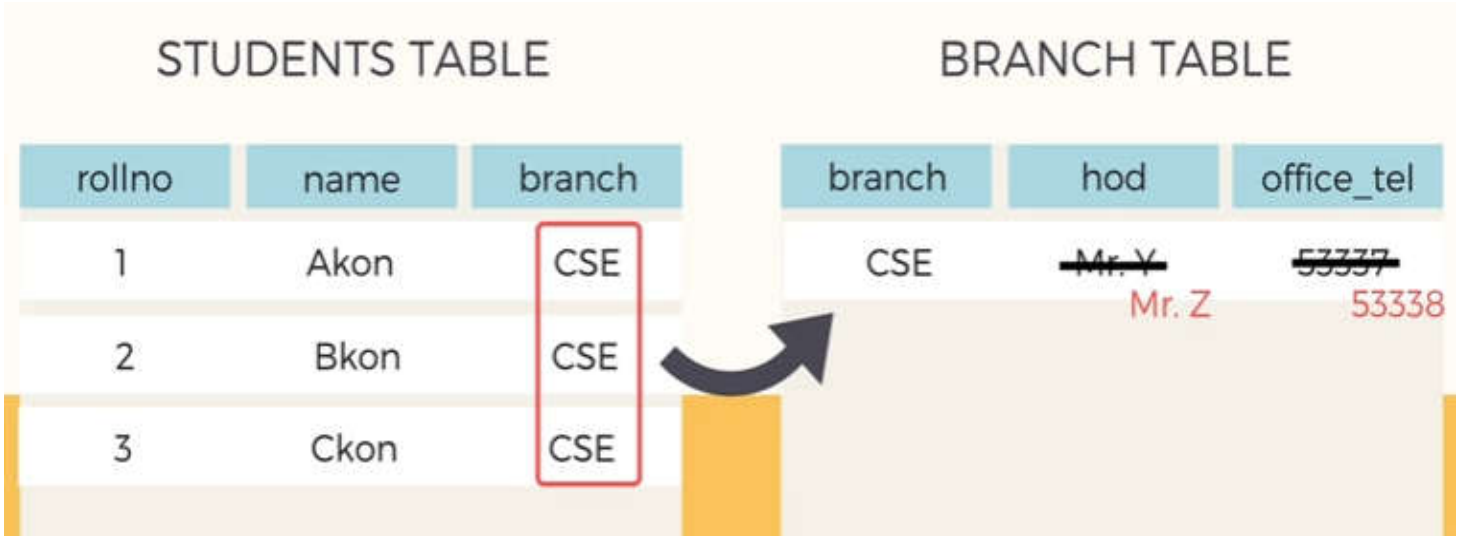
Data Redundancy

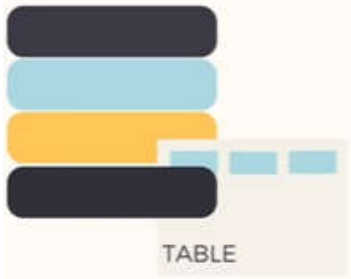
- Repetition of data hence needs extra space.



How Normalization will solve all these problems?


STUDENTS TABLE
rollno
name
branch
1
Akon
CSE
2
Bkon
CSE
3
Ckon
CSE
BRANCH TABLE
branch
hod
office_tel
CSE
Mr. Y
53337


STUDENTS TABLE
rollno
name
branch
1
Akon
CSE
2
Bkon
CSE
3
Ckon
CSE
BRANCH TABLE
branch
hod
office_tel
CSE
Mr. Y
53337
Mr. Z
53338



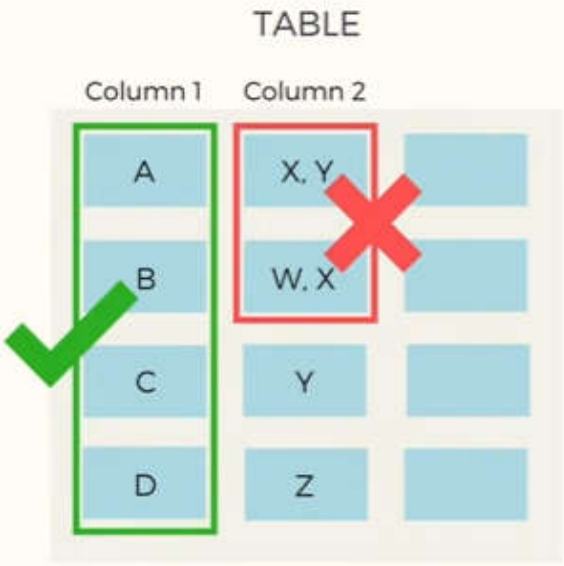
Types of Normalization

1. 1st Normal Form
2. 2nd Normal Form
3. 3rd Normal Form
4. BCNF

1st Normal Form

Every Table in your Database should at least follow the 1st Normal Form, always.

or Stop using Database!



RULE 1

- Each Column should contain atomic values.
- Entries like X, Y and W, X violate this rule.

TABLE

DOB	Name	
26-10-89	A	
13-2-92	SK	
16-11-65	SA	
R	8-9-86	

RULE 2

- A Column should contain values that are of the same type.
- Do not inter-mix different types of values in any column.

TABLE

DOB	Name	Name
26-10-89	A	A
13-2-92	S	K
16-11-65	S	A
8-9-86	R	A

RULE 3

- Each column should have unique name.
- Same names leads to confusion at the time of data retrieval

TABLE

Roll_no	F_Name	L_Name
3	A	A
4	S	K
1	S	A
2	R	A

RULE 4

- Order in which data is saved doesn't matter.
- Using SQL query, you can easily fetch data in any order from a table.

STUDENTS TABLE

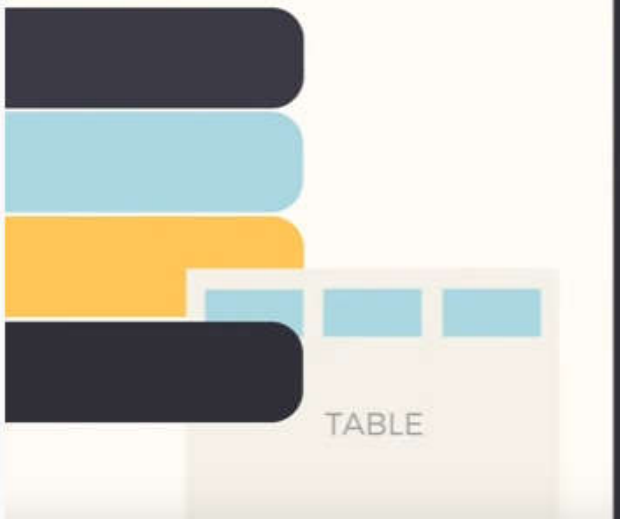
rollno	name	subject
101	Akon	OS, CN
103	Ckon	JAVA
102	Bkon	C, C++

Violation of 1 NF

STUDENTS TABLE

rollno	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	JAVA
102	Bkon	C
102	Bkon	C++

2nd Normal Form



For a table to be in the Second Normal Form...

- It should be in 1st Normal Form
- And, It should not have any Partial Dependencies.

What is Partial Dependency?

STUDENTS TABLE

student_id	name	reg_no	branch	address
1	Akon	CSE-18	CSE	TN
2	Akon	IT-18	IT	AP
3	Bkon	CSE-18	CSE	HR
4	Ckon	CSE-18	CSE	MH

As the student_id in this table will be unique, it can be used easily to fetch any data.

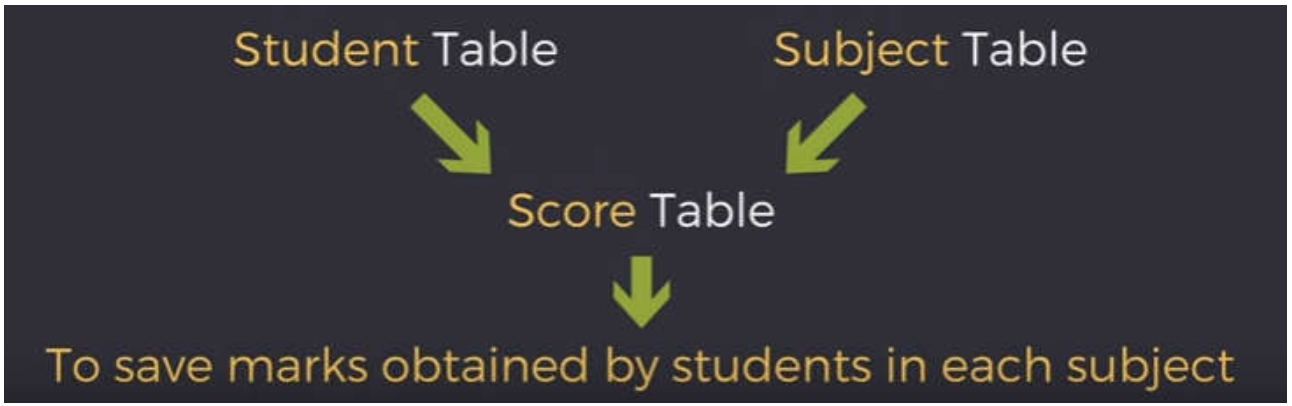
STUDENTS TABLE

student_id	name	reg_no	branch	address
10	Akon	CSE-18	CSE	TN

To get any data from the TABLE we can have PRI KEY

This is Dependency or

Functional Dependency



SCORE_ID CANNOT BE PRI KEY

SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	1	1	82	Mr. J
2	1	2	77	Mr. C++
3	2	1	85	Mr. J
4	2	2	82	Mr. C++
5	2	4	95	Mr. P

SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	10	1	82	Mr. J
2	10	2	77	Mr. C++
3	11	1	85	Mr. J
4	11	2	82	Mr. C++
5	11	4	95	Mr. P

SCORE TABLE

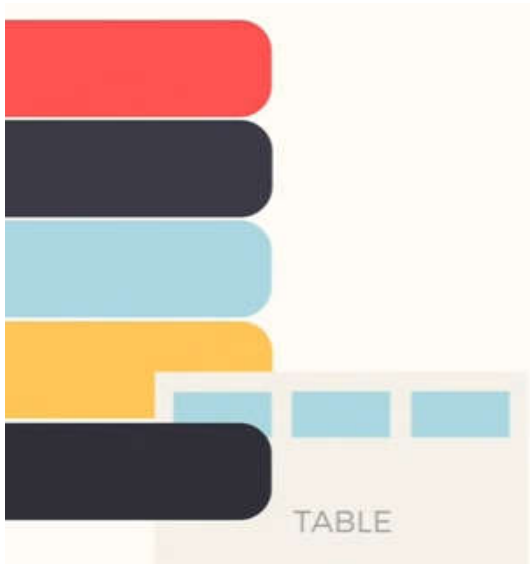
score_id	student_id	subject_id	marks	teacher
1	10 ?	1	82	Mr. J
2	10	2	77	Mr. C++
3	11 ?	1	85	Mr. J
4	11	2	82	Mr. C++
5	11	4	95	Mr. P

student_id + subject_id

can uniquely identify any row of data in SCORE table

Student **Subject**

Many to Many relationship.



Primary Key should be score_id

But student_id + subject_id together makes a more meaningful primary key.

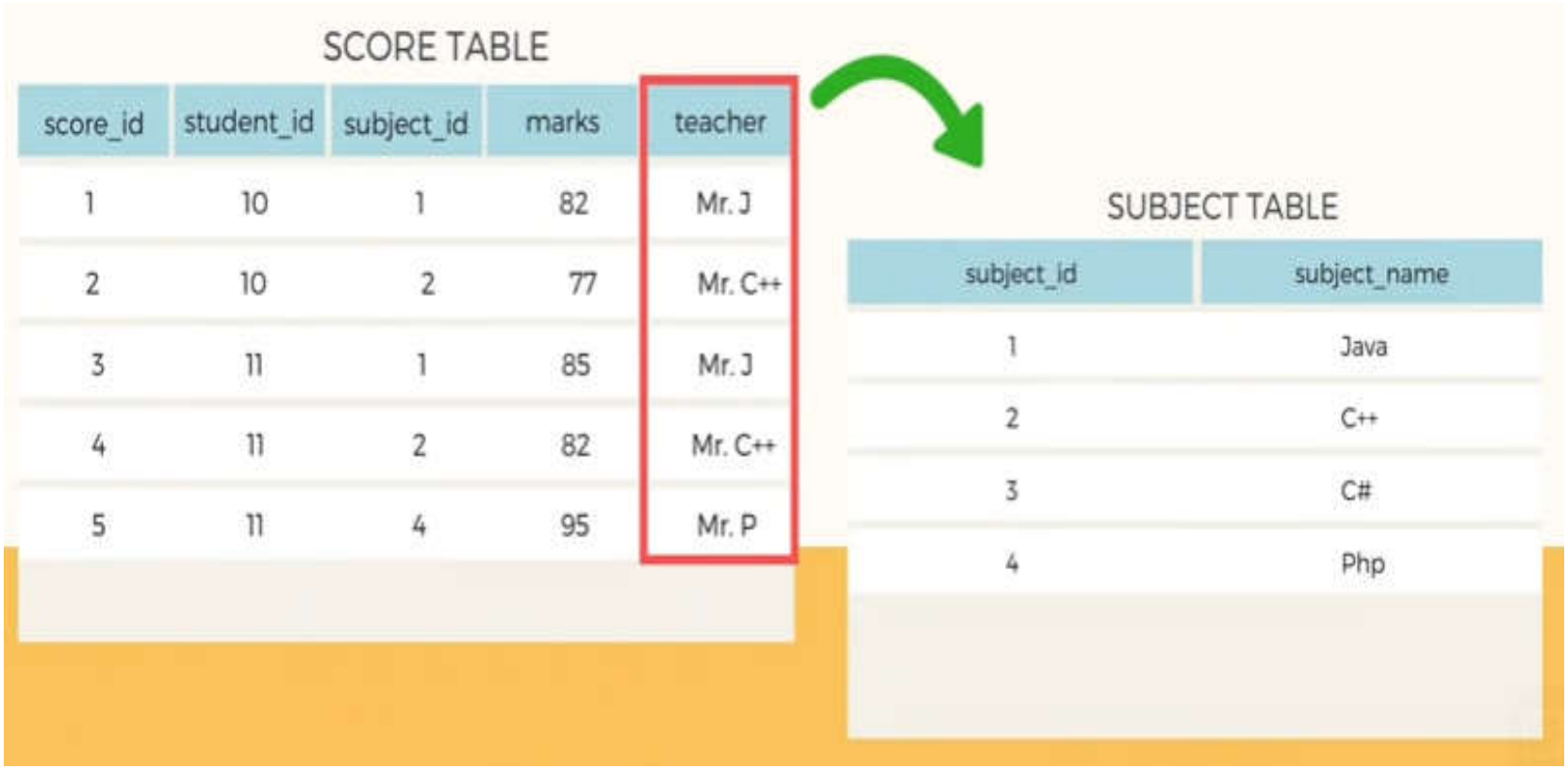
SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	10	1	82	Mr. J
2	10	2	77	Mr. C++
3	11	1	85	Mr. J
4	11	2	82	Mr. C++
5	11	4	95	Mr. P

teacher column only depends on subject and not on student.

This is Partial Dependency

Our objective is to remove teacher column from Score table.



SUBJECT TABLE

subject_id	subject_name	teacher
1	Java	Mr. J
2	C++	Mr. C++
3	C#	Mr. C#
4	Php	Mr. P

Teacher TABLE

teacher_id	teacher_name
1	Mr. J
2	Mr. C++
3	Mr. C#
4	Mr. P

Can even add more info. related to teachers like date of joining, salary etc.

2nd Normal Form (2NF)

* Must be in 1NF
* All the non prime attribute should be fully functional dependant on candidate key.

OR.

There should not be any partial dependancy

OR. **PARTIAL DEPENDANCY CONDITIONS TO BE CHECKED ON FD**

** LHS should be proper subset of candidate key &
** RHS should be non prime attribute.

Whether given Table satisfies 2NF ??

eg. R(ABCDEF)

$FD = \{C \to F, E \to A, EC \to D, A \to B\}$

$EC = FADB$

$\therefore EC^+ = ECFADB$

1 $\therefore CK = \{EC\}$

2 Prime attributes are attributes used to form CK

$\therefore$ Prime attributes $\Rightarrow$ E, C

3 $\therefore$ Non prime attributes $\Rightarrow$ A, B, D, F

$\therefore$ ** are verified.

4

** LHS should be proper subset of candidate key &
** RHS should be non prime attribute.

This table is not in 2NF

PARTIAL DEPENDANCY CONDITIONS TO BE CHECKED ON FD

* Identify for 2NF ?

R(A, B, C, D, E, F)

FD (A → B, B → C , C → D , D → E)

* Identify for 2NF ?

R(A, B, C, D, E, F)

FD (A → B, B → C , C → D , D → E)

$AB\underline{C}\underline{D}\underline{E}F^{+} = \{A, B, C, D, E, F\}$

$AF^{+} = \{A, B, C, D, E, F\}$ (SK)

$\therefore A^{+} = \{A, B, C, D, E\}$ ✗ SK

$F^{+} = \{F\}$ ✗ SK

$\therefore$ AF is CK

P.A. = { A, F } N.P.A = { B, C, D, E }

No P.A on RHS $\therefore$ Only CK is AF

$\therefore$ Partial dependency ?

For A → B. $\Rightarrow$ PD.

$\therefore$ This is not in 2NF

** LHS should be proper subset of candidate key &
** RHS should be non prime attribute.

Identify for 2NF ?

R(A, B, C, D)

FD = { AB → CD , C → A , D → B }

Identify for 2NF ?

R(A,B,C,D)

FD = { AB → CD , C → A , D → B }

~~A B C D~~$^+$ = { A, B, C, D }

AB^+ = { A, B, C, D } (SK)

A^+ = { A } (✗ SK)

B^+ = { B } (✗ SK) ∴ AB is CK

PA = { A, B }

∵ PA → A present on RHS of FD.

∴ CB^+ = { C, B, A, D } (SK)

C^+ = { A, C } (✗ SK)

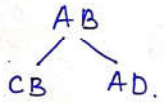
B^+ = { B } (✗ SK) ⇒ ∴ CB is CK

∴ PA = { A, B, C }

AD^+ = { A, D, B, C }

A^+ = (✗ SK) , D^+ = { D, B } (✗ SK)

∴ ⇒ AD is CK



** LHS should be proper subset of candidate key &
** RHS should be non prime attribute.

∴ PA = { A, B, C, D }

NPA = ϕ.

∵ ~~All~~ → No NPA ∴ This is in 2NF

3rd Normal Form

- It must satisfy following criteria
- Table should be in 2nd NF.
- There should not be any transitive dependancy in table

Should not be NPA → NPA

Eg.

T1

RN	Name of state	Name of City
1	MP	Indore
2	MH	A'nagar
3	UP	Ramgad
4	MH	A'nagar
5	AP	Vijayawada

Here CK is {RN}

PA = {RN} NPA = {Name of state, Name of city}

FD { RN → Name of state (FD1), Name of state → Name of city (FD2) }

Should not be NPA → NPA

(T1)

Eg.

RN	Name of state	Name of City
1	MP	Indore
2	MH	A'nagar
3	UP	Ramgad
4	MH	A'nagar
5	AP	Vijaywada

Here CK is {RN} PA = {RN} NPA = {Name of state, Name of city}

FD { RN → Name of state , Name of state → Name of city }

FD1 FD2

FD { RN → Name of state , Name of state → Name of city }

FD1 FD2

** For each FD

LHS must be CK or SK OR RHS is prime attribute

So for T1, FD1 ✓ OR X ⇒ ✓

FD2 X OR X ⇒ X

So this is not in 3rd NF.

* If RN → Name of city is provided in FD then we can conclude this table is in 3rd NF.

eg - T2

R(A B C D)

FD { AB → C , C → D }

CK = { AB }

[∵ AB^+ = ABCD]

PA = A, B

NPA = C, D

Should not be NPA → NPA

∵ For FD { ✓ OR X , X OR X }

✓ X

∴ This table is not in 3rd NF

** For each FD
LHS must be CK or SK OR RHS is prime attribute

eg (3)

R(A B C D)

FD { AB → CD , D → A }

∴ C, D, A at right side

∴ AB^+ = BCDA (with few test or checks)

i.e AB^+ = ABCD DB= ABCD Should not be NPA→ NPA

∴ PA = {A, B}, D

NPA = C

* * For each FD
LHS must be CK or SK OR RHS is prime attribute

∴ For FD { V OR ✓ , X OR ✓ }

∴ This table is in 3rd NF

Topic ..... 3NF. ..... Unit No. .....

Student table

Sid	Sname	DOB	State	Country	PIN
1	A	-	HR	IND	41
2	B	-	HR	IND	41
3	C	-	HR	IND	41
4	D	-	MH	IND	42

3NF????

CK = Sid

∵ Propersubset of CK → NPA ⇒ PD

For given 'R', No P.D. ∴ it is in 2NF

☆ if PIN → State, Country

∴ PIN updates ∴ State & country need to updated

Update Anomaly obtained

☆ ∴ NPA → NPA. (Condition for th Transitive Dependency) TD

☆☆☆ ∴ LHS is SK/CK or RHS is PA for 3NF

eg. R(A, B, C, D)

FD (A → B, B → C, C → D)

$AB^+ = \{A, B, C, D\}$

Transitive Depend TD present

$R(A, B, C, D, E, F)$

FD $\{AB \rightarrow CDEF,\ BD \rightarrow F\}$

$\Rightarrow$ $ABCDEF^+ = \{A, B, C, D, E, F\}$

$AB^+ = \{A, B, C, D, E, F\}$.

SK & CK $= AB^+$.

PA $= \{A, B\}$

$\therefore$ Only one CK

NPA $= \{C, D, E, F\}$

For $AB \rightarrow CDEF$.

PA $\rightarrow$ NPA $\Rightarrow$ ✗ TD

BD $\rightarrow$ F

NPA $\rightarrow$ NPA $\Rightarrow$ ✓ TD

This is not in 3NF

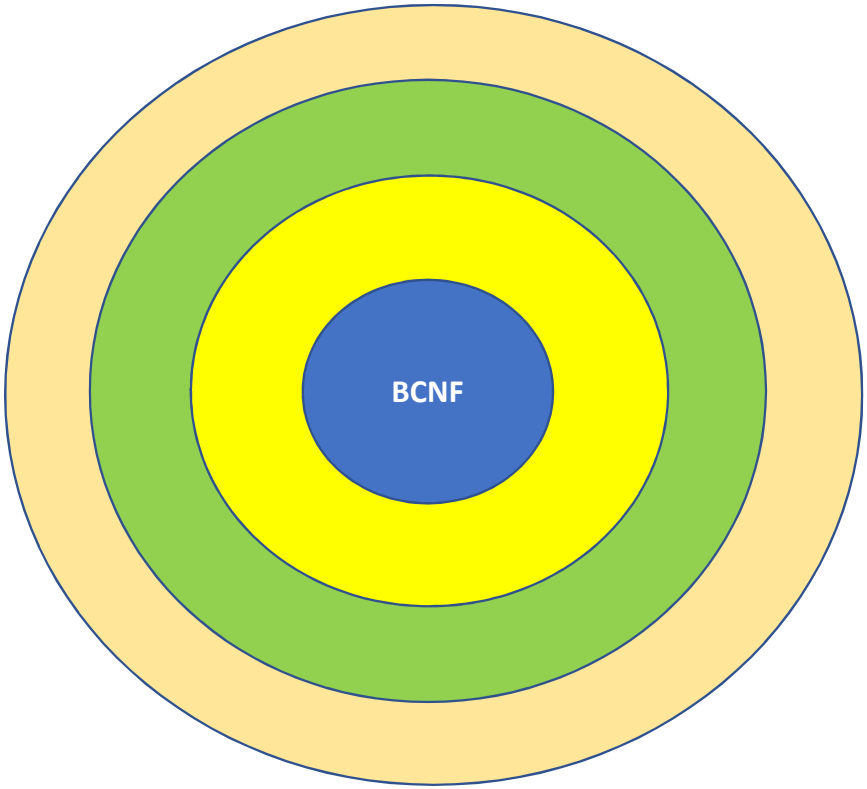
3NF????

Should not be NPA→ NPA

BCNF

(Boyce Codd Normal Form)

- **It must be in 3NF**
- **LHS of each FD must be CK or SK**



CK={Stud_rn, VoterID}

FD {Stud_rn →Sname,
Stud_rn→ Saddress,
VoterID→ Sname,
VoterID→ Stud_rn }

Stud_rn	Sname	Saddress	VoterID
1	A	22	A11
2	B	21	B22
3	C	24	C33

BCNF (Boyce Codd Normal Form)

R(A, B, C)

FD { A → B, B → C, C → A }

⇒ Since 3NF is quite less productive for overlapping CK. eg. AB, BC, CD, DB.

Cond^n → 1) Must be in 3NF

2) For nontrivial FD X → Y
X is superkey.

BCNF????

SK ON LHS.....

∴ $AB^+ = \{A, B, C\}$

SK $AC^+ = \{A, B, C\}$

(SK) (CK) $A^+ = \{A, B, C\}$

∴ PA = {A, B, C}

SK, CK ∴ $C^+ = \{A, B, C\}$

SK, CK $B^+ = \{A, B, C\}$

Obtain the Highest Normal Form for given relation

R(A, B, C, D, E)

FD = { A → BCDE, BC → ACE, D → E }

HIGHEST FORM?

⇒ $(ABCDE)^+ = \{A, B, C, D, E\}$

$A^+ = \{A, B, C, D, E\}$ — SK, CK

$BC^+ = \{B, C, A, E, D\}$ — SK, CK

$B^+ = \{B\}$ × SK

$C^+ = \{C\}$ × SK.

PA = { A, B, C }

NPA { D, E }

$AC^+ = \{A, C\}$ × SK.

$BA^+ = \{B, A, C, D, E\}$. SK

$B^+ = \{B\}$ × SK.

$A^+ = \{A, B, C, D, E\}$ SK

∴ BA is NOT CK

* Verification *			
BCNF	✓	✓	×
3NF	✓	✓	×
2NF	✓	✓	✓
1NF.	✓	.	

∴ Highest Normal Form is 2NF

SK ON LHS…..

Should not be NPA → NPA

Obtain the NF and Convert to BCNF for

For R(A,B,C,D,E,F) with

FD(AB-> C, C-> D, C-> E, E-> F, F-> A)

Ck={AB, FB, EB, CB}

PA={A,B,C,E,F},NPA={D}

(use lossless decomposition)

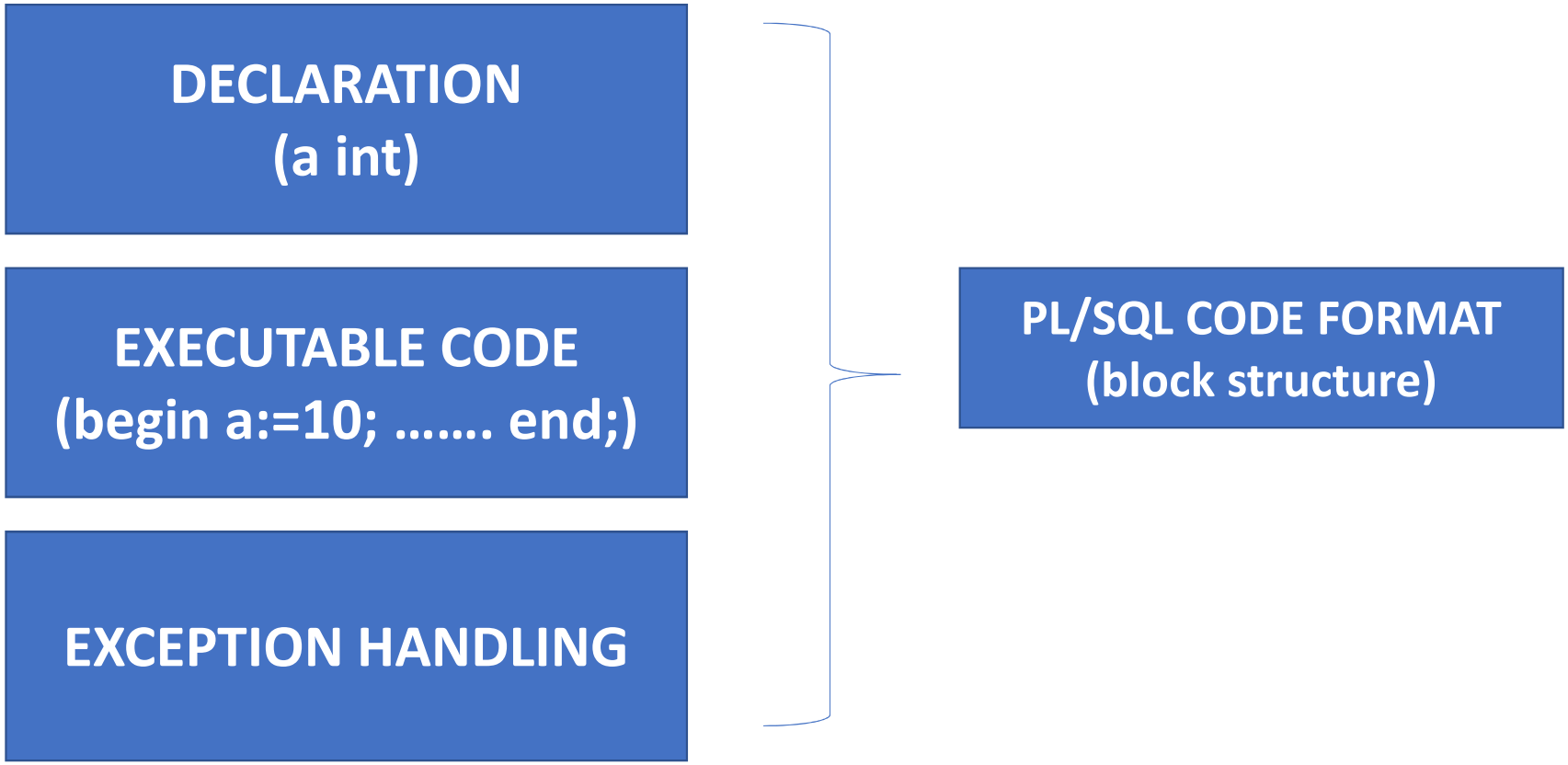
PL/SQL stands for Procedural Language/Structured Query Language.
PL/SQL is Procedural Language extension of SQL.
(TELLS NOT ONLY 'WHAT' BUT 'HOW'…..)

Advantages of PL/SQL

1) PL/SQL not only supports SQL data manipulation but also provides **conditional checking**, **branching** and **looping**.
2) PL/SQL sends an entire block of statements to the Oracle engine at one time. This in turn **reduces network traffic**.
3) Applications written in PL/SQL are **portable** to any computer hardware and operating system where Oracle is present.
4) PL/SQL **handles errors or exceptions** effectively during the execution of a PL/SQL program.
5) PL/SQL blocks can be **stored** in the database and **reused**.
6) PL/SQL provides features like **Triggers, Functions, Cursors, Procedure.**

Sr. No.	Key	SQL	PL/SQL
1	Definition	SQL, is Structural Query Language for database.	PL/SQL is a programming language using SQL for a database.
2	Variables	SQL has no variables.	PL/SQL has variables, data types etc.
3	Control Structures	SQL has no FOR loop, if control and similar structures.	PL/SQL has FOR loop, while loop, if controls and other similar structures.
4	Operations	SQL can execute a single operation at a time.	PL/SQL can perform multiple operation at a time.
5	Language Type	SQL is a declarative language.	PL/SQL is a procedural language.
6	Embedded	SQL can be embedded in a PL/SQL block.	PL/SQL can also be embedded in SQL code.
6	Interaction	SQL directly interacts with database server.	PL/SQL does not directly interacts with database server.
7	Orientation	SQL is data oriented language.	PL/SQL is application oriented language.
8	Objective	SQL is used to write queries, create and execute DDL and DML statments.	PL/SQL is used to write program blocks, functions, procedures, triggers and packages.

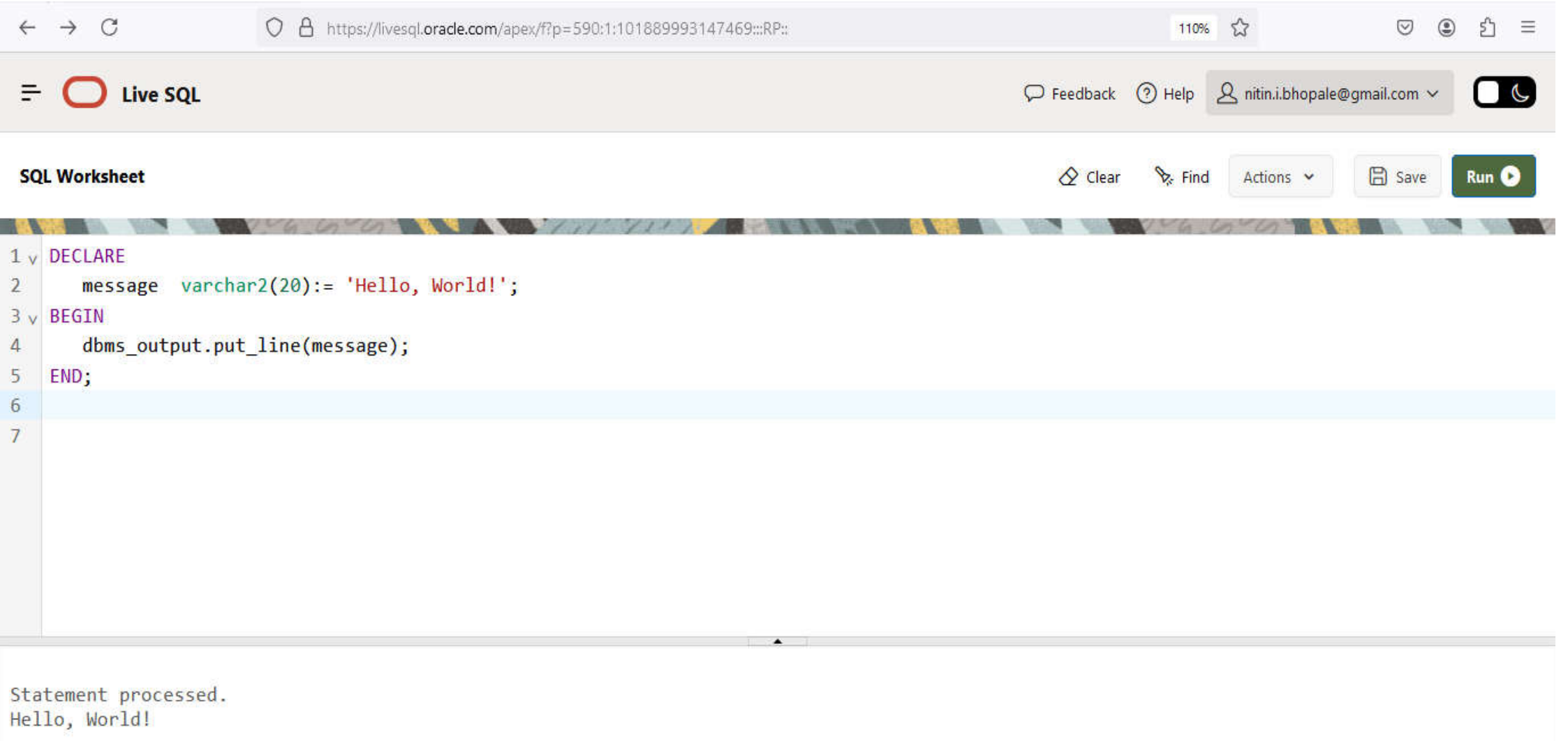
PL/SQL


DECLARATION
(a int)
EXECUTABLE CODE
(begin a:=10; ……. end;)
EXCEPTION HANDLING
PL/SQL CODE FORMAT
(block structure)

S.No	Date Type & Description
1	**Numeric** Numeric values on which arithmetic operations are performed.
2	**Character** Alphanumeric values that represent single characters or strings of characters.
3	**Boolean** Logical values on which logical operations are performed.
4	**Datetime** Dates and times.

S.No	Data Type & Description
1	**PLS_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
2	**BINARY_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
3	**BINARY_FLOAT** Single-precision IEEE 754-format floating-point number
4	**BINARY_DOUBLE** Double-precision IEEE 754-format floating-point number
5	**NUMBER(prec, scale)** Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0
6	**DEC(prec, scale)** ANSI specific fixed-point type with maximum precision of 38 decimal digits

7	**DECIMAL(prec, scale)** IBM specific fixed-point type with maximum precision of 38 decimal digits
8	**NUMERIC(pre, secale)** Floating type with maximum precision of 38 decimal digits
9	**DOUBLE PRECISION** ANSI specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
10	**FLOAT** ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
11	**INT** ANSI specific integer type with maximum precision of 38 decimal digits


https://livesql.oracle.com/apex/f?p=590:1:101889993147469:::RP::
110%
Live SQL
Feedback
Help
nitin.i.bhopale@gmail.com
SQL Worksheet
Clear
Find
Actions
Save
Run
1 DECLARE
2 message varchar2(20):= 'Hello, World!';
3 BEGIN
4 dbms_output.put_line(message);
5 END;
6
7
Statement processed.
Hello, World!

Program for addition of two numbers

```
DECLARE
A number:=5;
B number:=6;
C number;
BEGIN
C:=A+B;
dbms_output.put_line('The addition of two numbers is : '||C);
END;
```

If Input accepted from user:

```
DECLARE
A number; B number; C number; (OR A int; B int; C int; )
BEGIN
A:=&A; (for user defined values OR A:=:A)
B:=&B;
C:=A+B;
dbms_output.put_line('The addition of two numbers is : '||C);
END;
```


https://livesql.oracle.com/apex/f?p=590:1:5002118354497:::RP::
110%
Live SQL
Feedback
Help
nitin.i.bhopale@gmail.com
SQL Worksheet
Clear
Find
Actions
Save
Run
1 DECLARE
2 A number:=5;
3 B number:=6;
4 C number;
5 BEGIN
6 C:=A+B;
7 dbms_output.put_line('the addition of two numbers is : '||C);
8 END;
9
Statement processed.
the addition of two numbers is : 11
2023 Oracle · Live SQL 23.3.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ♥ using Oracle APEX · Privacy · Terms of Use

To find out whether number is **even** or **odd**.

```
DECLARE
nm number ;
BEGIN
nm:=:nm ;
IF (mod(nm,2)=0) THEN
dbms_output.put_line(nm || ' is Even Number');
ELSE
dbms_output.put_line(nm || ' is Odd Number');
END IF;
END;
```

```
DECLARE
    -- Global variables
    num1 number := 95;
    num2 number := 85;
BEGIN
    dbms_output.put_line('Outer Variable num1: ' || num1);
    dbms_output.put_line('Outer Variable num2: ' || num2);
    DECLARE
        -- Local variables
        num1 number := 195;
        num2 number := 185;
    BEGIN
        dbms_output.put_line('Inner Variable num1: ' || num1);
        dbms_output.put_line('Inner Variable num2: ' || num2);
    END;
END;
```

```
Statement processed.
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185
```

```
DECLARE
a number(2) := 10;
BEGIN
a:= 10;
IF( a < 20 ) THEN
dbms_output.put_line('a is less than 20 ' );
END IF;
dbms_output.put_line('value of a is : ' || a);
END;
```

Program for comparison of two numbers :

```
DECLARE
no1 number;
no2 number;
BEGIN
no1:=:no1;
no2:=:no2;
IF (no1>no2) THEN
dbms_output.put_line(no1|| ' is a Greater Number');
ELSE IF (no2>no1) THEN
dbms_output.put_line(no2|| ' is a Greater Number');
ELSE
dbms_output.put_line(no1|| ' and ' ||no2 || ' are Equal');
END IF;
END IF;
END;
```

Program to print the numbers 1 to 5

```
DECLARE
i number:=0;
BEGIN
LOOP
i:=i+1;
dbms_output.put_line(i);
IF (i>=5) THEN
EXIT;
END IF;
END LOOP;
END;
```

https://livesql.oracle.com/apex/f?p=590:1:16876282845835::NO:RP:: 170%

Live SQL Feedback Help nitin.i.bhopale@gmail.com

QL Worksheet Clear Find Actions Save Run

```
1  DECLARE
2  i number:=0;
3  BEGIN
4  LOOP
5  i:=i+1;
6  dbms_output.put_line(' value'||i);
7  IF (i>=5)
8  THEN EXIT;
9  END IF;
10 END LOOP;
11 END;
```

tatement processed.
value1
value2
value3
value4
value5

Program to print 1 to 7 numbers using FOR LOOP.

```
DECLARE
i number;
BEGIN
FOR i IN 1..7 LOOP
dbms_output.put_line(i);
END LOOP;
END;
```

Program to print 1 to 10 numbers in Reverse order using FOR LOOP.

```
DECLARE
i number;
BEGIN
FOR i IN REVERSE 1..10 LOOP
dbms_output.put_line(i);
END LOOP;
END;
```

```
DECLARE
   i number(1);
   j number(1);
BEGIN
   << outer_loop >>
   FOR i IN 1..3 LOOP
      << inner_loop >>
      FOR j IN 1..3 LOOP
         dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
      END loop inner_loop;
   END loop outer_loop;
END;
```

```
i is: 1 and j is: 1
i is: 1 and j is: 2
i is: 1 and j is: 3
i is: 2 and j is: 1
i is: 2 and j is: 2
i is: 2 and j is: 3
i is: 3 and j is: 1
i is: 3 and j is: 2
i is: 3 and j is: 3
```

Program to print the sum of 1 to 50 numbers using EXIT WHEN condition.

```
DECLARE
i number(3) := 1;
ans number(5):=0;
BEGIN
LOOP
ans:=ans+i;
i := i + 1;
EXIT WHEN i >=50;
END LOOP;
dbms_output.put_line('Sum of 1 to 50 numbers is: ' || ans);
END;
```

Sum of 1 to 50 numbers is: 1225

Program to print Fibonacci series

```
DECLARE
fno number:=0;
sno number:=1;
tno number:=0;
i number:=0;
maxval number:=0;
BEGIN
maxval:=:maxval;
dbms_output.put_line(fno);
dbms_output.put_line(sno);
LOOP
tno:=fno+sno;
dbms_output.put_line(tno);
fno:=sno;
sno:=tno;
i:=i+1;
EXIT WHEN i>=maxval-2;
END LOOP;
END;
```

Assigning SQL Query Results to PL/SQL Variables

Cursor

- Oracle creates a memory area, known as **context area**, for processing an SQL statement, which contains all information needed for processing the statement; for example, number of rows processed, etc.
- A **Cursor** is a **pointer** to this context area. PL/SQL controls the context area through a **Cursor**. A Cursor holds the rows (one or more) returned by a SQL statement.
- Oracle has a pre-defined area in memory kept aside, in which Cursors are opened. Hence the Cursor's size will be limited by the size of this pre-defined area.
- Write SELECT statement while creating a Cursor. Cursors are memory areas that allow access to the data present in the table without disturbing the actual contents of the table.

1) Implicit cursors

- **Implicit cursors** are automatically created by Oracle whenever an SQL statement is executed and when there is no explicit cursor for the statement. Programmers **cannot control** the implicit cursors and the information in it.
- **SQL cursor attributes**

Attribute	Description
%FOUND	Returns **TRUE** if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns **FALSE**.
%NOTFOUND	The logical opposite of %FOUND. It returns **TRUE** if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns **FALSE**.
%ISOPEN	Always returns **FALSE** for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
%ROWCOUNT	Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Program to print number of rows deleted

```
DECLARE
row_del number;
BEGIN
delete from emp;
row_del:=SQL%ROWCOUNT;
dbms_output.put_line('Number of rows deleted are : ' || row_del);
END;
```

- *Number of rows deleted are 14*
  Statement processed

```
DECLARE
total_rows number(2);
no number;
BEGIN
no:=:no;
UPDATE emp SET sal = sal + 500 where deptno=no;
IF SQL%NOTFOUND THEN
dbms_output.put_line('no customers selected');
ELSIF SQL%FOUND THEN
total_rows := SQL%ROWCOUNT;
dbms_output.put_line( total_rows || ' employees are updated');
END IF;
END;
```

Output : If the input no is 20

5 employees are updated

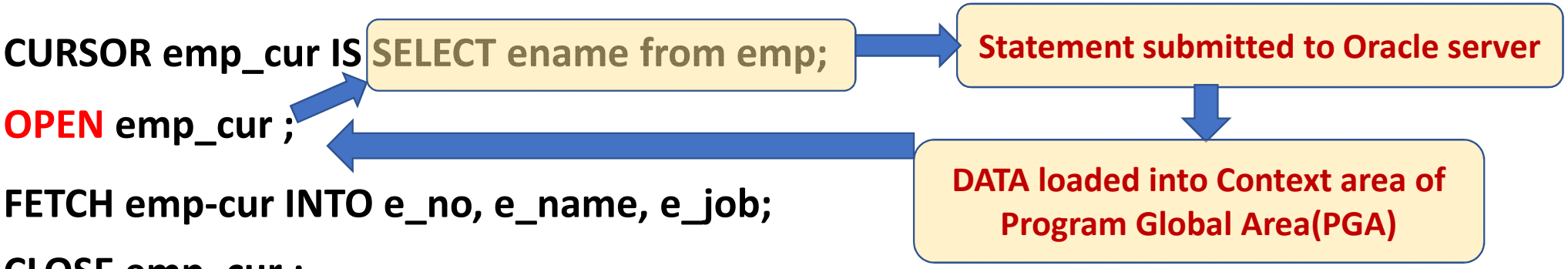
2) Explicit cursors

- **Explicit Cursors** are user defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block.

Working with an Explicit Cursor involves four steps:

- Declaring the cursor for initializing in the memory
- Opening the cursor for allocating memory
- Fetching the cursor for retrieving data
- Closing the cursor to release allocated memory

1. **DECLARE**
2. **OPEN**
3. **FETCH**
4. **CLOSE**

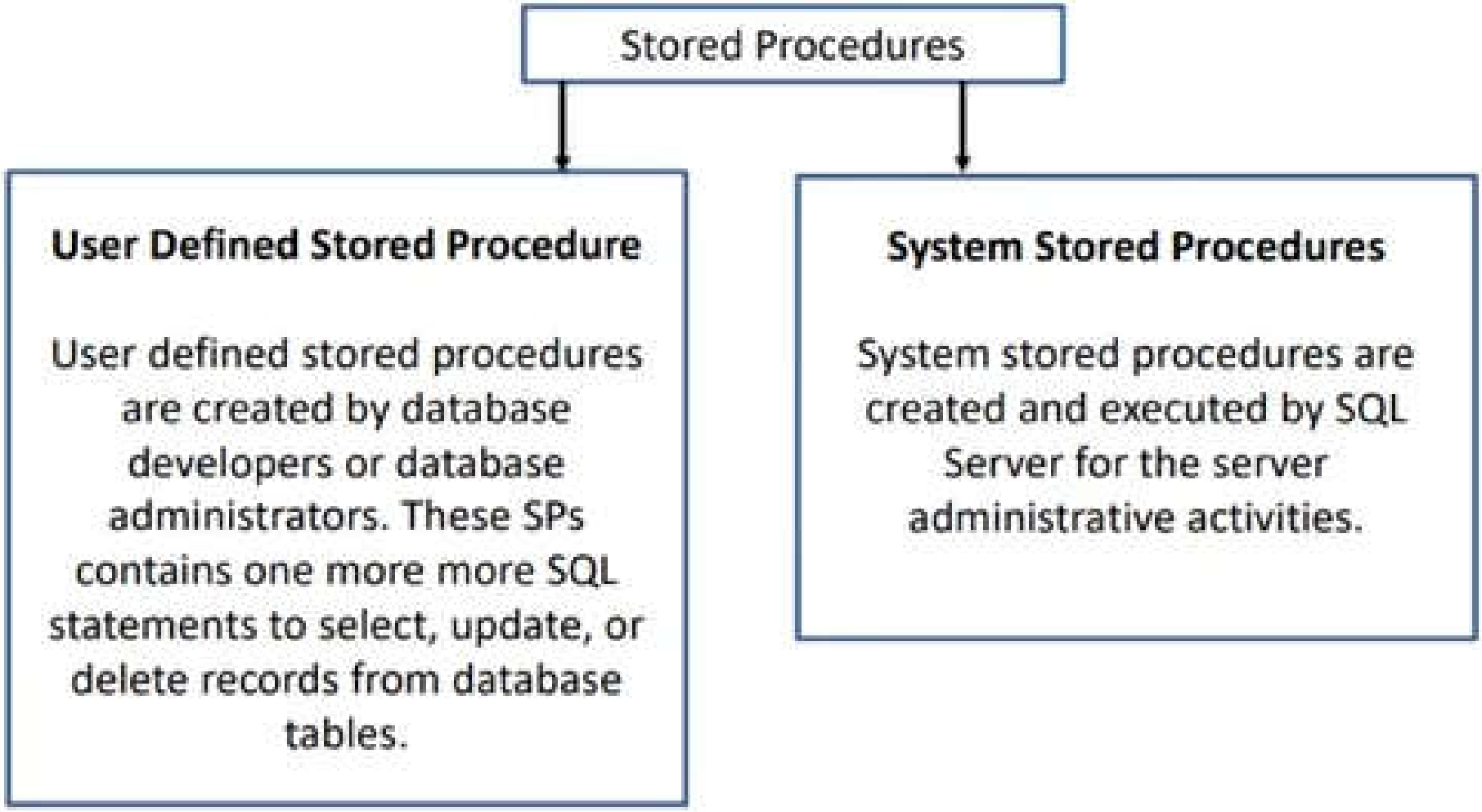


```
DECLARE
e_no emp.empno%type; (If the data type of the column or variable changes, there is no need to modify the declaration code.)
e_name emp.ename%type;
CURSOR e_cur IS select empno,ename from emp;
BEGIN
OPEN e_cur;
LOOP
FETCH e_cur INTO e_no,e_name;
dbms_output.put_line(e_no || ' ' || e_name);
EXIT WHEN e_cur%NOTFOUND;
END LOOP;
CLOSE e_cur;
END;
```

SQL Stored Procedures

- A stored procedure is a **database object.**
- A stored procedure is a **series of declarative SQL statements.**
- A stored procedure can be stored in the database and can be **reused over and over again.**
- Parameters can be passed to a stored procedure, so that the stored procedure can act based on the parameter value(s).
- SQL Server creates an **execution plan and stores it in the cache.**

Types of stored procedures



- **Procedures**
  A **Subprogram** that performs a specific action or task is called as **Procedure**. In other words **Procedure** is a logically grouped set of SQL and PL/SQL statements that perform a specific task.
- Procedures are stored in the Oracle database. They are invoked or called by any PL/SQL block that appears within an application.
- **Procedures consists of following three parts:**
  1) **Declarative Part** : It contains the declaration of cursors, constants, variables, exceptions, subprograms. These objects are local to the procedure.
  2) **Executable Part** : The executable part is a PL/SQL block consisting of SQL and PL/SQL statements that assign values to control execution and manipulate data. The action that the Procedure is expected to perform is coded here.
  3) **Exception handling** : This part contains code that performs actions to deal with exceptions that may be raised during the execution of code in the executable part.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `name`()
BEGIN
select * from localdb.person;
END
=> Call name;

CREATE DEFINER=`root`@`localhost` PROCEDURE `name`()
BEGIN
select * from localdb.person where id=2;
END

=>Call name;
```

Parameterised Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `name`(in n int)
BEGIN
select * from localdb.person where id=n;
END
```

=> Call name(2);

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `name`(out records int)
BEGIN
select count(*) into records from localdb.person where id=2;
END
```

⇒Call name(@records);

⇒Select @records as totalrec;

Parameterised Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `name`(inout records int,in n int)
BEGIN
select count(*) into records from localdb.person where id=n;
END
```

⇒Call name(@rec,3);

⇒Select @rec as totalrec;

Creating a Procedure

- CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter_name [IN | OUT | IN OUT] data type [, ...])]
  {IS | AS}
  BEGIN
  < procedure_body >
  END procedure_name;

Where,

- ***procedure-name*** specifies the name of the procedure.
- [**OR REPLACE**] option allows modifying an existing procedure.
- The optional parameter list contains name, mode and types of the parameters. **IN** represents that value will be passed from outside and **OUT** represents that this parameter will be used to return a value outside of the procedure.
- ***procedure-body*** contains the executable part.
- The **AS** keyword is used instead of the **IS** keyword for creating a standalone procedure.

Executing a Standalone Procedure

```
mysql> CREATE PROCEDURE disp()
   -> BEGIN
   -> select * from user;
   -> END;
   -> /
mysql> call disp/
```

fname	lname	city
Mahesh	Patil	Pune
Nitin	Patil	Pune
Ramesh	Pawar	Pune
Nilesh	Deshmukh	Pune
Onkar	Gunjal	Kopargaon
Kunal	Dhorde	Kopargaon

8 rows in set (0.07 sec)

```
mysql> CREATE PROCEDURE search_name(IN inputname long)
    ->        BEGIN
    ->       select * from user where fname=inputname;
    ->       END;
    ->      /

Query OK, 0 rows affected (0.00 sec)

mysql> call search_name("Nitin")/
+-------+-------+------+
| fname | lname | city |
+-------+-------+------+
| Nitin | Patil | Pune |
+-------+-------+------+
```

Functions

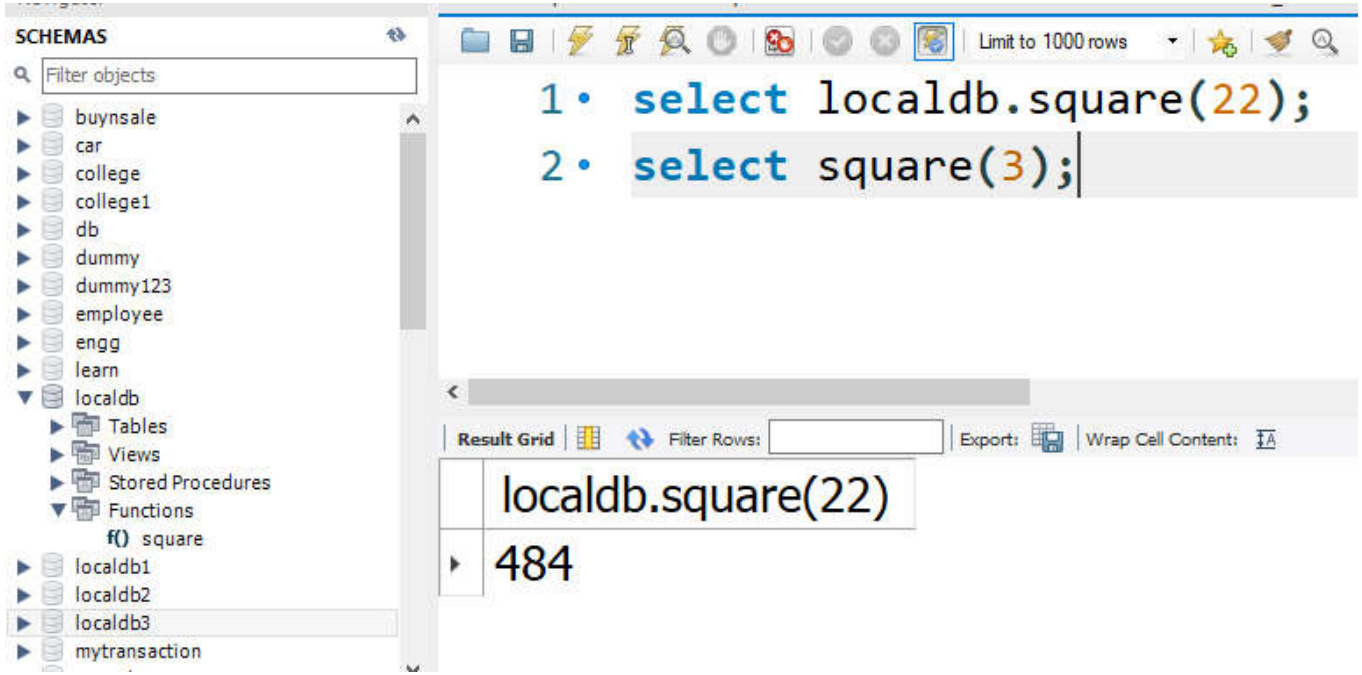
Function is a **subprogram** that is used for the purpose of performing a specific task. The main use of Function is **code reusability**. Functions and Procedures are almost same, except that Function have a RETURN clause i.e. a Function must return a value back to the caller.

- **Advantages of Functions**
  1. Code Reusability feature can be used.
  2. It saves time and cost.
  3. Increases the flexibility of the program.
  4. Works better in client-server type of operations.
  5. Memory space required is less.
  6. It returns value to the calling program.
  7. Arguments can be passed to Functions to get the desired result.

- **String Functions**
- CHAR(), CONCAT(), LOWER() /LCASE(), SUBSTRING(), SUBSTR(), UPPER()/UCASE(), TRIM(), LENGTH()
- **Numeric Functions :**
- MOD(), POWER()/POW(), ROUND(), SIGN(), SQRT(), TRUNCATE()
- **Date and Time Functions :**
- CURDATE()/CURRENT_DATE()/CURRENT_DATE, DATE(), MONTH(), YEAR(), NOW(), SYSDATE()
- **Aggregate Functions :**
- Sum(), count(), min(), max(), avg()

```
1 • CREATE FUNCTION `addition` (a int)
2   RETURNS INTEGER
3   BEGIN
4   declare sq int;
5   set sq= a*a;
6   RETURN sq;
7   END
```

Execution on MySQL workbench



```
create or replace function adder(n1 in number, n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
/
```

```
DECLARE
  n3 number(2);
BEGIN
  n3 := adder(11,22);
  dbms_output.put_line('Addition is: ' || n3);
END;
/
```

Execution on MySQL command line client

```
Addition is: 33

Statement processed.

0.05 seconds
```

S.NO	Function	Procedure
1.	Functions always return a value after the execution of queries.	The procedure can return a value using “IN OUT” and “OUT” arguments.
2.	In SQL, those functions having a DML statement can not be called from SQL statements. But autonomous transaction functions can be called from SQL queries.	A procedure can not be called using SQL queries.
3.	Each and every time functions are compiled they provide output according to the given input.	Procedures are compiled only once but they can be called many times as needed without being compiled each time.
4.	A Function can not return multiple result sets.	A procedure is able to return multiple result sets.

5.	The function can be called using Stored Procedure.	While procedures cannot be called from function.
6.	A function used only to read data.	A procedure can be used to read and modify data.
7.	The return statement of a function returns the control and function's result value to the calling program.	While the return statement of the procedure returns control to the calling program, it can not return the result value.
8.	The function does not support try-catch blocks.	Procedure supports try-catch blocks for error handling.

9.	A function can be operated in the SELECT statement.	While it can't be operated in the SELECT statement.
10.	Functions do not permit transaction management.	It allows transaction management.
11.	In functions, we can use only a table variable. Temporary tables can not be created in function.	In procedures, we can use temporary tables or table variables to store temporary data.

Difference between PL/SQL Function and Procedure :

Sr. No.	PL/SQL Function	PL/SQL Procedure
1	A Function **must return** a value.	A Procedure **cannot return** a value.
2	Functions **can be called** or used by the normal SQL statements.	Procedures **cannot be called** or used by the normal SQL statements.
3	Functions are considered as expressions.	Procedures are not considered as expressions.
4	Function is **not** a **precompiled execution plan**.	Procedure **is** a **precompiled execution plan**.
5	Functions are traditionally the smaller, more specific pieces of code.	Procedures are bigger and heavily used pieces of code.

Whenever you want to return some value and have to use that value further, go for function.
If you want to return some value as the end result, go for procedure.

Sr. No.	Key	Static SQL	Dynamic SQL
1	Database Access	In Static SQL, database access procedure is predetermined in the statement.	In Dynamic SQL, how a database will be accessed, can be determine only at run time.
2	Efficiency	Static SQL statements are more faster and efficient.	Dynamic SQL statements are less efficient.
3	Compilation	Static SQL statements are compiled at compile time.	Dynamic SQL statements are compiled at run time.
4	Application Plan	Application Plan parsing, validation, optimization and generation are compile time activities.	Application Plan parsing, validation, optimization and generation are run time activities.
5	Use Cases	Static SQL is used in case of uniformly distributed data.	Dynamic SQL is used in case of non-uniformly distributed data.
6	Dynamic Statements	Statements like EXECUTE IMMEDIATE, EXECUTE, PREPARE are not used.	Statements like EXECUTE IMMEDIATE, EXECUTE, PREPARE are used
7	Flexibility	Static SQL is less flexible.	Dynamic SQL is highly flexible.

Thank you !!!