STORED PROCEDURE, QBE

DATABASE MANAGEMENT SYSTEM

Sujan Tamrakar

STORED PROCEDURE

- A.k.a Proc, Sproc, Stopro, Persistent Stored Procedures
- Program modules that are stored by the DBMS at the db server
- Set of SQL statements with an assigned name, which are stored in a relational database management system as a group, so it can be **reused** and **shared** by multiple programs
- Can be functions or procedures
- One can pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

STORED PROCEDURE

- Useful in following circumstances:
  - If a db program is needed by several applications, it can be stored at the server & invoked by any of the application programs. This reduces duplication of effort & improve software modularity.
  - Executing a program at server can be fast
  - These procedures can enhance the modeling power provided by views by allowing more complex types of derived data to be made available to db users

STORED PROCEDURE

- Sproc may return result sets. Such results can be processed using **cursors** by other sproc, by associating a result set locator or applications
- Sproc may also contain declared variables for processing data & cursors that allow it to loop through multiple rows in table.
- Sproc flow control statements typically include if, while loop, repeat & case statements.
- Sproc can receive variables, return results or modify variables & return them, depending on how & where the variable is declared.

STORED PROCEDURE

Syntax:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Execute a Stored Procedure:

```
EXEC procedure_name;
```

STORED PROCEDURE

Ex:

```
CREATE PROCEDURE uspGetAddress
AS
SELECT * FROM USERINFO
GO
```

Calling it:

```
EXEC uspGetAddress;
```

OR

```
uspGetAddress;
```

STORED PROCEDURE

Ex:

```
USE CollegeDatabase;
GO
CREATE PROCEDURE humanResources
    @lastname nvarchar(50),
    @firstname nvarchar(50)
AS
SELECT * FROM EmployeeDepartment WHERE fname = @firstname AND lname= @lastname
GO
```

Calling it:

```
EXEC humanResources @firstname='sujan', @lastname ='tamrakar';
EXEC humanResources 'tamrakar', 'sujan';
```

STORED PROCEDURE

Ex:

```
CREATE PROCEDURE uspGetAddress @city nvarchar(30)
AS
SELECT * FROM USERINFO WHERE CITY=@city
GO
```

[using wildcard % □ where city like @city+'%']

Calling it:

```
EXEC uspGetAddress @city='Pokhara'
```

QBE (QUERY BY EXAMPLE)

- It is the name of both DML and an early DB system that included this language.
- It has a 2D (two dimensional) syntax. Queries look like table.
- In SQL (1D language), queries are written in one line. A 2D language requires 2D for its expression.
- QBE queries are expressed ‘by example’.
- Instead of giving a procedure for obtaining the desired answer, the user gives example of what is desired.
- The system generalizes this example to compute the answer to the query.

QBE (QUERY BY EXAMPLE)

- Skeleton tables:
  - We express queries in QBE by skeleton tables.
  - Rather than clutter the display with all skeletons, the user selects those skeletons needed for a given query and fills in the skeletons with example rows.
  - An example row consists of constants and example elements, which are domain variables.
  - To avoid confusion between the two, QBE uses underscore (_) before domain variables as in _x and lets constant appear without any qualification.
  - This convention is in contrast to those in most other languages where constants are quoted and variables appear without any qualification. [int roll = 5; string name='sujan';]

QBE (QUERY BY EXAMPLE)

- Queries on relation:
  - To find all loan numbers at the Pokhara branch, we bring up skeleton for loan relation and fill in it as;

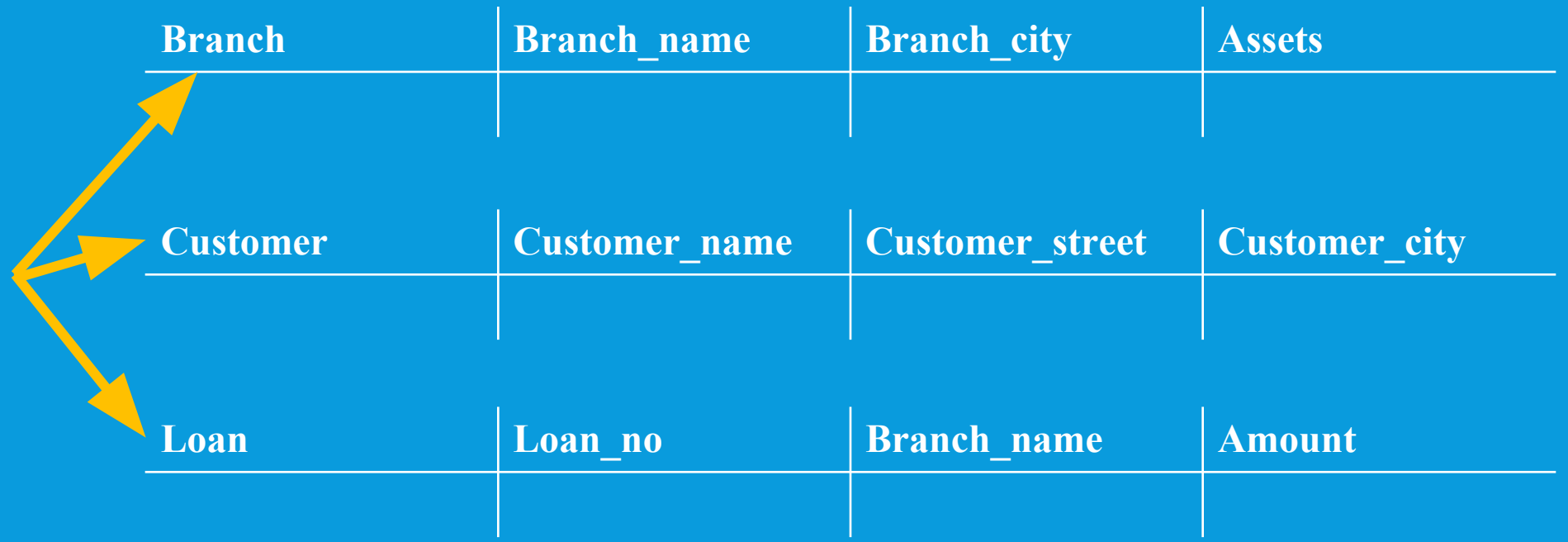
Loan	Loan_no	Branch_name	Amount
	P._x	Pokhara	

Relation name

P □ Print x □ domain variable _ □ representation of variable

Pokhara □ constant

QBE (QUERY BY EXAMPLE)



QBE skeleton tables for Bank example.

QBE (QUERY BY EXAMPLE)

- To display entire loan relation, we can create a single row consisting of P. in every field or shortly by placing a single P. in column headed by relation name.

Loan	Loan_no	BranchName	Amount
P.			

- Finding the loan numbers of all loans with a loan amount more than 700.

Loan	Loan_no	BranchName	Amount
	P.		>700

P. Or P._x

QBE (QUERY BY EXAMPLE)

Queries on Several Relations

QBE allows queries that span several different relations.

Ex: Suppose we want to find the names of all customers who have a loan from Pokhara branch. This can be written as;

Loan	Loan_no	BranchName	Amount
	_x	Pokhara	

Borrower	customer_name	Loan_no
	P._y	_x

First it finds tuples in loan relation and then corresponding customer from borrower relation.

THANK YOU!