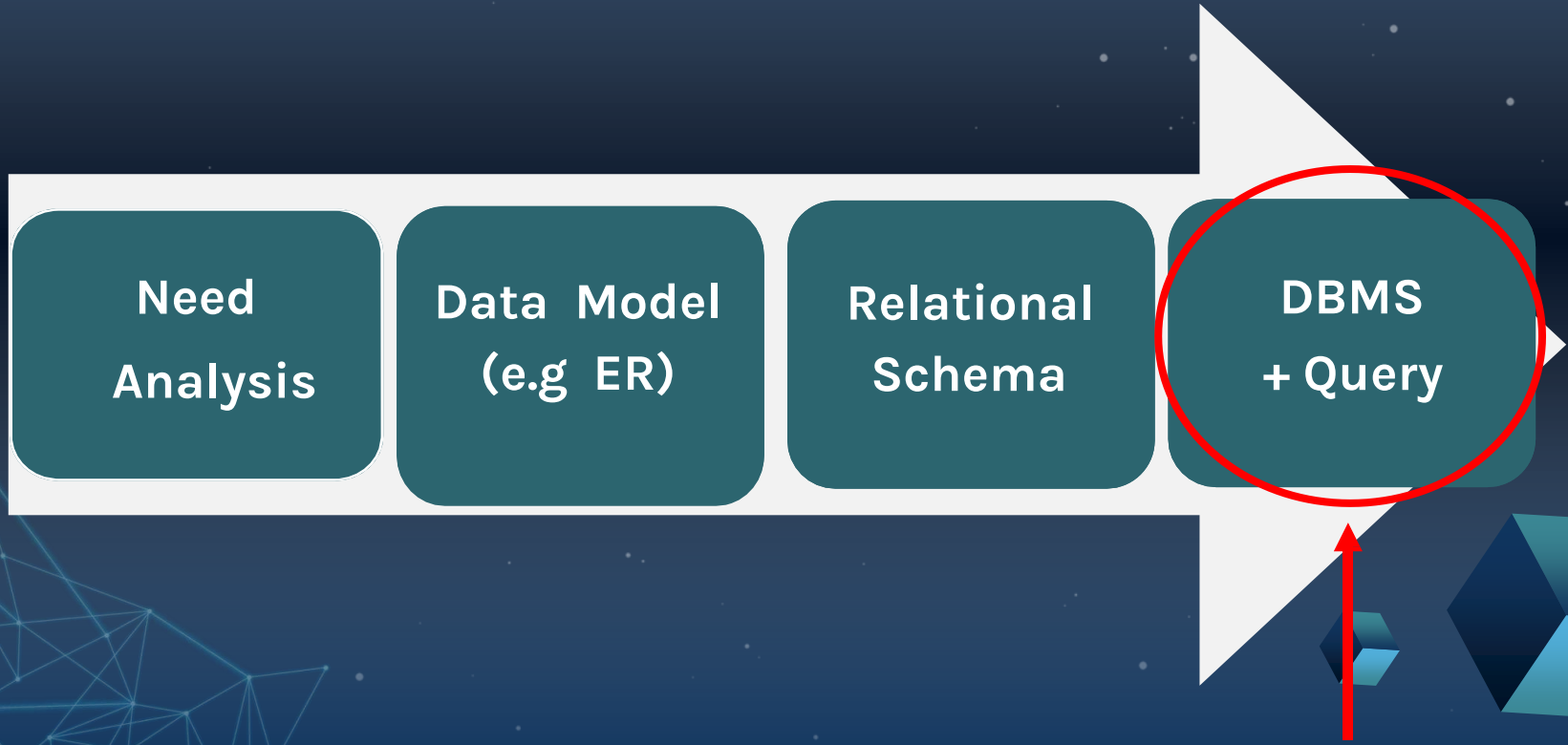
Structure Query Language (SQL)

Ahmadi Irmansyah Lubis

Review : Database Design

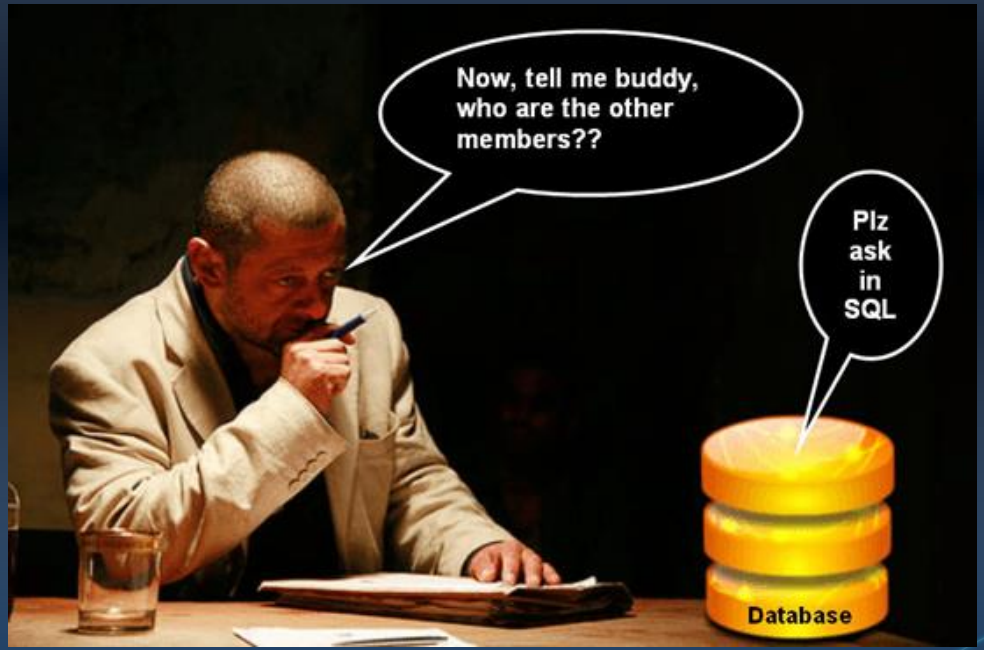


Structure Query Language (SQL)

- SQL (pronounced "ess-que-el") stands for Structured Query Language.
- SQL is used to communicate with a database.
- According to ANSI (American National Standards Institute), it is the standard language for relational database management systems.
- SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.
- Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.
- Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system.
- However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database.

Structure Query Language (SQL)

- SQL is the standard language to relational database management system



Brief History

- **1970** – Dr. Edgar F. "Ted" Codd of IBM is known as the father of relational databases. He described a relational model for databases.
- **1974** – Structured Query Language appeared.
- **1978** – IBM worked to develop Codd's ideas and released a product named System/R.
- **1986** – IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software which later came to be known as Oracle.

Recommended Tutorial on SQL

- W3Schools
  https://www.w3schools.com/sql/sql_intro.asp
- SQL Course

  http://www.sqlcourse.com/
- Tutorials Point

  https://www.tutorialspoint.com/sql/index.htm
- Guru 99
  https://www.guru99.com/sql.html

SQL Parts

DDL	• Data definition language (DDL): creation of tables and views
DML	• Data manipulation language (DML): ad-hoc queries and updates
DCL	• Data control language (DCL): controlling access to data in a database

SQL Illustration

DDL

Foundations

DML

Contents

DCL

Access

SQL Parts

- **DDL :**

```
CREATE TABLE Account
  (AccNumber integer NOT NULL,
  Owner varchar(50),
  Balance currency,
  AccType varchar(10),
  PRIMARY KEY (Number));
```

- **DML :**

```
SELECT * FROM Account WHERE Type = "checking ";
```

- **DCL :**

```
GRANT all ON banking
TO teller IDENTIFIED BY ‘password’;
```

Data Definition Language (DDL)

Data Definition Language

The SQL DDL describe specification about :

- Schema for each relation
- Domain of each atribute
- Integrity constraints
- Index set maintained for each relation
- Security and authorization information for each relation
- Physical storage structure of each relation on disk

Basic Domain Types

- **char(n) :** a fixed-length character string with user-specified length n.
- **varchar(n):** a variable-length characters with user-specified maximum length n

What is the difference char(n) and varchar(n)?

Data	Memory allocation	
	char(10)	varchar(10)
du	10 bytes	2 bytes
dunia	10 bytes	5 bytes
duniaku	10 bytes	7 bytes

Basic Domain Types (Cont)

- **int, mediumint, smallint, tinyint.**
- **numeric(p,d) :** a fixed-point number with user-specified precision. The number consists of p digit (plus sign), and d digits are to the right of the decimal point

  Example : numeric (3,1)

  44.5 (it will be stored exactly)

  444.5 (it will not be stored exactly)

  0.32 (it will not be stored exactly)

- **real, double precision.** Floating-point and double-precision floating- point numbers with machine-dependent precision.
- **float(n).** Floating-point number, with precision at least n digits
- **tinyblob, blob, mediumblob, longblob.**

Basic Domain Types (Cont)

- **date :** contain 4 digit year, month, and day
  - Example: **date** '2016-7-27'
- **time:** contain hour, minute, and second
  - Example: **time** '09:00:30'
    time '09:00:30.75'
- **timestamp**: date and time
  - Example: **timestamp** '2016-7-27 09:00:30.75'
- **null**
  - Null includes in all attribute type
  - If we declare an attribute **not null,** it prevents null value for the attribute

Relation Scheme Sample

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

loan (loan_number, branch_name, amount)

Borrower (customer_name, loan_number)

Account (account_number, branch_name, balance)

Depositor (customer_name, account_number)

Creating Table Command

create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

- r : table name
- A : field in table r
- D : data type domain for field A

Example: branch (**branch_name**, branch_city, assets)

```
create table branch
  (  branch_name varchar(15) not null,
     branch_city varchar(30),
     assets integer,
     primary key (branch_name)  )
```

Integrity Constraints in Creating Table

Integrity constraints include :

not null

primary key (A_1, ..., A_n)

check (P), P must be obeyed by all tuples

Example: Declare check asset must positive number

```
create table branch
        ( branch_name varchar (15),
        branch_city varchar (30),
        assets     integer,
        primary key (branch_name),
        check (assets >= 0))
```

Integrity Constraints in Creating Table

Foreign key (A1, ..., An) **references** R

Example:

customer (**customer_name**, customer_street, customer_city) \
loan (**loan_number**, branch_name, amount)
borrower (**customer_name**, **loan_number**)

- Make borrower table which describe relationship customer and loan
- create table borrower (

  customer_name varchar(30),
  loan_number number (8),
  primary key (customer_name, loan_number)
  foreign key (customer_name)
  references customer (customer_name),
  foreign key (loan_number)
  references loan)

Integrity Constraints in Creating Table

CASCADE

- Whenever rows in the master (referenced) table are deleted (or updated), the respective rows of the child (referencing) table with a matching foreign key column will be deleted (or updated) as well.
- This is called a cascade on delete (or cascade on update)

```
create table borrower (customer_name varchar(30),
        loan_number number(8),
      primary key (customer_name, loan_number)
        foreign key (customer_name)
        references customer (customer_name),
        foreign key (loan_number)
        references loan ON DELETE CASCADE)))
```

Alter Table Commands

Add an attribute to an existing table. All tuple in the relation are assigned null as the value for the new attribute

- **alter table** r **add** A D
- **alter table** borrower **add** b_date DATE

Rename attribute :

- **alter table** r **change** A A_new
- **alter table** borrower **change** b_date birth_date DATE

Drop attributes from a table :

- **alter table** r **drop** A
- **alter table** borrower **drop** birth_date

Add or remove a constraint

- **alter table** *r* **add primary key** $(A_1, ..., A_n)$
- **alter table** *r* **add foreign key** $(A1, ..., An)$ **references** *R*

Default Value

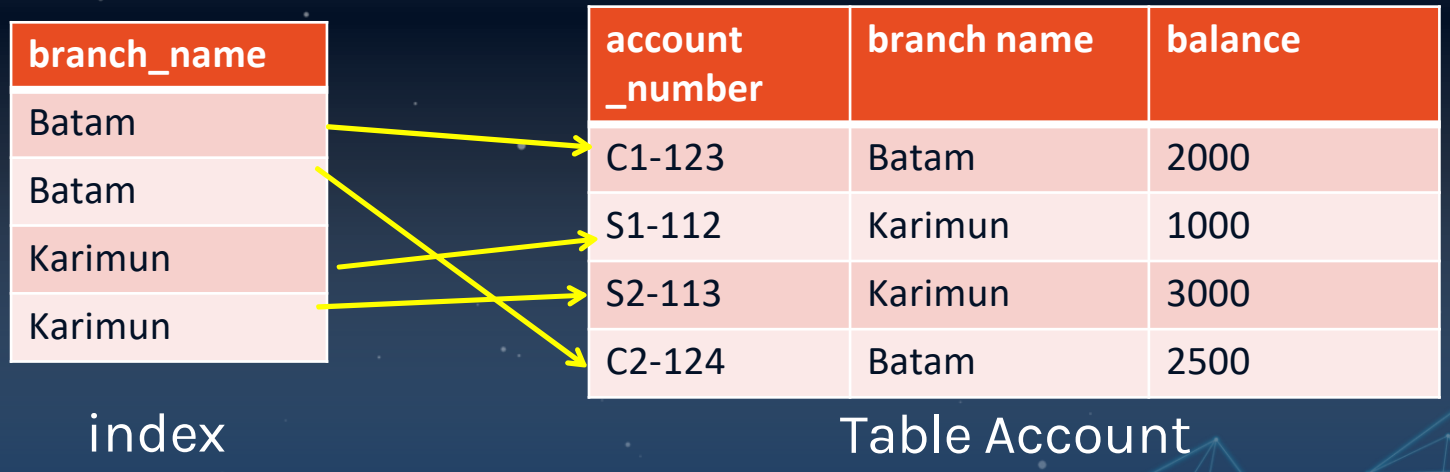
When we declare an attribute, we can add DEFAULT keyword, followed by NULL or a constant

Example :

- Gender CHAR(1) DEFAULT ‘?’
- Birthdate DATE DEFAULT DATE ‘0000-00-00’

Index

- An index can be created in a table to find data more quickly and efficiently.
- The users cannot see the indexes, they are just used to speed up searches/queries.
- Example : Creating index on **column branch name** because data branch_name is search frequently



branch_name
Batam
Batam
Karimun
Karimun

index

account _number	branch name	balance
C1-123	Batam	2000
S1-112	Karimun	1000
S2-113	Karimun	3000
C2-124	Batam	2500

Table Account

Index

Make single-column index

CREATE INDEX branch_idx on Account (branch_name)

Make unique index

CREATE UNIQUE INDEX branch_idx on

Account (branch_name)

Make composite index

CREATE INDEX branch_idx on Account(account_number, branch_name)

Implicit Indexes:

Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

Drop index

DROP INDEX branch_idx on Account

When Should Indexes Be Avoided ?

- ✓ Index should not be used on **small tables**.
- ✓ Tables that have **frequent**, large batch update or insert operations.
  - ✓ **Note: Updating a table with indexes takes more time** than updating a table without (because the indexes also need an update). So you should only create indexes **on columns (and tables)** that will be frequently searched against.
- ✓ Indexes should not be used on columns that contain a high number of **NULL values**.
- ✓ Columns that are **frequently manipulated** should not be indexed.

Data Manipulation Language

Data Modification (CRUD)

- Add tuple to table (*insert*)
- Delete specific tuple from table (*delete*)
- Modify specific value of tuple (*update*)

Insertion

INSERT INTO R(A1,...,An) **VALUES** (v1,...,vn)

- Add tuple with value vi for attribute Ai, for i=1,...,n

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.

INSERT INTO R **VALUES** (v1,...,vn)

Two samples below have the same result:

- **insert into** account (branch-name, balance, account-number) **values** ('Perryridge', 1200, 'A-9732')
- **insert into** *account*
  values ('A-9732', 'Perryridge',1200) **---- correct order!**

Insertion

Two samples below are same :

- **insert into** *account (branch-name, account-number)* **values** (‘Perryridge’, ‘A-9732’)
- **insert into** *account (branch-name, account-number,balance)* **values** (‘Perryridge’, ‘A-9732’, NULL)

Deletion

DELETE FROM <Table Name> **WHERE** <condition>

- Delete specific tuple from table where condition(t) is true
- Example: remove all tuple in an account table in branch name Perryridge
- **delete from** account **where** branch_name = 'Perryridge'

What is the meaning of *delete from account ?*

Updates

Modify specific value of tuple

- **update** R **set** attribute = expression **where** <condition>

Example : pay interest rate 5% for account which have balance more than $1000. Update account balance **!**

- **update** account **set** balance = balance * 1.05 **where** balance > 1000

- **update** account **set** balance = 100.000 **where** balance > 1000

Updates

Question :

Pay interest rate 6% for account which have balance more than $1000.

Pay interest rate 5% for account which have balance less or equal than $1000.

Update account balance !

Answer :

With two commands update:

1. **update** *account* **set** *balance* = *balance* * 1.06 where *balance* > 10000
2. **update** *account* **set** *balance* = *balance* * 1.05
   where *balance* ≤ 10000

Another answer is case statement

Case Statement

Question :

Pay interest rate 6% for account which have balance more than $1000.

Pay interest rate 5% for account which have balance less or equal than $1000. Update account balance !

Answer :

```
update account
  set balance =
          case
               when balance <= 1000 then balance *1.05
            else   balance * 1.06
        end
```

Thanks!

Do you have any questions?