UNIVERSITY OF WESTMINSTER

7BUIS030W
Data System Concepts and Fundamentals

Lecture -6

Lecture-5 Outline

Physical database design and implementation, relational model, keys, introduction to SQL, creating and dropping tables using DDL, using constraints

Relational model

- An approach to managing data using a structure and language.
- The relational model is based on the mathematical concept of a relation, which is physically represented as a **table.**
- All data is logically structured within relations (tables)

Relational model

- The model helps to visualize database structure.
- Allows a high degree of data independence
- Provides substantial grounds for dealing with data consistency and redundancy problems

Relational Data structure

Relation:

A table with columns and rows

Attribute:

Named column of a relation

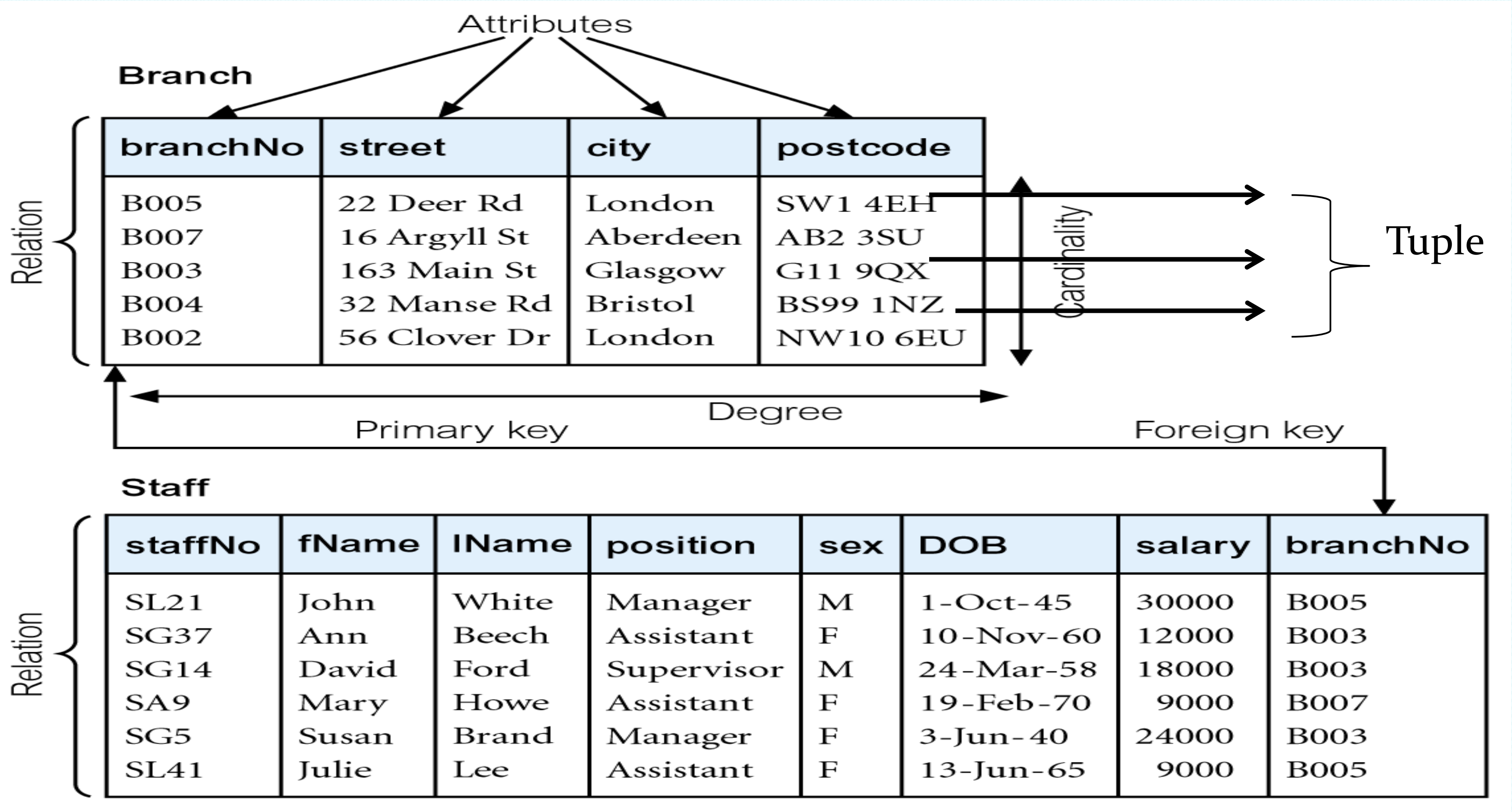
Tuple:

A row of a relation

Domain:

Set of allowable values for one or more attributes

Relational data structure



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relational data structure

Attribute	Domain name	Meaning	Domain Definitions
branchNo	BranchNumbers	The set of all possible branch numbers	Character, size 4
street	StreetNames	The set of all street names in the country	Character, size 25
city	CityNames	The set of all city names in the country	Character, size 15
postcode	postcodes	The set of all postcodes in the country	Character, size 8
sex	Gender	The gender of a person	Character, size 1
DOB	DateOfBirth	Possible values of staff birth dates	Date format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	7 digits

Relational Data structure

Degree:

Number of attributes it contains

Eg: branch relation has 4 attributes or degree is 4

Each row of the table is a four-tuple containing four values

Cardinality:

Number of tuples contained in a relation

Relational Data structure

Terminologies revisited:

Formal terms	Alternative-1	Alternative-2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Properties of relations

- Relation name is distinct from all other relation names in relational schema.
- Each cell of relation contains exactly one atomic (single) value.
- Each attribute has a distinct name.
- Values of an attribute are from the same domain.
- Order of attributes has no significance.(because relation is a set)
- Each tuple is distinct; no duplicate tuples.
- Order of tuples has no significance, theoretically.

Relational keys

Since no duplicate tuples within a relation, relational keys are used to identify each tuple.

Superkey: An attribute or set of attributes that uniquely identifies a tuple within a relation.

Eg: branchNo, city, postcode

Candidate Key

- Superkey (K) such that no proper subset is a superkey within the relation.
- In each tuple of R, values of K uniquely identify that tuple (uniqueness).
- In other words candidate keys are minimal super keys.

Eg: branchNo is a candidate key where a city is not

Relational keys

Primary Key

- Candidate key selected to identify tuples uniquely within relation.

Alternate Keys

- Candidate keys that are not selected to be primary key.

Eg: If branchNo is the PK then postcode is the alternate key

Foreign Key

- Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Eg: branchNo in both staff and branch relation

Null values

Null

- Represents value for an attribute that is currently unknown or not applicable for tuple.
- Deals with incomplete or exceptional data.
- Represents the absence of a value and is not the same as zero or spaces, which are values.

Integrity Constraints

Entity Integrity

- In a base relation, no attribute of a primary key can be null.

Referential Integrity

- If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.
- For eg: it is not possible to create a branch no exclusively for the staff relation if it does not appear in the branch relation

- General Constraints
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.

Relational model development

- Databases allow us to store and filter data to find specific information.
- A database can be queried using a variety of methods, one of which is using query languages
- A Query language is a written language used only to write specific queries.
- This is a powerful tool as the user can define precisely what is required in a database.

Structured Query Language-SQL

- SQL is the only standard database language to gain wide acceptance
- SQL is an example of a **transform-oriented language** – A language designed to use relations to transform inputs to required outputs

Structured Query Language-SQL

Objectives of SQLto gain wide acceptance

Ideally, SQL should allow user to :

- Create the database and relation structures
- Perform basic data managements tasks such as:
  - Insertion
  - Modification
  - Deletion

of data from the relations

- Perform both simple and complex queries

Structured Query Language-SQL

Components of SQL:

Data Definition Language(DDL) for defining the database structure and controlling access to data

Data Manipulation Language (DML) for retrieving and updating data

Structured Query Language-SQL

Components of SQL:

CREATE ALTER DROP RENAME TRUNCATE COMMENT	Data Definition Language (DDL)

SELECT
INSERT
UPDATE
DELETE
MERGE
	Data Manipulation Language (DML)

SQL Data Types

Data type	Declaration	Description
Character	CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
Character	VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). Can store up to 255 characters. **Note:** any greater value than 255 it will be converted to a TEXT type
Boolean	BOOLEAN	Consist of distinct truth values ‘True’ or ‘False’
Exact numeric	NUMERIC, DECIMAL,INTEGER, SMALL INT, BIGINT	To define numbers with an exact representation
Approximate numeric	FLOAT, REAL, DOUBLE PRECISION	To define numbers that do not have an exact representation
datetime	DATE,TIME, TIMESTAMP	Used to define points in time to a certain degree of accuracy
interval	INTERVAL	Used to represent periods of time
Large objects	CHARACTER LARGE OBJECTS,BINARY LARGE OBJECT	Data types that hold a large amount of data

SQL Data Types-numbers

Data type	Description
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

SQL Data Types- date

Data type	Description
DATE()	A date. Format: YYYY-MM-DD ***Note:*** *The supported range is from '1000-01-01' to '9999-12-31'*
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS ***Note:*** *The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'*
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS ***Note:*** *The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC*
TIME()	A time. Format: HH:MI:SS ***Note:*** *The supported range is from '-838:59:59' to '838:59:59'*
YEAR()	A year in two-digit or four-digit format. ***Note:*** *Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069*

Creating tables in SQL

```
CREATE TABLE table_name
( column1 data_type(size),
   column2 data_type(size),
    column3 data_type(size), .... );
```

table_name: name of the table.

column1 name of the first column.

data_type: Type of data we want to store in the particular column.

For example,**int** for integer data.

size: Size of the data we can store in a particular column. For example if for a column we specify the data_type as int and size as 10 then this column can store an integer number of maximum 10 digits.

Creating tables in SQL

Example-1

Create a table named Students with three columns, ROLL_NO, NAME and SUBJECT.

```
CREATE  TABLE  Students
 ( ROLL_NO int(3),
    NAME varchar(20),
    SUBJECT varchar(20)
 );
```

Creating tables in SQL

Example-2

Create a table named "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City.

```
CREATE TABLE Persons
 (
      personID   int,
      LastName  varchar(255),
      FirstName  varchar(255),
      Address     varchar(255),
      City           varchar(255)
   );
```

Creating tables in SQL- Constraints

- **SQL constraints** are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table.
- This ensures the accuracy and reliability of the data in the table.
- If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level.
  - Column level constraints apply to a column, and table level constraints apply to the whole table.

Creating tables in SQL- Constraints

- The following constraints are commonly used in SQL:
- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE**- Ensures that all values in a column are different
- **PRIMARY KEY**- A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY**- Uniquely identifies a row/record in another table
- **CHECK**- Ensures that all values in a column satisfies a specific condition
- **DEFAULT**- Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

Creating tables in SQL-
SQL NOT NULL Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

Creating tables in SQL- SQL NOT NULL Constraint

Example-3

Modify the query in example -2 to ensure that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

```
CREATE TABLE Persons
(
      personID   int NOT NULL,
      LastName  varchar(255) NOT NULL,
      FirstName  varchar(255) NOT NULL,
      Age             int
  );
```

Creating tables in SQL-
SQL UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Creating tables in SQL-
SQL UNIQUE Constraint

Example-4

Create a table for orders with columns “”

```
CREATE TABLE Persons (
    personID    int NOT NULL UNIQUE,
     LastName  varchar(255) NOT NULL,
     FirstName  varchar(255),
     Age            int
  );
```

Creating tables in SQL-
SQL PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

Creating tables in SQL-

SQL Primary Key Constraint

Example-5

Modify the query in example -4 to create a Primary key constraint on the “Persons" table

```
CREATE TABLE Persons
(
      personID    int NOT NULL UNIQUE,
      LastName  varchar(255) NOT NULL,
      FirstName  varchar(255),
      Age             int,
      constraint   p_pid_pk PRIMARY KEY (personID)
  );
```

Creating tables in SQL-
SQL Foreign KEY Constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Creating tables in SQL-
SQL Foreign KEY Constraint

- Example-6

Create a table named “orders" that contains 3 columns: orderID, orderNo, personID.

- Notice that the “personID" column in the "Orders" table points to the “personID" column in the "Persons" table.
- The “personID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.
- The “personID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Creating tables in SQL- SQL Foreign KEY Constraint

- Example-6

Create a table named “orders" that contains 3 columns: orderID, orderNo, personID.

```
CREATE TABLE Orders
(
   orderID     int NOT NULL,
   orderNo    int NOT NULL,
   personID  int,
   constraint o_oid_pk PRIMARY KEY (orderID),
   constraint o_pid_fk FOREIGN KEY (personID) REFERENCES Persons(personID)
);
```

The SQL DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

DROP TABLE *table_name*;

Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

The SQL TRUNCATE TABLE Statement

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

TRUNCATE TABLE *table_name*;

The SQL RENAME TABLE Statement

The rename table statement is used to change the name of the table.

RENAME TABLE *table_name to table_name2*;

RENAME TABLE Emp to Employee;

RENAME TABLE Dept to Department;

The SQL ALTER TABLE Statement

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

The SQL ALTER TABLE Statement

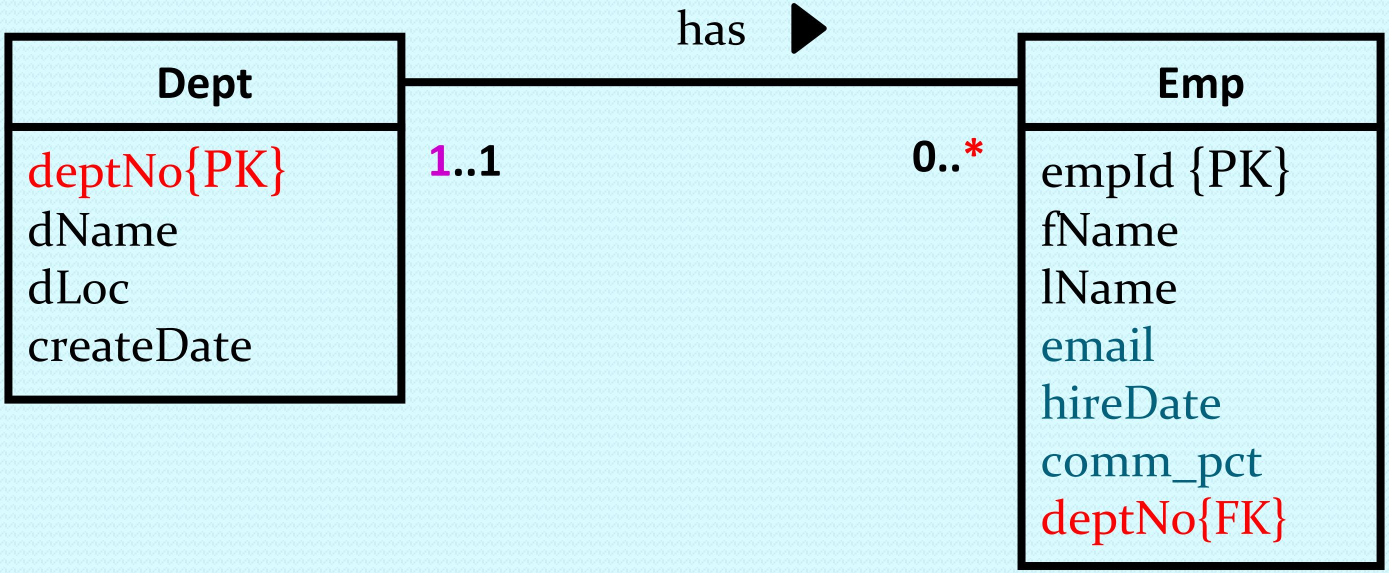
```
ALTER TABLE Emp ADD dateOfBirth date;-- add column


ALTER TABLE Emp
CHANGE COLUMN dateOfBirth DOB date; --change column

ALTER TABLE Emp DROP COLUMN DOB;--delete column
```

Creating Table in SQL

Exercise: For the following logical ERD create the table for the database



has ▶

Dept

deptNo{PK}
dName
dLoc
createDate

1..1

0..*

Emp

empId {PK}
fName
lName
email
hireDate
comm_pct
deptNo{FK}

```
CREATE TABLE Dept
(
        deptNo        INT(4),
        dName         VARCHAR(20) UNIQUE NOT NULL
        dLoc          VARCHAR(30) NOT NULL
        createDate    TIMESTAMP,
        constraint    d_dno_pk PRIMARY KEY (deptNo)
)
ENGINE=InnoDB
```

(Note: InnoDB is a storage engine that ensures referential integrity, it is the default storage engine on the Uni setup)

```
CREATE TABLE Emp
(
        empId          INT(6),
        fName          VARCHAR(50) NOT NULL,
        lName          VARCHAR(50) NOT NULL,
        email          VARCHAR(100) UNIQUE NOT NULL,
        hireDate       DATE,
        comm_pct       DECIMAL(2,2),
        deptNo         INT(4) NOT NULL,
        constraint     e_eid_pk PRIMARY KEY (empId),
        constraint     e_dno_fk FOREIGN KEY (deptNo)
        references     Dept(deptNo)
);
```

Not Null Constraint needed