CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

 pk.profgiri  /in/ponguru  @ponguru  Ponnurangam.kumaraguru

Update Operations on Relations

INSERT a tuple

DELETE a tuple

MODIFY a tuple

Integrity constraints should not be violated by the update operations

Several update operations may have to be grouped together

Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints

Possible violations for each operation

INSERT may violate any of the constraints:

- Domain constraint:
  - if one of the attribute values provided for the new tuple is not of the specified attribute domain
- Key constraint:
  - if the value of a key attribute in the new tuple already exists in another tuple in the relation
- Referential integrity:
  - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
- Entity integrity:
  - if the primary key value is null in the new tuple

Operation:
Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.
Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Operation:
Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.
Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Operation:
Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.
Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

Possible violations for each operation

DELETE may violate only referential integrity:

- If the primary key value of the tuple being deleted is referenced from other tuples in the database
  - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL
    - RESTRICT option: reject the deletion
    - CASCADE option: by deleting tuples that reference the tuple that is being deleted
    - SET NULL option: set the foreign keys of the referencing tuples to NULL
- One of the above options must be specified during database design for each foreign key constraint

Operation:
Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.
Result: This deletion is acceptable and deletes exactly one tuple.

Operation:
Delete the EMPLOYEE tuple with Ssn = '999887777'.
Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Operation:
Delete the EMPLOYEE tuple with Ssn = '333445555'.
Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

What are the results?

Possible violations for each operation

UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified

Any of the other constraints may also be violated, depending on the attribute being updated:

- Updating the primary key (PK):
  - Similar to a DELETE followed by an INSERT
  - Need to specify similar options to DELETE
- Updating a foreign key (FK):
  - May violate referential integrity
- Updating an ordinary attribute (neither PK nor FK):
  - Can only violate domain constraints

Operation:
Update the salary of the EMPLOYEE tuple with Ssn = '999887777' to 28000.
Result: Acceptable.

Operation:
Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 1.
Result: Acceptable.

Operation:
Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7.
Result: Unacceptable, because it violates referential integrity.

Operation:
Update the Ssn of the EMPLOYEE tuple with Ssn = '999887777' to '987654321'.
Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.

Schema and Catalog Concepts in SQL (cont'd.)

`CREATE SCHEMA` statement

```
CREATE SCHEMA COMPANY AUTHORIZATION 'Jsmith';
```

Catalog

Named collection of schemas in an SQL environment

SQL also has the concept of a cluster of catalogs

Authorization is to make the owner of the Schema

The CREATE TABLE Command in SQL

Specifying a new relation

- Provide name of table
- Specify attributes, their types and initial constraints

Can optionally specify schema:

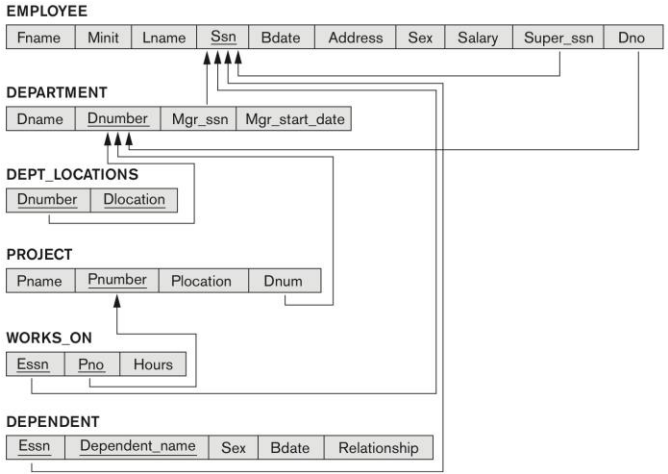
```
CREATE TABLE COMPANY.EMPLOYEE ...
```

or

```
CREATE TABLE EMPLOYEE ...
```

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1)

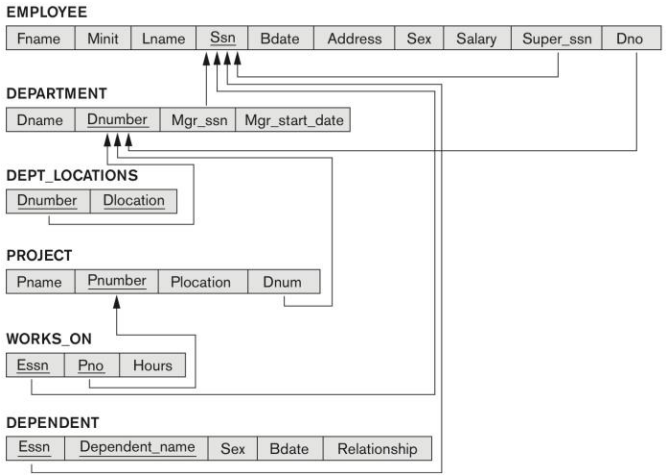
```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```



continued on next slide

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1)-continued

```
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```



This Lecture

Attribute Data Types and Domains in SQL

Basic **data types**

- **Numeric** data types
  - Integer numbers: `INTEGER`, `INT`, and `SMALLINT`
  - Floating-point (real) numbers: `FLOAT` or `REAL`, and `DOUBLE PRECISION`
- **Character-string** data types
  - Fixed length: `CHAR(n)`, `CHARACTER(n)`
  - Varying length : `VARCHAR(n)`, `CHAR VARYING(n)`, `CHARACTER VARYING(n)`

Data type	Range	Storage
BIGINT	-2^{63} (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)	8 Bytes
INT	**-2^{31} (-2,147,483,648) to $2^{31}-1$ (2,147,483,647)**	**4 Bytes**
SMALLINT	-2^{15} (-32,768) to $2^{15}-1$ (32,767)	2 Bytes
TINYINT	0 to 255	1 Byte

Attribute Data Types and Domains in SQL (cont'd.)

Bit-string data types [0s, 1s]

- Fixed length: `BIT(n)`
- Varying length: `BIT VARYING(n)`

Boolean data type

- Values of `TRUE` or `FALSE` or `NULL`

DATE data type

- Ten positions
- Components are `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD
- Multiple mapping functions available in RDBMSs to change date formats

Attribute Data Types and Domains in SQL (cont'd.)

Additional data types

- **Timestamp** data type
- Includes the `DATE` and `TIME` fields
  - Plus a minimum of six positions for decimal fractions of seconds
  - Optional `WITH TIME ZONE` qualifier
- **INTERVAL** data type
  - Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp; useful for scheduling, project timelines, etc.
- **DATE, TIME, Timestamp, INTERVAL** data types can be **cast** or converted to string formats for comparison.

```
SELECT DATE '2025-10-28' + INTERVAL '5' DAY AS new_date;
```

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum string length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data

https://www.w3schools.com/sql/sql_datatypes.asp

Numeric Data Types

Data type	Description
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)

https://www.w3schools.com/sql/sql_datatypes.asp

Date & Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	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:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

https://www.w3schools.com/sql/sql_datatypes.asp

Attribute Data Types and Domains in SQL (cont'd.)

Domain

- Name used with the attribute specification
- Makes it easier to change the data type for a domain that is used by numerous attributes
- Improves schema readability
- Example:

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
CREATE DOMAIN CPI_DATA AS REAL CHECK
(value >= 0 AND value <= 10);
```

Attribute Data Types and Domains in SQL (cont'd.)

Domain

```
CREATE DOMAIN AgeDomain AS INT
CHECK (VALUE BETWEEN 0 AND 120);

CREATE TABLE Person (
  name VARCHAR(50),
  age AgeDomain
);
```

Attribute Data Types and Domains in SQL (cont'd.)

TYPE

User Defined Types (UDTs) are supported for object-oriented applications. (See Ch.12) Uses the command:

```
CREATE TYPE AUDIO  AS BLOB (1M) [Binary Large OBject]

CREATE TYPE SalaryRange AS (
  min_salary DECIMAL(10,2),
  max_salary DECIMAL(10,2)
);
```

Specifying Constraints in SQL

Basic constraints:

Relational Model has 3 basic constraint types that are supported in SQL:

Key constraint: A primary key value cannot be duplicated

Entity Integrity Constraint: A primary key value cannot be null

Referential integrity constraints : The “foreign key “ must have a value that is already present as a primary key, or may be null.

Specifying Attribute Constraints

Other Restrictions on attribute domains:

Default value of an attribute

```
DEFAULT <value>
```

NULL is not permitted for a particular attribute (NOT NULL)

`CHECK` clause

```
Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

CHECK (Dept_create_date <= Mgr_start_date);
```

Specifying Attribute Constraints

CHECK clause

```
Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

CREATE TABLE Accounts (
  account_id INT PRIMARY KEY,
  balance DECIMAL(10,2),
  min_balance DECIMAL(10,2),
  CHECK (balance >= min_balance)
);
```

Specifying Attribute Constraints

```
CREATE TABLE Students (
  student_id INT PRIMARY KEY,
  age INT CHECK (age >= 5 AND
age <= 25)
);
```

Specifying Key and Referential Integrity Constraints

`PRIMARY KEY` clause

Specifies one or more attributes that make up the primary key of a relation

```
Dnumber INT PRIMARY KEY;
```

`UNIQUE` clause

Specifies alternate (secondary) keys (called CANDIDATE keys in the relational model).

```
Dname VARCHAR(15) UNIQUE;
```

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.

https://www.w3schools.com/sql/sql_unique.asp

Specifying Key and Referential Integrity Constraints (cont'd.)

`FOREIGN KEY` clause

- Default operation: reject update on violation
- Attach **referential triggered action** clause
  - Options include `SET NULL`, `SET CASCADE`, and `SET DEFAULT`
  - Action taken by the DBMS for `SET NULL` or `SET DEFAULT` is the same for both `ON DELETE` and `ON UPDATE`
  - `CASCADE` option suitable for some propagation needs to be done [Manager leaving organization, or somebody stepping down as manager]

Giving Names to Constraints

Using the Keyword **`CONSTRAINT`**

- Name a constraint
- Useful for later altering

```
CREATE TABLE EMPLOYEE
    ( … ,
      Dno          INT            NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( … ,
      Mgr_ssn CHAR(9)          NOT NULL       DEFAULT '888665555',
      … ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( … ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE CASCADE           ON UPDATE CASCADE);
```

Default attribute values and referential integrity triggered action specification (Fig. 6.2)

Basic Retrieval Queries in SQL

`SELECT` statement

- One basic statement for retrieving information from a database

SQL allows a table to have two or more tuples that are identical in all their attribute values [Results from the query]

- Unlike relational model (relational model is strictly set-theory based)
- Tuple-id may be used as a key

The SELECT-FROM-WHERE Structure of Basic SQL Queries

Basic form of the `SELECT` statement:

SELECT <attribute list>
FROM <table list>
WHERE <condition>;

where

- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

The SELECT-FROM-WHERE Structure of Basic SQL Queries (cont'd.)

Logical comparison operators

=, <, <=, >, >=, and <>

Projection attributes

Attributes whose values are to be retrieved

Selection condition

Boolean condition that must be true for any retrieved tuple. Selection conditions include join conditions (see Ch.8) when multiple relations are involved.

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname='John' AND Minit='B' AND Lname='Smith';
```

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname='John' AND Minit='B' AND Lname='Smith';
```

Bdate	Address
1965-01-09	731Fondren, Houston, TX

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname='Research' AND Dnumber=Dno;
```

Basic Retrieval Queries

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname='Research' AND Dnumber=Dno;
```

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

Where condition

```
mysql> select fname, lname, address from employee, department where dname='research' AND dno = dnumber;
+----------+---------+--------------------------+
```

Where condition

```
mysql> select fname, lname, address from employee, department where dname='research' AND dno = dnumber;
+----------+---------+-------------------------+
| fname    | lname   | address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.00 sec)
```

Where condition

```
mysql> select fname, lname, address from employee, department where dname='research' AND dno = dnumber;
+----------+---------+-------------------------+
| fname    | lname   | address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.00 sec)

mysql> select fname, lname, address from employee, department where dno = dnumber AND dname='research';
```

Where condition

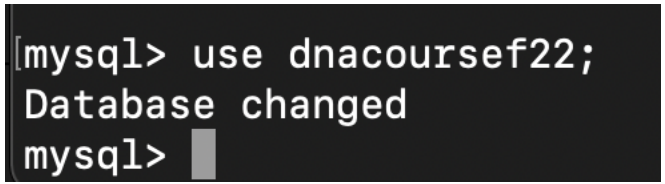
```
mysql> select fname, lname, address from employee, department where dname='research' AND dno = dnumber;
+----------+---------+-------------------------+
| fname    | lname   | address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.00 sec)

mysql> select fname, lname, address from employee, department where dno = dnumber AND dname='research';
+----------+---------+-------------------------+
| fname    | lname   | address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.01 sec)
```

Case In-sensitive

```
mysql> SELECT Fname, Lname, Address FROM EMPLOYEE, DEPARTMENT WHERE Dname='Research' AND Dnumber=Dno;
+----------+---------+-------------------------+
| Fname    | Lname   | Address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.00 sec)
```

Hands-on for some basics

mysql> use dnacoursef22;



Hands-on for some basics

mysql> show tables;

```
mysql> show tables;
+------------------------+
| Tables_in_dnacoursef22 |
+------------------------+
| DEPARTMENT             |
| DEPENDENT              |
| DEPT_LOCATIONS         |
| EMPLOYEE               |
| PROJECT                |
| WORKS_ON               |
+------------------------+
6 rows in set (0.01 sec)
```

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM EMPLOYEE, DEPARTMENT, PROJECT
WHERE DNUM=DNUMBER AND Mgr_ssn=Ssn AND
Plocation = 'Stafford’;

```
mysql> SELECT Pnumber, Dnum, Lname, Address, Bdate
    -> FROM EMPLOYEE, DEPARTMENT, PROJECT
    -> WHERE DNUM=DNUMBER AND Mgr_ssn=Ssn AND
    -> Plocation = 'Stafford';
+---------+------+---------+-----------------------+------------+
| Pnumber | Dnum | Lname   | Address               | Bdate      |
+---------+------+---------+-----------------------+------------+
|      10 |    4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |
|      30 |    4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |
+---------+------+---------+-----------------------+------------+
2 rows in set (0.00 sec)
```

Unspecified WHERE Clause and Use of the Asterisk

```
Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;
```

Unspecified WHERE Clause and Use of the Asterisk

Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;

```
mysql> Select Ssn
    -> FROM EMPLOYEE;
+-----------+
| Ssn       |
+-----------+
| 123456789 |
| 333445555 |
| 453453453 |
| 666884444 |
| 888665555 |
| 987654321 |
| 987987987 |
| 999887777 |
+-----------+
8 rows in set (0.00 sec)
```

Unspecified WHERE Clause and Use of the Asterisk

```
Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;
```

Unspecified WHERE Clause and Use of the Asterisk

Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;

```
mysql> SELECT SSN, DNAME
    -> FROM EMPLOYEE, DEPARTMENT;
+-----------+----------------+
| SSN       | DNAME          |
+-----------+----------------+
| 123456789 | Research       |
| 123456789 | Headquarters   |
| 123456789 | Administration |
| 333445555 | Research       |
| 333445555 | Headquarters   |
| 333445555 | Administration |
| 453453453 | Research       |
| 453453453 | Headquarters   |
| 453453453 | Administration |
| 666884444 | Research       |
| 666884444 | Headquarters   |
| 666884444 | Administration |
| 888665555 | Research       |
| 888665555 | Headquarters   |
| 888665555 | Administration |
| 987654321 | Research       |
| 987654321 | Headquarters   |
| 987654321 | Administration |
| 987987987 | Research       |
| 987987987 | Headquarters   |
| 987987987 | Administration |
| 999887777 | Research       |
| 999887777 | Headquarters   |
| 999887777 | Administration |
+-----------+----------------+
24 rows in set (0.00 sec)
```

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)

Specify an asterisk (*)

Retrieve all the attribute values of the selected tuples

```
SELECT *
FROM EMPLOYEE
WHERE Dno = 5;
```

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)

Specify an asterisk (*)

Retrieve all the attribute values of the selected tuples

```
SELECT *
FROM EMPLOYEE
WHERE Dno = 5;
```

```
mysql> SELECT *
    -> FROM EMPLOYEE
    -> WHERE Dno = 5;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                 | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M    |  40000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX   | F    |  25000 | 333445555 |   5 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
4 rows in set (0.00 sec)
```

```
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dno=Dnumber;
```

```
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dno=Dnumber;
```

Attributes from both tables

```
mysql> SELECT *
    -> FROM EMPLOYEE, DEPARTMENT
    -> WHERE Dname='Research' AND Dno=Dnumber;
+----------+-------+----------+-----------+------------+--------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
| Fname    | Minit | Lname    | Ssn       | Bdate      | Address                  | Sex | Salary | Super_ssn | Dno | Dname    | Dnumber | Mgr_ssn   | Mgr_start_date |
+----------+-------+----------+-----------+------------+--------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
| John     | B     | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston TX  | M   |  30000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Franklin | T     | Wong     | 333445555 | 1965-12-08 | 638 Voss, Houston TX     | M   |  40000 | 888665555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Joyce    | A     | English  | 453453453 | 1972-07-31 | 5631 Rice, Houston TX    | F   |  25000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Ramesh   | K     | Narayan  | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX  | M   |  38000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
+----------+-------+----------+-----------+------------+--------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
4 rows in set (0.00 sec)
```

```
SELECT *
FROM EMPLOYEE, DEPARTMENT;
```

SELECT *
FROM EMPLOYEE, DEPARTMENT;

Attributes from both tables

```
mysql> SELECT *
    -> FROM EMPLOYEE, DEPARTMENT;
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dname	Dnumber	Mgr_ssn	Mgr_start_date
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Research	5	333445555	1988-05-22
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Administration	4	987654321	1995-01-01
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Headquarters	1	888665555	1981-06-19
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Research	5	333445555	1988-05-22
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Administration	4	987654321	1995-01-01
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Headquarters	1	888665555	1981-06-19
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Research	5	333445555	1988-05-22
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Administration	4	987654321	1995-01-01
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Headquarters	1	888665555	1981-06-19
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Research	5	333445555	1988-05-22
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Administration	4	987654321	1995-01-01
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Headquarters	1	888665555	1981-06-19
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Research	5	333445555	1988-05-22
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Administration	4	987654321	1995-01-01
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Headquarters	1	888665555	1981-06-19
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Research	5	333445555	1988-05-22
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Administration	4	987654321	1995-01-01
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Headquarters	1	888665555	1981-06-19
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Research	5	333445555	1988-05-22
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Administration	4	987654321	1995-01-01
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Headquarters	1	888665555	1981-06-19
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Research	5	333445555	1988-05-22
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Administration	4	987654321	1995-01-01
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Headquarters	1	888665555	1981-06-19

```
24 rows in set (0.00 sec)
```

Activity

Modify CREATE Table for PROJECT, WORKS_ON & DEPENDENT for CONSTRAINT, DEFAULT, ON DELETE, ON UPDATE

```
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in