Introduction to SQL

History

- IBM Sequel language developed as part of System R project at the IBM San Jose Research Laboratory
  - Renamed Structured Query Language (SQL)
  - ANSI and ISO standard SQL:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL:1999 (language name became Y2K compliant!)
  - SQL:2003
- Comm. systems offer SQL-92 features, plus varying feature sets from later standards and special proprietary features.
  - Not all examples here may work on your particular system

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

Create Table Construct

- An SQL relation is defined using the **create table** command:
- **create table** *r*(*A*1 *D*1, *A*2 *D*2, ..., *An Dn,*

  (integrity-constraint1),

  ...

  (integrity-constraintk))
  - *r* is the name of the relation
  - each *Ai* is an attribute name in the schema of relation *r*
  - *Di* is the data type of values in the domain of attribute *Ai*
- Example: **create table** *instructor*(

  ID **char**(5),

  name **varchar**(20) **not null,**

  *dept_name***varchar**(20),

  *salary***numeric**(8,2))
- **insert into** *instructor* **values** (‘10211’, ’Smith’, ’Biology’, 66000);
- **insert into** *instructor* **values** (‘10211’, null, ’Biology’, 66000);

Integrity Constraints in Create Table

- **not null**
- **primary key**(*A*1, ..., *An*)
- **foreign key** (*A*m, ..., *An*) **references** *r*
- Example: **create table** *instructor*(

  ID **char**(5),
  name **varchar**(20) **not null,**
  dept_name **varchar**(20),
  salary **numeric**(8,2),
  primary key (*ID*),
  foreign key *(dept_name*) **references** *department))*

- **primary key** declaration on an attribute automatically ensures **not null**

More Relation Definitions

- **create table** *student*(

  ID **varchar**(5) **primary key**,
  name **varchar**(20) not null,
  dept_name **varchar**(20),
  tot_cred **numeric**(3,0),
  foreign key *(dept_name*) **references** *department)*);

- **create table** *takes*(

  ID **varchar**(5) **primary key**,
  course_id **varchar**(8),
  sec_id **varchar**(8),
  semester **varchar**(6),
  year **numeric**(4,0),
  grade **varchar**(2),
  foreign key (*ID*) **references** *student,*
  foreign key (*course_id, sec_id, semester, year*) **references** *section*);

- **create table** *course*(

  course_id **varchar**(8) **primary key**,

  title **varchar(**50),

  dept_name **varchar**(20),

  credits **numeric**(2,0),

  foreign key *(dept_name*) **references** *department)*);

- Some foreign keys may cause errors
  - Specified either via:
    - Circular references, or
    - they refer to a table that has not yet been created

```
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),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
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) );
```

Figure 4.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 3.7.

```
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber           INT             NOT NULL,
          Dlocation         VARCHAR(15)     NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
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) );
```

Figure 4.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 3.7.

Drop and Alter Table Constructs

- **drop table**
- **alter table**
- **alter table** *r* **add** *A D*

  A is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
- All tuples in the relation are assigned *null* as the value for the new attribute.
- **alter table** *r* **drop** *A*

  where *A* is the name of an attribute of relation *r*

- Dropping of attributes not supported by many databases.

INSERT, DELETE, and UPDATE Statements to modify the database

- INSERT: Specify the relation name and a list of values for the tuple

```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

```
U3B:  INSERT INTO   WORKS_ON_INFO ( Emp_name, Proj_name,
                    Hours_per_week )
      SELECT        E.Lname, P.Pname, W.Hours
      FROM          PROJECT P, WORKS_ON W, EMPLOYEE E
      WHERE         P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

The DELETE Command

- Removes tuples from a relation
  - Includes a `WHERE` clause to select the tuples to be deleted

```
U4A:  DELETE FROM   EMPLOYEE
      WHERE         Lname='Brown';

U4B:  DELETE FROM   EMPLOYEE
      WHERE         Ssn='123456789';

U4C:  DELETE FROM   EMPLOYEE
      WHERE         Dno=5;

U4D:  DELETE FROM   EMPLOYEE;
```

The UPDATE Command

- Modify attribute values of one or more selected tuples
- Additional **`SET`** clause in the `UPDATE` command
  - Specifies attributes to be modified and new values

```
U5:   UPDATE   PROJECT
      SET      Plocation = 'Bellaire', Dnum = 5
      WHERE    Pnumber=10;
```

Basic Retrieval Queries in SQL

- `SELECT` statement: for retrieving information from a database

  select *A*1, *A*2, ..., *An*

  from *r*1, *r*2, ..., *rm*

  where *P*

  - *Ai* represents an attribute
  - *Ri* represents a relation
  - *P* is a predicate.
- SQL allows a table to have two or more tuples that are identical in all their attribute values
  - Unlike relational model (RA)
  - Multiset or bag behavior
  - DISTINCT option makes result table: set of tuples

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.

Structure of Basic SQL Queries

- Logical comparison operators: =, <, <=, >, >=, and <>
- **Projection attributes (project of RA)**
  - Attributes whose values are to be retrieved
  - Example: find the names of all instructors:

    select *name*

    from *instructor*

NOTE: SQL names are case insensitive (i.e., you may use upper-or lower-case letters.)E.g., *Name*≡*NAME*≡*name*

- **Selection condition (select of RA)**
  - Boolean condition that must be true for any retrieved tuple

The select Clause

SQL allows duplicates in relations as well as in query results.

To force the elimination of duplicates, insert the keyword **distinct** after select.

Find the names of all departments with instructor, and remove duplicates

select distinct *dept_name*

from *instructor*

The keyword **all** specifies that duplicates not be removed.

select all *dept_name*

from *instructor*

The select Clause

An asterisk in the select clause denotes “all attributes”

select *

from *instructor*

The **select** clause can contain arithmetic expressions involving the operation, +, –, ∗, and /, and operating on constants or attributes of tuples.

The query:

select *ID, name, salary/12*

from *instructor*

would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.

The where Clause

where clause specifies conditions the result must satisfy

Corresponds to the **selection predicate** of the RA.

To find all instructors in Comp. Sci. dept with salary > 80000

select *name*

from *instructor*

where *dept_name*='Comp. Sci.'**and** *salary* > 80000

Comparison results can be combined using the logical connectives **and, or,** and **not.**

Comparisons can be applied to results of arithmetic expressions.

The from Clause

The **from** clause lists the relations involved in the query

Corresponds to the Cartesian product operation of the relational algebra.

Find the Cartesian product *instructor X teaches*

select ∗

from *instructor, teaches*

generates every possible instructor –teaches pair, with all attributes from both relations.

Figure 3.6
One possible database state for the COMPANY relational database schema.

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

Figure 3.6
One possible database state for the COMPANY relational database schema.

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

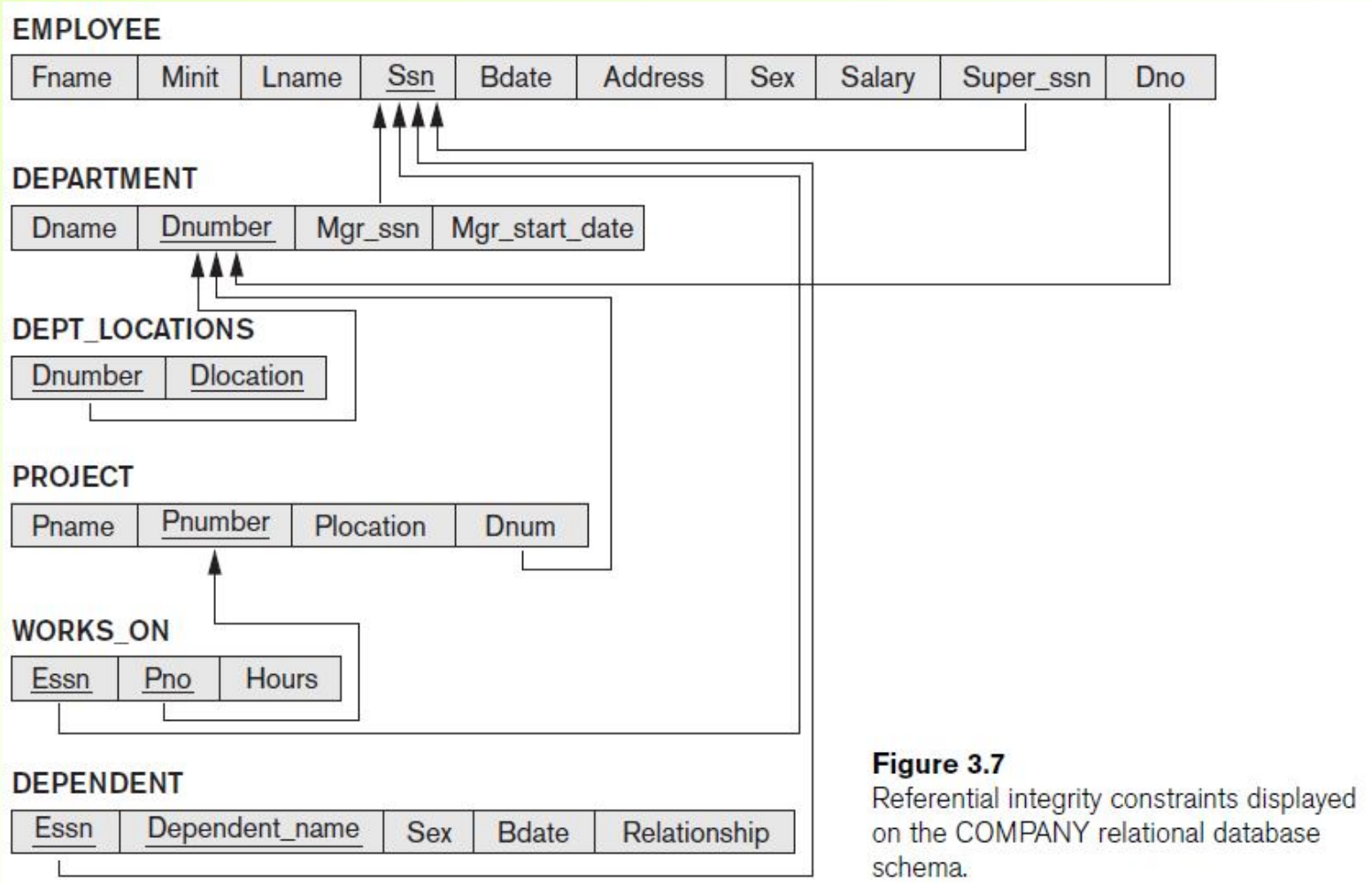

Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(a)

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

(b)

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

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';
```

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;
```

Examples

$$\pi_{BDATE,ADDRESS}\,(\sigma_{FNAME='John'\ AND\ MINIT='B'\ AND\ LNAME='Smith'}\,(EMPLOYEE))$$

Examples

Q3: For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor

- **SELECT** E.FNAME, E.LNAME, S.FNAME, S.LNAME
  FROM EMPLOYEE **AS** E, EMPLOYEE **AS** S
  WHERE E.SUPERSSN=S.SSN;
- declare alternative relation names E and S, called **aliases** or **tuple variables,** for the EMPLOYEE relation.
- EMPLOYEE **AS** E(FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO)

Examples

- **Q1B:**
- **SELECT** E.FNAME, E.NAME, E.ADDRESS

  FROM EMPLOYEE E, DEPARTMENT D

  WHERE D.NAME='Research' **AND** D.DNUMBER=E.DNUMBER;
- **Q1C:**
- **SELECT** * **FROM** EMPLOYEE **WHERE** DNO=5;
- **Q1D:**
- **SELECT** *

  FROM EMPLOYEE, DEPARTMENT

  WHERE DNAME='Research' **AND** DNO=DNUMBER;

Retrieve all attributes of an EMPLOYEE and attributes of the DEPARTMENT (s)he works in for every employee of 'Research' dept

Examples

Q10A:

SELECT * **FROM** EMPLOYEE,DEPARTMENT;

query Q1D retrieves all the attributes of an EMPLOYEE and the attributes of the DEPARTMENT he or she works in for every employee of the 'Research' department;

Q10A specifies the CROSS PRODUCT of the EMPLOYEE and DEPARTMENT relations.