BASIC SQL

LECTURE OUTLINE

SQL Data Definition and Data Types

Specifying Constraints in SQL

Basic Retrieval Queries in SQL

Set Operations in SQL

BASIC SQL

Structured Query Language

Considered one of the major reasons for the commercial success of relational databases

Statements for data definitions, queries, and updates

- Both DDL and DML
- Core specification plus specialized extensions

Terminology:

Relational Model	SQL
relation	**table**
tuple	**row**
attribute	**column**

Syntax notes:

- Some interfaces require each statement to end with a semicolon.
- SQL is not case-sensitive.

SQL DATA DEFINITION

CREATE statement

•Main SQL command for data definition

SQL schema

•Identified by a **schema name**

•Includes an **authorization identifier** (owner)

•Components are **descriptors** for each schema element

•Tables, constraints, views, domains, and other constructs

CREATE SCHEMA statement

• CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

CREATE TABLE COMMAND

Specify a new relation

- Provide name
- Specify attributes and initial constraints
- **Base tables** (**base relations**)
  - Relation and its tuples are physically stored and managed by DBMS

Can optionally specify schema:

- CREATE TABLE COMPANY.EMPLOYEE ...

or

- CREATE TABLE EMPLOYEE ...

Include information for each column (attribute) plus constraints

- Column name
- Column type (domain)
- Key, uniqueness, and null constraints

BASIC DATA TYPES

INTINTEGERSMALLINT, BIGINT

REALDOUBLE , FLOAT

Numeric data types DECIMAL(*n,m*), DEC(*n,m*), NUMERIC(*n,m*), NUM(*n,m*)

- Integer numbers: , ,
- Floating-point (real) numbers: ,
- Fixed-point numbers:

- **Character-string** data types
  - Fixed length: CHAR(*n*)CHARACTER(*n*)
  - Varying length: , VARCHAR(*n*)CHAR VARYING(*n*), CHARACTER VARYING(*n*), LONG VARCHAR

Large object data types

- Characters: , CLOBCHAR LARGE OBJECT , CHARACTER LARGE OBJECT
- Bits: , BLOBBINARY LARGE OBJECT

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

MORE DATA TYPES

Additional data types

- TIMESTAMPdata 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 absolut value of a date, time, or timestamp

Columns can be declared to be NOT NULL

Columns can be declared to have a default value

•Assigned to column in any tuple for which a value is not specified

Example

```
CREATE TABLE EMPLOYEE (
...
NICKNAME VARCHAR(20) DEFAULT NULL,
...
        Province CHAR(2) NOT NULL DEFAULT 'ON',
...
);
```

```
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);
[redacted]
CREATE TABLE DEPARTMENT
        ( Dname          VARCHAR(15)        NOT NULL,
          Dnumber        INT                NOT NULL,
          Mgr_ssn        CHAR(9)            NOT NULL,
          Mgr_start_date DATE                         );
[redacted]
```

Figure 4.1
SQL CREATE TABLE

[redacted]

DOMAINS IN SQL

Name used in place of built-in data type

Makes it easier to change the data type used by numerous columns Improves schema readability

Example:

```
CREATE DOMAIN SIN_TYPE AS CHAR(9);
```

SPECIFYING KEY CONSTRAINTS

PRIMARY KEY clause

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

Dnumber INT NOT NULL PRIMARY KEY,

•Primary key attributes must be declared NOT NULL

UNIQUE clause

•Specifies alternate (candidate) keys

Dname VARCHAR(15) UNIQUE;

•May or may not allow null values, depending on declaration

If no key constraints, two or more tuples may be identical in *all* columns.

•SQL deviates from pure relational model!

•Multiset (bag) behaviour

REFERENTIAL CONSTRAINTS

FOREIGN KEY clause

FOREIGN KEY (Dept) REFERENCES DEPARTMENT (Dnum),

- Default operation: reject update on violation
- Attach **referential triggered action** clause in case referenced tuple is deleted
- Options include SET NULL, CASCADE, and SET DEFAULT

Foreign key declaration must refer to a table already created

SPECIFYING TUPLE CONSTRAINTS

Some constraints involve several columns

CHECK clause at the end of a CREATE TABLE statement

•Apply to each tuple individually

Example

- CHECK (Dept_create_date <= Mgr_start_date)

EXAMPLE

Recall Employee example:

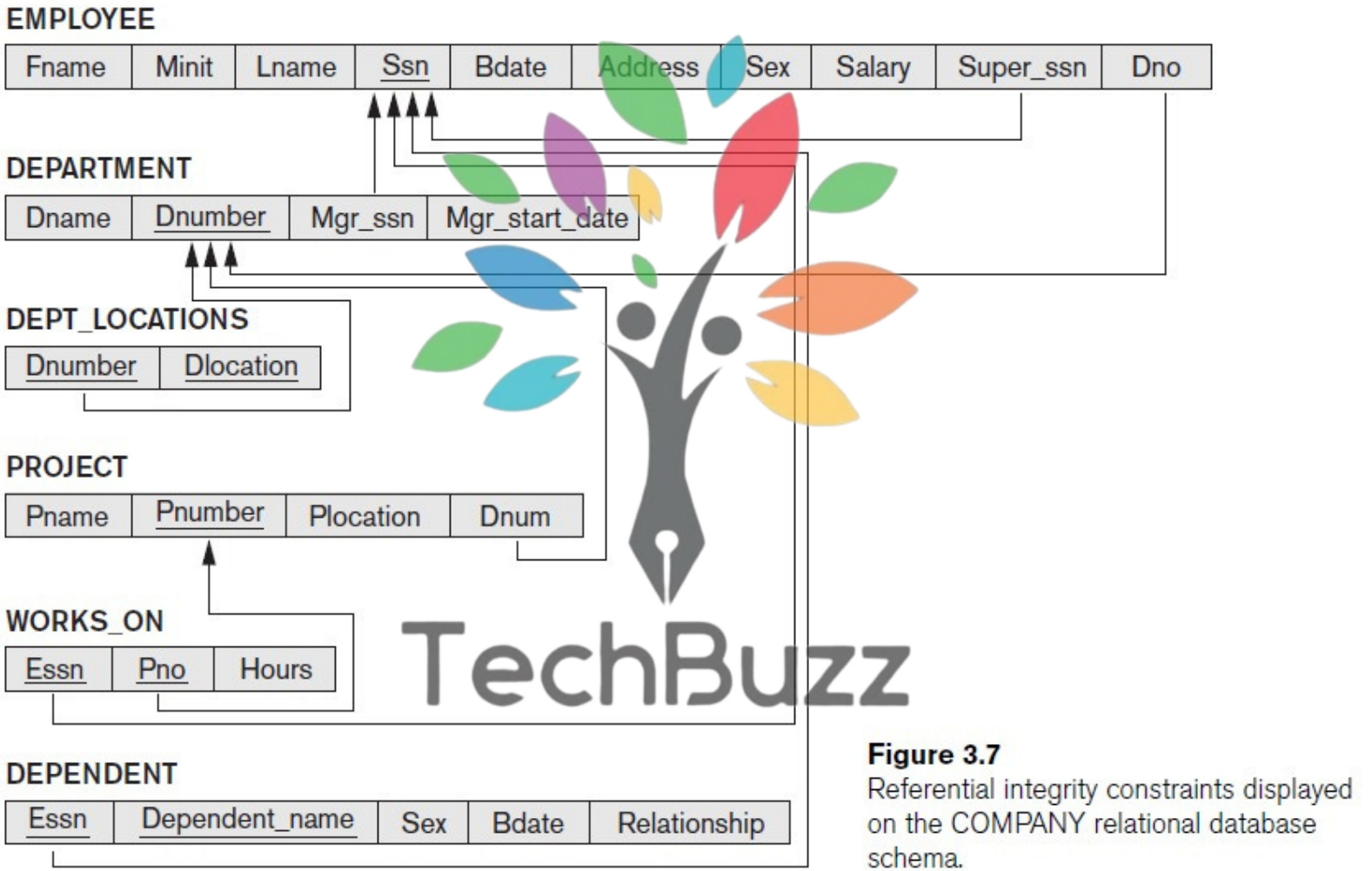

Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

```
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);
```

Figure 4.2
Example illustrating how default attribute values and referential integrity triggered actions are specified in SQL.

```
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.

BASIC SQL RETRIEVAL QUERIES

All retrievals use SELECT statement:

```
SELECT <return list>
FROM <table list>
[ WHERE <condition> ] ;
```

where

<return list>	is a list of expressions or column names whose values are to be retrieved by the query
<table list>	is a list of relation names required to process the query
<condition>	is a Boolean expression that identifies the tuples to be retrieved by the query

Example

```
SELECT title, year, genre
FROM Film
WHERE director = 'Steven Spielberg' AND year > 1990;
```

Omitting WHERE clause implies all tuples selected.

SEMANTICS FOR 1 RELATION

1.Start with the relation named in the FROM clause

2.Consider each tuple one after the other, eliminating those that do not satisfy the WHERE clause.

•Boolean condition that must be *true* for any retrieved tuple

•Logical comparison operators

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

3.For each remaining tuple, create a return tuple with columns for each expression (column name) in the SELECT clause.

•Use SELECT * to select all columns.

Film

title	genre	year	director	minutes	budget gross	
The Company Men	drama	2010	John Wells	104	15,000,000	4,439,063
Lincoln	biography	2012	Steven Spielberg	150	65,000,000	181,408,467
War Horse	drama	2011	Steven Spielberg	146	66,000,000	79,883,359
Argo	drama	2012	Ben Affleck	120	44,500,000	135,178,251

SELECT-FROM-WHERE SEMANTICS

What if there are several relations in the FROM clause?

1. Start with cross-product of all relation(s) listed in the FROM clause.
   - Every tuple in *R1* paired up with every tuple in *R2* paired up with …
2. Consider each tuple one after the other, eliminating those that do not satisfy the WHERE clause.
3. For each remaining tuple, create a return tuple with columns for each expression (column name) in the SELECT clause.

Steps 2 and 3 are just the same as before.

```
SELECT actor, birth, movie
FROM Role, Person
WHERE actor = name and birth > 1940;
```

Role

actor	movie	persona
Ben Affleck	Argo	Tony Mendez
Alan Arkin	Argo	Lester Siegel
Ben Affleck	The Company Men	Bobby Walker
Tommy Lee Jones	The Company Men	Gene McClary

Person

name	birth	city
Ben Affleck	1972	Berkeley
Alan Arkin	1934	New York
Tommy Lee Jones	1946	San Saba

AMBIGUOUS COLUMN NAMES

Same name may be used for two (or more) columns (in different relations)

- Must **qualify** the column name with the relation name to prevent ambiguity

Customer			
custid	name	address	phone

Sale		
saleid	date	custid

LineItem			
saleid	product	quantity	price

```
SELECT name, date, product, quantity
FROM Customer, Sale, LineItem
WHERE price > 100 AND Customer.custid = Sale.custid AND
        Sale.saleid = LineItem.saleid;
```

Note

- If SELECT clause includes custid, it must specify whether to use Customer.custid or Sale.custid *even though the values are guaranteed to be identical*.

2-RELATION SELECT-FROM-WHERE

```
SELECT award, actor, persona, Role.movie
FROM Honours, Role
WHERE category = 'actor' AND winner = actor
AND Honours.movie = Role.movie
```

Honours

movie	award	category	winner	
Lincoln	Critic's Choice	actor	Daniel Day-Lewis	
Argo	Critic's Choice	director	Ben Affleck	
Lincoln	SAG	supporting actor	Tommy Lee Jones	
Lincoln	Critic's Choice	screenplay	Tony Kushner	
War Horse	BMI Film music		John Williams	

Role

actor	movie	persona
Ben Affleck	Argo	Tony Mendez
Tommy Lee Jones	Lincoln	Thaddeus Stevens
Daniel Day-Lewis	The Boxer	Danny Flynn
Daniel Day-Lewis	Lincoln	Abraham Lincoln

Honours.movie	award	category	winner		actor	Role.movie	persona
Lincoln	Critic's Choice	actor	Daniel Day-Lewis		Ben Affleck	Argo	Tony Mendez
Lincoln	Critic's Choice	actor	Daniel Day-Lewis		Tommy Lee Jones	Lincoln	Thaddeus Stevens
Lincoln	Critic's Choice	actor	Daniel Day-Lewis		Daniel Day-Lewis	The Boxer	Danny Flynn
Lincoln	Critic's Choice	actor	Daniel Day-Lewis		Daniel Day-Lewis	Lincoln	Abraham Lincoln
Argo	Critic's Choice	director	Ben Affleck		Ben Affleck	Argo	Tony Mendez
Argo	Critic's Choice	director	Ben Affleck		Tommy Lee Jones	Lincoln	Thaddeus Stevens
Argo	Critic's Choice	director	Ben Affleck		Daniel Day-Lewis	The Boxer	Danny Flynn
Argo	Critic's Choice	director	Ben Affleck		Daniel Day-Lewis	Lincoln	Abraham Lincoln
Lincoln	SAG	supporting actor	Tommy Lee Jones		Ben Affleck	Argo	Tony Mendez
Lincoln	SAG	supporting actor	Tommy Lee Jones		Tommy Lee Jones	Lincoln	Thaddeus Stevens
Lincoln	SAG	supporting actor	Tommy Lee Jones		Daniel Day-Lewis	The Boxer	Danny Flynn
...							

RECALL SAMPLE TABLES

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 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;
```

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.

(c)

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

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';
```

TABLES AS SETS IN SQL

Duplicate tuples may appear in query results

- From duplicates in base tables
- From projecting out distinguishing columns

Keyword **DISTINCT** in the SELECT clause eliminates duplicates

Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

```
Q11:   SELECT   ALL Salary
       FROM     EMPLOYEE;

Q11A:  SELECT   DISTINCT Salary
       FROM     EMPLOYEE;
```

SET OPERATIONS

Result treated as a set (no duplicates)

- **UNION**, **EXCEPT** (difference),**INTERSECT**

Corresponding multiset (bag) operations:

- ,ȚNIONALLEXCEPTALLINTERSECTALL

Arguments must be *union-compatible*

- Same number of columns
- Corresponding columns of same type

Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

```
Q4A:  (SELECT   DISTINCT Pnumber
       FROM     PROJECT, DEPARTMENT, EMPLOYEE
       WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn
                AND Lname='Smith' )
       UNION
      ( SELECT  DISTINCT Pnumber
       FROM     PROJECT, WORKS_ON, EMPLOYEE
       WHERE    Pnumber=Pno AND Essn=Ssn
                AND Lname='Smith' );
```

OTHER OPERATORS

Standard arithmetic operators:

•Addition (+), subtraction (–), multiplication (*), and division (/) **[NOT] LIKE** comparison operator

•Used for string **pattern matching**

•Percent sign (%) matches zero or more characters

•Underscore (_) matches a single character

e.g., to also match Tommy Lee Jones as supporting actor:

```
SELECT award, actor, persona, Role.movie
FROM Honours, Role
WHERE category LIKE '%actor' AND winner = actor AND
Honours.movie = Role.movie;
```

[NOT] BETWEEN comparison operator

```
WHERE year BETWEEN 1990 AND 2010
```

equivalent to `WHERE year >= 1990 AND YEAR <= 2010`

LECTURE SUMMARY

Introduction to SQL

- Comprehensive language
- Data definition including constraint specification
- Basic SELECT-FROM-WHERE
- Set operators