CS 564: Database Management Systems

Lecture 3: SQL Basics II

Xiangyao Yu

1/29/2024

Announcements

Form groups for later assignments

- Due on **Feb 5 (Monday), 11:59pm**

Assignment #1. SQL

- Due on **Feb 7 (Wednesday), 11:59pm**

Module A1: SQL

SQL: Basics I

SQL: Basics II

- Multi-table operations

Advanced SQL I

Advanced SQL II

Outline of This Lecture

Integrity constraint

Multiple Relations

- Foreign keys
- Join

Outline of This Lecture

Integrity constraint

Multiple Relations

- Foreign keys
- Join

Integrity Constraint (IC)

Integrity Constraint (IC): a condition specified on a database schema and restricts the data that can be stored in an instance of the database.

Example from Lecture 2:

```
CREATE TABLE Product (
    pid INTEGER PRIMARY KEY,
    name CHAR(30) UNIQUE,
    category CHAR(20),
    price REAL,
    manufacturer CHAR(20)
);
```

Integrity Constraint (IC)

Integrity constraint specification

- Integrity constraints are specified when the user defines a database schema

Integrity constraint enforcement

- When a database application runs, the DBMS checks for violations and disallows changes to the data that violate the specified ICs (or make compensating changes to the data to ensure all ICs are satisfied)

Outline of this Lecture

Integrity constraint

Foreign Keys

Multi-table queries

- Join

Multiple Relations – Example

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

Primary Key: sid (Student ID)

Create Relation “Student”

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

```
CREATE TABLE Students (
        sid         CAHR(20),
        name        CHAR(30),
        age         INTEGER,
        gpa         REAL,
        UNIQUE (name, age),
        CONSTRAINT StudentsKey PRIMARY KEY (sid)
);
```

Cannot have two rows with identical name and age

Create Relation “Student”

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

```
CREATE TABLE Students (
        sid         CAHR(20),
        name        CHAR(30),
        age         INTEGER,
        gpa         REAL,
        UNIQUE (name, age),
        CONSTRAINT StudentsKey PRIMARY KEY (sid)
);
```

Equivalent

```
CREATE TABLE Students (
        sid         CAHR(20) PRIMARY KEY,
        name        CHAR(30),
        age         INTEGER,
        gpa         REAL,
        UNIQUE (name, age)
);
```

Multiple Relations – Example

Enrolled

cid	grade	studid
EE102	B	53666
CS101	C	53831
CS102	B	53832
EE101	A	53650

Primary Key: (cid, studid)

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

Primary Key: sid

Multiple Relations – Example

Enrolled

cid	grade	studid
EE102	B	53666
CS101	C	53831
CS102	B	53832
EE101	A	53650

Primary Key: (cid, studid)
Foreign Key: studid

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

Primary Key: sid

Create Relation “Enrolled”

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

```
CREATE TABLE Enrolled (
        studid    CAHR(20),
        cid       CHAR(20),
        grade     CHAR(10),
        PRIMARY KEY (studid, cid),
);
```

Primary key contains two attributes

Create Relation “Enrolled”

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

FOREIGN KEY *(<list of attributes>)*
REFERENCES *<relation> (<attributes>)*

Referenced attributes must be declared **PRIMARY KEY** (default) or **UNIQUE**

```
CREATE TABLE Enrolled (
        studid    CAHR(20),
        cid       CHAR(20),
        grade     CHAR(10),
        PRIMARY KEY (studid, cid),
        FOREIGN KEY (studid) REFERENCES Students (sid)
);
```

Primary key contains two attributes

Foreign key that refers to Students

Foreign Key Constraint

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

The foreign key in **Enrolled** must match the primary key in **Students**

- Must have the same number of columns and compatible data type
- The column names can be different

A foreign key (i.e., *studid*) must **reference a valid, existing key** in the referenced table (i.e., Students.*sid*)

Foreign Key Constraint Violation

Foreign key constraint can be violated in the following two cases:

- A new row is inserted to relation **Enrolled** and references value that does not exist in **Students**
- A deletion or update to **Students** causes some tuples of **Enrolled** to dangle

FK Constraint Violation – Insert to Enrolled

A new row is inserted to relation **Enrolled** and references value that does not exist in **Students**

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A
Inserted row: **60000**	**EE101**	**A**

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

The **INSERT** command is simply rejected

FK Constraint Violation – Delete/Update Students

A deletion or update to **Students** causes some tuples of **Enrolled** to dangle

Now the FK does not exist in Students

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
~~53832~~	~~Guldu~~	~~12~~	~~2.0~~

Row deletion
(update can be considered as deletion + insertion)

FK Constraint Violation – Delete/Update Students

A deletion or update to **Students** causes some tuples of **Enrolled** to dangle

Now the FK does not exist in Students

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
~~53832~~	~~Guldu~~	~~12~~	~~2.0~~

Row deletion
(update can be considered as deletion + insertion)

Possible solutions

- Disallow the deletion of the **Students** row
- Delete rows in **Enrolled** that refer to the deleted **Students** row
- Set corresponding *studid* to some existing default *sid*
- Set the corresponding *studid* column to *null*

Enforce Foreign Key Constraints

Solution 1: Disallow the deletion of the **Students** row (default)

- ON DELETE/UPDATE **NO ACTION**

```
CREATE TABLE Enrolled (
    studid    CAHR(20),
    cid       CHAR(20),
    grade     CHAR(10),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students
              ON DELETE CASCADE
              ON UPDATE NO ACTION
);
```

Enforce Foreign Key Constraints

Solution 1: Disallow the deletion of the **Students** row (default)

- ON DELETE/UPDATE **NO ACTION**

Solution 2: Delete/update rows in **Enrolled** that refer to the deleted **Students** row

- ON DELETE/UPDATE **CASCADE**

```
CREATE TABLE Enrolled (
    studid    CAHR(20),
    cid       CHAR(20),
    grade     CHAR(10),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students
              ON DELETE CASCADE
              ON UPDATE NO ACTION
);
```

Enforce Foreign Key Constraints

Solution 1: Disallow the deletion of the **Students** row (default)

- ON DELETE/UPDATE **NO ACTION**

Solution 2: Delete/update rows in **Enrolled** that refer to the deleted **Students** row

- ON DELETE/UPDATE **CASCADE**

Solution 3: Set corresponding *studid* to some existing default *sid*

- ON DELETE/UPDATE **SET DEFAULT**

```
CREATE TABLE Enrolled (
    studid    CAHR(20),
    cid       CHAR(20),
    grade     CHAR(10),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students
              ON DELETE CASCADE
              ON UPDATE NO ACTION
);
```

Requires default value in Students

E.g., sid CHAR(20) **DEFAULT** ‘53666’

Enforce Foreign Key Constraints

Solution 1: Disallow the deletion of the **Students** row (default)

- ON DELETE/UPDATE **NO ACTION**

Solution 2: Delete/update rows in **Enrolled** that refer to the deleted **Students** row

- ON DELETE/UPDATE **CASCADE**

Solution 3: Set corresponding *studid* to some existing default *sid*

- ON DELETE/UPDATE **SET DEFAULT**

Solution 4: Set the corresponding *studid* values to *null*

- ON DELETE/UPDATE **SET NULL**

```
CREATE TABLE Enrolled (
    studid    CAHR(20),
    cid       CHAR(20),
    grade     CHAR(10),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students
              ON DELETE CASCADE
              ON UPDATE NO ACTION
);
```

Requires default value in Students

E.g., sid CHAR(20) **DEFAULT** ‘53666’

Enforce Foreign Key Constraints

Solution 1: Disallow the deletion of the **Students** row (default)

- ON DELETE/UPDATE **NO ACTION**

Solution 2: Delete/update rows in **Enrolled** that refer to the deleted **Students** row

- ON DELETE/UPDATE **CASCADE**

Solution 3: Set corresponding *studid* to some existing default *sid*

- ON DELETE/UPDATE **SET DEFAULT**

Solution 4: Set the corresponding *studid* values to *null*

- ON DELETE/UPDATE **SET NULL**

```
CREATE TABLE Enrolled (
    studid    CAHR(20),
    cid       CHAR(20),
    grade     CHAR(10),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students
              ON DELETE CASCADE
              ON UPDATE NO ACTION
);
```

Requires default value in Students

E.g., sid CHAR(20) **DEFAULT** ‘53666’

Can specify the actions on delete and update separately

Example

```
CREATE TABLE Enrolled (
        studid      CAHR(20),
        cid         CHAR(20),
        grade       CHAR(10),
        PRIMARY KEY (studid, cid),
        FOREIGN KEY (studid) REFERENCES Students
                ON DELETE CASCADE
                ON UPDATE NO ACTION
);
```

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What happens if the following update is performed?

```
UPDATE   Students
SET      sid = 50001
WHERE    sid = 53666
```

Example

```
CREATE TABLE Enrolled (
        studid      CAHR(20),
        cid         CHAR(20),
        grade       CHAR(10),
        PRIMARY KEY (studid, cid),
        FOREIGN KEY (studid) REFERENCES Students
                ON DELETE CASCADE
                ON UPDATE NO ACTION
);
```

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What happens if the following update is performed?

```
UPDATE   Students
SET      sid = 50001
WHERE    sid = 53666
```

The update will be rejected due to "**ON UPDATE NO ACTION**"

Example

```
CREATE TABLE Enrolled (
        studid      CAHR(20),
        cid         CHAR(20),
        grade       CHAR(10),
        PRIMARY KEY (studid, cid),
        FOREIGN KEY (studid) REFERENCES Students
                ON DELETE CASCADE
                ON UPDATE NO ACTION
);
```

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What happens if the following deletion is performed?

```
DELETE
FROM    Students
WHERE   sid = 53832
```

Example

```
CREATE TABLE Enrolled (
        studid      CAHR(20),
        cid         CHAR(20),
        grade       CHAR(10),
        PRIMARY KEY (studid, cid),
        FOREIGN KEY (studid) REFERENCES Students
                ON DELETE CASCADE
                ON UPDATE NO ACTION
);
```

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
~~53832~~	~~CS102~~	~~B~~
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
~~53832~~	~~Guldu~~	~~12~~	~~2.0~~

What happens if the following deletion is performed?

```
DELETE
FROM    Students
WHERE   sid = 53832
```

The corresponding row in **Enrolled** is also deleted due to "**ON DELETE CASCADE**"

Outline of this Lecture

Integrity constraint

Foreign Keys

Multi-table queries

- Join

Multi-Table Queries

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

- The query cannot be answered by a single relation
- Must combine data from both relations

Semantics of SQL: Single Table

SELECT $a_1, a_2, \ldots, a_k$
FROM R_1
WHERE conditions

answer := {}
for t_1 **in** R_1 **do**
  if conditions
    then *answer := answer* $\cup$ $\{(a_1,\ldots,a_k)\}$
return *answer*

Semantics of SQL: Multiple Tables

SELECT $a_1, a_2, \ldots, a_k$
FROM $R_1, R_2, \ldots, R_n$
WHERE conditions

answer := {}
for t_1 **in** R_1 **do**
    for t_2 in R_2 do
        …..
            for t_n in R_n do
                if conditions
                then *answer* := *answer* ∪ $\{(a_1,\ldots,a_k)\}$
return *answer*

Cross product

Multi-Table Queries

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

```
SELECT    S.name
FROM      Students S, Enrolled E
WHERE     S.sid = E.studid
    AND E.grade = 'A'
    AND E.cid = 'EE101'
```

Join

Multi-Table Queries

Enrolled

studid	cid	grade
53666	EE102	B
53831	CS101	C
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

```
SELECT    S.name
FROM      Students S, Enrolled E
WHERE     S.sid = E.studid
    AND E.grade = 'A'
    AND E.cid = 'EE101'
```

Join

Good Style

```
SELECT    name
FROM      Students, Enrolled
WHERE     sid = studid
    AND grade = 'A'
    AND cid = 'EE101'
```

Bad Style

Semantics: Select-From-Where

1. Start with the cross product of all the relations in the **`FROM`** clause
2. Apply the conditions from the **`WHERE`** clause
3. Project onto the list of attributes and expressions in the **`SELECT`** clause
4. If **`DISTINCT`** is specified, eliminate duplicate rows

Multi-Table Queries – Simplified Example

```
SELECT    S.name
FROM      Students S, Enrolled E
WHERE     S.sid = E.studid
  AND E.grade = 'A'
  AND E.cid = 'EE101'
```

Enrolled

studid	cid	grade
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

Step 1: Cross Product (Enrolled x Students)

E.studid	E.cid	E.grade
53832	CS102	B
53832	CS102	B
53832	CS102	B
53650	EE101	A
53650	EE101	A
53650	EE101	A

S.sid	S.name	S.age	S.gpa
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0

Multi-Table Queries – Example

```
SELECT    S.name
FROM      Students S, Enrolled E
WHERE     S.sid = E.studid
  AND E.grade = 'A'
  AND E.cid = 'EE101'
```

Enrolled

studid	cid	grade
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

Step 1: Cross Product (Enrolled x Students)

E.studid	E.cid	E.grade	S.sid	S.name	S.age	S.gpa
53832	CS102	B	50000	Dave	19	3.3
53832	CS102	B	53650	Smith	19	3.8
53832	CS102	B	53832	Guldu	12	2.0
53650	EE101	A	50000	Dave	19	3.3
53650	EE101	A	53650	Smith	19	3.8
53650	EE101	A	53832	Guldu	12	2.0

Step 2: Filtering (S.sid = E.studid AND E.grade = 'A')

E.studid	E.cid	E.grade	S.sid	S.name	S.age	S.gpa
53650	EE101	A	53650	Smith	19	3.8

Rows satisfying both conditions in the WHERE clause

Multi-Table Queries – Example

```
SELECT    S.name
FROM      Students S, Enrolled E
WHERE     S.sid = E.studid
  AND E.grade = 'A'
  AND E.cid = 'EE101'
```

Enrolled

studid	cid	grade
53832	CS102	B
53650	EE101	A

Students

sid	name	age	gpa
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0

What are the names of students who got A in EE101?

Step 1: Cross Product (Enrolled x Students)

E.studid	E.cid	E.grade
53832	CS102	B
53832	CS102	B
53832	CS102	B
53650	EE101	A
53650	EE101	A
53650	EE101	A

S.sid	S.name	S.age	S.gpa
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0
50000	Dave	19	3.3
53650	Smith	19	3.8
53832	Guldu	12	2.0

Step 2: Filtering (S.sid = E.studid AND E.grade = 'A')

E.studid	E.cid	E.grade
53650	EE101	A

S.sid	S.name	S.age	S.gpa
53650	Smith	19	3.8

Step 3: Projection (SELECT S.name)

S.name
Smith

Semantics of SQL

The query processor will **almost never** evaluate the query this naïve way

SQL is a **declarative** language

The DBMS figures out the most efficient way to compute it

We will discuss this later in the course when we talk about *query optimization*

Multi-Table Query – Exercise

Enrolled

studid	cid	grade
53666	EE102	90
53831	CS101	85
53832	CS102	90
53650	EE101	95

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What are the IDs of students who got higher grade in EE101 than EE102?

Multi-Table Query – Exercise

Enrolled

studid	cid	grade
53666	EE102	90
53831	CS101	85
53832	CS102	90
53650	EE101	95

Students

sid	name	age	gpa
50000	Dave	19	3.3
53666	Jones	18	3.4
53688	Smith	18	3.2
53650	Smith	19	3.8
53831	Madayan	11	1.8
53832	Guldu	12	2.0

What are the IDs of students who got higher grade in EE101 than EE102?

```
SELECT    E1.studid
FROM      Enrolled E1, Enrolled E2
WHERE     E1.studid = E2.studid
  AND     E1.cid = 'EE101'
  AND     E2.cid = 'EE102'
  AND     E1.grade > E2.grade;
```

Join

Jupyter Notebook

Summary

Integrity constraint (IC)

- Specification and enforcement of ICs

Foreign keys

- Declaring foreign keys, foreign key constraint, ways to enforce foreign key constraints

Multi-table queries

- Semantic of multi-table queries