**Data Creation**

CREATE TABLE Students (sid CHAR(20),
                      name CHAR(30),
                      login CHAR(20),
                      age INTEGER,
                      gpa REAL);

INSERT INTO Students (sid, name, login, age, gpa)
VALUES(53688, 'Smith', 'smith@ee', 18, 3.2);

DELETE FROM Students WHERE Students.name = 'Smith';

UPDATE Students SET Students.age = Students.age + 1,
                     Students.gpa = Students.gpa - 1
WHERE Students.sid = 53688;

**Table Update**

CREATE TABLE Students (sid CHAR(20),
                      name CHAR(30),
                      login CHAR(20),
                      age INTEGER,
                      gpa REAL);

INSERT INTO Students (sid, name, login, age, gpa)
VALUES(53688, 'Smith', 'smith@ee', 18, 3.2);

DELETE FROM Students WHERE Students.name = 'Smith';

UPDATE Students SET Students.age = Students.age + 1,
                     Students.gpa = Students.gpa - 1
WHERE Students.sid = 53688;
Table Update

```sql
UPDATE Students S
SET S.gpa = S.gpa - 1
WHERE S.gpa >= 3.3
```

Integrity Constraints

- An IC
  - is a condition specified on a database schema
  - restricts the data that can be stored in an instance
  - is enforced by DBMS

Integrity Constraints

- Why do we need them?
Integrity Constraints

- Every table has a key (guaranteed!)
- Why?

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

Integrity Constraints

- Candidate key

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

Integrity Constraints

- Primary key

```
CREATE TABLE Students (sid CHAR(20),
    name CHAR(30),
    login CHAR(20),
    age INTEGER,
    gpa REAL
    UNIQUE (name, age),
    CONSTRAINTS StudentKey PRIMARY KEY (sid)
);
```
Integrity Constraints

- Foreign key

```
Student(sid: string, name: string, login: string, age: integer, gpa: real)
Enrolled(studentid: string, cid: string, grade: string)
```

- How do we ensure only certain students can enroll?

---

Integrity Constraints

- Foreign key on a table can refer to itself.
- Example?
  - Employee and Manager

---

Integrity Constraints

- Not NULL constraint
- When do we need this?
Integrity Constraints

CREATE TABLE Enrolled (studentid CHAR(20),
cid CHAR(20),
grade CHAR(10) NOT NULL,
PRIMARY KEY (studentid, cid),
FOREIGN KEY (studentid) REFERENCES Students );

General Constraints

• Business rules
  – Only students with GPA > 2.5 are allowed to enroll
• Other examples
• Why are they not IC’s?

Enforcing Integrity Constraints

INSERT INTO Students (sid, name, login, age, gpa)
VALUES(null, 'Smith', 'smith@ee', 18, 3.2)

What other constraints can be violated by insertion?
Enforcing Integrity Constraints

INSERT INTO Students (sid, name, login, age, gpa)
VALUES(null, 'Smith', 'smith@ee', 18, 3.2)

What other constraints can be violated by insertion?
- domain
- unique
- primary key
- foreign key

Enforcing Integrity Constraints

DELETE FROM Students S
WHERE S.name='Smith'

What other constraints can be violated by deletion?
- foreign key
Enforcing Integrity Constraints

UPDATE Students S
SET S.gpa = S.gpa - 0.1
WHERE S.gpa >= 3.3

What other constraints can be violated by update?

Enforcing Foreign Key Constraints

• Insertion
  – What happens if the studentid is invalid when inserting a new enrollment record?

• Deletion
  – What happens when deleting a student record?
Enforcing Foreign Key Constraints

• Deletion
  – What happens when deleting a student record?
    • Do not delete.

```sql
CREATE TABLE Enrolled (studentid CHAR(20),
cid CHAR(20),
grade CHAR(10),
PRIMARY KEY (studentid, cid),
FOREIGN KEY (studentid) REFERENCES students
   ON DELETE NO ACTION
);
```

Enforcing Foreign Key Constraints

• Deletion
  – What happens when deleting a student record?
    • Remove the record in the Student table
    • Remove all records in the Enrollment table with the given StudentID.

```sql
CREATE TABLE Enrolled (studentid CHAR(20),
cid CHAR(20),
grade CHAR(10),
PRIMARY KEY (studentid, cid),
FOREIGN KEY (studentid) REFERENCES students
   ON DELETE CASCADE
);
```

Enforcing Foreign Key Constraints

• Deletion
  – What happens when deleting a student record?
    • Remove the record in the Student table
    • Set the StudentID to NULL for all records in Enrollment with the given StudentID.

```sql
CREATE TABLE Enrolled (studentid CHAR(20),
cid CHAR(20),
grade CHAR(10),
PRIMARY KEY (studentid, cid),
FOREIGN KEY (studentid) REFERENCES students
   ON DELETE SET NULL
);
```
Enforcing Foreign Key Constraints

• Deletion
  — What happens when deleting a student record?
    • Remove the record in the Student table
    • Set the StudentID to the default value for all records in Enrollment with the given StudentID.

```
CREATE TABLE Enrolled (studentid CHAR(20) DEFAULT '53666',
    cid CHAR(20),
    grade CHAR(10),
    PRIMARY KEY (studentid, cid),
    FOREIGN KEY (studentid) REFERENCES students
    ON DELETE SET DEFAULT
);
```

Enforcing Foreign Key Constraints

• Update
  — What happens when updating the ID of a student?

• Update
  — What happens when updating the ID of a student?
    • ON UPDATE NO ACTION
    • ON UPDATE CASCADE
    • ON UPDATE SET DEFAULT
    • ON UPDATE SET NULL
Enforcing Foreign Key Constraints

- Update
  - What happens when updating the ID of a student?

    CREATE TABLE Enrolled (studentid CHAR(20),
      cid CHAR(20),
      grade CHAR(10),
      PRIMARY KEY (studentid, cid),
      FOREIGN KEY (studentid) REFERENCES students
      ON DELETE CASCADE
      ON UPDATE NO ACTION
    );

Data Retrieval

- SELECT *
  FROM Students S
  WHERE S.age < 18

Data Retrieval

- SELECT S.name, S.login
  FROM Students S
  WHERE S.age < 18
SQL (Structured Query Language)

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] tbl_name [(col_name,...)]
VALUES [(expression | DEFAULT),...,]
[ ON DUPLICATE KEY UPDATE col_name=expression, ...]
```

or
```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] tbl_name [(col_name,...)]
SELECT ...
```

or
```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] tbl_name
SET col_name=expression | DEFAULT, ...
[ ON DUPLICATE KEY UPDATE col_name=expression, ...]
```

SQL (Structured Query Language)

```
DELETE [LOW_PRIORITY] [QUICK] FROM table_name
[WHERE where_definition]
[ORDER BY ...]
[LIMIT rows]
```

or
```
DELETE [LOW_PRIORITY] [QUICK] table_name[-?] [, table_name[-?] ...]
FROM table_references
[WHERE where_definition]
```

or
```
DELETE [LOW_PRIORITY] [QUICK]
FROM table_name[-?] [, table_name[-?] ...]
USING table_references
[WHERE where_definition]
```

SQL (Structured Query Language)

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name=expr [, col_name2=expr2 ...]
[WHERE where_definition]
[ORDER BY ...]
[LIMIT rows]
```

or
```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, col_name ...
SET col_name=expr [, col_name2=expr2 ...]
[WHERE where_definition]
```
SQL (Structured Query Language)

```
SELECT [DISTINCT | ALL]
    column, column [AS alias] [IN (SELECT)] [DISJOINT | INTERSECT | UNION | UNION ALL | UNION DISTINCT]
FROM table
WHERE [condition] [AND | OR] [condition]
GROUP BY column [AS alias] [IN (SELECT)] [DISJOINT | INTERSECT | UNION | UNION ALL | UNION DISTINCT]
HAVING [condition] [AND | OR] [condition]
ORDER BY column [AS alias] [IN (SELECT)] [DISJOINT | INTERSECT | UNION | UNION ALL | UNION DISTINCT]
```

Quiz #3

- Course Project
  - Decide if you are working alone or a team of 2
  - Provide your idea for a database project
  - Begin listing some database requirements