kdeg
knowledge and data engineering group

Information Management and Data Engineering

CS4D2a – 4CSLL1 – CS3041

Database Constraints

Séamus Lawless

seamus.lawless@scss.tcd.ie

Integrity v Security

- Integrity and Security are related but they are not the same thing
  - Integrity is concerned with *accidental* corruption
  - Security is concerned with *deliberate* corruption
- Integrity
  - Integrity Constraints
- Security
  - Security Policies
  - Access Control

Relational Model Constraints

- Constraints expressed in the Relational Schema
  - *Explicit Constraints*
- Constraints that cannot be expressed in the Relational Schema
  - *Semantic Constraints*
  - Can be expressed in SQL in some cases
  - Usually enforced by the application programs

Integrity Constraints

- Three types of integrity constraint are considered part of the relational model:
  - Key
  - Entity Integrity
  - Referential Integrity
- The DBMS must be able to enforce these constraints

Key Constraint

- Specifies that there may not be any duplicate entries in key attributes
  - Primary Key
  - Candidate Keys
- Keys are used to uniquely identify a tuple
  - Having a duplicate value in a Key implies that we cannot uniquely identify some tuples

Entity Integrity Constraint

- Specifies that there may not be any NULL values in the Primary Key attribute
  - The Primary Key is used to uniquely identify each tuple in a relation
  - Having a NULL in a Primary Key implies that we cannot identify some tuples

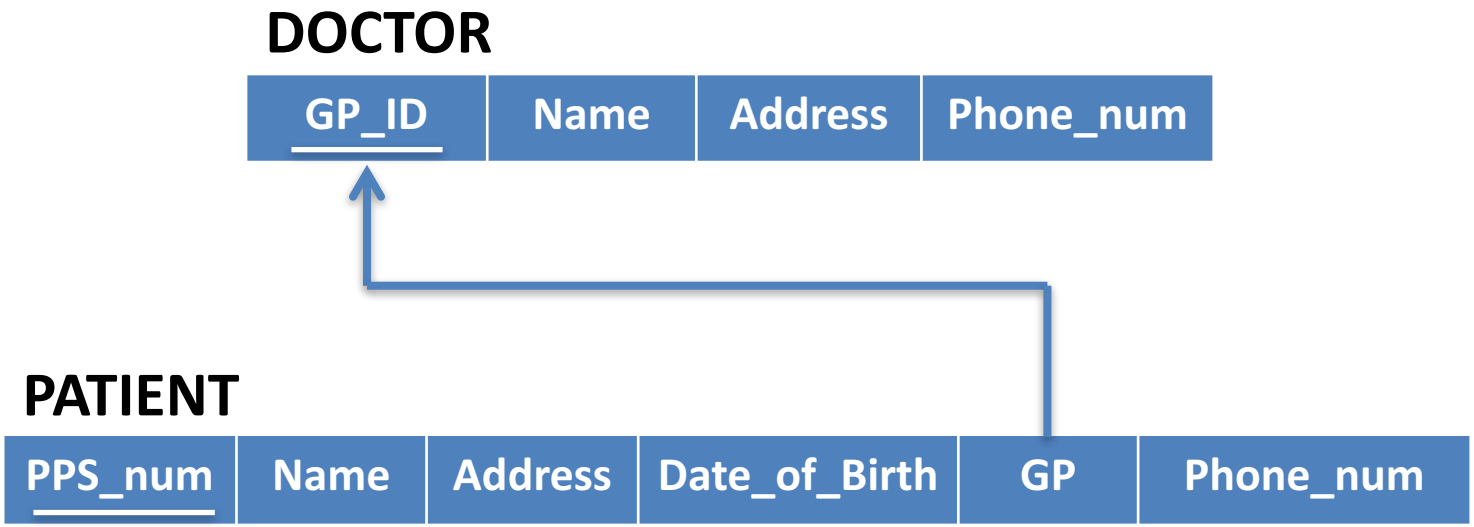
Referential Integrity

- Entity constraints are specified on individual relations
- Referential Integrity constraints are specified between two relations
  - Maintains consistency among tuples in the two relations
  - A tuple in one relation that refers to another relation, must refer to an *existing tuple* in that relation
- A Foreign Key formally specifies a Referential Integrity Constraint between two relations

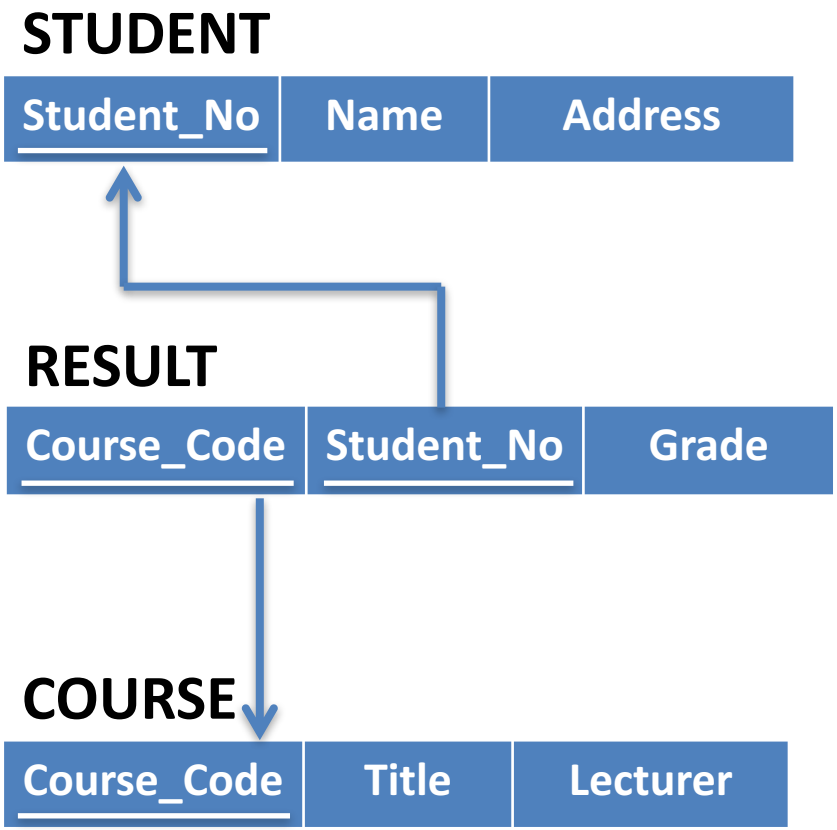
NULL Keys

- As per the Entity Integrity constraint
  - No part of a Primary Key can be NULL
- However, Foreign Keys in certain circumstances may be NULL
  - A decision must be made during schema design as to whether it is valid for the foreign key to be NULL at any point

NULL Keys


DOCTOR
GP_ID
Name
Address
Phone_num
PATIENT
PPS_num
Name
Address
Date_of_Birth
GP
Phone_num

NULL Keys


STUDENT
Student_No
Name
Address
RESULT
Course_Code
Student_No
Grade
COURSE
Course_Code
Title
Lecturer

Referential Integrity

- When defining an attribute as a Foreign Key
  - You must also specify whether or not the foreign key is allowed to contain NULLS
- In the case of a composite Foreign Key
  - if the Foreign Key is allowed to contain NULLS
  - then either all the component attributes should be NULL or none of them NULL
    - in order to enforce referential integrity

Constraint Violation

- There are three basic operations that modify the state of relations in a DB
  - Insert
  - Update
  - Delete
- These operations should not violate the integrity constraints specified for the DB
  - Key, Entity, Referential

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

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

Constraint Violation - Insert

- Insert provides a list of attribute values for a new tuple t that is to be added to relation R
- Inserts can violate all the integrity constraints that we have discussed
  - Key
  - Entity Integrity
  - Referential Integrity

Example

- Can we insert the following into EMPLOYEE?
  - <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4>
- Answer:

Example

- Can we insert the following into EMPLOYEE?
  - <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4>
- Answer:

Example

- Can we insert the following into EMPLOYEE?
  - <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F, 28000, ‘987654321’, 7>
- Answer:

Example

- Can we insert the following into EMPLOYEE?
  - <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4>
- Answer:

Constraint Violation - Delete

- To specify a deletion, a condition on the attributes of a relation is created which selects one or more tuples to be deleted
- The Delete operation can only violate the Referential Integrity constraint

Example

- Can we delete the following?
  - Any tuples in WORKS_ON with Essn = ‘999887777’ and Pno = 10.
- Answer:

Example

- Can we delete the following?
  - Any tuples in EMPLOYEE with Ssn = '999887777'
- Answer:

Example

- Can we delete the following?
  - Any tuples in EMPLOYEE with Ssn = '333445555'
- Answer:

Cascading Deletes

- An option to address Delete operations which violate Referential Integrity is to *cascade*, or propagate, the deletion
- For instance, in example 2:
  - Delete any tuples in EMPLOYEE with Ssn = '999887777'
- The DBMS could automatically delete the offending tuples from WORKS_ON
  - in addition to the original tuple in EMPLOYEE
  - This must be implemented carefully, as it can lead to unintentional loss of data

Constraint Violation - Update

- An Update operation is used to change the values of one or more attributes of a relation
- To specify an update, a condition on the attributes of a relation is created which selects one or more tuples to be modified
- Updates can violate all the integrity constraints that we have discussed
  - Key
  - Entity Integrity
  - Referential Integrity

Example

- Can we perform the following?
  - Update the salary of any EMPLOYEE tuples with Ssn = ‘999887777’ to 28000
- Answer:

Example

- Can we perform the following?
  - Update the Dno of any EMPLOYEE tuples with Ssn = ‘999887777’ to 1
- Answer:

Example

- Can we perform the following?
  - Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7
- Answer:

Example

- Can we perform the following?
  - Update the Ssn of any EMPLOYEE tuples with Ssn = ‘999887777’ to ‘987654321’
- Answer:

Cascading Updates

- As with Delete, an option to address Update operations which violate Referential Integrity is to *cascade*, or propagate, the update
- For instance:
  - Update the Ssn of any EMPLOYEE tuples with Ssn = '333445555' to '123123123'
- The DBMS could automatically update the relations which have a Foreign Key to Ssn
  - WORKS_ON, DEPARTMENT, DEPENDENT and EMPLOYEE itself

Alternatives to Cascading

- The alternatives to the cascading of updates or deletes are:
  - Rejection of the update or delete as long as foreign key references exist
  - Update of the corresponding foreign key to NULL
  - Update of the corresponding foreign key to some default value

Constraints in SQL

- Constraints specified as part of relation, or table, definition are called “table constraints”
  - they are specified on each table individually
- They are typically specified during table creation in the CREATE TABLE statement
  - can be added later using ALTER TABLE
- Constraints that affect more than one table are called Assertions

PRIMARY KEY

- The PRIMARY KEY constraint specifies the attribute(s) that forms the Primary Key
  - For a single attribute, the constraint can directly follow the attribute specification
    - Dnumber INT PRIMARY KEY
  - composite keys can be specified at the end of the CREATE TABLE statement
    - PRIMARY KEY (Dnumber, Dlocation)

UNIQUE

- As we have seen, there is often more than one candidate key in a relation
- Secondary keys can be specified using the UNIQUE constraint
  - For a single attribute, the constraint can directly follow the attribute specification
    - Engine_num INT UNIQUE
  - composite secondary keys can be specified at the end of the CREATE TABLE statement
    - UNIQUE (Licence_Yr, Licence_Mth, Licence_Day)

NOT NULL

- By default SQL allows NULLs as attribute values
  - a NOT NULL constraint may be specified if NULLs are not permitted for a specific attribute
  - this is always the case for any attribute that forms part of the Primary Key

```
CREATE TABLE Person
(PPS char(8) NOT NULL PRIMARY KEY,
Fname varchar(255) NOT NULL,
Lname varchar(255),
Phone int);
```

More Complex Constraints

- Can be specified using:
  - CHECK
  - ASSERTION
  - TRIGGER

CHECK

- More complex constraints can be specified using the CHECK clause
  - used to restrict the values that can be entered for an attribute
- Each CHECK is specified on one or more attributes from a single table
- The CHECK is performed for every tuple that is inserted or modified

CHECK Clause

- CHECK clauses are specified within the CREATE TABLE statement
- They can be specified on an individual attribute
  - Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21)
- or on multiple attributes from the same table
  - CHECK (Dept_create_date <= Mgr_start_date)

Referential Integrity

- Referential Integrity is specified using the FOREIGN KEY clause
  - specified at the end of the CREATE TABLE statement
    - FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
  - can also have composite Foreign Keys
    - FOREIGN KEY(artist, album) REFERENCES ALBUM(artist, name)

Referential Integrity Violation

- As discussed earlier, Referential Integrity can be violated on update, insert or delete
  - Default action in SQL is to reject the operation
- it is possible to specify an alternate action by attaching a clause to each Foreign Key
  - SET NULL
  - CASCADE
  - SET DEFAULT

Referential Integrity Violation

- Each action must be qualified with either
  - ON DELETE
  - ON UPDATE

```
CREATE TABLE EMPLOYEE
(Fname VARCHAR(15) NOT NULL,
Lname VARCHAR(15) NOT NULL,
Ssn CHAR(9) NOT NULL PRIMARY KEY,
Super_ssn CHAR(9),
Dno INT NOT NULL DEFAULT 1,
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn) ON DELETE SET
NULL ON UPDATE CASCADE,
FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) ON DELETE SET
DEFAULT ON UPDATE CASCADE);
```

Naming Constraints

- Constraints can be named using CONSTRAINT
  - Names must be unique within the schema

```
CREATE TABLE movie
( movie_id INT NOT NULL PRIMARY KEY,
  title varchar(255) NOT NULL,
  genre CHAR(10),
  CONSTRAINT check_movie_type CHECK
  (genre IN ('Horror', 'Action', 'Other')));
```

Assertions

- An Assertion is a stand-alone constraint in a schema
  - used to specify a restriction that affects more than one table
- Table constraints (using CHECK) can be used to specify multiple table constraints, however, it is better practice to use Assertions:
  - Table constraints are only evaluated if, and only if, the table to which it is attached has some data

Assertion Syntax

- The general form of the ASSERTION command is:

  CREATE ASSERTION <assertion-name>
  CHECK (<search-condition>)

- ASSERTIONs:
  - are associated with the relations in question
  - are evaluated before an operation can be performed on those relations
  - are violated if false the and operation is not allowed
  - define valid states of a DB
  - are actually stored as rows in the ASSERTIONS table which is part of the system catalog

Evaluation of Assertions

- Assertions are checked at the end of each SQL statement
  - a transaction can be more than one SQL statement
  - Assertion evaluation can be deferred until the end of a transaction, but is always evaluated prior to the completion of a transaction
- If an assertion fails, the DBMS returns an error message and the SQL statement is rejected

Assertions

MOVIE

ID	Title	Director	Genre	Cost_Price	Sale_Price	...

MUSIC

ID	Title	Artist	Genre	Cost_Price	Sale_Price	...

```
CREATE ASSERTION maximum_inventory
CHECK((SELECT SUM(Cost_Price) from MOVIE) +
        (SELECT SUM(Cost_Price) from MUSIC)
        < 500000);
```

Triggers

- Triggers are Event-Condition-Action rules
  - allow constraints to be checked on specified events and resulting actions to be invoked
- Triggers are only tested when certain events occur
  - e.g. insert, update etc.
- When triggered, a specified condition is tested
  - If the condition does not hold, then no further action is taken in response to the event
  - If the condition is satisfied, defined actions associated with the trigger are performed by the DBMS

Triggers

- General form:

  CREATE TRIGGER <trigger name>

  (**AFTER** I **BEFORE**) <triggering events> ON <table name>

  [**FOR EACH ROW**]

  [**WHEN** <condition>]

  <trigger actions> ;

- <trigger event>
  - INSERT I DELETE I UPDATE [OF <column name> { , <column name> }]

Triggers

```
CREATE TRIGGER Total_Salary
AFTER DELETE ON EMPLOYEE
FOR EACH ROW
WHEN (OLD.Dno IS NOT NULL)
        UPDATE DEPARTMENT
        SET Total_salary = Total_salary –
OLD.Salary
        WHERE Dno = OLD.Dno;
```

Triggers

```
CREATE TRIGGER Inform_Accounts
BEFORE INSERT OR UPDATE OF Salary, Supervisor_ssn
        ON EMPLOYEE
FOR EACH ROW
WHEN (NEW.Salary > (SELECT Salary FROM EMPLOYEE
                        WHERE Ssn = NEW.Supervisor_ssn))
        salary_violation(NEW.Supervisor_ssn, NEW.Ssn);
```

Assertions v Triggers

- Assertions
  - do not modify the data, only check certain conditions
- Triggers
  - are more powerful because the can check conditions and also modify the data
  - are linked to specific tables and specific events
- All assertions can be implemented as triggers
- Not all triggers can be implemented as assertions
- Oracle does not have assertions