Relational Data Model – Part 2

Integrity Constraints

Integrity Constraints over Relations

- Constraints are **conditions** that must hold on *all* relations in a database state.
- The *main types* of constraints in the relational data model include:
  1. **Domain constraints**;
  2. **Key constraints**;
  3. **Entity integrity constraints**;
  4. **Referential integrity constraints**.

(1) Domain Constraints

- Every value in a tuple must be from the **domain of its attribute**.
  - INT
  - VARCHAR
  - DATE
  - SMALLINT
  - NOT NULL

(2) Key Constraints - Observation

- We observe that: **data does not occur independently from one another within individual relations.**

- No two students have the same student ID:

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com
...	...	...	...

- No two enrolments have the same student ID, the same course number in the same semester:

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016
...	...	...	...	...

(2) Key Constraints - Definitions

- Let $R(A_1, \ldots, A_n)$ be a relation schema.
- A **superkey** SK of R is a subset of attributes of R, i.e., $SK \subseteq \{A_1, \ldots, A_n\}$, such that
  - no two distinct tuples in $r(R)$ can have the same value for SK.
- A superkey SK of R is **minimal** if there is no other superkey $SK' \subset SK$ held on R. A minimal superkey is also known as a **candidate key**.
- A **primary key** PK of R is a minimal superkey of R, (i.e., a primary key is one of the candidate keys). If a relation has only one candidate key then that would be the primary key.

(2) Key Constraints - Example

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com
460	Tyrion	11/09/1987	tyrion@hotmail.com

- Is {DoB} a superkey of STUDENT? **No!**
- Is {StudentID, DoB} a superkey of STUDENT? **Yes!**
- Is {StudentID, DoB} a candidate key of STUDENT? **No!**
- Is {StudentID} a candidate key of STUDENT? **Yes!**
- Can {StudentID} be chosen as a primary key of STUDENT? **Yes!**
- Can {DoB} be chosen as a primary key of STUDENT? **No!**

(2) Key Constraints - Example

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016
458	COMP1130	2015 S1	inactive	20/02/2015

- Is {CourseNo, Semester} a superkey of ENROL? **No!**
- Is {StudentID, CourseNo, Semester} a candidate key of ENROL? **Yes!**
- Can {StudentID, CourseNo} be chosen as a primary key of ENROL? **No!**

(3) Entity Integrity Constraints

- Specifying a primary key also invokes the entity integrity constraint.
- **null** is a special value, which represents the value of an attribute that may be unknown or inapplicable.
- The **entity integrity constraint** states that **no primary key value can be NULL**.
  - This is because primary key values are used to *identify* individual tuples in a relation.
- **Note:** Other attributes of *R* may be constrained to disallow null values, even though they are not attributes in the primary key.

(3) Entity Integrity Constraints – Example

- If StudentID is specified as the primary key of Student, then the following relation violates the entity integrity constraint.

Student			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
NULL	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

- How about the case when Email is the primary key of Student?

Answer: **The relation does not violate the entity integrity constraint.**

(4) Referential Integrity Constraints - Observation

- We observe that: **data does not occur independently from one another across relations.**
- Every course number appearing in ENROL must exist in COURSE:

STUDENT

StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

COURSE

No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6

ENROL

StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016

(4) Referential Integrity Constraints - Definition

- We use $t[A]$ to denote the value of attribute A in tuple t.

  Example: For the tuple t=(459,Fran,11/09/1987,frankk@gmail.com), t[Name]=Fran and t[DoB]=11/09/1987.

- A **referential integrity constraint** specifies a reference between **two** relations, while the previous constraints involve **only one** relation.
- Let R_1 and R_2 be relation schemas in a database schema S, and R_2 has the primary key $\{B_1, \ldots, B_n\}$.
- A **foreign key** on R_1 is a statement $[A_1, \ldots, A_n] \subseteq R_2[B_1, \ldots, B_n]$ restricting states of S to satisfy the following property:
  - for each tuple $t \in r(R_1)$ there exists a tuple $t' \in r(R_2)$ with $t[A_i] = t'[B_i]$ for $i = 1, \ldots, n$.
- R_1 is called the **referencing relation** and R_2 is called the **referenced relation**.

(4) Referential Integrity Constraints – Example

- What foreign keys can be established in the database StuEnrol?

Student			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

Course		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6

Enrol				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016

(4) Referential Integrity Constraints – Example

- In this case, we can establish the following foreign keys on ENROL:
  1. [CourseNo]$\subseteq$ COURSE[No];
  2. [StudentID]$\subseteq$ STUDENT[StudentID].

- This database state satisfies the above two foreign keys because
  - for each tuple t_1 in ENROL, there is a tuple t_2 in COURSE such that the CourseNo value in t_1 is the same with the No value in t_2;
  - for each tuple $t_1^{'}$ in ENROL, there is a tuple $t_2^{'}$ in STUDENT such that the StudentID value in $t_1^{'}$ is the same with the StudentID value in $t_2^{'}$.

(4) Referential Integrity Constraints – Question

- If the database STUENROL is slightly changed as follows, does this database still satisfy the foreign keys in the previous example?

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

COURSE		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2600	2016 S2	active	11/06/2016

(4) Referential Integrity Constraints – Question

Answer: **The following database does not satisfy the foreign key of ENROL: [CourseNo]⊆ COURSE[No].**

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

COURSE		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2600	2016 S2	active	11/06/2016

Constraint Violations

- There are three basic operations that can change a database state:
  - **Insert**: insert one or more new tuples in a relation;
  - **Delete**: delete tuples in a relation;
  - **Update** (or **Modify**): change the values of attributes in existing tuples.
- Whenever these operations are applied, the integrity constraints specified in a database schema **should not be violated**.
- However,
  - Insert may violate ...
  - Delete may violate ...
  - Update may violate ...