Relational Model

CSC 343
Winter 2021

MICHAEL LIUT (MICHAEL.LIUT@UTORONTO.CA)
ILIR DEMA (ILIR.DEMA@UTORONTO.CA)

DEPARTMENT OF MATHEMATICAL AND COMPUTATIONAL SCIENCES
UNIVERSITY OF TORONTO MISSISSAUGA

UNIVERSITY OF TORONTO MISSISSAUGA

UNIVERSITY OF TORONTO MISSISSAUGA

Describing Data: Data Models

A **data model** is a collection of concepts for describing data.

A **schema** is a description of a particular collection of data, using a given data model.

The **relational model of data** is the most widely used today.

- Main concept: ***relation***, basically a table with rows and columns.
- Use tables to represent data and relationships.
- Every relation has a ***schema***, which describes the columns, or attributes.

UNIVERSITY OF TORONTO MISSISSAUGA

Relational Model

Proposed by Edgar F. Codd in 1970 as a data model which strongly supports data independence.

Made available in commercial DBMSs in 1984.

- It is not easy to implement data independence efficiently and reliably!

It is based on (a variant of) the mathematical notion of ***relation***.

Relations are represented as tables.

Relational Database: Definitions

Relational Database: a set of relations.

Relation: made of two parts

1. **Instance** → a table with rows and columns.
2. **Schema** → specifies name of a relation, and, name and type of each column.

# of Rows = cardinality
# of Columns = degree/arity

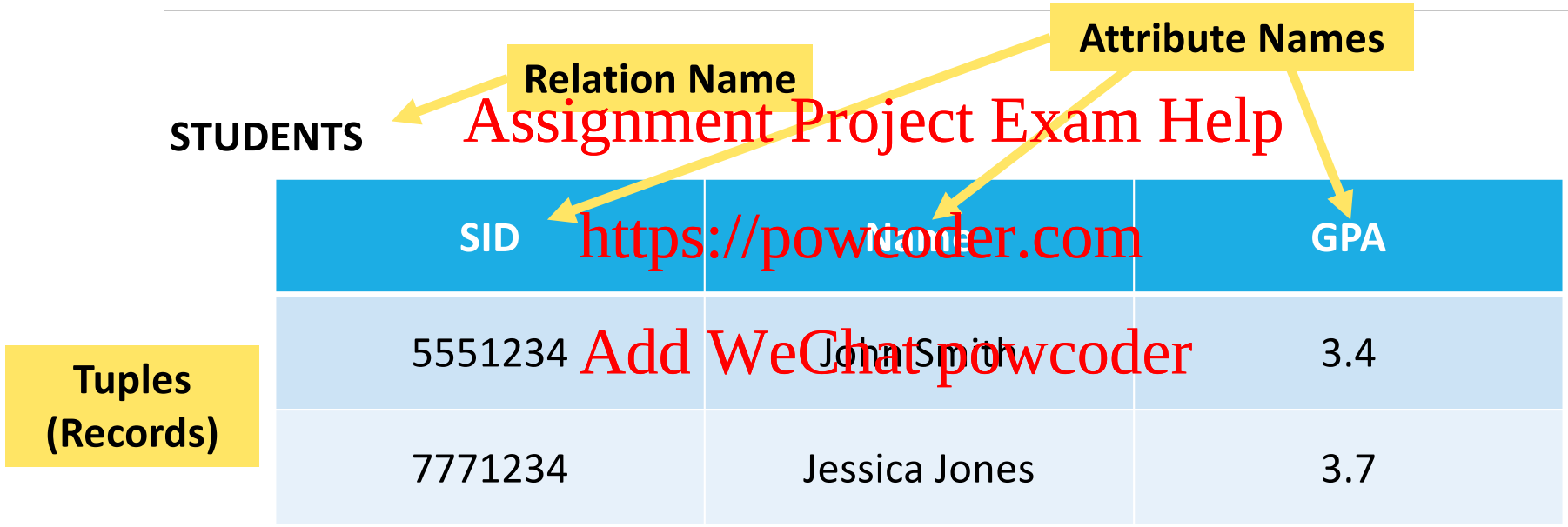
To make it easier, think of a relation as a ***set*** of rows and ***tuples***

- i.e. all rows are distinct!

rows, not columns!!!

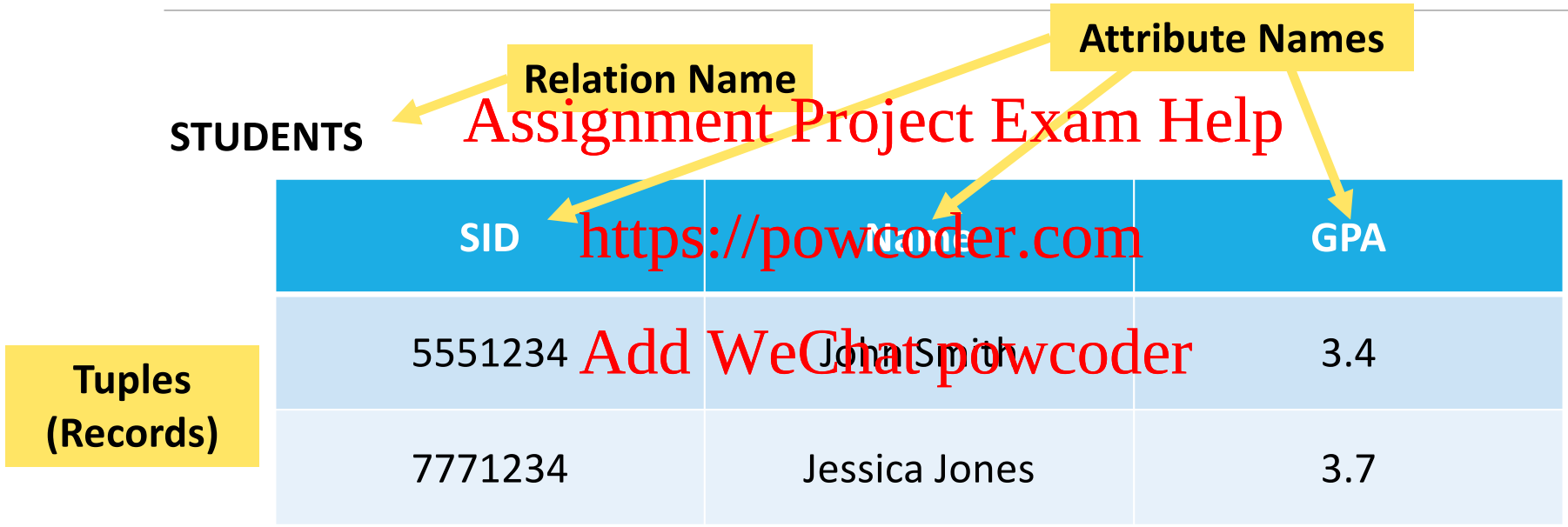
e.g. Students(sid: integer, name: string, gpa: real)

A Relation is a Table



The set of the permitted values for an attribute is called the attribute of **domain**.
e.g. domain(SID) = {5551234, 7771234}.

A Relation is a Table



Cardinality = 2 Degree = 3 **Note:** all rows are distinct!

Question?

UNIVERSITY OF TORONTO MISSISSAUGA

Do all entries in a column, in a relation instance, have to be distinct?

Pause the video and think about this for a moment.

NO! Unless we rule the column to be "unique" columns do not need to be distinct.
e.g. multiple students can have the same name or same birth date.

Relational Data Model

UNIVERSITY OF TORONTO MISSISSAUGA

Relation Schema → relation name and attribute list.

- Optionally: types of attributes. For example:
  - Students (id, name)
  - Students (id: string, name: string)

Relation → set of tuples conforming to a schema.

- e.g. {(5551234, John Smith), (7771234, Jessica Jones)…}

Database → set of relations.

Database Schema → set of all relation schemas in the database.

UNIVERSITY OF TORONTO MISSISSAUGA

Clarification of Terminology

1. **Relation** is a Table.
2. **Attribute** is a column.
3. **Tuple** is a row.
4. **Arity/Degree** of a relation is the number of attributes (columns).
5. **Cardinality** of a relation is the number of tuples (rows).

Why Relations?

Very simple model.

Often matches how we think about data.

Abstract model that underlies SQL, the most important database language!

Relations are Unordered

The order of tuples is irrelevant. Tuples may be stored in any arbitrary order.

e.g. instructor relation with unordered tuples

ID	name	dept_name	salary
12345	Einstein	Physics	95000
45411	Turing	Comp Sci	72500
35521	Mozart	Music	45000
98002	Dexter	Biology	67200
45556	Aristotle	Physics	86000
72331	Khan	History	55500
42399	Taylor	Mathematics	78800
53440	Bond	Chemistry	110000

Database

Information about enterprise is broken up into parts:

- instructor
- student
- advisor

Bad design:

univ (instructor_ID, dept_name, salary, student_ID, ...)

The need for NULL values.
e.g. represent a student with no instructor

Normalization Theory deals with how to design "good" relational schemas.

Repetition of information is bad!
e.g. two students have the same instructor

Relational Query Languages

A major strength of the relational model: supports simple, powerful querying of data.

Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.

- The key: precise *semantics* for relational queries.

- Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

The SQL Query Language

UNIVERSITY OF TORONTO MISSISSAUGA

Developed by IBM (system R) in the 1970s.

There was a need for a standard since it was used by many vendors.

Standards:

- SQL–86
- SQL–89 (minor revisions)

- SQL–92 (major revisions)
- SQL–99 (major extensions)
- …
- SQL-2011 (major extensions, specifically in temporal databases)
- SQL:2016 (8th revision of the standard, a lot of JSON functionality)

Temporal Database

UNIVERSITY OF TORONTO MISSISSAUGA

Efficiently and effectively store data relating to instances of time.

Timestamps are stored in RDBMSs, but are inefficient.

Two major methods (attributes) of tracking time:

1. Transaction Time
2. Valid Time

Temporal Database

UNIVERSITY OF TORONTO MISSISSAUGA

1. Transaction Time
   - The time period during which a row is committed to or recorded in the database.

2. Valid Time
   - The time period during which a row is regarded as correctly reflecting reality by the user of the database.

The combination of the attributes 'Transaction Time' and 'Valid Time' forms **Bitemporal Time**

Transaction Time Example

UNIVERSITY OF TORONTO MISSISSAUGA

ID	name	dept_name	transaction_from	transaction_to
12345	Einstein	Physics	January 2000	∞
45411	Turing	Comp Sci	September 1986	August 2000
35521	Mozart	Music	October 1999	∞
98002	Dexter	Biology	May 2018	∞
45556	Aristotle	Physics	January 2018	∞

Valid Time Example

UNIVERSITY OF TORONTO MISSISSAUGA

ID	name	dept_name	valid-to
12345	Einstein	Physics	∞
45411	Turing	Comp Sci	August 2000
35521	Mozart	Music	∞
98002	Dexter	Biology	August 2018
45556	Aristotle	Physics	December 2018

Database Schemas in SQL

SQL is primarily a query language.

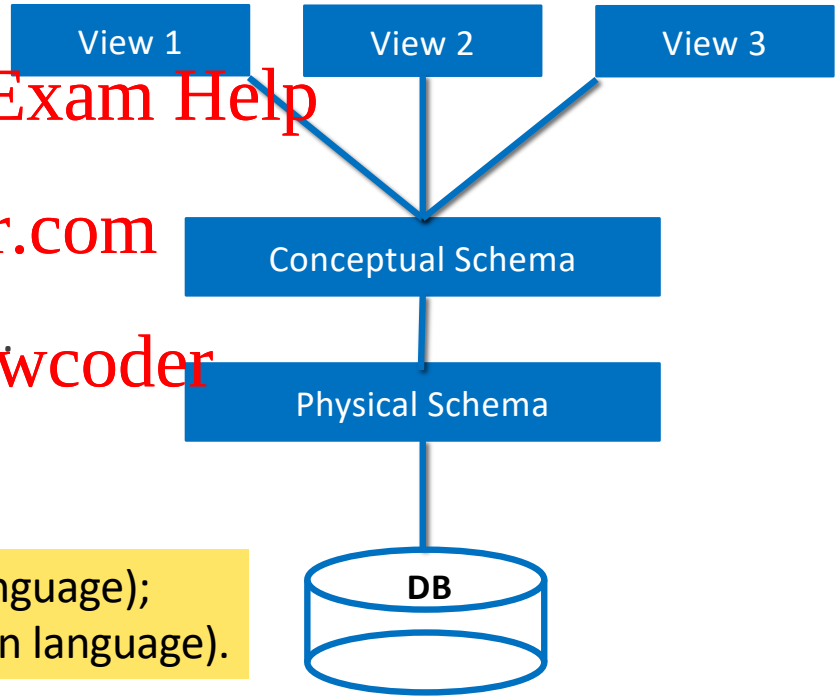
- Used for retrieving information from a database.

SQL also includes a data-definition component for describing schemas.

Levels of Abstraction

Many views, single conceptual (logical) schema and physical schema.

- Views describe how users see the data.
- Conceptual schema defines logical structure.
- Physical schema describes the files and indexes used.



- Schemas are defined using DDL (data definition language);
- Data is modified/queried using DML (data manipulation language).

UNIVERSITY OF TORONTO MISSISSAUGA

Example: University Database

Conceptual Data:

- Students(sid:string, name:string, login:string, age:integer, gpa:real)
- Courses(cid:string, cname:string, credits:integer)
- Enrolled (sid:string, cid:string, grade:string)

Physical Schema:

- Relations stored as unordered files.
- Index on first column of Students.

External Schema (i.e. View):

- Course_info(cid:string, enrollment:integer)

Integrity Constraints

UNIVERSITY OF TORONTO MISSISSAUGA

An *integrity constraint* is a property that must be satisfied by all meaningful database instances.

A constraint can be seen as *a predicate*.

- A database is *legal* if it satisfies all integrity constraints.

Types of constraints:

1. Intra-relational constraints
   - e.g. *domain constraints* and *tuple constraints*.
2. Inter-relational constraints
   - The most common is *referential constraint*.

Rationale For Integrity Constraints

UNIVERSITY OF TORONTO MISSISSAUGA

Useful for describing the application in greater detail.

Contributes to *data quality* (avoids data entry errors).

An element in the design process (discussed later).

Used by the system in choosing a strategy for query processing

Note: It is not the case that all desirable properties of the data in a database can be described by means of integrity constraints.
e.g. “data in the relation **Instructor** must be correct”

Tuple and Domain Constraints

A *tuple constraint* expresses conditions on the values of each tuple, independently of other tuples.

e.g. **(GPA > 3.0)** or **Net = Amount – Deduction**

A *domain constraint* is a tuple constraint that involves a single attribute.

e.g.. **(GPA $\leq$ 4.0) AND (GPA $\geq$ 0.0)**

UNIVERSITY OF TORONTO MISSISSAUGA

Unique Values For Tuples

RegNum	Surname	FirstName	BirthDate	Program
22335	Jones	Jenny	04/12/89	History
22567	Jones	Robby	17/02/91	Engineering
43754	Jones	Joe	28/03/91	Computing
38267	Smith	Melissa	13/08/90	Computing
17382	Smith	Melissa	08/09/90	English

Registration number identifies students

i.e. there is no pair of tuples with the same value for **RegNum**.

Personal data could identify students as well

i.e. there is no pair of tuples with the same values for all of **Surname**, **FirstName**, **BirthDate**.

Keys

A *key* is a set of attributes that uniquely identifies tuples in a relation.

More Precisely:

- A set of attributes K is a ***superkey*** for a relation R if R cannot contain two distinct tuples t_1 and t_2 s.t. $t_1[K] = t_2[K]$;
- K is a ***candidate key*** for R is K is a minimal superkey.
  - There exists no other superkey K' of R that is contained in K as a proper subset.
  - i.e. $K' \subset K$

Example

RegNum	Surname	FirstName	BirthDate	Program
22335	Jones	Jenny	04/12/89	History
22567	Jones	Robby	17/02/91	Engineering
43754	Jones	Joe	28/03/91	Computing
38267	Smith	Melissa	13/08/90	Computing
17382	Smith	Melissa	08/09/90	English

RegNum is a key!

i.e. **RegNum** is a *superkey* and contains a sole attribute, thus, it is minimal.

{Surname, FirstName, BirthDate} is another key!

Question?

RegNum	Surname	FirstName	BirthDate	Program
22335	Jones	Jenny	04/12/89	History
22567	Jones	Robby	17/02/91	Engineering
43754	Jones	Joe	28/03/91	Computing
38267	Smith	Melissa	13/08/90	Computing
17382	Smith	Melissa	08/09/90	English

Can {Surname, Program} be a key?

NO! It may not be apparent with this example, however, there could be combinations of Surnames and Programs that are the same.

i.e. You could potentially have students with the same Surname in the same Program.

UNIVERSITY OF TORONTO MISSISSAUGA

Another Question? Foreshadowing!

RegNum	Surname	FirstName	BirthDate	Program
22335	Jones	Jenny	04/12/89	History
22567	Jones	Robby	17/02/91	Engineering
43754	Jones	Joe	28/03/91	Computing
38267	Smith	Melissa	13/08/90	Computing
17382	Smith	Melissa	08/09/90	English

Any ideas on how we would address this issue?

i.e. How do we ensure that only a certain set of keys can be used to obtain the data sets that we want?

Normalization and Database Design (we will discuss in future!)

Existence of Keys

Relations are sets;

- each relation is composed of **distinct** tuples.

The whole set of attributes for a relation defines a ***superkey***.

Each relation has a key, which is a set of all its attributes, or a subset thereof.

Existence of Keys

Keys guarantee that each piece of data in the database can be accessed.

Keys are a major feature of the Relational Model

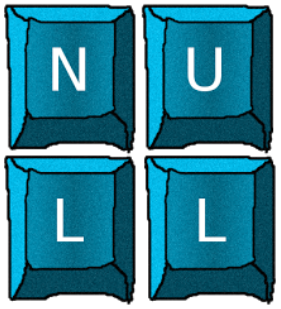
- Allow us to say that it is "value-based".

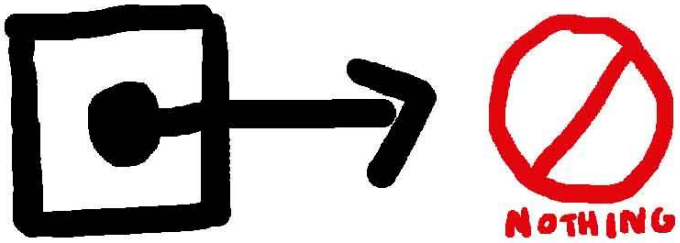
Keys and Null Values

The presence of NULL values negatively impacts keys:

- unique identification is no longer guaranteed.

- no assistance in establishing correspondence between data in different relations.





Keys and Null Values

RegNum	Surname	FirstName	BirthDate	Program
NULL	Jones	Robby	NULL	Computing
43754	Jones	Melissa	28/03/91	Engineering
38267	Smith	Melissa	NULL	NULL
NULL	Smith	Melissa	08/09/90	Engineering

Questions to think about?

1. How do we access the 1st tuple?
2. Are the 3rd and 4th tuples the same?

Keys and Null Values

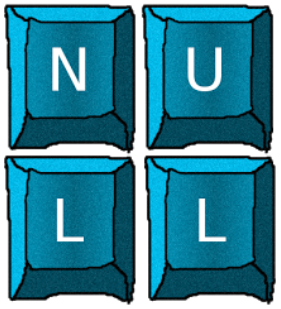
The presence of NULL values negatively impacts keys:

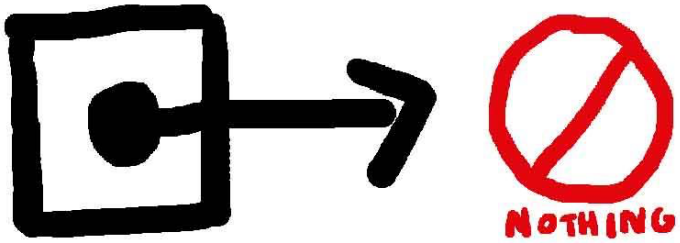
- unique identification is no longer guaranteed.

Maps to Q1

Maps to Q2

- no assistance in establishing correspondence between data in different relations.





Primary Keys

UNIVERSITY OF TORONTO MISSISSAUGA

In general, the presence of NULL values in key has to be limited.

- In most cases (and preferably) non-existent!

Each relations must have a ***primary key***, where NULL values are not allowed to occur in any attribute.

References between relations are realized through primary keys.

Notation:

- The attribute of the primary key are underlined!

Primary Keys

UNIVERSITY OF TORONTO MISSISSAUGA

Notation:

- The attribute of the primary key are underlined!

RegNum	Surname	FirstName	BirthDate	Program
22567	Jones	Robby	NULL	Computing
43754	Jones	Melissa	28/03/91	Engineering
38267	Smith	Melissa	NULL	NULL
17383	Smith	Melissa	08/09/90	Engineering

UNIVERSITY OF TORONTO MISSISSAUGA

DO We Always Have Primary Keys?

Ordinarily, we have reasonable primary keys.

e.g. Client/Account Number, Student Number, SIN, etc...

There potentially could be multiple keys, however, one will be designated as the primary.

UNIVERSITY OF TORONTO MISSISSAUGA

Recap

A set of fields is a ***key*** for a relation if:

1. No two distinct tuples can have the same value in all key fields, and
2. This is not true for any subset of the key.

◦ If false, then a ***superkey***.

If there are more than one key for a relation, one is selected to be the ***primary key***.

STUDENTS

SID	Name	GPA
5551234	John Smith	3.4
7771234	Jessica Jones	3.7

e.g. SID is a key for Students. The tuple {SID, GPA} is a superkey.

Primary and Candidate Keys

"For a given student and course, there is a single grade." vs. "Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade."

Be careful to define Integrity Constraints (ICs) correctly at design time.

ICs are checked when data is updated.

Enrolled(SID, CID, grade)

Enrolled(<u>SID</u>, <u>CID</u>, grade)

Enrolled(<u>SID</u>, CID, grade)

- Key {CID, grade}

Primary and Candidate Keys

Be Careful!

- If an IC is used carelessly it can prevent the storage of database instances that arise in practice!

UNIVERSITY OF TORONTO MISSISSAUGA

Foreign Keys

Pieces of data in different relations are correlated by means of values of primary keys.

Referential integrity constraints are imposed in order to guarantee that the values refer to existing tuples in the referenced relation.

A ***foreign key*** requires that the values on a set X of attributes of a relation R_1 must appear as values for the primary key of another relation R_2.

- i.e. set of attributes in one relation that is used to 'refer' to a tuple in another relation. These must correspond to primary key of the second relation! Like a 'logical pointer'!

Referential Integrity

e.g. **SID** is a foreign key referring to **Students**:

Enrolled(**SID**: string, **CID**: string, **grade**: string)

- If all foreign key constraints are enforced, ***referential integrity*** is achieved.

  i.e. No dangling references exist!

Referential Integrity

Only students listed in the **Student** relation should be allowed to enroll for courses.

Enrolled

SID	CID	Grade
22555	Biology101	A
23555	History407	B
23455	English201	A
22555	Genetics304	C

Students

SID	LastName	Login	Age	GPA
22555	White	white@sci	21	3.4
23555	Jones	jones@his	20	3.2
23455	Taylor	taylor@eng	20	3.6

Enforcing Referential Integrity

Consider the relations: **Students** and **Enrolled**

- *SID* in **Enrolled** is a foreign key that references **Students**.

QUESTION?!

What should be done if an **Enrolled** tuple with a non-existent *SID* is inserted?

REJECT IT!

Enforcing Referential Integrity

ANOTHER QUESTION?!

If a **Students** tuple is deleted, what should occur?

1. Delete all **Enrolled** tuples that reference it.
2. Prevent deletion of a **Students** tuple that it is being referenced.
3. Set SID in **Enrolled** tuples that refer to it to a *default* SID.
4. Set SID in **Enrolled** tuples that refer to it to NULL.

Similar if the primary key of **Students** tuple is updated.

This is called "Cascading"

Integrity Constraints (a.k.a. ICs)

RECALL
Semantics is the meaning or relationship of meaning of a sign or set of signs.

ICs are based upon the semantics of the real-world enterprise.
- This is being described in the database relations.

We CAN check a database instance to see if an IC is violated.

HOWEVER

We CANNOT infer that an IC is true by looking at an instance.
- An IC is a statement about all possible instances.

Key and ***Foreign Key*** ICs are the most common; more general ICs are also supported.

Where do ICs Come From?

UNIVERSITY OF TORONTO MISSISSAUGA

Let's work through an example!

Information:

3 relations: Offences, Officers, and Cars

1. Offences: {Code, Date, Officer, Dept, Registration}
2. Officers: {RegNum, Surname, FirstName}

3. Cars: {Registration, Dept, Owner}

Now let's add some data!

Offences

Code	Date	Officer	Dept	Registration
83746	25/05/2016	567	45	416 YYZ
29374	26/05/2016	984	45	747 LHR
39374	28/05/2016	984	45	416 YYZ
27483	28/05/2016	567	45	747 LHR
95848	28/05/2016	363	82	333 VCE

Officers

RegNum	Surname	FirstName
567	Hopkins	John
984	Walker	Johnny
363	Ross	George

Cars

Registration	Dept	Owner
416 YYZ	45	Michael Liut
747 LHR	45	Jessica Jones
333 VCE	82	Giovanni Saputo
647 YUL	45	Michael Liut

Offences

Code	Date	Officer	Dept	Registration
83746	25/05/2016	567	45	416 YYZ
29374	26/05/2016	984	45	747 LHR
39374	28/05/2016	984	45	416 YYZ
27483	28/05/2016	567	45	747 LHR
95848	28/05/2016	363	82	333 VCE

Officers

RegNum	Surname	FirstName
567	Hopkins	John
984	Walker	Johnny
363	Ross	George

Cars

Registration	Dept	Owner
416 YYZ	45	Michael Liut
747 LHR	45	Jessica Jones
333 VCE	82	Giovanni Saputo
647 YUL	45	Michael Liut

NOTE:

Offences[Officer] ⊆ Officers[RegNum]

Offences[Registration, Dept] ⊆ Cars[Registration,Dept]

Violation of Foreign Keys

UNIVERSITY OF TORONTO MISSISSAUGA

Offences

Code	Date	Officer	Dept	Registration
83746	25/05/2016	912	37	416 YYZ
29374	26/05/2016	912	45	747 LHR

Officers

RegNum	Surname	FirstName
567	Hopkins	John
984	Walker	Johnny

Cars

Registration	Dept	Owner
416 YYZ	45	Michael Liut
747 LHR	45	Jessica Jones
333 VCE	82	Giovanni Saputo

Referential Constraints

UNIVERSITY OF TORONTO MISSISSAUGA

General comments that should be noted:

Referential Constraints play an important role in making the Relational Model value-based.

Care is needed in case of referential constraints that involve two or more attributes.

Examples

Accidents

Code	Dept1	Registration1	Dept2	Registration2
6002	45	416 YYZ	82	333 VCE
6437	45	747 LHR	82	333 VCE

Cars

Registration	Dept	Owner	…
416 YYZ	45	Michael Liut	…
747 LHR	45	Jessica Jones	…
333 VCE	82	Giovanni Saputo	…
647 YUL	45	Michael Liut	…

Here we have two ***Referential Constraints*** for **Accidents**: {Registration1, Dept1} to **Cars** and {Registration2, Dept2} to **Cars**.

Citations, Images and Resources

UNIVERSITY OF TORONTO MISSISSAUGA

Database Management Systems (3rd Ed.), Ramakrishnan & Gehrke

Some content is based off the slides of Dr. Fei Chiang - http://www.cas.mcmaster.ca/~fchiang/

https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf

http://www.clipartpanda.com/clipart_images/key-clip-art-159-3522839

https://lh3.ggpht.com/EDtqA1VCeZlAbp2d9IxSePGy2QKAMjEEnER8TJrhxmDA443kLmlDZFMOJqtvY8JRrEo=w300

https://s3.amazonaws.com/images.katehaddleston.com/galleries/d6970e47-10f6-40a3-8745-fca33c1fc217/1428388432254_null_pointer_2_large

http://www.netanimations.net/Animated-dancing-red-question-mark-picture-moving.gif