Australian National University

W2 – Housekeeping

1. Weekly online quiz due 12noon Wednesday after watching online lectures.
2. Online lectures vs In-person lectures
3. Lecture slides uploaded on Thursday morning. Can you find the slides?
4. The PostgreSQL documentation is recommended for additional reading. It includes an excellent SQL reference.
   `https://www.postgresql.org/docs/15/index.html`
5. Class representatives: Jiabao Han, Lingwei Wu, Junling Yu, Neda Andromeda

Week 2

Relational Data Model and Data Definition Language

Week 2

- Self Study Review
- Quiz Feedback
- Superkey, Candidate Key and Primary Key
- Referential Integrity Constraints
- SQL: Data Definition Language

A Review of your Self Study

- W02L01: Relational Data Model: Schema and State
- W02L02: Relational Data Model: Integrity Constraints
- W02L03: SQL Data Definition Language - **Please Practise**

Review of Schema and State

Relation/Table, Relation Schema, Relational Database Schema and Relational Database State

Relation v.s. Table (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

- Correspondence of informal and formal terms:

INFORMAL TERMS	FORMAL TERMS
Table	Relation
Column	Attribute
Data type	Domain
Row	Tuple
Table definition	Relation schema

- **How many tuples and attributes does the table ENROL have?**
  3 tuples and 5 attributes.
- **In the relational data model, the order of tuples in a relation is not important but the order of the attributes in a relation is important?**
  Yes.

Relation Schema – Example

- Consider a relation schema ENROL
  - ENROL(StudentID: `INT`, CourseNo: `STRING`, Semester: `STRING`, Status: `STRING`, EnrolData: `DATE`).

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

Relational Database Schema – Example

- A **relational database schema** S is
  - a set of relation schemas $S = \{R_1, \ldots, R_m\}$, and
  - a set of integrity constraints IC.

STUDENT			
StudentID	Name	DoB	Email

COURSE		
No	Cname	Unit

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

Relational Database State – Example

- A **relational database state** of S is a set of relations such that
  - there is just one relation for each relation schema in S, and
  - all the relations satisfy the integrity constraints IC.

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

Relational Database State – Example

- A **relational database state** of S is a set of relations such that
  - **there is just one relation for each relation schema in** S

Relation schema

STUDENT			
StudentID	Name	DoB	Email

Relation

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

- **Can there be multiple relations that correspond to the same relation schema in a relational database state?**
  No.

Australian National University

Review of Integrity Constraints

- Domain constraints
- Key constraints
- Entity integrity constraints
- Referential integrity constraints

Domain Constraints

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

Key constraints

- A bunch of keys: superkey, candidate key, primary key

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!**

Entity Integrity Constraints

- 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.
  - 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.**

Referential Integrity Constraints

- A referential integrity constraint specifies a reference between two relations.
- 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

  - Similarly, every student ID appearing in ENROL must exist in STUDENT.

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 ...

Review of Data Definition language

- CREATE TABLE, ALTER TABLE, DROP TABLE
- INT, FLOAT, NUMERIC, CHAR, DATE
- NOT NULL, DEFAULT, CHECK
- UNIQUE, PRIMARY KEY, FOREIGN KEY
- INDEX

please practice

Week 2

- Self Study Review
- Quiz Feedback
- Superkey, Candidate Key and Primary Key
- Referential Integrity Constraints
- SQL: Data Definition Language

Australian National University

Thanks for your feedback!

Australian National University

Quiz-Q4

Q4 Version 3 (latest)

Question 1

Not yet answered

Marked out of 0.10

Consider the relation schemas where we have made bold the primary keys:

STUDENT(StudentID, Name, DoB, Email)

ENROL(StudentID, CourseNo, Semester, Status, EnrolDate)

The foreign key StudentID in ENROL references StudentID in STUDENT.

Which of the following statements is incorrect?

- a. There may exist StudentID values in STUDENT that do not appear in ENROL.
- b. StudentID must be a superkey in STUDENT.
- c. The StudentID values in ENROL must be distinct.

Australian National University

Quiz-Q6 statistics

Analysis of responses

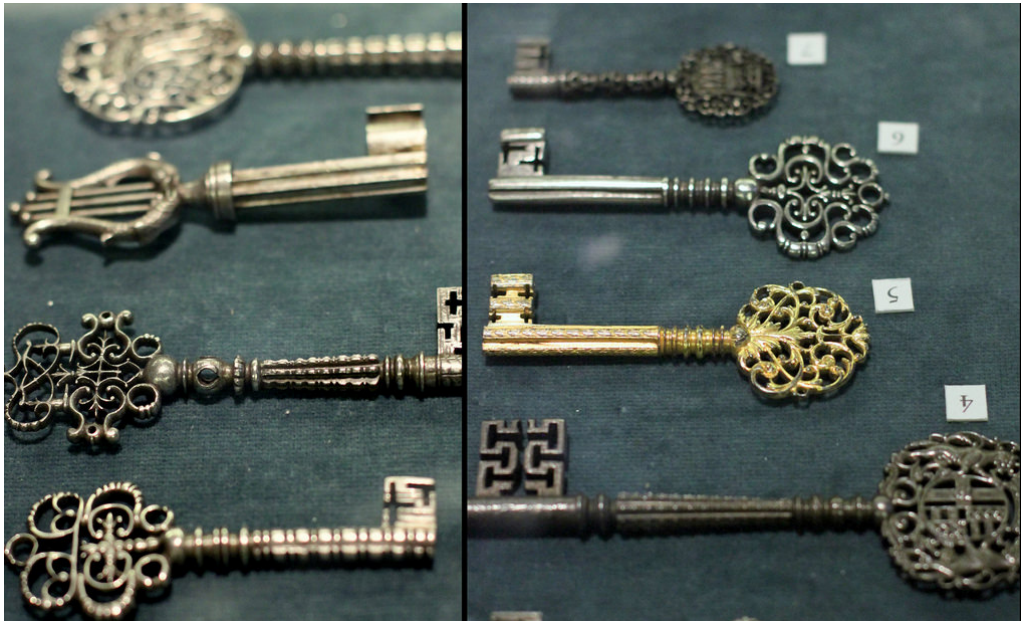
Part of question	Response	Partial credit	Count	Frequency
102655863	Relational Data Mode (Part 1): Schema and State	0.00%	61	16.27%
102655864	Relational Data Model (Part 2): Integrity Constraints	0.00%	143	38.13%
102655865	SQL (Part 1): SQL and Data Definition Language	0.00%	90	24.00%
102655866	None of the above	100.00%	155	41.33%

Week 2

- Self Study Review
- Quiz Feedback
- Superkey, Candidate Key and Primary Key
- Referential Integrity Constraints
- SQL: Data Definition Language

A Bunch of Keys

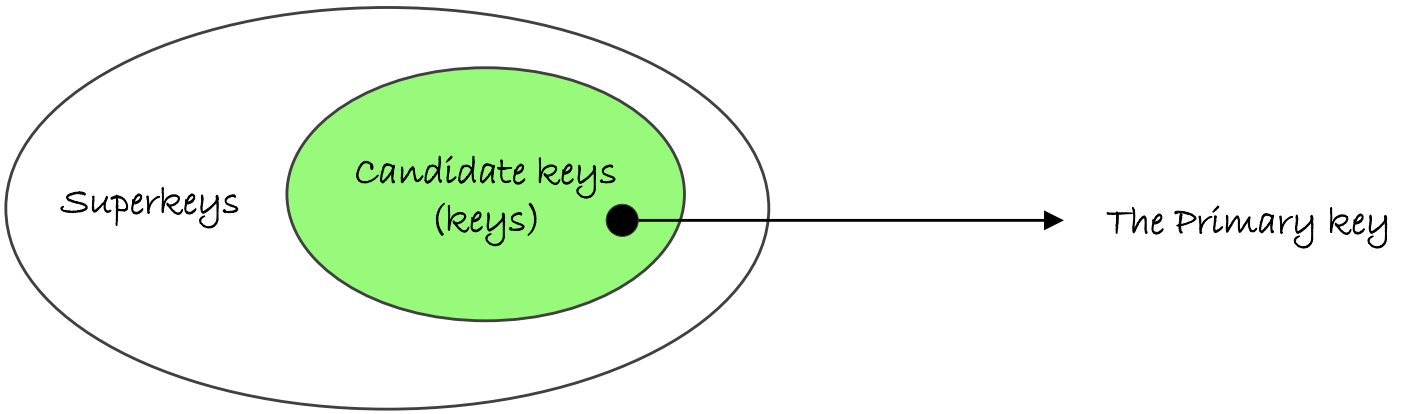
Superkey, Candidate key, Primary key and Foreign key



(Ashmolean Museum @ the University of Oxford `www.ashmolean.org/`)

A Bunch of Keys

- A *subset of the attributes* of a relation schema R is a **superkey** if it uniquely identifies any tuple in $r(R)$.
- A superkey K is called a **candidate key** if no proper subset of K is a superkey. That is, if you take any of the attributes out of K, then it is not enough to uniquely identify tuples.
- The **primary key** is chosen from the candidate keys and the primary key is one of the candidate keys.



- **Every candidate key must be a superkey in the same relation schema? Yes.**

Keys - Example 1

- Assume that a relation schema R(A, B, C, D) has only two candidate keys {A,B} and {C}.

- Is it possible that {A} is a SK?

  Answer: Impossible, otherwise {A,B} is not a candidate key (minimal SK).

- Is it possible that {B, C} is a SK?

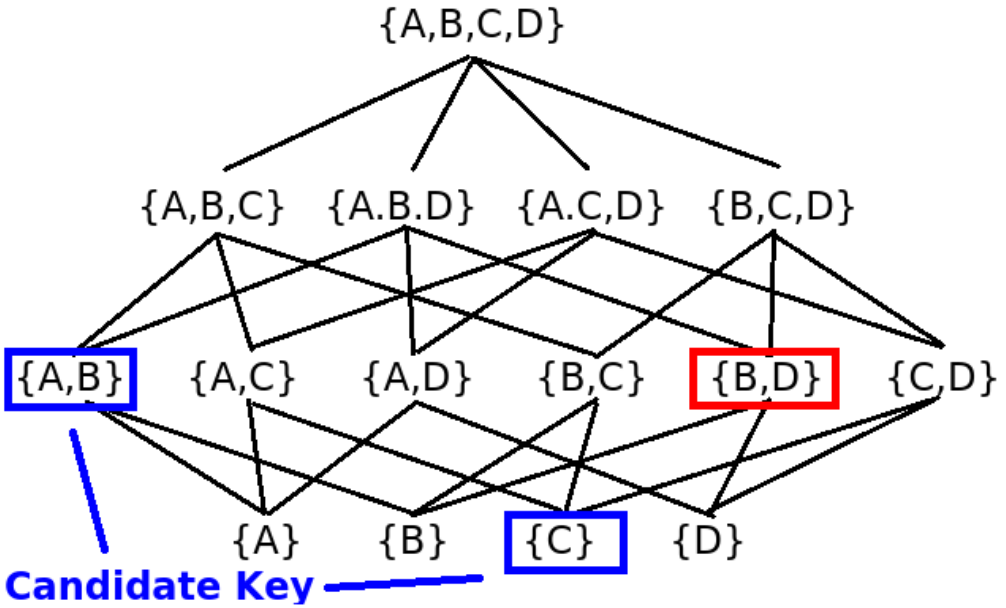
  Answer: {B, C} must be a SK because {C} is a candidate key.

- Is it possible that {B, D} is a SK? (tricky)

  Answer: {B, D} cannot be a SK because {B, D} does not have any candidate key as its subset.

Keys - Example 1

- Assume that a relation schema R(A, B, C, D) has only two candidate keys {A,B} and {C}.



- If it possible that {B, D} is a SK? (tricky)

  Answer: {B, D} cannot be a SK because {B, D} does not has any candidate key as its subset.

Keys – Example 2

- No two courses have the same **No** $\Rightarrow$ {No} is a superkey (**SK**) of COURSE.

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

- No two students have the same **StudentID** $\Rightarrow$ {StudentID} is a **SK** of STUDENT.
- No two students have the same **Email** $\Rightarrow$ {Email} is a **SK** of STUDENT.

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
...	...	...	...

Keys – Example 2

- {StudentID} is a **SK** of STUDENT and {Email} is also a **SK** of STUDENT.

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
...	...	...	...

- What are all **SKs** of STUDENT?

  For STUDENT, a SK can be any subset of attributes which includes StudentID or any subset of attributes which includes Email, e.g., {StudentID}, {StudentID, Name}, {StudentID, Email}, . . .

- What are **candidate keys** of STUDENT?

  For STUDENT, {StudentID} and {Email} are two candidate keys.

- What about the **primary key** of STUDENT?

  For STUDENT, the primary key can be chosen as either {StudentID} or {Email}.

Keys – Example 3

- No two enrolments have the same **StudentID**, the same **CourseNo** in the same **Semester** $\Rightarrow$ {StudentID , CourseNo, Semester} is a SK of ENROL.

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
...	...	...	...	...

Keys – Example 3

- {StudentID , CourseNo, Semester} is a SK of ENROL.

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

- What are all **SKs** of ENROL?

  For ENROL, a SK can be any subset of attributes which includes all StudentID, CourseNo and Semester, e.g., {StudentID , CourseNo, Semester}, {StudentID , CourseNo, Semester, Status}, ...

- What are **candidate keys** of ENROL?

  For ENROL, {StudentID , CourseNo, Semester} is the only candidate key.

- What about the **primary key** of ENROL?

  For ENROL, the primary key can only be {StudentID , CourseNo, Semester}.

keys – Exercise

- Find out candidate keys of BOOKING from the following schema of an ACCOMMODATION database held in a relational DBMS:
  - HOTEL(hotelNo, hotelName, city) with the primary key {hotelNo},
  - ROOM(roomNo, hotelNo, type, price) with the primary key {roomNo, hotelNo},
  - GUEST(guestNo, guestName, guestAddress) with the primary key {guestNo},
  - BOOKING(guestNo, hotelNo, date, roomNo).
- **Some additional constraints are as follows:**
  1. A booking can be made for one day only.
  2. A guest can make several bookings in a hotel for different days.
  3. A guest cannot make two or more bookings in the same hotel for the same day.
  4. A guest can make two or more bookings in different hotels for the same day.
  5. A booking cannot be in joint names. In other words a booking can only be held in the name of one guest.

Keys – Exercise

- BOOKING(guestNo, hotelNo, date, roomNo).
  1. A booking can be made for one day only.
  2. A guest can make several bookings in a hotel for different days.
  3. A guest cannot make two or more bookings in the same hotel for the same day.
  4. A guest can make two or more bookings in different hotels for the same day.
  5. A booking cannot be in joint names. In other words a booking can only be held in the name of one guest.
- Is {guestNo, hotelNo, date} a minimal SK and hence a candidate key?.
  - Is {guestNo, hotelNo, date} is a SK? Yes because of (3).
  - Is {guestNo, hotelNo} a SK? No because of (2).
  - Is {guestNo, date} a SK? No because of (4).
  - Is {hotelNo, date} a SK? No because a hotel usually has multiple rooms (indicated by the fact that ROOM(roomNo, hotelNo, type, price) has the primary key {roomNo, hotelNo}).
- Thus {guestNo, hotelNo, date} a minimal SK and hence a candidate key.

Keys – Exercise

- BOOKING(guestNo, hotelNo, date, roomNo).
  1. A booking can be made for one day only.
  2. A guest can make several bookings in a hotel for different days.
  3. A guest cannot make two or more bookings in the same hotel for the same day.
  4. A guest can make two or more bookings in different hotels for the same day.
  5. A booking cannot be in joint names. In other words a booking can only be held in the name of one guest.
- Is {guestNo, hotelNo, roomNo} a candidate key?
  No, it is not even a SK because of (2).
- Is {guestNo, date, roomNo} a candidate key?
  No, it is not even a SK because of (4).
- Is {hotelNo, date, roomNo} a candidate key?
  Yes, it is a SK because of (3) and (5) and no proper subset of {hotelNo, date, roomNo} is a SK, hence {hotelNo, date, roomNo} is a candidate key.

Keys – Exercise

- BOOKING(guestNo, hotelNo, date, roomNo).
  1. A booking can be made for one day only.
  2. A guest can make several bookings in a hotel for different days.
  3. **A guest is not allowed to make more than one booking for the same day even in the different hotels**.
  4. A booking cannot be in joint names. In other words a booking can only be held in the name of one guest.
- Is {guestNo, hotelNo, date} a minimal SK and hence a candidate key?.
  - Is {guestNo, hotelNo, date} is a SK? Yes because of (3).
  - Is {guestNo, hotelNo} a SK? No because of (2).
  - Is {guestNo, date} a SK? **Yes because of (3)**.
- Thus {guestNo, hotelNo, date} is no longer a **minimal** SK and hence cannot be a candidate key.
- Now {guestNo, date} is a minimal SK and hence a candidate key.
- Note that {hotelNo, date, roomNo} is also a minimal SK and hence a candidate key.

Week 2

- Self Study Review
- Quiz Feedback
- Superkey, Candidate Key and Primary Key
- Referential Integrity Constraint
- SQL: Data Definition Language

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 $R_1[A_1, \ldots, A_n] \subseteq R_2[B_1, \ldots, B_n]$ 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**.

Referential integrity constraints – Example 1

- In ENROL, [CourseNo]$\subseteq$ COURSE[No] and
  [StudentID]$\subseteq$ STUDENT[StudentID].

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
456	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016

Referential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: Does the above database satisfy the foreign key of ENROL: [StudentID]$\subseteq$ STUDENT[StudentID]?

Yes.

Rreferential integrity constraints – Example 1

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
458	COMP3600	2016 S2	active	11/06/2016

Question: Does the above database satisfy the foreign key of ENROL: [CourseNo]$\subseteq$ COURSE[No]?

No, because COMP3600 does not exist as a No value in COURSE.

Referential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we delete the first tuple in STUDENT?
No, because it will violate the foreign key of ENROL: [StudentID]$\subseteq$ STUDENT[StudentID]

Rreferential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we delete the first tuple in ENROL?
Yes.

Referential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we update COMP2400 to be COMP6240 in COURSE?
No, because it will violate the foreign key of ENROL: [CourseNo]$\subseteq$ COURSE[No].

Referential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we insert a new course COMP3600 Algorithms with 6 units in COURSE?

Yes.

Referential integrity constraints – Example 1

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
458	COMP2400	2016 S2	active	11/06/2016

Question: The foreign key StudentID in Enrol references StudentID in Student. The StudentID values in Enrol must be distinct?
No.

Referential integrity constraints – Example 2

- Identify **foreign keys**, if any, in HOTEL, ROOM, BOOKING and GUEST relations.
  - HOTEL(hotelNo, hotelName, city) with the primary key {hotelNo},
  - ROOM(roomNo, hotelNo, type, price) with the primary key {roomNo, hotelNo},
  - GUEST(guestNo, guestName, guestAddress) with the primary key {guestNo},
  - BOOKING(guestNo, hotelNo, date, roomNo).

- **Answer**:
  - ROOM: [hotelNo] $\subseteq$ HOTEL[hotelNo];
  - BOOKING: [hotelNo] $\subseteq$ HOTEL[hotelNo],
    [guestNo] $\subseteq$ GUEST[guestNo],
    [roomNo, hotelNo] $\subseteq$ ROOM[roomNo, hotelNo].

Example 2 – A Common Pitfall

- Consider the following relation schemas:
  - ROOM(roomNo, hotelName, type, price) with the primary key {roomNo, hotelName},
  - BOOKING(guestNo, date, roomNo, hotelName).

ROOM			
roomNo	hotelName	type	price
01	Sydney	twin	200
02	Sydney	single	100
01	Canberra	single	150

BOOKING			
guestNo	date	roomNo	hotelName
P1	30/07/2018	02	Sydney
P2	31/07/2018	01	Canberra

Now we add the following foreign key constraint:

- BOOKING[roomNo, hotelName]⊆ROOM[roomNo, hotelName]
- Is the above **equivalent** to:
  BOOKING[roomNo]⊆ROOM[roomNo], and
  BOOKING[hotelName]⊆ROOM[hotelName] **?**

Example 2 – A Common Pitfall

ROOM			
roomNo	hotelName	type	price
01	Sydney	twin	200
02	Sydney	single	100
01	Canberra	single	150

BOOKING			
guestNo	date	roomNo	hotelName
P1	30/07/2018	01	Sydney
P2	31/07/2018	02	Canberra

- The above relations satisfy the foreign keys:
  - BOOKING[roomNo]$\subseteq$ROOM[roomNo], and
  - BOOKING[hotelName]$\subseteq$ROOM[hotelName]

  but does not satisfy the foreign key:
  - BOOKING[roomNo, hotelName]$\subseteq$ROOM[roomNo, hotelName]

Referential integrity constraints – Example 2

- ROOM: [hotelNo] $\subseteq$ HOTEL[hotelNo];
- BOOKING: [hotelNo] $\subseteq$ HOTEL[hotelNo],[guestNo] $\subseteq$ GUEST[guestNo], [roomNo, hotelNo] $\subseteq$ ROOM[roomNo, hotelNo].
- Is it possible to make a booking in the BOOKING relation in the name of a person who is not listed in the GUEST relation?

  Answer: Impossible because in BOOKING, [guestNo] $\subseteq$ GUEST[guestNo], i.e., the guestNo value of BOOKING must exist as a guestNo value of GUEST.
- Is it possible to add a new room in the ROOM relation to a hotel that is not listed in the HOTEL relation?

  Answer: Impossible because in ROOM: [hotelNo] $\subseteq$ HOTEL[hotelNo], i.e., the hotelNo value of ROOM must exist as a hotelNo value of HOTEL.
- Is it possible to add a new hotel without any bookings or room information to the ACCOMMODATION database?

  Answer: Possible because none of the attributes in HOTEL(hotelNo, hotelName, city) references to any attribute in ROOM, GUEST and BOOKING.

Week 2

- Self Study Review
- Quiz Feedback
- Superkey, Candidate Key and Primary Key
- Referential Integrity Constraint
- SQL: Data Definition Language

Data Definition Language

Data Definition Language – Relation Schema

- Create a relation schema ENROL
  - **Enrol**(StudentID: `INT`, CourseNo: `STRING`, Semester: `STRING`, Status: `STRING`, EnrolData: `DATE`)

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

- The **CREATE TABLE** statement is used to create a new relation schema by specifying its name, its attributes and, *optionally*, its constraints.

```
CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE);
```

Data Definition Language – CREATE TABLE

- Create a relation schema ENROL
  - **Enrol**(StudentID, CourseNo, Semester, Status, EnrolData)

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

- Can we use the following **CREATE TABLE** statement to create the above relation schema?

```
CREATE TABLE Enrol(StudentID, CourseNo, Semester, Status,
EnrolDate);
```

- **No** because the data type is required for each attribute.

Data Definition Language – CREATE TABLE

- Create a relation schema ENROL
  - **Enrol**(StudentID: `INT`, CourseNo: `STRING`, Semester: `STRING`, Status: `STRING`, EnrolData: `DATE`)

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

- None of the following **CREATE TABLE** statements is correct.

1. ```
   CREATE TABLE Enrol(StudentID INT; CourseNo VARCHAR(20);
   Semester VARCHAR(50); Status VARCHAR(50); EnrolDate DATE);
   ```
2. ```
   CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
   Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE,);
   ```
3. ```
   CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
   Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE),
   ```
   ```

Data Definition Language – CREATE TABLE

- Create a relation schema ENROL
  - **Enrol**(StudentID: `INT`, CourseNo: `STRING`, Semester: `STRING`, Status: `STRING`, EnrolData: `DATE`)

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

- The correct **CREATE TABLE** statement

```
CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE);
```

- What about the following two **CREATE TABLE** statements?

```
create table Enrol(StudentID int, CourseNo varchar(20),
Semester varchar(50), Status varchar(50), EnrolDate date);
```

```
CREATE TABLE enrol(studentiD INT, courseno VARCHAR(20),
semester VARCHAR(50), status VARCHAR(50), enroldate DATE);
```

Data Definition Language – CREATE TABLE

- Create a relation schema ENROL
  - **Enrol**(StudentID: `INT`, CourseNo: `STRING`, Semester: `STRING`, Status: `STRING`, EnrolData: `DATE`)

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

- PostgreSQL switches **CREATE TABLE** statements to lower case **unless we use double quotes**.

```
create table enrol(studentid int, courseno varchar(20),
semester varchar(50), status varchar(50), enroldate date);
```

```
u1024708=> \d enrol
            Table "public.enrol"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 studentid | integer               |
 courseno  | character varying(20) |
 semester  | character varying(50) |
 status    | character varying(50) |
 enroldate | date                  |
```

Data Definition Language – CREATE TABLE

- Can we create two relation schemas with the same name in the same database?

```
CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE);
```

```
create table enrol(studentid int, courseno varchar(20),
semester varchar(50), status varchar(50), enroldate date);
```

- **No** with the following error message.

```
u1024708=> create table enrol(studentid int, courseno varchar(20),
u1024708(> semester varchar(50), status varchar(50), enroldate date);
ERROR:  relation "enrol" already exists
```

Data Definition Language – CREATE TABLE

- Can we create the following two relation schemas in the same database?

```
u1024708=> CREATE TABLE Enrol(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE);
CREATE TABLE
u1024708=> CREATE TABLE "Enrol"(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50), EnrolDate DATE);
CREATE TABLE
```

- Yes. Enrol and "Enrol" are different.

```
u1024708=> \dt
              List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+----------
 public | Enrol          | table | u1024708
 public | enrol          | table | u1024708
```

Data Definition Language – Relational Database Schema

- A **relational database schema** S is
  - a set of relation schemas $S = \{R_1, \ldots, R_m\}$, and
  - a set of integrity constraints IC.

STUDENT			
StudentID	Name	DoB	Email

COURSE		
No	Cname	Unit

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

Data Definition Language – Domain Constraints

STUDENT			
StudentID	Name	DoB	Email

COURSE		
No	Cname	Unit

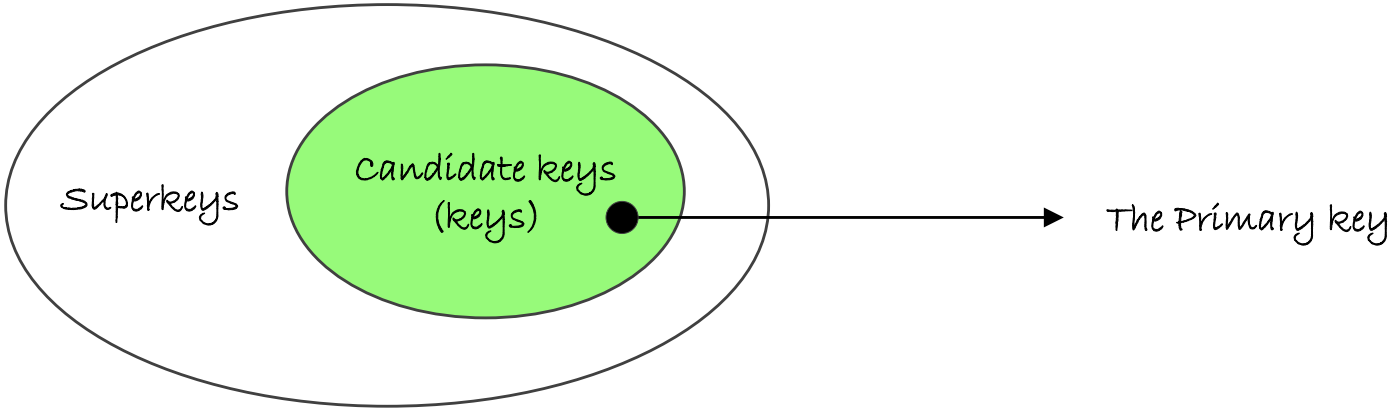
ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

```
CREATE TABLE STUDENT(StudentID INT, Name VARCHAR(50), DoB Date,
Email VARCHAR(100));

CREATE TABLE COURSE(No VARCHAR(20), Cname VARCHAR(50), Unit SMALLINT);

CREATE TABLE ENROL(StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50));
```

Data Definition Language – Key Constraints



- **UNIQUE:** uniquely identify each tuple in a table.

  Every superkey is UNIQUE. Should we specify UNIQUE for every superkey?

STUDENT			
StudentID	Name	DoB	**Email**

Data Definition Language – Candidate Key

STUDENT			
StudentID	Name	DoB	**Email**

- **UNIQUE:** uniquely identify each tuple in a table.
  Specify UNIQUE for every candidate key.
- For example, {StudentID} and {Email} are two candidate keys for STUDENT.

```
CREATE TABLE Student
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100),
        UNIQUE(StudentID),
        UNIQUE(Email));
```

Data Definition Language – Candidate Key

ENROL				
StudentID	**CourseNo**	**Semester**	Status	EnrolDate

- {StudentID, CourseNo, Semester} is a candidate key of ENROL.

```
CREATE TABLE Enrol
       (StudentID INT ,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        EnrolDate DATE,
        UNIQUE(StudentID, CourseNo, Semester));
```

Data Definition Language – Primary Key

STUDENT			
StudentID	Name	DoB	**Email**

- **PRIMARY KEY:** Specify PRIMARY KEY the primary key.
- For example, {StudentID} and {Email} are two candidate keys for STUDENT, and {StudentID} is selected as the primary key.

```
CREATE TABLE STUDENT
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100),
        PRIMARY KEY(StudentID),
        UNIQUE(Email));
```

Data Definition Language – Primary Key

ENROL				
StudentID	**CourseNo**	**Semester**	Status	EnrolDate

- {StudentID, CourseNo, Semester} is the primary key of ENROL.

```
CREATE TABLE Enrol
       (StudentID INT ,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        EnrolDate DATE,
        PRIMARY KEY(StudentID, CourseNo, Semester));
```

Data Definition Language – Primary Key

STUDENT			
StudentID	Name	DoB	**Email**

- Can we select multiple primary keys for the same relation schema?

```
CREATE TABLE STUDENT
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100),
        PRIMARY KEY(StudentID),
        PRIMARY KEY(Email));
```

- No because multiple primary keys for the same relation schema are not allowed.

Data Definition Language – Candidate Key

STUDENT			
StudentID	Name	DoB	**Email**

- Can we add multiple UNIQUE constraints for the same relation schema?

```
CREATE TABLE STUDENT
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100),
        UNIQUE(StudentID),
        UNIQUE(Email));
```

- Yes because multiple candidate keys (or superkeys) for the same relation schema are allowed.

Data Definition Language – Entity Integrity Constraints

- **Entity integrity constraints**: no primary key value can be NULL.
- Can the StudentID value be NULL?

```
CREATE TABLE ENROL
       (StudentID INT ,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        EnrolDate DATE,
        PRIMARY KEY(StudentID, CourseNo, Semester));
```

- No. None of the columns listed in the primary key can be NULL.

Data Definition Language – Entity Integrity Constraints

- What about UNIQUE constraints?
- Can the StudentID value be NULL?

```
CREATE TABLE Student
       (StudentID INT,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100),
        UNIQUE(StudentID),
        UNIQUE(Email));
```

- In PostgreSQL, two NULL values are not considered equal. That means even in the presence of a unique constraint it is possible to store duplicate rows that contain a null value in at least one of the constrained columns. **But other SQL databases might not follow this rule and be careful when developing applications that are intended to be portable.**

Data Definition Language – Referential Integrity Constraints

- Referential integrity constraints: the values in a column (or a group of columns) in one table must match the values appearing in some row of another table.

```
CREATE TABLE STUDENT( StudentID INT PRIMARY KEY, Name VARCHAR(50),
DoB Date, Email VARCHAR(100));

CREATE TABLE COURSE( No VARCHAR(20) PRIMARY KEY, Cname VARCHAR(50),
Unit SMALLINT);

CREATE TABLE ENROL( StudentID INT, CourseNo VARCHAR(20),
Semester VARCHAR(50), Status VARCHAR(50));
```

- Every StudentID appearing in ENROL must exist in STUDENT.
- Every CourseNo appearing in ENROL must exist in COURSE.

Data Definition Language – Foreign Key

```
CREATE TABLE Student
       ( StudentID INT PRIMARY KEY,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100));

CREATE TABLE Course
       ( No VARCHAR(20) PRIMARY KEY,
        Cname VARCHAR(50),
        Unit SMALLINT);

CREATE TABLE Enrol
       ( StudentID INT,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        FOREIGN KEY(StudentID) REFERENCES Student(StudentID),
        FOREIGN KEY(CourseNo) REFERENCES Course(No));
```

- Does {StudentID} in Student have to be the primary key of Student?

  Answer: In PostgreSQL, {StudentID} in Student must be either the primary key or form a unique constraint.

Attribute Constraints – Foreign Key

```
CREATE TABLE ENROL
       ( StudentID INT,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID),
        FOREIGN KEY(CourseNo) REFERENCES COURSE(No));

CREATE TABLE STUDENT
       ( StudentID INT PRIMARY KEY,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100));

CREATE TABLE COURSE
       ( No VARCHAR(20) PRIMARY KEY,
        Cname VARCHAR(50),
        Unit SMALLINT);
```

- Can we define ENROL before STUDENT and COURSE?

  Answer: No. ENROL has the foreign keys that reference STUDENT and COURSE.

Australian National University

Create Index (optional reading, will not be assessed)

CREATE INDEX constructs an index on the specified column(s) of the specified table.
In PostgreSQL, the index methods include B-tree, hash and others.

STUDENT

StudentID	Name	Age
111	Ava	30
222	Tom	25
333	John	35
444	Emily	35

COURSE

CourseNo	Name	Unit
ECON2102	Economics	6
COMP2400	Databases	6
BUSN2011	Accounting	6

ENROL

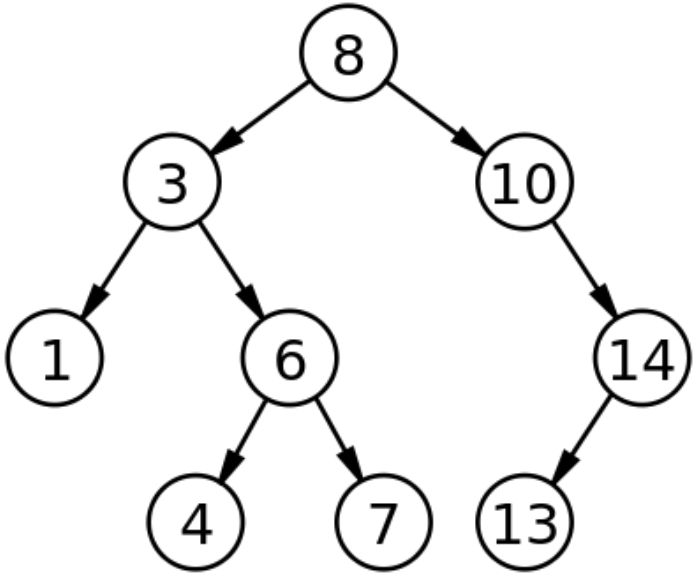
StudentID	CourseNo	Semester
111	BUSN2011	S2 2020
111	COMP2400	S2 2020
111	ECON2102	S2 2019
222	BUSN2011	S2 2020
222	COMP2400	S2 2020
333	BUSN2011	S2 2020
333	COMP2400	S2 2020
333	ECON2102	S2 2020

FK (StudentID) references STUDENT(StudentID)
FK (CourseNo) references COURSE(CourseNo)

`https://www.postgresql.org/docs/12/sql-createindex.html`

Create Index (optional reading, will not be assessed)

CREATE INDEX constructs an index on the specified column(s) of the specified table.
How to use '**B-tree**' (binary search tree) to construct an index?

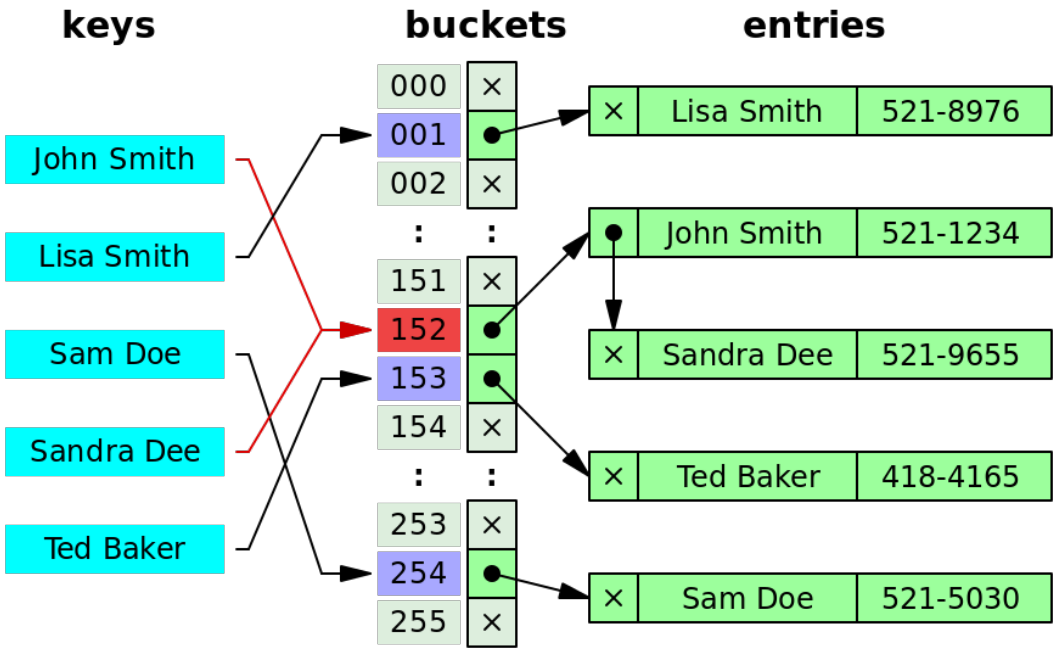


`https://en.wikipedia.org/wiki/Binary_search_tree`

Create Index (optional reading, will not be assessed)

CREATE INDEX constructs an index on the specified column(s) of the specified table.

How to use '**Hash Function**' to construct an index?



https://en.wikipedia.org/wiki/Hash_table