Week 2 Workshop

Housekeeping

1. Please attend the lab that you had registered for and the lab signup had been finalised. Lab swaps are not allowed unless there is a special consideration and an approval.
2. From Week 2 to Week 11, weekly online quiz is always due 23:59 pm Thursday after you watch the online lectures.
3. After Lab 1, If you still have any questions or issues about the lab environment, please bring your questions to the online drop-in sessions (Aug 6, Fri 3-5 pm) in Week 2.
4. An optional exercise website is available for our course

   `https://cs.anu.edu.au/dab/bench/db-exercises/`
5. Make effective use of Wattle discussion forum.
   - We strongly encourage you to ask your questions on the forum, and everyone in the class can benefit from the discussions and answers.
   - You should not post any solutions/results/ideas/interpretations related to assessment items (including assignments, quizzes, tests, exams).

Thanks for your feedback!

(1) Set, Tuple, Cartesian product of sets and Relation

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

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

Set – Example

- A set is a **collection** of **distinct** elements.

Set – Example

- A set is a **collection** of **distinct** elements.
- **Collection**: the elements in a set have no order.
  e.g., $\{A, B\} = \{B, A\}$

Set – Example

- A set is a **collection** of **distinct** elements.
- **Collection**: the elements in a set have no order.
  e.g., $\{A, B\} = \{B, A\}$
- **Distinct**: each element can not be in the set more than once.
  e.g., {A, A, B} is Not a set.
  Note that **multisets** allow to have duplicate elements.

Set – Example

- A set is a **collection** of **distinct** elements.
- **Collection**: the elements in a set have no order.
  e.g., $\{A, B\} = \{B, A\}$
- **Distinct**: each element can not be in the set more than once.
  e.g., {A, A, B} is Not a set.
  Note that **multisets** allow to have duplicate elements.
- **Cardinality**: the cardinality of a set is the number of elements of the set.

Tuple – Example

- A tuple is an **ordered** list of n elements.

Tuple – Example

- A tuple is an **ordered** list of n elements.
- **ordered**: the elements in a tuple have an order.
  e.g., $(A, B) \neq (B, A)$

Tuple – Example

- A tuple is an **ordered** list of n elements.
- **ordered**: the elements in a tuple have an order.
  e.g., $(A, B) \neq (B, A)$
- The same element can be in a tuple **more than once**.
  e.g., (A, A, B)is a tuple.

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order

- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}?

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order

- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}?

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order

- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}? Yes!

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order
- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}? Yes!
- Questino 2: {(A,B),(A,C)} = {(B,A),(A,C)}?

A Set of Tuples – Example

- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order
- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}? Yes!
- Questino 2: {(A,B),(A,C)} = {(B,A),(A,C)}?

A Set of Tuples – Example

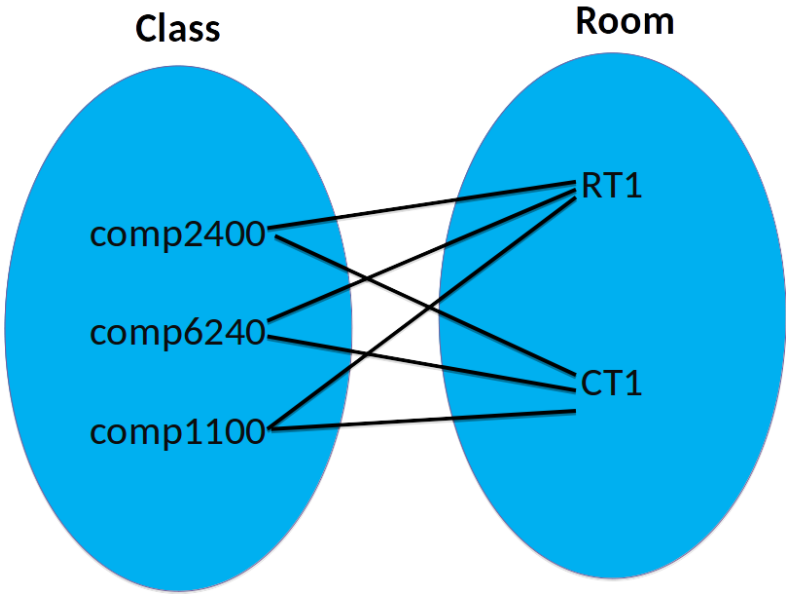
- A set of tuples is a collection of distinct tuples.
- **Set**:
  - the tuples in this set have no order.
  - each tuple can not be in the set more than once.
- **Tuple**:
  - the elements in a tuple have an order
- Question 1: {(A,B),(A,C)} = {(A,C),(A,B)}? Yes!
- Questino 2: {(A,B),(A,C)} = {(B,A),(A,C)}? No!

Cartesian product – Examples

- Let *Class* and *Room* be two sets:
  - $Class = \{comp2400, comp6240, comp1100\}$
  - $Room = \{RT1, CT1\}$
- What is the Cartesian product of $Class \times Room$?

Cartesian product – Examples

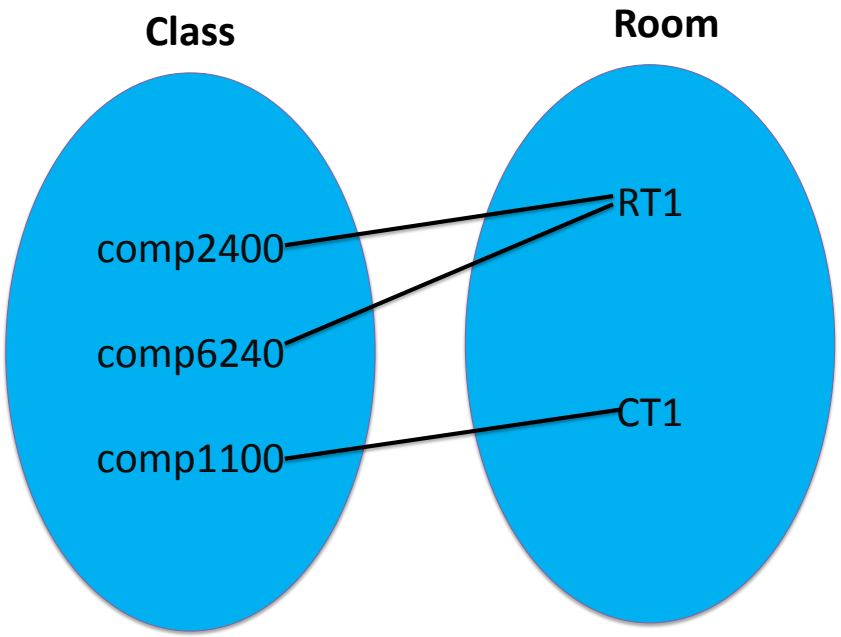
- Let *Class* and *Room* be two sets:
  - $Class = \{comp2400, comp6240, comp1100\}$
  - $Room = \{RT1, CT1\}$
- What is the Cartesian product of $Class \times Room$?
- $Class \times Room = \{(c, r) | c \in Class, r \in Room\}$
  $= \{(comp2400, RT1), (comp2400, CT1), (comp6240, RT1),$
  $(comp6240, CT1), (comp1100, RT1), (comp1100, CT1)\}$



Class	Room
comp2400	RT1
comp2400	CT1
comp6240	RT1
comp6240	CT1
comp1100	RT1
comp1100	CT1

Relations – Examples

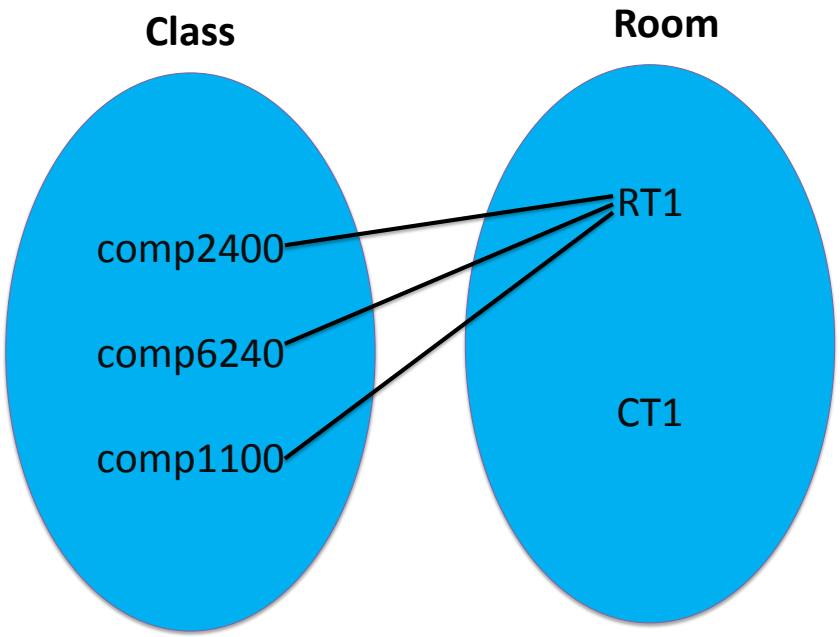
- $R_1 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, CT1)\}$



Class	Room
comp2400	RT1
comp6240	RT1
comp1100	CT1

Relations – Examples

- $R_2 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, RT1)\}$



Class	Room
comp2400	RT1
comp6240	RT1
comp1100	RT1

Relations – Examples

- Let *Class* and *Room* be two sets:
  - $Class = \{comp2400, comp6240, comp1100\}$
  - $Room = \{RT1, CT1\}$
- $Class \times Room = \{(c, r) | c \in Class, r \in Room\}$ = $\{(comp2400, RT1), (comp2400, CT1), (comp6240, RT1), (comp6240, CT1), (comp1100, RT1), (comp1100, CT1)\}$
- $R_1 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, CT1)\}$
- $R_2 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, RT1)\}$
- **What is the relationship of R_1 and R_2 with $Class \times Room$?**

Relations – Examples

- Let *Class* and *Room* be two sets:
  - $Class = \{comp2400, comp6240, comp1100\}$
  - $Room = \{RT1, CT1\}$
- $Class \times Room = \{(c, r) | c \in Class, r \in Room\}$ = $\{(comp2400, RT1), (comp2400, CT1), (comp6240, RT1), (comp6240, CT1), (comp1100, RT1), (comp1100, CT1)\}$
- $R_1 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, CT1)\}$
- $R_2 = \{(comp2400, RT1), (comp6240, RT1), (comp1100, RT1)\}$
- **What is the relationship of R_1 and R_2 with $Class \times Room$?**

  Answer: R_1, R_2 are the subsets of $Class \times Room$.
  R_1, R_2 and $Class \times Room$ are all sets of tuples.

(2) Relation/Table, Relation Schema, Relation Database Schema and Relation 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

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?**

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.

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 important but the order of the attributes in a relation is not important?**

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 important but the order of the attributes in a relation is not important?**
  No.

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

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?**

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

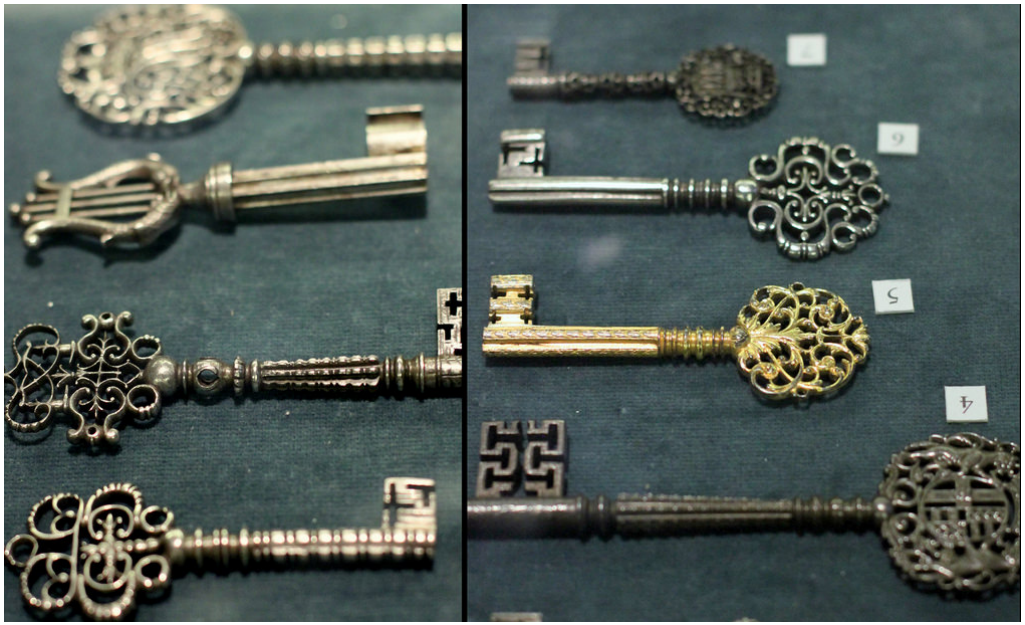
STUDENT			
StudentID	Name	DoB	Email

  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.

(2) Superkey, Candidate key, Primary key and Foreign key



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

A Bunch of Keys

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 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.

A Bunch of Keys

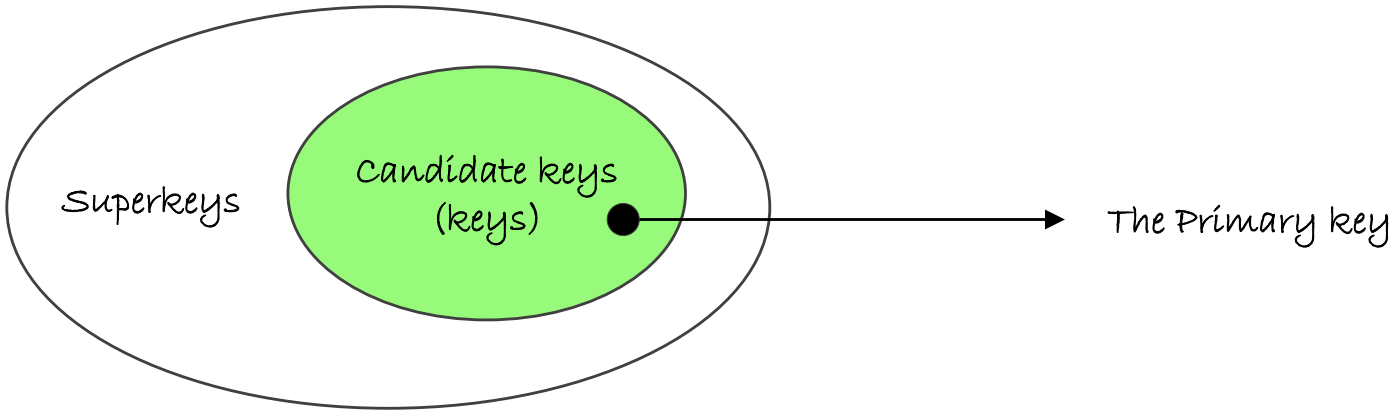
- A *subset of the attributes* of a relation schema R is a **superkey** if it uniquely identifies any tuple in $r(R)$.
- A superkey K is called a **candidate key** if no proper subset of K is a superkey. That is, if you take any of the attributes out of K, then it is not enough to uniquely identify tuples.

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.

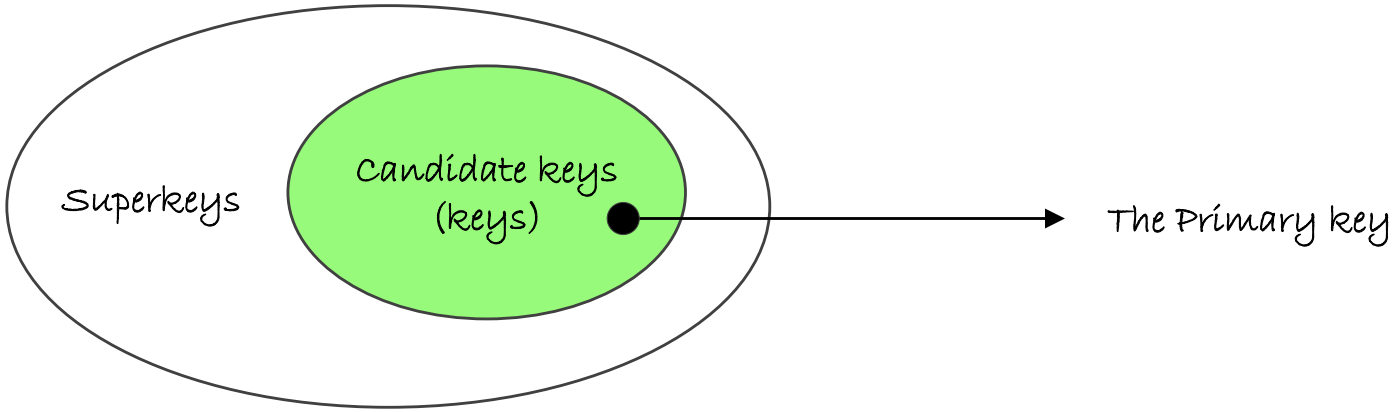
A Bunch of Keys

- 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.



A Bunch of Keys

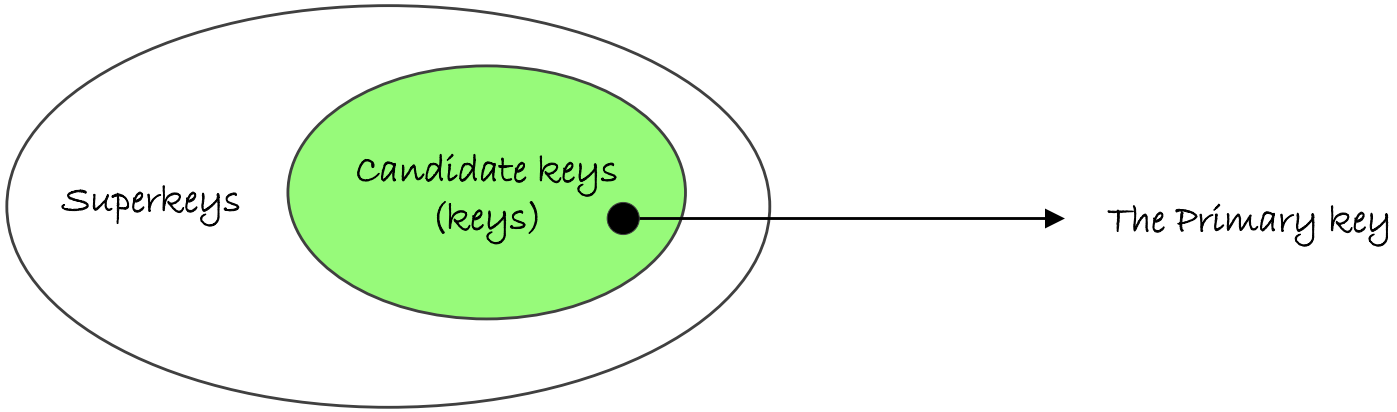
- A *subset of the attributes* of a relation schema R is a **superkey** if it uniquely identifies any tuple in $r(R)$.
- A superkey K is called a **candidate key** if no proper subset of K is a superkey. That is, if you take any of the attributes out of K, then it is not enough to uniquely identify tuples.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

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.**

Superkey – Example

- 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
...	...	...

Superkey – Example

- 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
...	...	...	...

Superkey, Candidate key and Primary key – Example

- {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
...	...	...	...

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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}, ...

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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.

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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}.

Superkey – Example

- 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
...	...	...	...	...

Superkey, Candidate key and Primary key – Example

- {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
...	...	...	...	...

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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}, ...

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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.

Superkey, Candidate key and Primary key – Example

- {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?

Superkey, Candidate key and Primary key – Example

- {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}.

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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:**

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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?.

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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}).

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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.

Superkey, Candidate key and Primary key – 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?.

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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).

Superkey, Candidate key and Primary key – 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?

Superkey, Candidate key and Primary key – 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)**.

Superkey, Candidate key and Primary key – 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)**.

Superkey, Candidate key and Primary key – 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 a candidate key.

Superkey, Candidate key and Primary key – 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 a candidate key.
- Now {guestNo, date} is a minimal SK and hence a candidate key.

Superkey, Candidate key and Primary key – 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 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.

Superkey, Candidate key and Primary key – Exercise

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

Superkey, Candidate key and Primary key – Exercise

- 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?

Superkey, Candidate key and Primary key – Exercise

- 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).

Superkey, Candidate key and Primary key – Exercise

- 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?

Superkey, Candidate key and Primary key – Exercise

- 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.

Superkey, Candidate key and Primary key – Exercise

- 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.

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

Superkey, Candidate key and Primary key – Exercise

- 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.

- 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.

Superkey, Candidate key and Primary key – Exercise

- 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?

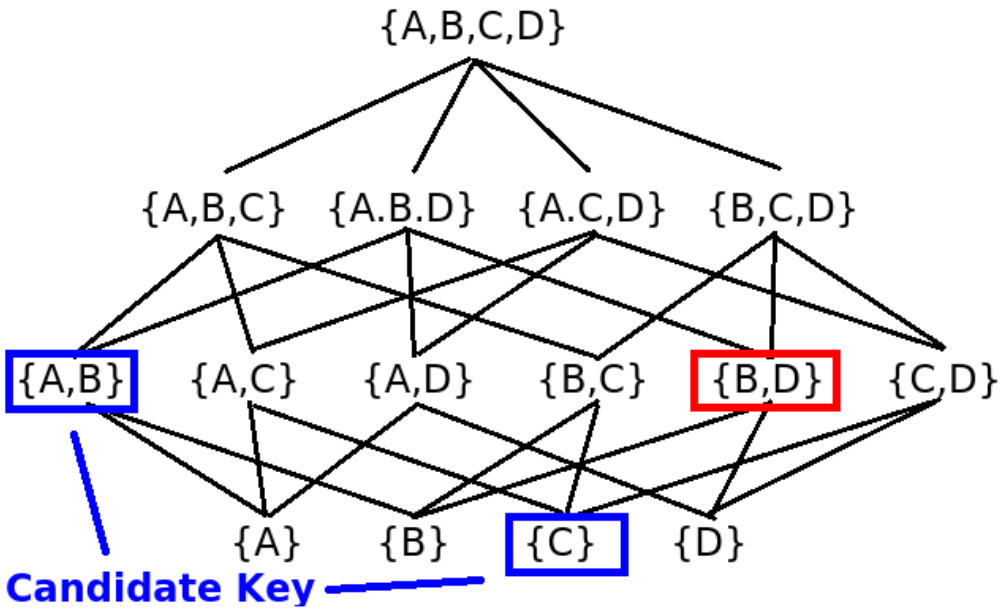
  Answer: {B, C} must be a SK because {C} is a candidate key.

- 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.

Superkey, Candidate key and Primary key – Exercise

- 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.

(4) Domain constraints, Key constraints, Entity integrity constraints and Referential integrity constraints.

Domain constraints, Key constraints and Entity integrity constraints

- **Domain constraints**: every value in a tuple must be from the **domain of its attribute**.
  e.g., INT, VARCHAR, DATE, NOT NULL, etc.

Domain constraints, Key constraints and Entity integrity constraints

- **Domain constraints**: every value in a tuple must be from the **domain of its attribute**.
  e.g., INT, VARCHAR, DATE, NOT NULL, etc.

- **Key constraints**: a bunch of keys (superkey, candidate key and primary key).

Domain constraints, Key constraints and Entity integrity constraints

- **Domain constraints**: every value in a tuple must be from the **domain of its attribute**.
  e.g., INT, VARCHAR, DATE, NOT NULL, etc.

- **Key constraints**: a bunch of keys (superkey, candidate key and primary key).

- **Entity integrity constraints**: no primary key value can be NULL.

Domain constraints, Key constraints and Entity integrity constraints

- **Domain constraints**: every value in a tuple must be from the **domain of its attribute**.
  e.g., INT, VARCHAR, DATE, NOT NULL, etc.

- **Key constraints**: a bunch of keys (superkey, candidate key and primary key).

- **Entity integrity constraints**: no primary key value can be NULL.

Referential integrity constraints – Example

- 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).

Referential integrity constraints – Example

- 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];

Referential integrity constraints – Example

- 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].

Referential integrity constraints – Example

- 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].

Foreign Key (referential integrity) – Example

- ROOM: [hotelNo] $\subseteq$ HOTEL[hotelNo];

Foreign Key (referential integrity) – Example

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

Foreign Key (referential integrity) – Example

- 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?

Foreign Key (referential integrity) – Example

- 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.

Foreign Key (referential integrity) – Example

- 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?

Foreign Key (referential integrity) – Example

- 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.

Foreign Key (referential integrity) – Example

- 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?

Foreign Key (referential integrity) – Example

- 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.

Foreign Key (referential integrity) – Example

- 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.

Foreign Key (referential integrity) – Example

- 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

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

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

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

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

Yes.

Foreign Key (referential integrity) – 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
458	COMP3600	2016 S2	active	11/06/2016

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

Foreign Key (referential integrity) – 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
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.

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we delete the first tuple in STUDENT?

Foreign Key (referential integrity) – 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
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]

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we delete the first tuple in ENROL?

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

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

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

Question: Can we update COMP2400 to be COMP6240 in COURSE?

Foreign Key (referential integrity) – 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
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].

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

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

Foreign Key (referential integrity) – 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
458	COMP2400	2016 S2	active	11/06/2016

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

Yes.

Foreign Key (referential integrity) – 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
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?

Foreign Key (referential integrity) – 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
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.

Foreign Key (referential integrity) – 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

Foreign Key (referential integrity) – 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]$\subseteq$ROOM[roomNo, hotelName]

Foreign Key (referential integrity) – 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] **?**

Foreign Key (referential integrity) – 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

Foreign Key (referential integrity) – 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]

Foreign Key (referential integrity) – 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]

(5) SQL: Data Definition Language
(*v.s.* Relation Schema + Integrity Constraints)

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

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

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);
```

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

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

- Which of the following **CREATE TABLE** statements is/are 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

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.

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);`

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,);
   ```

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);
```

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);
```

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
```

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

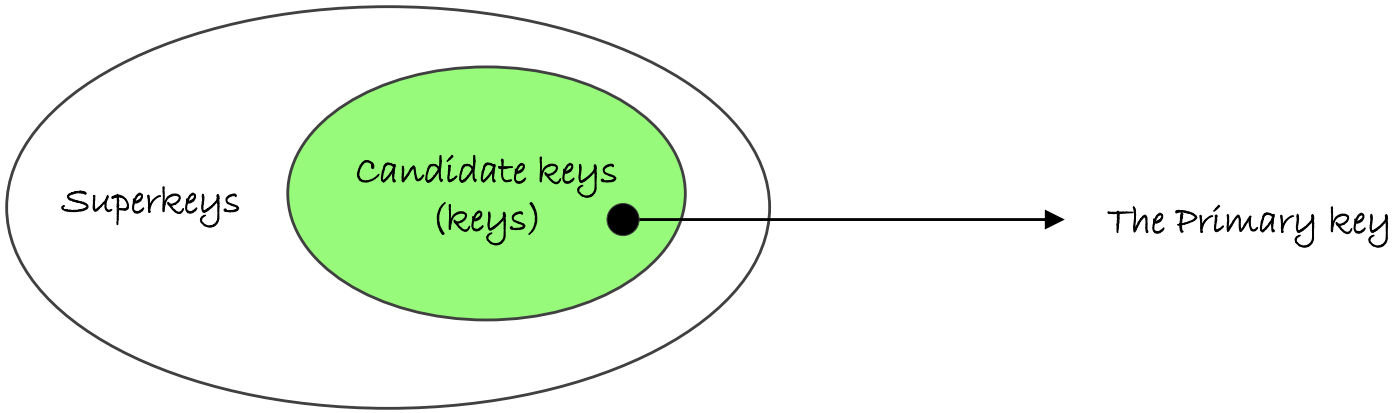
ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate

```
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 – Key Constraints

STUDENT			
StudentID	Name	DoB	**Email**

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

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));
```

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.

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));
```

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?

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));
```

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));
```

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?

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?

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.

Create Index (optional reading, will not be accessed)

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

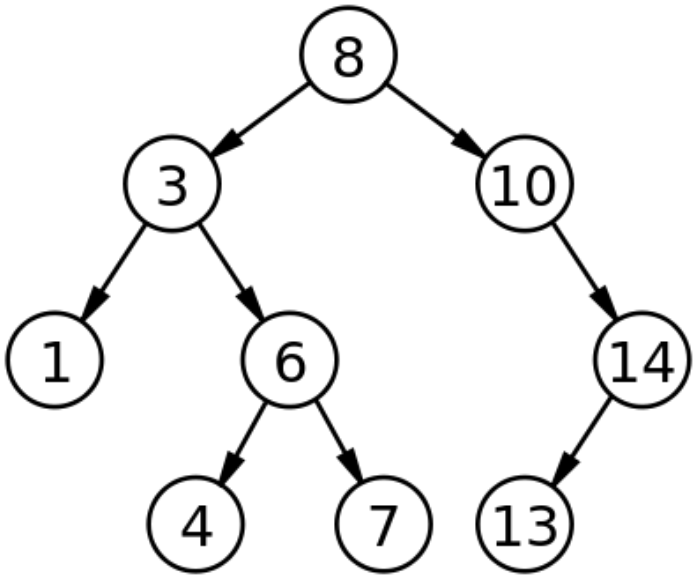
StudentID	CourseNo	Semester
111	BUSN2011	S2 2020
111	COMP2400	S2 2020
111	ECON2102	S2 2019
222	BUSN2011	S2 2020
222	COMP2400	S2 2020
333	BUSN2011	S2 2020
333	COMP2400	S2 2020
333	ECON2102	S2 2020

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 accessed)

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?

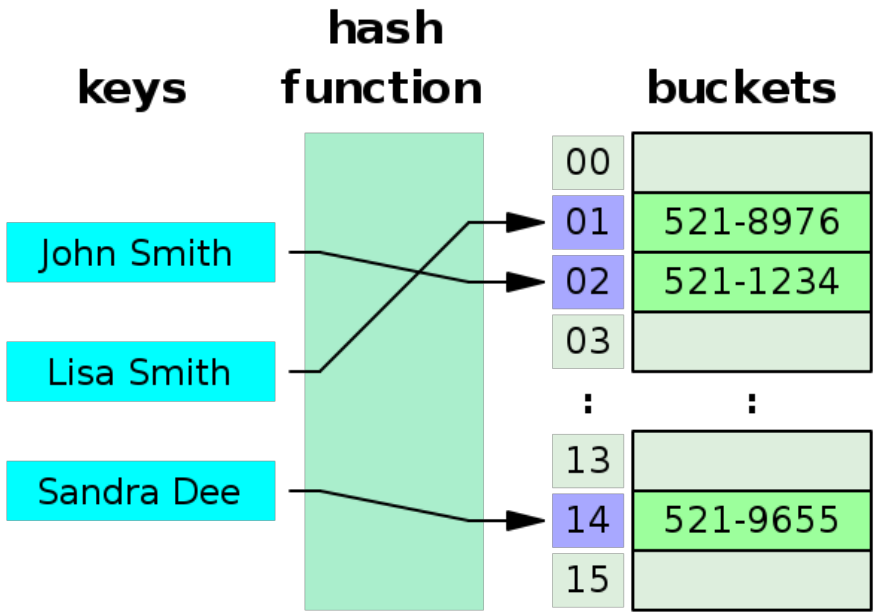


`https://en.wikipedia.org/wiki/Binary_search_tree`

Create Index (optional reading, will not be accessed)

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

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

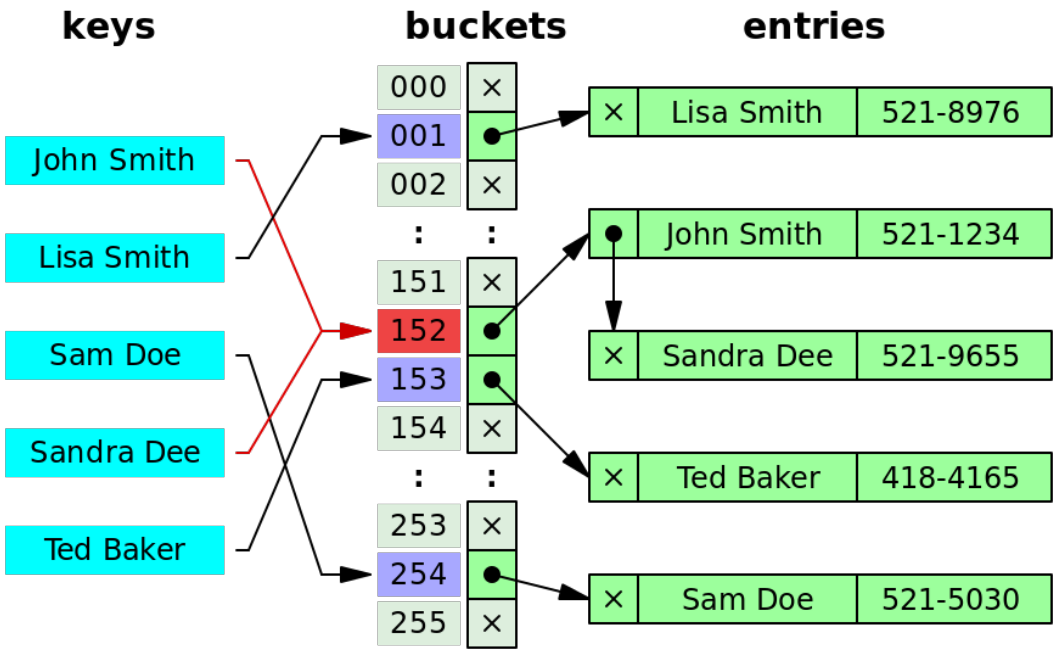


https://en.wikipedia.org/wiki/Hash_table

Create Index (optional reading, will not be accessed)

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

(credit cookie) René Descartes and the Cartesian Product

https://en.wikipedia.org/wiki/Ren%C3%A9_Descartes

René Descartes

René Descartes (Renatus Cartesius, 1596–1650) was a French

René Descartes

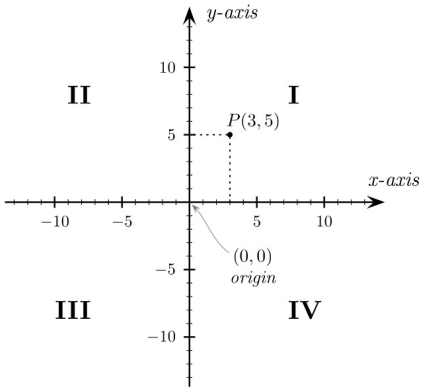
René Descartes (Renatus Cartesius, 1596–1650) was a French

- **Philosopher**: Cogito Ergo Sum (“I think, therefore I am”)

René Descartes

René Descartes (Renatus Cartesius, 1596–1650) was a French

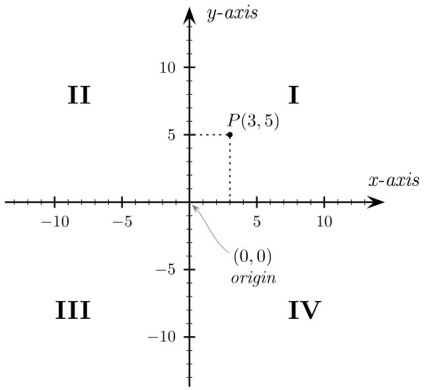
- **Philosopher**: Cogito Ergo Sum (“I think, therefore I am”)
- **Mathematician**: Cartesian coordinate system (Cartesian Product?)



René Descartes

René Descartes (Renatus Cartesius, 1596–1650) was a French

- **Philosopher**: Cogito Ergo Sum (“I think, therefore I am”)
- **Mathematician**: Cartesian coordinate system (Cartesian Product?)



- **Scientist**: “contact” lenses