CS 5530

Database Systems
Spring 2020

Relational Model

Keys

Library

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379
1005	978-0394823379
1006	978-0062278791

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Library

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

Phones

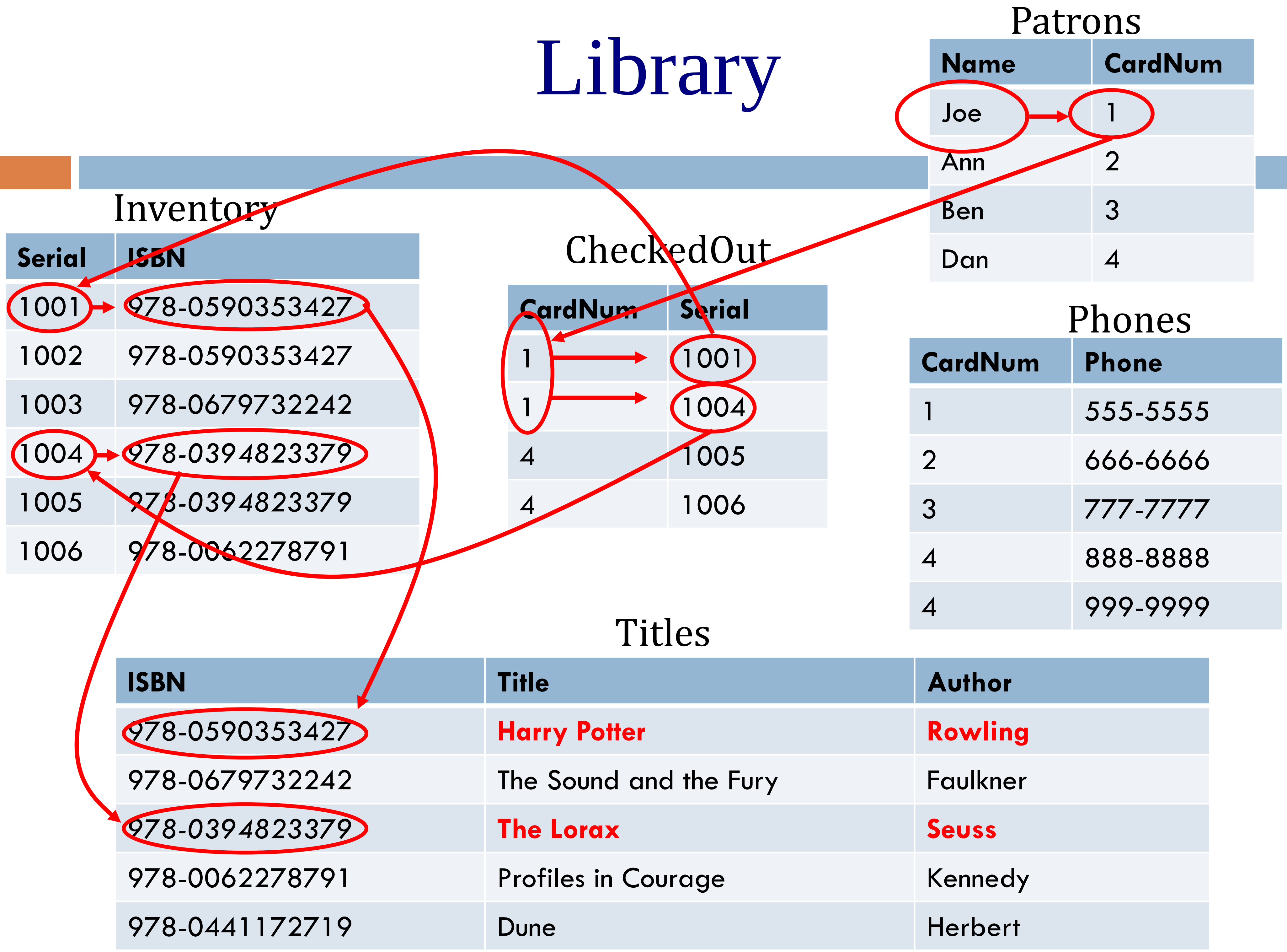
CardNum	Phone
1	555-5555
2	666-6666
4	999-9999

How do we find all the books checked out by Joe?

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Library



Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379
1005	978-0394823379
1006	978-0062278791

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

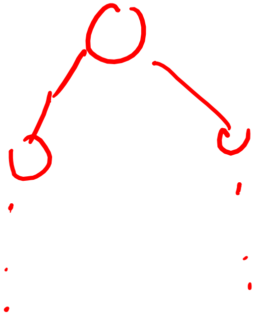
Phones

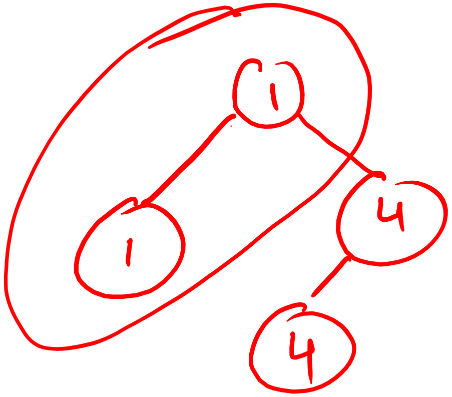
CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

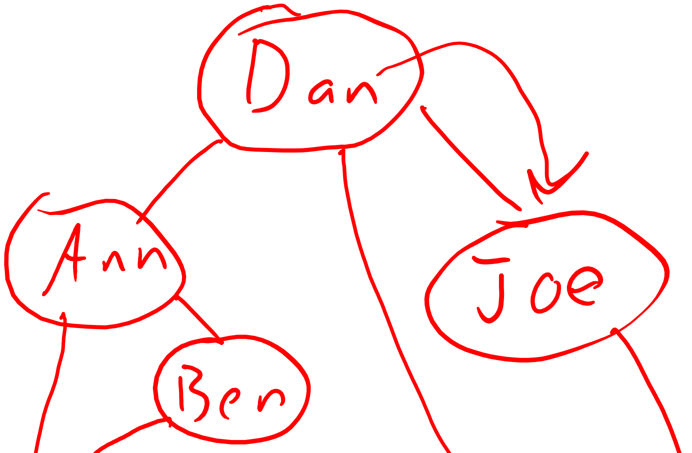
Titles

ISBN	Title	Author
978-0590353427	**Harry Potter**	**Rowling**
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	**The Lorax**	**Seuss**
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Don’t Worry – It’s Fast







Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379
1005	978-0394823379
1006	978-0062278791

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Definitions

- **Attribute**: a name and a type (column heading)

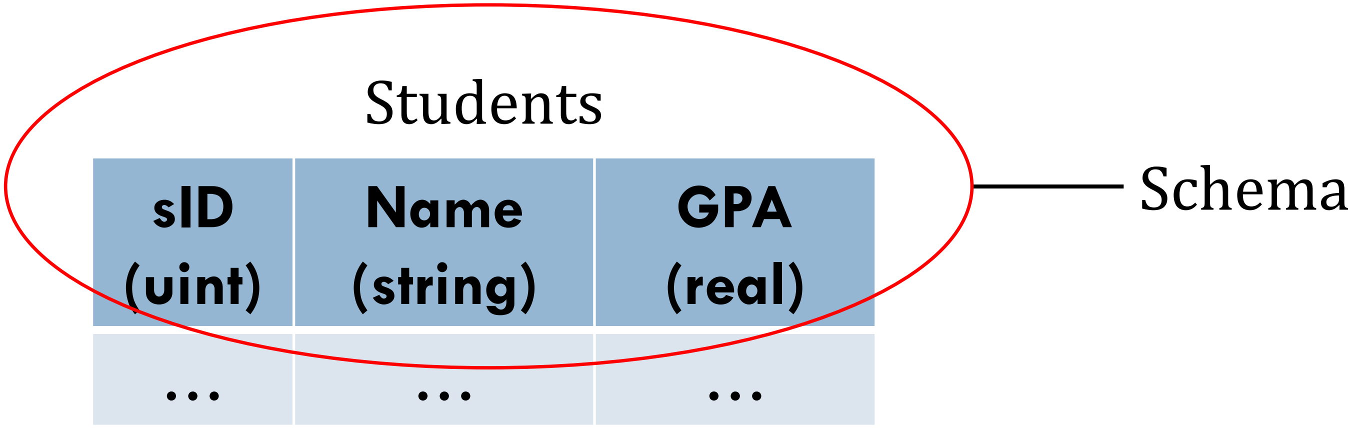
Students

Attribute

sID (uint)	Name (string)	GPA (real)
…	…	…

Definitions

- **Schema**: Table name + a set of attributes
  - Specifies the structure/rules of a table



Students

sID (uint)	Name (string)	GPA (real)
…	…	…

- A schema does *not* specify any values

Definitions

- **Instance**: the values in a table
    - A set of *tuples*

Students

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Instance

Definitions

- **Instance**: the values in a table
  - A set of tuples

- **Tuple**: one row

Students

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Tuple

Definitions

- **Relation**: a.k.a “table”
  - Schema + instance

Students

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Definitions

- **Relation**: a.k.a "table"
  - A schema + instance

Relation1

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Relation2

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0

Keys

- Keys uniquely identify each tuple
  - Critical for the DBMS’ underlying operations
  - Defines the meaning of the relation

key

sID (uint)	Name (string)	GPA (real)
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

key

Make	Model	…
Toyota	F1	
Toyota	Camry	
Subaru	WRX	
Subaru	F1	

Keys

- As a DB designer, you will define keys for each table
- …but we need formal definitions before SQL

Definitions

- **Superkey**:

- A set of attributes is a *superkey* if no two rows are allowed to have the same values in those columns (no duplicates)

A_1	A_2	A_3
x	4	q
y	4	p
x	3	x

Subsets	SK?
A_1	NO
A_2	NO
A_3	?
$A_1 A_2$	?
A_1, A_3	↓
A_2, A_3	
A_1, A_2, A_3	yes
$\{\}$	

Enrolled

sI	cID	Grade

{sID, cID}

{sID, cID, Grade}

Students

sID	Name	GPA

{sID}

{sID, ..., ...}

Definitions

- **Key**:

- A set of fields is a *key* if:
  - it is a superkey
  - no proper subset of its fields is a superkey

Definitions

- **Key**:

- A set of fields is a *key* if:
  - it is a superkey
  - no proper subset of its fields is a superkey

- (The empty set is not usually considered a superkey)

Keys

- What are the possible key(s)?

$\{A_1, A_2\}$

$\{A_1\}$

$\{A_2\}$

A_1	A_2	A_3
x	4	q
y	4	p
x	3	x

Attribute Set	SK	Key
$\{A_1\}$	No	→
$\{A_2\}$	No	→
$\{A_3\}$	Yes	Yes
$\{A_1\ A_2\}$	Yes	yes
$\{A_1\ A_3\}$	Yes	No
$\{A_2\ A_3\}$	Yes	↓
$\{A_1\ A_2\ A_3\}$	Yes	

Keys

- What are the possible key(s)?

A_1	A_2	A_3
x	4	q
y	4	p
x	3	x

Attribute Set	SK	Key
$\{A_1\}$	No	No
$\{A_2\}$	No	No
$\{A_3\}$	Yes	Yes
$\{A_1\ A_2\}$	Yes	Yes
$\{A_1\ A_3\}$	Yes	No
$\{A_2\ A_3\}$	Yes	No
$\{A_1\ A_2\ A_3\}$	Yes	No

Keys

- DB design works the other way around
  - Start with keys, not data
  - Keys define what data is valid

Keys

- DB designer's job: problem description → schemas

Keys

- What should be the key for the Enrolled schema?

Enrolled

sID	cID	Grade

- sID = student ID

- cID = course ID

- No instance data given – use human understanding of relation

Keys

- What should be the key for the Enrolled schema?

Enrolled

sID	cID	Grade

Keys

- What is the key for the Enrolled schema?
  - {sID, cID}

classes

cID Name when

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Enrolled

sID	cID	Grade
1	CS3500	B
2	CS3500	A+
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+

Primary/Candidate Key

- DBA specifies one key as the **primary key**

- If a schema has more than one key, they are called **candidate keys**

Candidate Keys

- We can specify additional uniqueness constraints (candidate keys)

Enrolled

sID	cID	Grade

1 X A

```
PK(sID, cID)
UQ(sID, Grade)
```

Candidate Keys

- We can specify additional uniqueness constraints (candidate keys)

Enrolled

sID	cID	Grade

```
PK(sID, cID)
UQ(sID, Grade)
```

- "A student can not take a course twice, and can not receive the same grade twice"

Exercise

- Provide the primary key (PK) plus any additional candidate keys (UQ) for the `Enrolled` table:

sID	cID	Grade

1. “A student can not take the same course twice, and no two students can receive the same grade in the same course”

2. “A student can earn multiple grades in a class”

3. “A student can only take one course and receive a single grade for that course, and courses have maximum enrollment of one”

Exercise

- "A student can not take the same course twice, and no two students can receive the same grade in the same course"

PK(sID, cID)

UQ(cID, Grade)

sID	cID	Grade
1	x	A
2	x	A

Exercise

- “A student can retake a course for a different grade”

pk(sID, cID, Grade)

sID	cID	Grade

Exercise

- "A student can only take one course and receive a single grade for that course, and courses have a maximum enrollment of one"

pk(sID)
uq(cID)

pk(sID, cID)

sID	cID	Grade

Primary vs. Candidate Keys

```
PK(sID, cID)                  UQ(sID, cID)
```

Or

```
UQ(sID, Grade)                PK(sID, Grade)
```

- “A student can not take a course twice, and can not receive the same grade in different courses”

Primary vs. Candidate Keys

```
PK(sID, cID)
UQ(sID, Grade)
```

Or

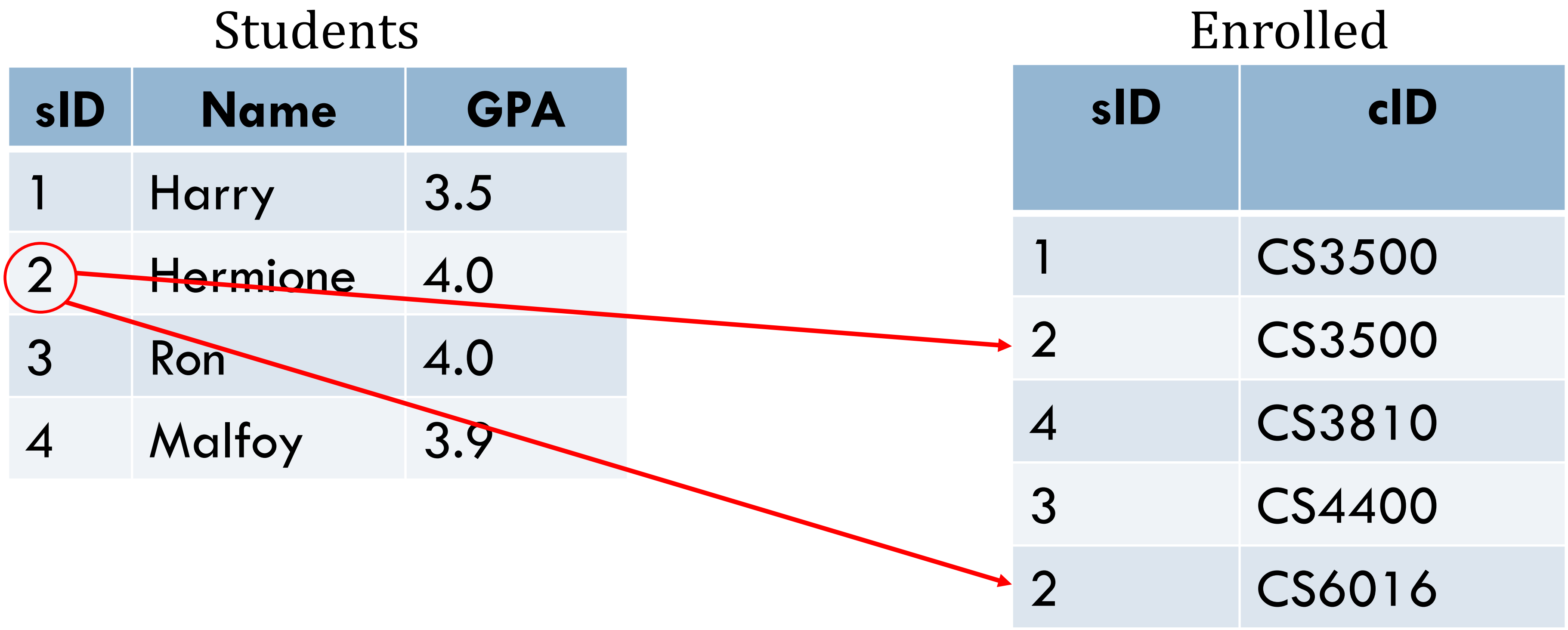
```
UQ(sID, cID)
PK(sID, Grade)
```

- “A student can not take a course twice, and can not receive the same grade in different courses”

- PK should be small, ideally integer type(s)

Keys

- Keys relate tuples from different tables



Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Enrolled

sID	cID
1	CS3500
2	CS3500
4	CS3810
3	CS4400
2	CS6016

Quiz

- What happens?

```
Thing* t1 = new Thing();
Thing* t2 = t1;
delete(t1);
cout << t2->x;
```

a) `t2->`~~`value`~~x is printed
[✓ b) crash]
c) compiler error
[✓ d) no way to know what happens]

Quiz

- What happens?

```
Thing* t1 = new Thing();
Thing* t2 = t1;
delete(t1);
cout << t2->x;
```

a) `t2->value` is printed
b) crash
c) compiler error
d) no way to know what happens

Quiz

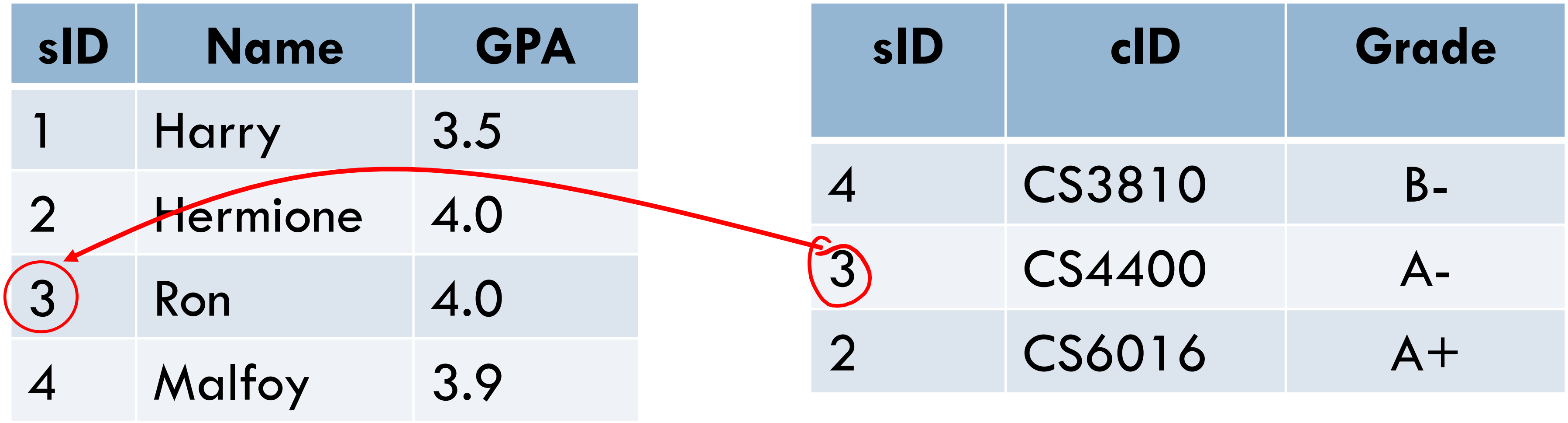
- What happens?

```
Thing* t1 = new Thing();
Thing* t2 = t1;
delete(t1);
cout << t2->x;
```

a) `t2->value` is printed
b) crash - what we *want* to happen (with useful message)
c) compiler error
d) no way to know what happens

Ponder…

- Pointer analogy



Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+

Ponder…

- What happens if I delete the student “Ron”?

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+

Ponder…

- What happens if I delete the student "Ron"?

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+

meaningless

Referential Integrity

- **Referential Integrity**:
  - References between values should always be meaningful
  - This is an *invariant* – should hold true at all times

Referential Integrity

- **Referential Integrity**:
  - References between values should always be meaningful
  - This is an *invariant* – should hold true at all times
- C++ lets us violate this
- Java enforces it by taking away control

Referential Integrity

- Option 1:
  - Whenever we remove a record from **`Students`**, run another command to update all **`Enrolled`** with that sID

Referential Integrity

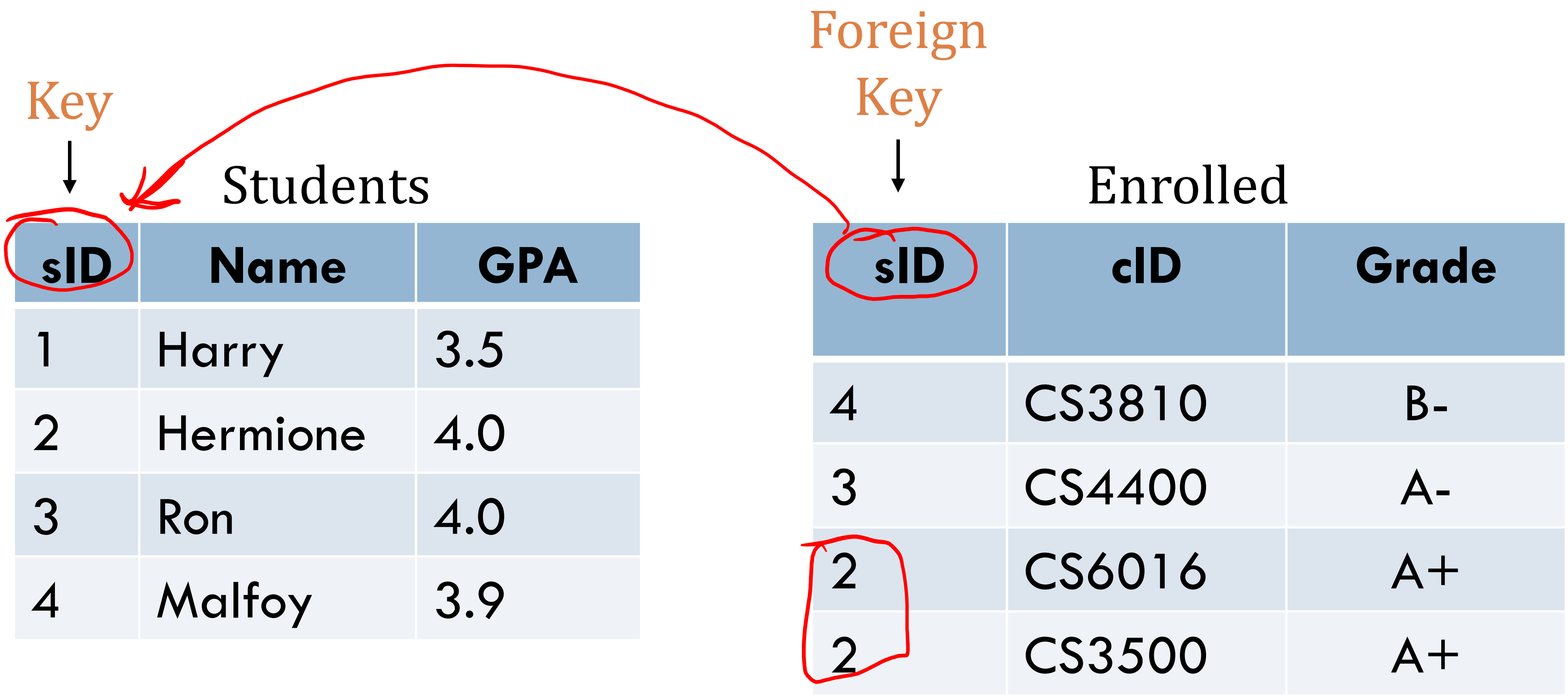
- Option 1:
  - Whenever we remove a record from **`Students`**, run another command to update all **`Enrolled`** with that sID
  - Invariant briefly broken
  - What if more tables reference **`Students`**?

Referential Integrity

- Option 2:
  - Let the DBMS update **`Enrolled`** automatically
  - SQL provides some support

Foreign Key

- **Foreign Key**:
  - Attribute whose values are a key in another table
  - Think of it as a “pointer”



Key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Foreign Key

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+
2	CS3500	A+

Foreign Key

- **Foreign Key**:
  - Not necessarily unique itself

Key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
3	Ron	4.0
4	Malfoy	3.9

Foreign Key

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+
2	CS3500	A+

SQL (ish)

```
CREATE TABLE Enrolled(
  sID int,
  cID char(6),
  grade char(2),
  PRIMARY KEY (sID, cID),
  FOREIGN KEY (sID) REFERENCES Students(sID))
```

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	3.5
3	Ron	4.0
4	Malfoy	3.9

Enrolled

sID	cID	Grade
1	CS3500	A
2	CS3500	B
4	CS3810	B-
3	CS4400	A-

Foreign Key Constraint

- If the *referenced* key is modified, what should we do in the *referencing* table?

- SQL gives a few options

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A+

Delete Record

- 1. Delete corresponding record(s)

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
~~Dan~~	~~4~~

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
~~4~~	~~888-8888~~
~~4~~	~~999-9999~~

Delete Record

- 1. Delete corresponding record(s)

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
~~Dan~~	~~4~~

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
~~4~~	~~888-8888~~
~~4~~	~~999-9999~~

- If the referencing record has no meaning on its own

Nullify Key

- 2. Nullify foreign key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
NULL	CS4400	A-
2	CS6016	A+

Nullify Key

- 2. Nullify foreign key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
NULL	CS4400	A-
2	CS6016	A+

- If we want to keep the data, but “unlink” it
  - We can still analyze, e.g. average historic GPA

Nullify Key

- 2. Nullify foreign key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
NULL	CS4400	A-
2	CS6016	A+

- But! – This is bad design
  - If `{sID, cID}` is primary key for `Enrolled`

Nullify Key

- 2. Nullify foreign key

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	4.0
~~3~~	~~Ron~~	~~4.0~~
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
NULL	CS4400	A-
2	CS6016	A+

- But! – This is bad design
  - If `{sID, cID}` is primary key for `Enrolled`
- Add a unique ID to `Enrolled` table

Disallow

- 3. Disallow changes to referenced table
  - Try to delete Joe – SQL reports error

Patrons

Name	CardNum
~~Joe~~	~~1~~
Ann	2
Ben	3
Dan	4
steve	5

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Disallow

- 3. Disallow changes to referenced table
  - Try to delete Joe – SQL reports error

Patrons

Name	CardNum
~~Joe~~	~~1~~
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

- If we need to take some action first
  - Contact Joe, get his books back

Foreign Keys

- What if we try to enroll a non-existent student?
  - SQL will reject it

Students

sID	Name	GPA
1	Harry	3.5
2	Hermione	3.5
3	Ron	4.0
4	Malfoy	3.9

Enrolled

sID	cID	Grade
4	CS3810	B-
3	CS4400	A-
2	CS6016	A
5 (?)	CS4150	E