

Database Systems (資料庫系統)

11/28/2022

Database #1

Introduction to Computer (計算機概論)

Winston Hsu, NTU

1

Course Goals

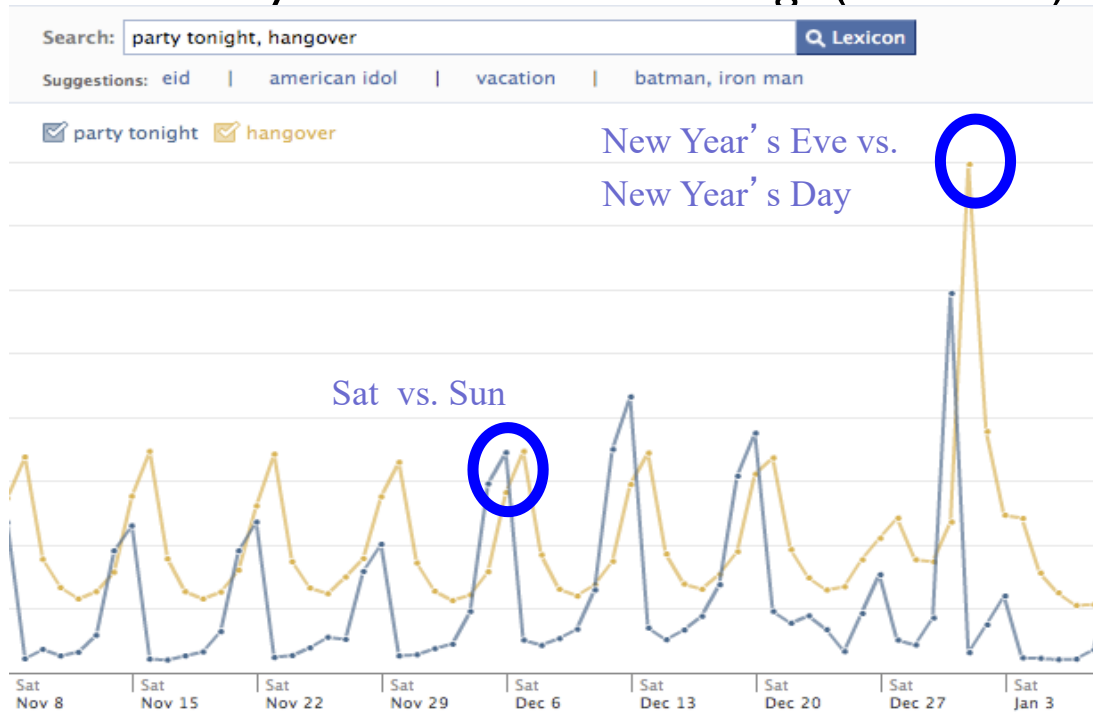
- A very preliminary course in database systems.
- Learning objective
 - **Use** a relational database
 - Build a relational database
 - Advanced issues – data analytics

Supplemental database software

- Commercial: Microsoft SQL Server
- Open-source: PostgreSQL, MySQL

2

Data Warehouse Example: “Party” vs. “Hangover” over Perabytes of Unstructured Logs (Facebook)



3

Outline

- Why do we need a DBMS (Database Management System)?
- What can a DBMS do for an application?
- Why study database systems?
- Data Models: Overview of a Relational Model
- Levels of Abstraction in a DBMS
- Sample Queries in DBMS
- Structure of a DBMS
- Large-scale Database (*)

4

Why DBMS?

- Suppose that you want to build an university database. It must store the following information:
 - **Entities:** Students, Professors, Classes, Classrooms
 - **Relationships:** Who teaches what? Who teaches where? Who teaches whom?

```
SELECT S.name  
FROM Students S  
WHERE S.sid = 123456
```

5

Database Management System (DBMS)

- DBMS is software to store and manage data, so applications don't have to worry about them.
- What can a DBMS do for applications?
 - Can you think of them?

6

What can DBMS do for applications?

- Store huge amount of data (e.g., TB+) over a long period of time
- Allow apps to query and update data
 - Query: what is Mary's grade in the "Operating System" course?
 - Update: enroll Mary in the "Database" course
- Protect from unauthorized access.
 - Students cannot change their course grades.
- Protect from system crashes
 - When some system components fail (hard drive, network, etc.), database can be restored to a good state.

7

More on what can DBMS do for applications?

- Protect from incorrect inputs
 - Mary has registered for 100 courses
- Support concurrent access from multiple users
 - 1000 students using the registration system at the same time
- Allow administrators to easily change **data schema**
 - At a later time, add TA info to courses.
- Efficient database operations
 - Search for students with 5 highest GPAs

8

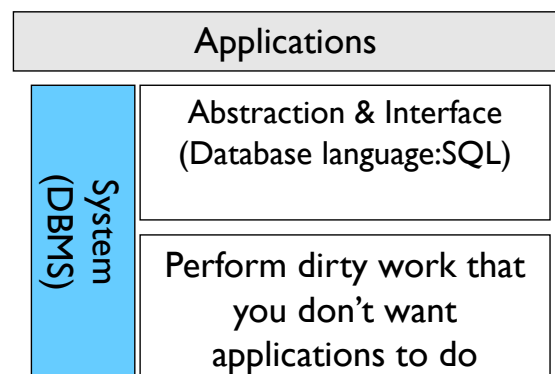
Alternative to Using a DBMS

- Store data as files in operating systems.
- Applications have to deal with the following issues:
 - 32-bit addressing (4GB) is insufficient to address 100GB+ data file
 - Write special code to support **different queries**
 - Write special code to protect data from **concurrent access**
 - Write special code to protect against **system crashes**
 - Optimize applications for **efficient access and query**
 - May often rewrite applications
- Easier to buy a DBMS to handle these issues

9

What can a DBMS do for applications?

- Define data: Data Definition Language (DDL)
- Access and operate on data: Data Manipulation Language (DML)
 - Query language
- Storage management
- Transaction Management
 - Concurrency control
 - Crash recovery
- Provide good security, efficiency, and scalability



10

Why Study Database Systems?

- They are everywhere.
 - Online stores, real stores
 - Banks, credit card companies
 - Passport control
 - Police (criminal records)
 - Airlines and hotels (reservations)
- DBMS vendors & products
 - Oracle, Microsoft (Access and SQL server), IBM (DB2), Sybase, ...
- Big Data Analytics (nowadays)
- Emerging new database (e.g., Graph)

11

Data Models

- A **data model** is a collection of concepts for describing data.
 - Entity-relation (ER) model
 - Relational model (main focus of this course)
- A **schema** is a description of data.
- The **relational model** is the most widely used data model.
 - A **relation** is basically a table with rows and columns of **records**.
 - Every relation has a **schema**, which describes the columns, or fields.

Students (Sid: String, Name: String, Login: String, age: Integer, ...)

12

Relational Model

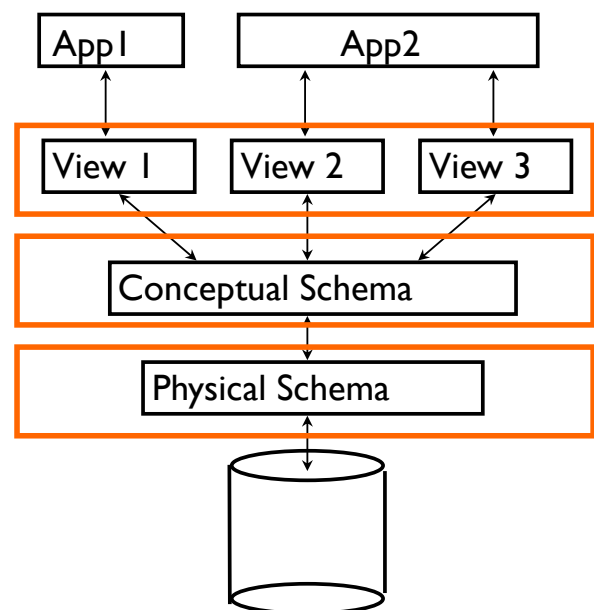
- The entire table shows an instance of the Students relation.
- The Students schema is the column heads
 - Students(Sid: String, Name: String, Login: String, age: Integer,...)

sid	name	email	age	gpa
53666	Jones	Jones@cs	18	3.4
53688	Smith	Smith@ee	18	3.2
53650	Joe	Joe@cs	19	2.5

13

Levels of Abstractions in DBMS

- Many **views**, one **conceptual schema** and one **physical schema**.
 - Conceptual schema defines logical structure
 - Relation tables
 - Physical schema describes the file and indexing used
 - Sorted file with B+ tree index
 - Views describe how applications (users) see the data
 - Relation tables but not store explicitly



14

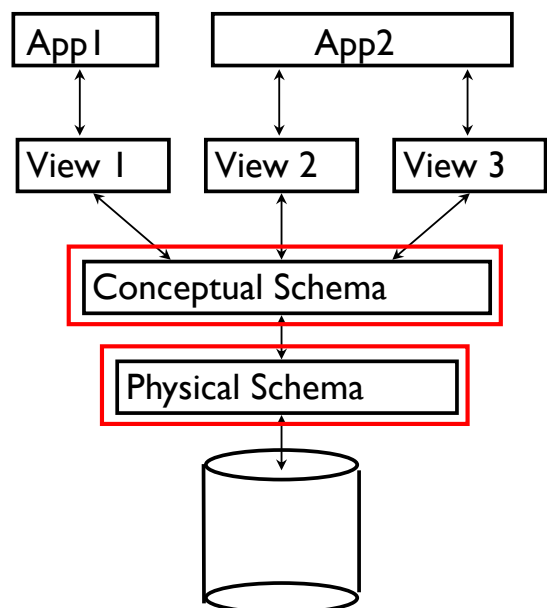
Example: University Database

- Conceptual schema:
 - Students (sid: string, name: string, login: string, age: integer, gpa:real)
 - Courses (cid: string, cname:string, credits:integer)
 - Enrolled (sid:string, cid:string, grade:string)
- Physical schema:
 - Relations stored as unordered files.
 - Index on first column of Students.
- View (External Schema):
 - Course_info(cid:string, cname: string, enrollment:integer)
 - Why?

15

Data Independence

- Three levels of abstraction provides **data independence**.
 - Changes in one layer only affect one upper layer.
 - E.g., applications are not affected by changes in conceptual & physical schema.



16

Queries in DBMS

- Sample queries on university database:
 - What is the name of the student with student ID 123456?
- The key benefits of using a relational database are
 - Easy to specify queries using a **query language**: Structured Query Language (SQL)

```
SELECT S.name  
FROM Students S  
WHERE S.sid = 123456
```

- Efficient query processor to get answer

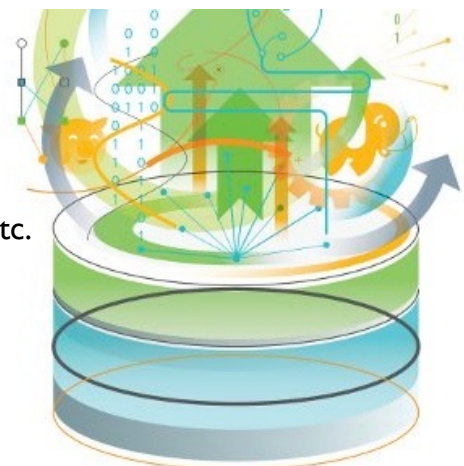
17

What Does “Big Data” Mean?

– Michael Stonebraker



- Big volume – scalability
- Big velocity – efficiency
- Big variety – diversity
- Current situation
 - Big volumes of data, but “small analytics”
 - Big analytics on big volumes of data
 - Clustering, regression, machine learning, etc.



Analysis and Indexing Solutions for Big Data are Mandatory for Future Developments

g a common vor-
the problems en-
design and identify
from among fami-
is.
at design patterns
es might be useful
not new. An exam-
; 1994 book *Design*
ined patterns use-
d programming.¹²
pattern language,
l Garlan's report,²³
ariety of architec-
r organizing soft-
ective. That these
may also be viewed
was noted earlier
is 1996 book *Pat-
e Architecture*.⁷ In
Pine-and-Filter

**If researchers
meet the
parallel challenge,
the future of IT
is rosy. If they
don't, it's not.**

circuits, and high-pe-
puting. We then focu
the patterns in the ap-
scribed earlier. Figur
sults of our pattern m
Computational an-
terns can be hierarch
to define an applica-
software architecture
pattern language for
sign must at least spe-
from high-level archite-
software implementa-
Mattson et al's 2004
Parallel Programming
such attempt to sys-
programming using a
language. We combin
and computational pa-
earlier in our pattern
ally sit on top of the

Asanoyic et al. A view of the parallel computing landscape. Communications of the ACM, 2010.

Hive – Simplifying Hadoop based on SQL

[Thusoo, Hive ApacheCon 2008]

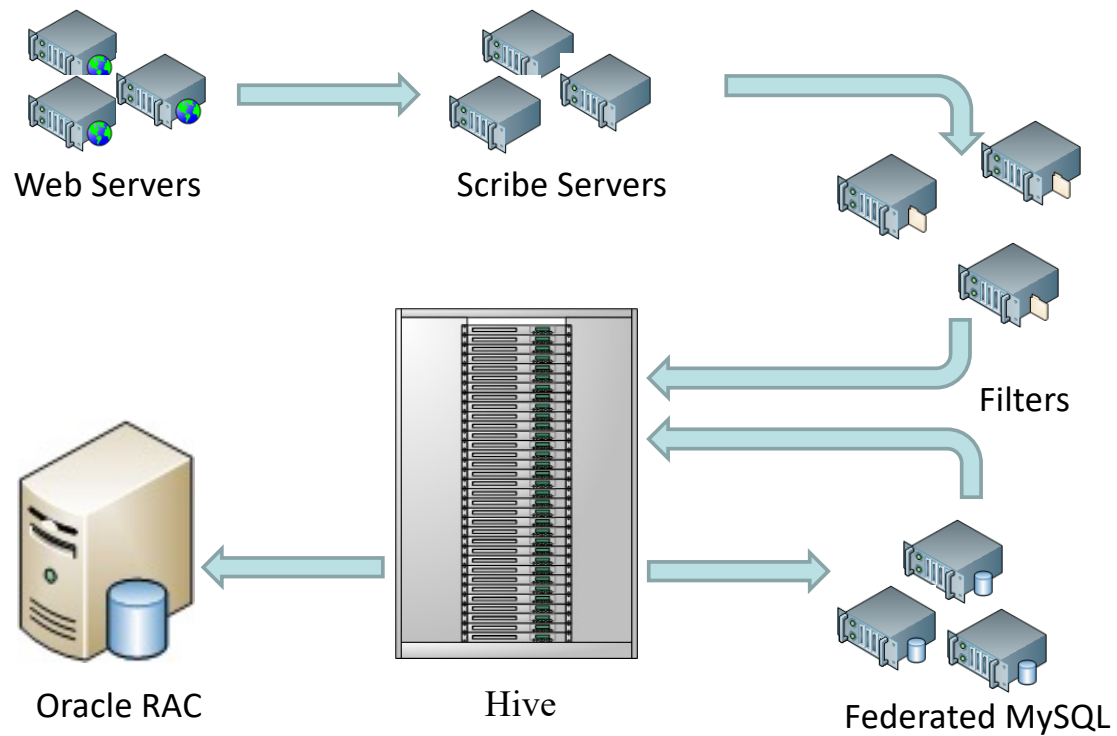
```
hive> select key, count(1) from kv1 where key > 100  
group by key;
```

vs.

```
$ cat > /tmp/reducer.sh  
uniq -c | awk '{print $2"\t"$1}'  
$ cat > /tmp/map.sh  
awk -F '\001' '{if($1 > 100) print $1}'  
$ bin/hadoop jar contrib/hadoop-0.19.2-dev-  
streaming.jar -input /user/hive/warehouse/kv1 -  
mapper map.sh -file /tmp/reducer.sh -file  
/tmp/map.sh -reducer reducer.sh -output  
/tmp/largekey -numReduceTasks 1  
$ bin/hadoop dfs -cat /tmp/largekey/part*
```

Data Warehousing at Facebook Today

[Thusoo, Hive ApacheCon 2008]



Relational Model

Chapter 3

Lecture Outline

- Quick intro to Relational Model
 - Definitions: schema, instance, tuple, field, domain, etc.
 - Basic SQL Commands (Data Definition Language)
 - Integration Constraints

23

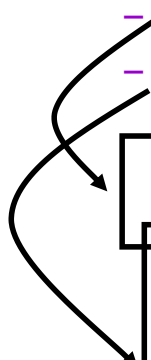
Relational Model

- Mostly widely used model
 - Vendors: Oracle, Microsoft, IBM (DB2), Sybase, ...
 - Open source: MySQL
- Simple
 - A relational database is a collection of **relations**.
 - Each relation is a **table** with rows and columns.
- Why do people like it?
 - Simple tabular data representation, easy to understand.
 - **Ease of expressing complex query (using SQL) on the data**
 - Efficient query evaluation (using query optimization)

24

Example of a Relation

- This is a “Students relation”.
- A relation has two parts:
 - **Schema** defines column heads of the table.
 - **Instance** contains the data rows (called **tuples** or **records**) of the table.



The diagram illustrates the components of a relation. A curved arrow points from the 'Schema' definition in the list to the header row of the table. Another curved arrow points from the 'Instance' definition to the data rows of the table.

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

25

Relational Schema (I)

- A relational schema specifies:
 - **name** of the relation: Students
 - names of **fields**: (sid, name, login, age, gpa)
 - **domain** of each field: it is type of possible **values** (some of built-in types in SQL are integer, real, string, and date.)
- A field can also be called an **attribute** or a **column**.

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

26

Relational Schema (2)

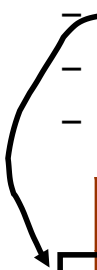
- We can refer to the field by
 - Field name (more common): the order of fields does not matter.
 - Position of the field (less common): the order of fields matters.
- A relational schema can be written using the following notation:
 - relation-name (field-name-1: domain-name-1, field-name-2: domain-name-2, ..., field-name-n: domain-name-n)
 - Example:

Students(sid: string, name: string, login: string, age: integer, gpa: real)

27

Relational Instance

- A relation instance contains a set of **tuples**
 - A relation instance is referred to as a **relation**.
 - A tuple can also be called a **record** or a **row**.
 - A relation instance is a set, and it has **no duplicate tuples**.
 - Order of tuples is **not** important.



sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

28

Degree and Cardinality

- **Degree** is the number of fields in schema (=5 in the table below)
- **Cardinality** is the number of tuples in relation (=3 in the table below)

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

29

Domain Constraints

- Values in the tuples' fields must satisfy the specified domain in the schema.
 - Similar to type matching in compilation
- Example of domain constraint violation:
 - Schema: Students(sid: string, name: string, login: string, age: integer, gpa: real)
 - A tuple: <sid: '50000', name: 38, login: 'dave@cs', age: 18.5, gpa: 3.3>
- There are other types of **integrity constraints**:
 - Key constraints, foreign key constraints, and general constraints.

30

Outline on SQL Basics

- History
- Basic commands
- Integrity constraints

31

SQL History

- It is a query language for relational databases.
- Developed by IBM (system R) in the 1970s
- Need for a standard since it is used by many database vendors.
- Two standard organizations
 - ANSI (American National Standard Institutes)
 - ISO (International Organization for Standardization)
- Standards:
 - SQL-86, SQL-89, SQL-92, SQL-99 (current standard)

32

SQL Basic Commands

- **create table:** create a table
- **drop table:** delete a table
- **alter table:** alter a field in a table
- **insert:** add a tuple
- **delete:** delete a tuple
- **update:** change field values in a tuple

33

SQL: create table

create table Students (sid **char(20)**, name **char(20)**,
login **char(10)**, age **integer**, gpa **real**)

- Use built-in types: integer, real, char(#)
 - Similar to type definition in programming language
 - You can define your own types (describe in CH5)

sid	name	login	age	gpa

34

SQL: delete table

drop table Students

35

SQL: alter table

- Add a new field in a table

alter table Students **add** dept char(20)

sid	name	login	age	gpa	dept

- Delete a field in a table

alter table Students **drop** gpa

sid	name	login	age	dept

36

SQL: insert

insert into Students (sid, name, login, age, gpa)
values ('53688', 'Smith', 'smith@cs', 18, 3.2)
or you can omit the fields

insert into Students
values ('53688', 'Smith', 'smith@cs', 18, 3.2)

sid	name	login	age	gpa
53688	Smith	Smith@cs	18	3.2

37

SQL: delete

delete from Students **as** S **where** S.name = 'Smith'

or you can omit as and the tuple variable S

delete from Students **where** name = 'Smith'

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

38

SQL: update

update Students S

set S.age = S.age + 1, S.gpa = S.gpa – 1

where S.sid = 53688

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18 19	3.2 2.2
53650	Smith	Smith@math	19	3.7

39

Integrity Constraints (IC)

- **IC**: condition that must be true for **any** instance of the database; e.g., domain constraints.
- Why ICs?
 - Prevent data entry errors or command errors.
- ICs are specified when schema is defined, (create table)
- ICs are checked when relations are modified (add, remove, and update).
- A **legal** instance of a relation is one that satisfies all specified ICs.
 - Should not allow illegal instances.

40

Types of Integrity Constraints

- Domain constraint
- Key constraint
- Foreign key constraint (referential integrity)
- Other constraints

41

Key Constraint

- A **key** is a set of **minimal fields** that can **uniquely** identify a tuple in a relation.
 1. No two distinct tuples can have same values in all key fields, and
 2. It is minimal, (no subset of the key is another key).
 - Part 2 false? A **superkey**.
 - If $\#key > 1$, one of the keys is chosen (by DBA) to be the **primary key**.
- Why is this a constraint?
 - When a table is modified (e.g., by adding a new tuple), DBMS checks to make sure that the specified keys remain valid keys (e.g., the new tuple has a unique key value).

42

Examples of Keys

- Valid keys: {sid}, {name, age} and {login}
- Invalid keys: {name} and {age}
- Valid Superkeys: {sid, gpa}, {sid, age}, and {sid, name, login, age, gpa}

sid	name	login	age	gpa
50000	Dave	Dave@cs	19	3.2
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7
53831	Madayan	Madayan@music	11	1.8

43

Primary and Candidate Keys

- A relation can have many possible keys, called **candidate keys**. But only one is chosen as the **primary key**.
 - DBMS may create an index on primary key to optimize tuple lookup (using primary key).
- Specify keys in SQL:

create table Students

(sid **char(20)**, name **char(20)**, login **char(10)**, age **integer**, gpa **real**,

unique (name, age),

constraint StudentsKey primary key (sid))

constraint name

44

Foreign Key (Definition)

- Used when one table references another table.
- A foreign key is like a “**pointer**” in C, referencing a **unique tuple** / field in the **referenced relation**.
 - A foreign key constraint makes sure that there is **no dangling pointer**.
- Huh? Take a look at an example.

45

Foreign Key Constraint

- Specify constraint:
 - Only the Students listed in the Students relation can enroll for courses.
 - [studid] in [Enrolled] **refer** to some fields [sid] in [Students]
- Why is it a constraint?
 - What happens when we delete the 1st tuple from [Students]? Invalid studid
 - A tuple in Enrolled relation becomes invalid. Why?

They are related

sid	name	login	age	gpa	cid	grade	studid
53666	Jones	Jones@cs	18	3.3	Database I 41	B	53666
53688	Smith	Smith@cs	18	3.2	Topology I 12	A	53650
53650	Smith	Smith@math	19	3.7			

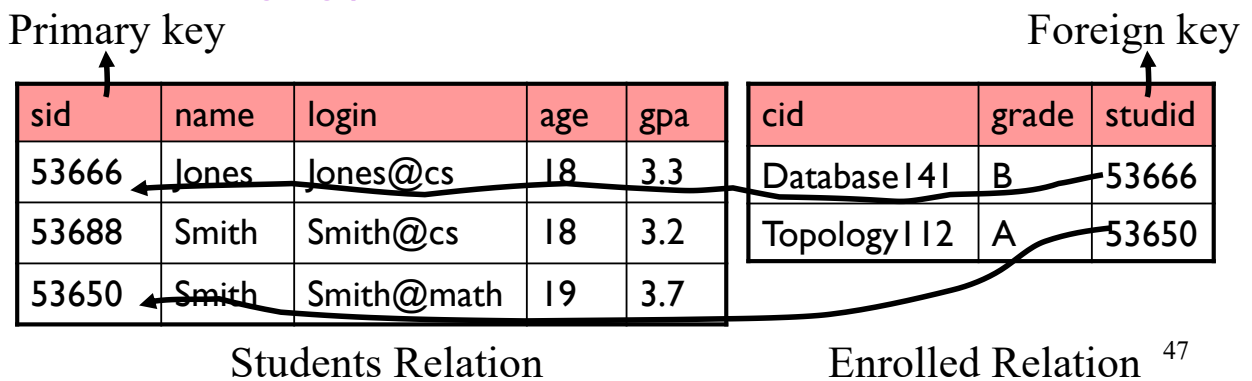
Students Relation

Enrolled Relation

46

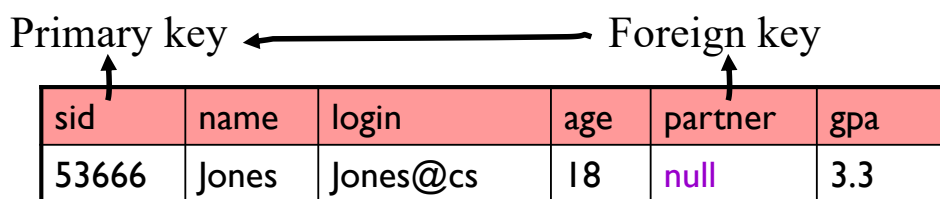
Foreign Key (Definition)

- Studid is called a **foreign key**.
- A foreign key is like a “**pointer**” in C, referencing a **unique tuple** / field in the **referenced relation**.
 - A foreign key constraint makes sure that there is **no dangling pointer**.



Self-Referral Foreign Key

- A foreign key can refer to the same relation.
- Example: each student could have a partner.
 - If a student hasn't found a partner, the value can be set to **null**.
 - It is ok to have null value in foreign key field.
 - But is it okay to have null value in primary key field?



More on Foreign Key

- The foreign key must refer to **primary key** in the referenced relation. Why?
 - Must be able to uniquely identify the tuple in the referenced relation.
- The foreign key needs not be a candidate key. Why?
 - Only used to uniquely identify a tuple in the referenced relation.
- If all foreign key constraints are enforced, **referential integrity** is achieved.
 - Think of a common example w/o referential integrity?
 - Bad links in HTML

49

Specify Foreign Keys in SQL

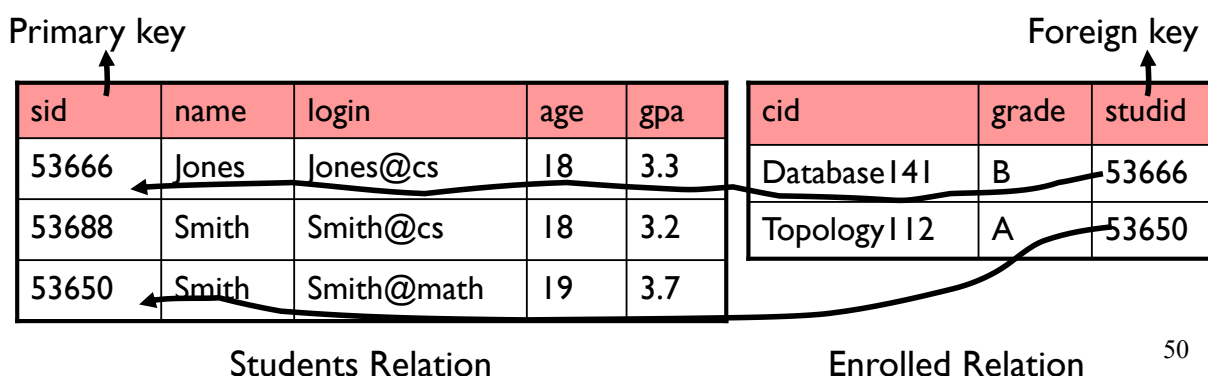
- Constraint: only students listed in the Students relation should be allowed to enroll for courses.

create table Enrolled

(studid char(20), cid char(20), grade char(20),

primary key (studid, cid),

foreign key (studid) references Students)



50

Foreign Key Constraint Violations

- When can they happen - delete? insert? update?

Primary key

Foreign key

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

cid	grade	studid
Database141	B	53666
Topology112	A	53650

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

cid	grade	studid
Database141	B	53666
Topology112	A	53650

History123	B	12345
------------	---	-------

insert

51

54321

Enforcing Referential Integrity

- What should be done if an Enrolled tuple with a non-existent student id is inserted?
 - Reject it!
- What should be done if a Students tuple is deleted while leaving a dangling enrolled tuple? (at least 4 possibilities)
 - Option 1: Also delete all Enrolled tuples that refer to it
 - Option 2: Disallow deletion
 - Option 3: Set studid in Enrolled tuples that refer to it to a default sid.
 - Option 4: Set studid in Enrolled tuples that refer to it to NULL.

Primary key

Foreign key

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

Students Relation

52

Referential Integrity in SQL

- Option 1: CASCADE
(also delete all tuples that refer to deleted tuple)
- Option 2: Default is NO ACTION (delete/update is rejected)
- Options $\frac{3}{4}$: SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled  
(studid CHAR(20) default "00000",  
cid CHAR(20),  
grade CHAR(2),  
PRIMARY KEY (studid,cid),  
FOREIGN KEY (studid)  
REFERENCES Students (sid)  
ON DELETE CASCADE  
ON UPDATE SET DEFAULT )
```

53

General Constraints

- An example : students ages must be over 16 years old.

```
create table Students (  
  sid char(20),  
  name char(20),  
  login char(10),  
  age integer,  
  gpa real,  
  unique (name, age),  
  constraint StudentsKey primary key (sid),  
  check (age > 16)  
)
```

54

More on General Constraints

- Two types
 - **Table constraint**: associate with a single table
 - **Assertions**: associate with multiple tables
 - if 1/2 of courses taken by a student has a grade = F in the Enrolled relation, the status of the student in the Students relation must be set to “In Trouble”.