

The Relational Model

CMU SCS 15-415
C. Faloutsos
Lecture #3
R & G, Chap. 3

Outline

- **Introduction**
- **Integrity constraints (IC)**
- **Enforcing IC**
- **Querying Relational Data**
- **ER to tables**
- **Intro to Views**
- **Destroying/altering tables**

Faloutsos 15-415

2

Why Study the Relational Model?

- **Most widely used model.**
 - Vendors: IBM/Informix, Microsoft, Oracle, Sybase, etc.
- **"Legacy systems" in older models**
 - e.g., IBM's IMS
- **Object-oriented concepts have recently merged in**
 - *object-relational model*
 - Informix->IBM DB2, Oracle 8i

Faloutsos 15-415

3

Relational Database: Definitions

- *Relational database*: a set of *relations*
- (relation = table)
- specifically

Faloutsos 15-415

4

Relational Database: Definitions

- *Relation*: made up of 2 parts:
 - *Schema*: specifies name of relation, plus name and type of each column.
 - *Instance*: a *table*, with rows and columns.
 - #rows = *cardinality*
 - #fields = *degree / arity*

Faloutsos 15-415

5

Relational Database: Definitions

- relation: a *set* of rows or *tuples*.
 - all rows are distinct
 - no order among rows (why?)

Faloutsos 15-415

6

Ex: Instance of Students Relation

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

- Cardinality = 3, arity = 5 ,
- all rows distinct
- Q: do values in a column need to be distinct?

Faloutsos 15-415

7

SQL - A language for Relational DBs

- **SQL*** (a.k.a. "Sequel"), standard language
- **Data Definition Language (DDL)**
 - create, modify, delete relations
 - specify constraints
 - administer users, security, etc.

* Structured Query Language

Faloutsos 15-415

8

SQL - A language for Relational DBs

- **Data Manipulation Language (DML)**
 - Specify *queries* to find tuples that satisfy criteria
 - add, modify, remove tuples

Faloutsos 15-415

9


SQL Overview

- **CREATE TABLE** <name> (<field> <domain>, ...)
- **INSERT INTO** <name> (<field names>)
VALUES (<field values>)
- **DELETE FROM** <name>
WHERE <condition>

Faloutsos 15-415

10

SQL Overview

- **UPDATE** <name>
SET <field name> =
<value>
WHERE <condition>
- **SELECT** <fields> 
FROM <name>
WHERE <condition>

Faloutsos 15-415

11

Creating Relations in SQL

- **Creates the Students relation.**

```
CREATE TABLE Students
(sid CHAR(20),
 name CHAR(20),
 login CHAR(10),
 age INTEGER,
 gpa FLOAT)
```

Faloutsos 15-415

12

Creating Relations in SQL

- **Creates the Students relation.**

–Note: the type (**domain**) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

Faloutsos 15-415

13

Table Creation (continued)

- **Another example:**

```
CREATE TABLE Enrolled
(sid CHAR(20),
 cid CHAR(20),
 grade CHAR(2))
```

Faloutsos 15-415

14

Adding and Deleting Tuples

- **Can insert a single tuple using:**

```
INSERT INTO Students
(sid, name, login, age, gpa)
VALUES
('53688', 'Smith', 'smith@cs',
18, 3.2)
```

Faloutsos 15-415

15

Adding and Deleting Tuples

- Can delete all tuples satisfying some condition (e.g., name = Smith):
DELETE
FROM Students S
WHERE S.name = 'Smith'

Powerful variants of these commands:
more later!

Outline

- Introduction
- Integrity constraints (IC)
- Enforcing IC
- Querying Relational Data
- ER to tables
- Intro to Views
- Destroying/altering tables

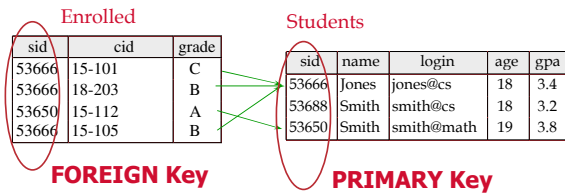
Keys

- Keys help associate tuples in different relations
- Keys are one form of integrity constraint (IC)

Enrolled			Students				
sid	cid	grade	sid	name	login	age	gpa
53666	15-101	C	53666	Jones	jones@cs	18	3.4
53666	18-203	B	53688	Smith	smith@cs	18	3.2
53650	15-112	A	53650	Smith	smith@math	19	3.8
53666	15-105	B					

Keys

- Keys help associate tuples in different relations
- Keys are one form of integrity constraint (IC)



Faloutsos 15-415

19

Primary Keys

- A set of fields is a **superkey** if:
 - No two distinct tuples can have same values in all key fields
- A set of fields is a **key** for a relation if :
 - minimal superkey

Faloutsos 15-415

20

Primary Keys

- what if >1 key for a relation?

Faloutsos 15-415

21

Primary Keys

- **what if >1 key for a relation?**
 - one of the keys is chosen (by DBA) to be the **primary key**. Other keys are called **candidate** keys..
 - Q: example?

Faloutsos 15-415

22

Primary Keys

- **E.g.**
 - *sid* is a key for Students.
 - What about *name*?
 - The set $\{sid, gpa\}$ is a superkey.

Faloutsos 15-415

23

Primary and Candidate Keys in SQL

- Possibly many **candidate keys** (specified using **UNIQUE**), one of which is chosen as the **primary key**.
- Keys must be used carefully!
- "For a given student and course, there is a single grade."

Faloutsos 15-415

24

Primary and Candidate Keys in SQL

```
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid))
vs.
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid),
UNIQUE (cid, grade))
```

Faloutsos 15-415

25

Primary and Candidate Keys in SQL

```
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid))
vs.
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid),
UNIQUE (cid, grade))
```

Q: what does this mean?

Faloutsos 15-415

26

Primary and Candidate Keys in SQL

```
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid))
vs.
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid),
UNIQUE (cid, grade))
```

"Students can take only one course, and no two students in a course receive the same grade."

Faloutsos 15-415

27

Foreign Keys

Enrolled

sid	cid	grade
53666	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B

Students

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

Faloutsos 15-415

28

Foreign Keys, Referential Integrity

- **Foreign key**: Set of fields 'referring' to a tuple in another relation.
 - Must correspond to the primary key of the other relation.
 - Like a 'logical pointer'.
- foreign key constraints enforce **referential integrity** (i.e., no dangling references.)

Faloutsos 15-415

29

Foreign Keys in SQL

Example: Only existing students may enroll for courses.

- *sid* is a foreign key referring to **Students**:

Enrolled

sid	cid	grade
53666	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B

Students

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

Faloutsos 15-415

30

Foreign Keys in SQL

```
CREATE TABLE Enrolled
(sid CHAR(20),cid CHAR(20),grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid) REFERENCES Students )
```

Enrolled

sid	cid	grade
53666	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B

Students

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

Outline

- Introduction
- Integrity constraints (IC)
- **Enforcing IC**
- Querying Relational Data
- ER to tables
- Intro to Views
- Destroying/altering tables

Enforcing Referential Integrity

- **Subtle issues:**
- **What should be done if an Enrolled tuple with a non-existent student id is inserted?**

Enrolled

sid	cid	grade
53666	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B

Students

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

Enforcing Referential Integrity

- **Subtle issues:**
- **What should be done if an Enrolled tuple with a non-existent student id is inserted? (*Reject it!*)**

Faloutsos 15-41534

Enforcing Referential Integrity

- **Subtle issues, cont'd:**
- **What should be done if a Student's tuple is deleted?**

Enrolled

sid	cid	grade
53666	15-101	C
53666	18-203	B
53650	15-112	A
53666	15-105	B

Students

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

Faloutsos 15-41535

Enforcing Referential Integrity

- **Subtle issues, cont'd:**
- **What should be done if a Students tuple is deleted?**
 - Also delete all Enrolled tuples that refer to it?
 - Disallow deletion of a Students tuple that is referred to?
 - Set sid in Enrolled tuples that refer to it to a *default sid*?
 - (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value *null*, denoting *'unknown'* or *'inapplicable'*.)

Faloutsos 15-41536

Enforcing Referential Integrity

- **Similar issues arise if primary key of Students tuple is updated.**

Faloutsos 15-415

37

Integrity Constraints (ICs)

- **IC: condition that must be true for *any instance of the database*; e.g., domain constraints.**
 - ICs are specified when schema is defined.
 - ICs are checked when relations are modified.

Faloutsos 15-415

38

Integrity Constraints (ICs)

- **A *legal* instance of a relation: satisfies all specified ICs.**
 - DBMS should not allow illegal instances.
- **we prefer that ICs are enforced by DBMS (as opposed to ?)**
 - Blocks data entry errors, too!

Faloutsos 15-415

39

Where do ICs Come From?

Faloutsos 15-415

40

Where do ICs Come From?

- the application!

Faloutsos 15-415

41

Where do ICs Come From?

- Subtle point:
- We can check a database instance to see if an IC is violated, but we can **NEVER** infer that an IC is true by looking at an instance.
 - An IC is a statement about *all possible* instances!
 - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.

Faloutsos 15-415

42

Where do ICs Come From?

- Key and foreign key ICs are the most common; more general ICs supported too.

Faloutsos 15-415

43

Outline

- **Introduction**
- **Integrity constraints (IC)**
- **Enforcing IC**
- **Querying Relational Data**
- **ER to tables**
- **Intro to Views**
- **Destroying/altering tables**

Faloutsos 15-415

44

ER to tables outline:

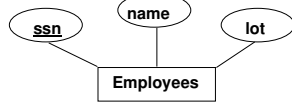
- **strong entities**
- **weak entities**
- **(binary) relationships**
 - 1-to-1, 1-to-many, etc
 - total/partial participation
- **ternary relationships**
- **ISA-hierarchies**
- **aggregation**

Faloutsos 15-415

45

Logical DB Design: ER to Relational

- (strong) entity sets to tables.

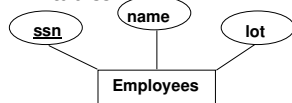


Faloutsos 15-415

46

Logical DB Design: ER to Relational

- (strong) entity sets to tables.



ssn	name	lot
123-22-3666	Attishoo	48
231-31-5368	Smiley	22
131-24-3650	Smethurst	35

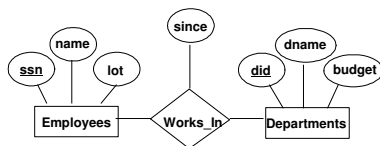
```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
```

Faloutsos 15-415

47

Relationship Sets to Tables

Many-to-many:

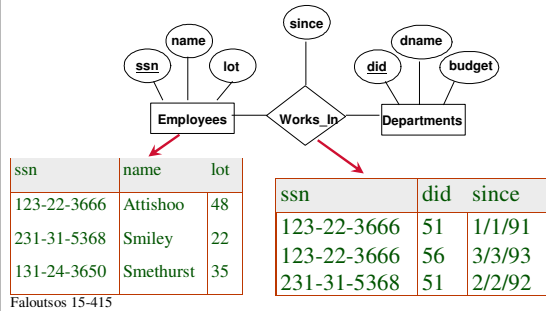


Faloutsos 15-415

48

Relationship Sets to Tables

Many-to-many:



Relationship Sets to Tables

- key of **many-to-many** relationships:
 - Keys from participating entity sets (as foreign keys).

```
CREATE TABLE Works_In(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
    REFERENCES Employees,
  FOREIGN KEY (did)
    REFERENCES Departments)
```

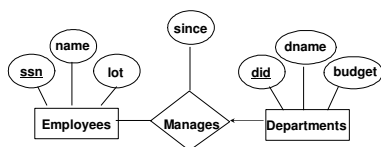
ssn	did	since
123-22-3666	51	1/1/91
123-22-3666	56	3/3/93
231-31-5368	51	2/2/92

Faloutsos 15-415

50

Review: Key Constraints in ER

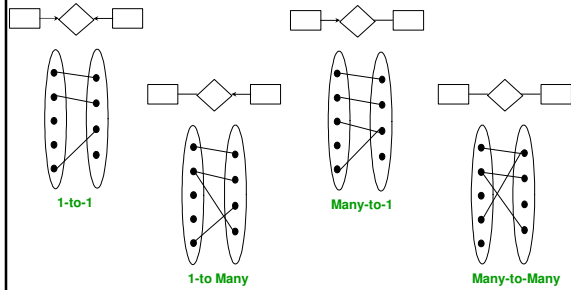
- 1-to-many:**



Faloutsos 15-415

51

Review: Key Constraints in ER



Faloutsos 15-415

52

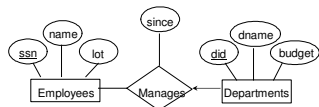
ER to tables - summary of basics

- **strong entities:**
 - key -> primary key
- **(binary) relationships:**
 - get keys from all participating entities - pr. key:
 - 1-to-1 -> either key (other: 'cand. key')
 - 1-to-N -> the key of the 'N' part
 - M-to-N -> both keys

Faloutsos 15-415

53

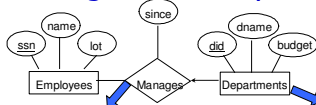
A subtle point (1-to-many)



Faloutsos 15-415

54

Translating ER with Key Constraints



```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,

  PRIMARY KEY (did),
  FOREIGN KEY (ssn)
  REFERENCES Employees,
  FOREIGN KEY (did)
  REFERENCES Departments)
```

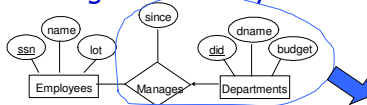
```
CREATE TABLE Departments(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  PRIMARY KEY (did),
)
```

Two-table-solution

Faloutsos 15-415

55

Translating ER with Key Constraints



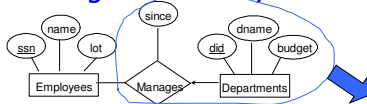
```
CREATE TABLE Dept_Mgr(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  dname CHAR(20),
  budget REAL,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn)
  REFERENCES Employees)
```

Single-table-solution

Faloutsos 15-415

56

Translating ER with Key Constraints



```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,

  PRIMARY KEY (did),
  FOREIGN KEY (ssn)
  REFERENCES Employees,
  FOREIGN KEY (did)
  REFERENCES Departments)
```

Vs.

```
CREATE TABLE Dept_Mgr(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  dname CHAR(20),
  budget REAL,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn)
  REFERENCES Employees)
```

Faloutsos 15-415

57

Pros and cons?

Drill:

What if the toy department has no manager (yet) ?

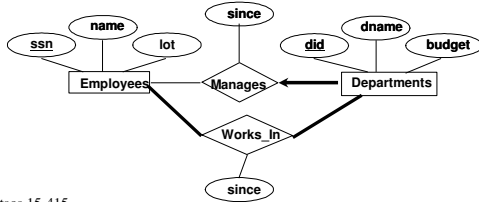
```
CREATE TABLE Dept_Mgr(  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  ssn CHAR(11),  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn)  
    REFERENCES Employees)
```

ER to tables outline:

- ✓ strong entities
- weak entities
- (binary) relationships
 - ✓ 1-to-1, 1-to-many, etc
 - total/partial participation
- ternary relationships
- ISA-hierarchies
- aggregation

Review: Participation Constraints

- Does every department have a manager?
 - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
 - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



Faloutsos 15-415

61

Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```

CREATE TABLE Dept_Mgr(
  did    INTEGER,
  dname  CHAR(20),
  budget REAL,
  ssn    CHAR(11) NOT NULL,
  since  DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE NO ACTION)
  
```

Faloutsos 15-415

62

Participation Constraints in SQL

- Total participation ('no action' -> do NOT do the delete)
- Ie, a department MUST have a manager

```

CREATE TABLE Dept_Mgr(
  did    INTEGER,
  dname  CHAR(20),
  budget REAL,
  ssn    CHAR(11) NOT NULL,
  since  DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE NO ACTION)
  
```

Faloutsos 15-415

63

Participation Constraints in SQL

- Partial participation, ie, a department may be headless

```
CREATE TABLE Dept_Mgr(  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  ssn CHAR(11) NOT NULL,  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE SET NULL)
```

Faloutsos 15-415

64

ER to tables outline:

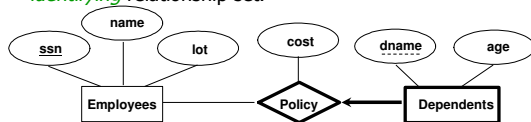
- ✓ strong entities
- • weak entities
- (binary) relationships
 - ✓ 1-to-1, 1-to-many, etc
 - ✓ total/partial participation
- ternary relationships
- ISA-hierarchies
- aggregation

Faloutsos 15-415

65

Review: Weak Entities

- A **weak entity** can be identified uniquely only by considering the primary key of another (*owner*) entity.
 - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
 - Weak entity set must have total participation in this *identifying* relationship set.

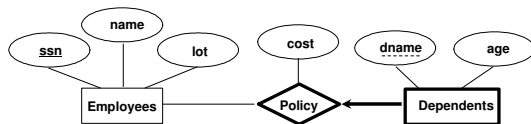


Faloutsos 15-415

66

Review: Weak Entities

How to turn 'Dependents' into a table?



Faloutsos 15-415

67

Translating Weak Entity Sets

- **Weak entity set and identifying relationship set are translated into a single table.**

```

CREATE TABLE Dep_Policy (
  dname CHAR(20),
  age   INTEGER,
  cost  REAL,
  ssn   CHAR(11) NOT NULL,
  PRIMARY KEY (dname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE CASCADE)
  
```

Faloutsos 15-415

68

Translating Weak Entity Sets

- **Weak entity set and identifying relationship set are translated into a single table.**
 - When the owner entity is deleted, all owned weak entities must also be deleted (-> 'CASCADE')

```

CREATE TABLE Dep_Policy (
  dname CHAR(20),
  age   INTEGER,
  cost  REAL,
  ssn   CHAR(11) NOT NULL,
  PRIMARY KEY (dname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE CASCADE)
  
```

Faloutsos 15-415

69

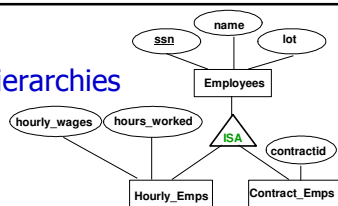
ER to tables outline:

- ✓ **strong entities**
- ✓ **weak entities**
- ✓ **(binary) relationships**
 - ✓ 1-to-1, 1-to-many, etc
 - ✓ total/partial participation
- **ternary relationships**
- ➔ **ISA-hierarchies**
- **aggregation**

Faloutsos 15-415

70

Review: ISA Hierarchies



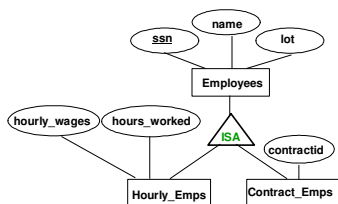
- **Overlap constraints:** Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- **Covering constraints:** Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)

Faloutsos 15-415

71

Drill:

- **What would you do?**



Faloutsos 15-415

72

Translating ISA Hierarchies to Relations

- **General approach: 3 relations: Employees, Hourly_Emps and Contract_Emps.**

- how many times do we record an employee?
- what to do on deletion?
- how to retrieve **all** info about an employee?

EMP (ssn, name, lot)

H_EMP(ssn, h_wg, h_wk) CONTR(ssn, cid)

Faloutsos 15-415

73

Translating ISA Hierarchies to Relations

- **Alternative: Just Hourly_Emps and Contract_Emps.**

– *Hourly_Emps*: ssn, name, lot, hourly_wages, hours_worked.

– Each employee **must be** in one of these two subclasses

EMP (ssn, name, lot)

H_EMP(ssn, h_wg, h_wk, name, lot) CONTR(ssn, cid, name, lot)

Notice: 'black' is gone!

Faloutsos 15-415

74

ER to tables outline:

- ✓ **strong entities**
- ✓ **weak entities**
- ✓ **(binary) relationships**
 - ✓ 1-to-1, 1-to-many, etc
 - ✓ total/partial participation
- ➔ **ternary relationships**
- ✓ **ISA-hierarchies**
 - **aggregation**

Faloutsos 15-415

75

Ternary relationships; aggregation

- rare
- keep keys of all participating entity sets
- (or: avoid such situations:
 - break into 2-way relationships or
 - add an auto-generated key)
-)

Faloutsos 15-415

76

Outline

- Introduction
- Integrity constraints (IC)
- Enforcing IC
- Querying Relational Data
- ER to tables
- Intro to Views
- Destroying/altering tables

Faloutsos 15-415

77

Views

- Virtual tables

```
CREATE VIEW
YoungActiveStudents(name,grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid=E.sid and S.age<21
```
- DROP VIEW

Faloutsos 15-415

78

Views and Security

- **DBA: grants authorization to a view for a user**
- **user can only see the view - nothing else**

Faloutsos 15-415

79

Outline

- **Introduction**
- **Integrity constraints (IC)**
- **Enforcing IC**
- **Querying Relational Data**
- **ER to tables**
- **Intro to Views**
- **Destroying/altering tables**

Faloutsos 15-415

80

Table changes

- **DROP TABLE**
- **ALTER TABLE, e.g.**
ALTER TABLE students
ADD COLUMN maiden-name CHAR(10)

Faloutsos 15-415

81

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used
 - Object-relational variant gaining ground
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
 - Two important ICs: primary and foreign keys
 - also: not null, unique
 - In addition, we *always* have domain constraints.
- Mapping from ER to Relational is (fairly) straightforward:

Faloutsos 15-415

82

ER to tables - summary of basics

- **strong entities:**
 - key -> primary key
- **(binary) relationships:**
 - get keys from all participating entities - pr. key:
 - 1:1 -> either key
 - 1:N -> the key of the 'N' part
 - M:N -> both keys
- **weak entities:**
 - strong key + partial key -> primary key
 - ON DELETE CASCADE

Faloutsos 15-415

83

ER to tables - summary of advanced

- **total/partial participation:**
 - NOT NULL; ON DELETE NO ACTION
- **ternary relationships:**
 - get keys from all; decide which one(s) -> prim. key
- **aggregation: like relationships**
- **ISA:**
 - 2 tables ('total coverage')
 - 3 tables (most general)

Faloutsos 15-415

84