

## The Relational Model

CMU SCS 15-415  
Tuesday, August 31, 2010  
Lecture #3 (Kesden)

R & G, Chap. 3

Slides from Christos Faloutsos

## Outline

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

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

15-415

3

## Relational Database: Definitions

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

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*

15-415

5

## Relational Database: Definitions

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

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?

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

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

15-415

9

## SQL Overview

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

15-415

10

## SQL Overview

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

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)
```

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.

15-415

13

## Table Creation (continued)

- **Another example:**

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

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)
```

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!**

15-415

16

## Outline

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

15-415

17

## 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					

Diagram showing connections from the Enrolled table to the Students table:

- Enrolled (53666, 15-101, C) connects to Student (53666, Jones, jones@cs, 18, 3.4)
- Enrolled (53666, 18-203, B) connects to Student (53688, Smith, smith@cs, 18, 3.2)
- Enrolled (53650, 15-112, A) connects to Student (53650, Smith, smith@math, 19, 3.8)

15-415

18

## 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					

**FOREIGN Key**      **PRIMARY Key**

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

15-415

20

## Primary Keys

- what if >1 key for a relation?

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?

15-415

22

## Primary Keys

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

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."

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))
```

### 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?

### 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."

### Foreign Keys

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					

### 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.)

### Foreign Keys in SQL

Example: Only existing students may enroll for courses.

- sid is a foreign key referring to Students:

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					

## 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

15-415

31

## Outline

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

15-415

32

## 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

15-415

33

## Enforcing Referential Integrity

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

15-415

34

## 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

15-415

35

## 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'.)

15-415

36

### Enforcing Referential Integrity

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

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.

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!

15-415

39

### Where do ICs Come From?

15-415

40

### Where do ICs Come From?

- the application!

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.

15-415

42

## Where do ICs Come From?

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

15-415

43

## Outline

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

15-415

44

## ER to tables outline:

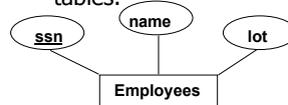
- strong entities
- weak entities
- (binary) relationships
  - 1-to-1, 1-to-many, etc
  - total/partial participation
- ternary relationships
- ISA-hierarchies
- aggregation

15-415

45

## Logical DB Design: ER to Relational

- (strong) entity sets to tables.

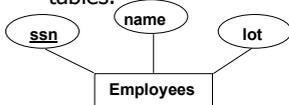


15-415

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

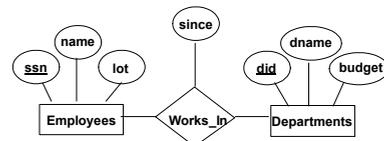
```
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
```

15-415

47

## Relationship Sets to Tables

Many-to-many:

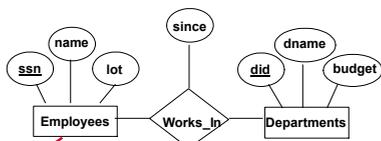


15-415

48

### Relationship Sets to Tables

Many-to-many:



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

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

15-415

49

### 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)
```

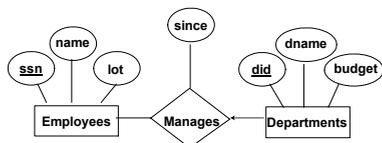
ssn	did	since
123-22-3666	51	1/1/91
123-22-3666	56	3/3/93
231-31-5368	51	2/2/92

15-415

50

### Review: Key Constraints in ER

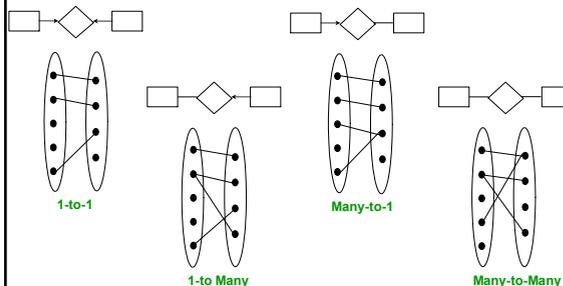
- 1-to-many:



15-415

51

### Review: Key Constraints in ER



15-415

52

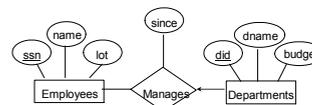
### ER to tables - summary of basics

- 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

15-415

53

### A subtle point (1-to-many)



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

15-41555

### 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

15-41556

### 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)
```

15-41557

## Pros and cons?

15-41558

## 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)
```

15-41559

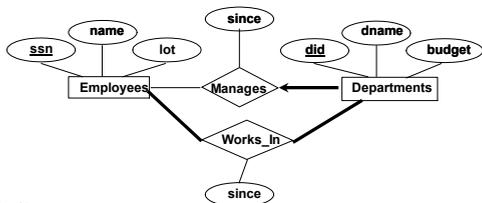
## ER to tables outline:

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

15-41560

### 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!)



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)
```

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)
```

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)
```

15-415

64

### ER to tables outline:

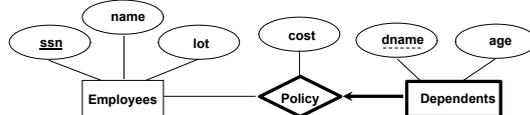
- ✓ strong entities
- weak entities
- (binary) relationships
  - ✓ 1-to-1, 1-to-many, etc
  - ✓ total/partial participation
- ternary relationships
- ISA-hierarchies
- aggregation

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.

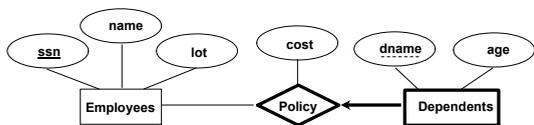


15-415

66

### Review: Weak Entities

How to turn 'Dependents' into a table?



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)
```

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)
```

15-415

69

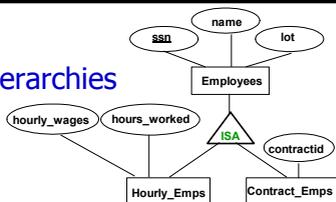
### ER to tables outline:

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

15-415

70

### Review: ISA Hierarchies



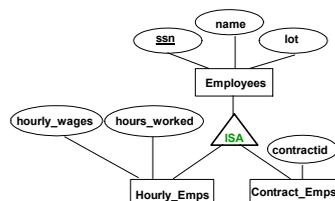
- Overlap constraints:** Can Joe be an Hourly\_Emps as well as a Contract\_Emps entity? (Allowed/disallowed)
- Covering constraints:** Does every Employees entity also have to be an Hourly\_Emps or a Contract\_Emps entity? (Yes/no)

15-415

71

### Drill:

- What would you do?



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)

blue	white	white
black	white	white
green	white	white

H\_EMP(ssn, h\_wg, h\_wk)

blue	white	white
green	white	white

CONTR(ssn, cid)

red	white	white
-----	-------	-------

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:

H\_EMP(ssn, h\_wg, h\_wk, name, lot)

blue	white	white	grey	grey
green	white	white	grey	grey

CONTR(ssn, cid, name, lot)

red	white	white	grey	grey
-----	-------	-------	------	------

Notice: 'black' is gone!

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

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
- )

15-415

76

### Outline

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

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

15-415

78

## Views and Security

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

15-415

79

## Outline

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

15-415

80

## Table changes

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

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:

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

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)

15-415

84