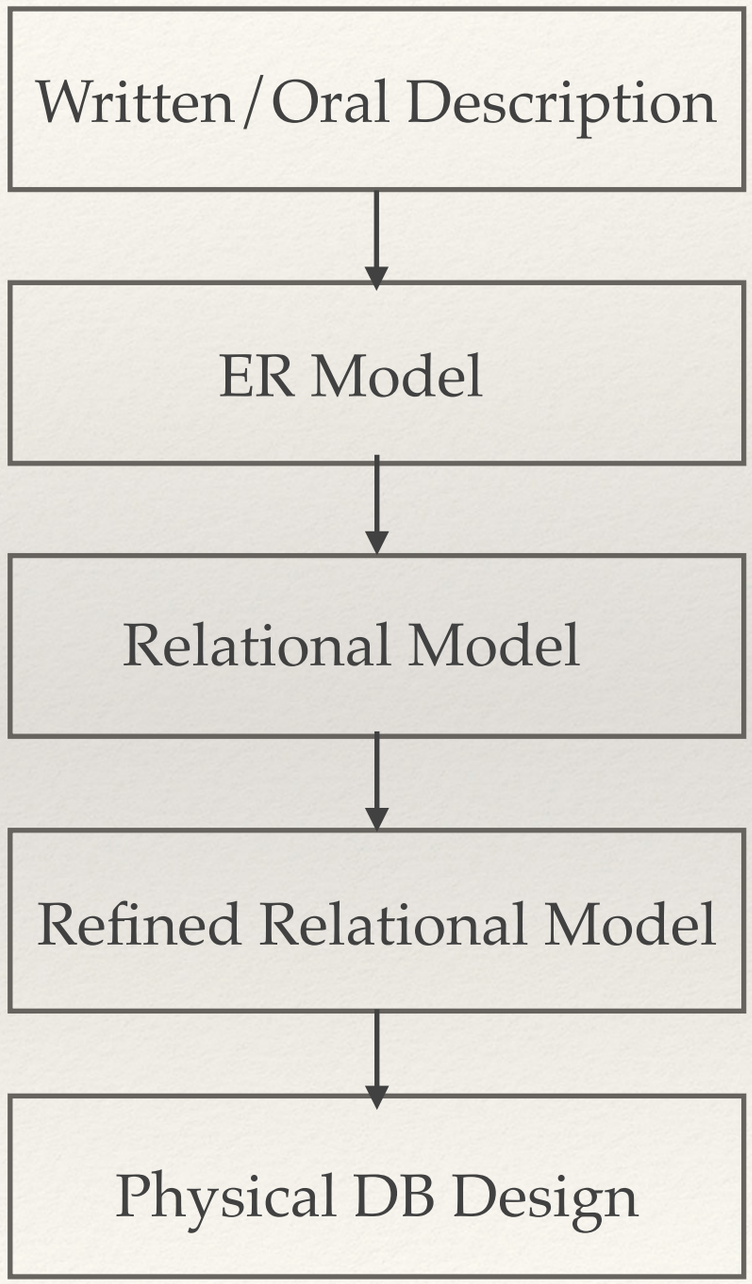
Relational Data Model: Part 1

Steps in Database Modeling

- Both data that's there and queries that may be asked
- ER useful for high level
- Concrete: Tables, Attributes
- Refine: Establishing relationships between attributes and tables
- Data Definition and Index structures
- Additional index structures



The Student-Courses Database

- Capture students and the courses they take.
- The courses have a unique number inside a department
- Students have a unique student identification
- We also want to capture the student's status (freshman, …, graduate) when they took the class
- Want to answer queries like: typical status of student for each course? Average enrollment for a course, etc.

Relational Model

3.1-3.3

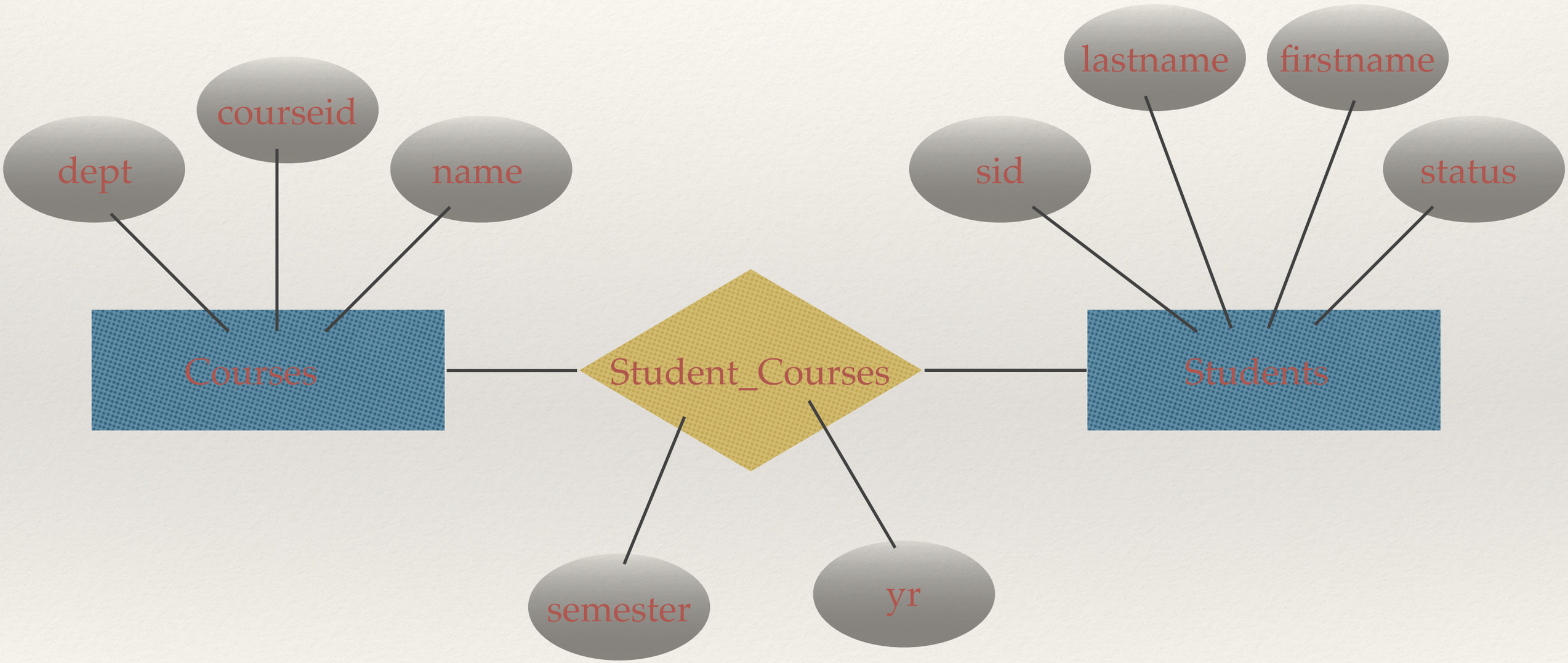
- Representing Entities and Relationships as tables
- Relationships between entities expressed as key-foreign key linkage
- Many other ER modeling concepts can be expressed as constraints on the attributes in the relational model
- Data Independence
- Describing data by its natural structure (not by a machine/ implementation dependent artifact)
- A query language independent of implementation details

Keys

- Candidate Key is any minimal subset of columns (attributes) that are unique in the table (relation)
- Primary Key is a “chosen” candidate key
- A relation must have a key (as all rows are distinct)
- Superkey is any superset of a key
- Foreign key is a attribute that whose values are primary key of another table

E-R model picture

2.2-2.3



Relational Model

Courses

dept	cid	name
CS	564	DBMS
CS	300	Prog II
Math	234	Calculus III
PolSci	104	US Govt

Students

sid	last_name	first_name	status
2016001	Clinton	Bill	5
2015001	Lincoln	Abraham	1
2014101	Obama	Barack	2
2012144	Bush	George	4

Student_Courses

dept	cid	sid	sem	year
CS	564	2016001	1	2016
PolSci	104	2015001	1	2016
PolSci	104	2014101	2	2015
Math	234	2012144	1	2013

Keys and Foreign Keys

Courses

dept	cid	name
CS	564	DBMS
CS	367	Data Structures
Math	234	Calculus III
PolSci	104	US Govt

Students

sid	last_name	first_name	status
2016001	Clinton	Bill	5
2015001	Lincoln	Abraham	1
2014101	Obama	Barack	2
2012144	Bush	George	4

Student_Courses

dept	cid	sid	sem	year
CS	564	2016001	1	2016
PolSci	104	2015001	1	2016
PolSci	104	2014101	2	2015
Math	234	2012144	1	2013

3 Kinds of Relationships

- 1-1 - captured in a single table
- 1-Many - can usually be captured in single table
- Many-Many - the most common - use separate table

Courses

dept	cid	name
CS	564	DBMS
CS	300	Programming II
Math	234	Calculus III
PolSci	104	US Govt

Courses_with_Instructors

dept	cid	name	instructor
CS	564	DBMS	Shatdal
CS	300	Prog. II	Smith
Math	234	Calculus III	Smith
PolSci	104	US Govt	Madison

Instructors

instr_id	instructor
101	Shatdal
102	Smith
103	Madison

Courses_Instructors

dept	cid	instr_id
CS	564	101
CS	367	102
Math	234	102
PolSci	104	103

ER Constraints

- Key Constraints can sometimes be represented by having the foreign key as the primary key (in a 1-N relationship).
- Participation constraint can be represented by having non-nullable values (e.g. every department must have a manager => manager value must be non-null)

IS-A/Class Hierarchies

2.4.4, 3.5.6

- Sub-categories of entities
- Movies
  - Animation; Mystery; Foreign
- Employees(ssn, name, lot)
  - Hourly(…, hourly_wage, hours_worked)
  - Contract(…, contractid)

Class Hierarchies

Employee

ssn	name	lot
101	Abe	1
201	George	2
301	Bill	1
202	James	2

Hourly

ssn	wages	hours
201	15	2
202	20	40

Contract

ssn	contractid
301	1001

Class Hierarchies

Contract

ssn	name	lot	contractid
301	Bill	1	1001

Regular

ssn	name	lot
101	Abe	1

Hourly

ssn	name	lot	wages	hours
201	George	2	15	2
202	James	2	20	40

Employees

ssn	name	lot	type	wages	hours	contractid
201	George	2	H	15	2	null
202	James	2	H	20	40	null
301	Bill	1	C	null	null	1001
101	Abe	1	R	null	null	null

Keys Revisited

- All rows are unique
- Candidate Key : Any subset of attributes that uniquely identifies a row
- [Primary] Key is the chosen candidate key
- Surrogate Key: a system or systematically generated key that uniquely identifies a row

Primary Key

```
db1=# create table students(sid int,
                            last_name varchar(20),
                            first_name varchar(20),
                            status smallint,
                            primary key (sid));
CREATE TABLE
db1=# insert into students values(1779, 'Washington', 'George', 1);
INSERT 0 1
db1=# insert into students values(1779, ‘Adams', 'John', 2);
ERROR:  duplicate key value violates unique constraint "students_pkey"
DETAIL:  Key (sid)=(1779) already exists.
```

Candidate Keys

- Most systems would let you identify these as uniqueness constraints

```
db1=# create table students(sid int, last_name varchar(20),
db1(#                               first_name varchar(20), status smallint,
db1(#                               primary key (sid),
db1(#                               unique (last_name, first_name));
CREATE TABLE
db1=# insert into students values(1779, 'Washington', 'George', 1);
INSERT 0 1
db1=# insert into students values(1781, 'Washington', 'George', 2);
ERROR:  duplicate key value violates unique constraint
"students_last_name_first_name_key"
DETAIL:  Key (last_name, first_name)=(Washington, George) already exists.
```

Surrogate Key

- To ensure uniqueness where none may exist
- To make keys more compact
- Do not change
- Uniform, Compatible
- No semantics
- Can't use for optimization
- Harder to enforce the natural keys

ckey	dept	cid	name
1	CS	564	DBMS
2	CS	367	Data Structures
3	Math	234	Calculus III
4	PolSci	104	US Govt

Referential Integrity

- Foreign Keys *references* must be valid
- NULL or a key from table being referenced
- How to maintain this integrity?

```
create table enrollment(
 dept varchar(10),
 cid smallint,
 sid int,
 sem smallint,
 yr smallint,
 primary key (dept, cid, sid),
 foreign key (dept, cid)
              references courses,
 foreign key (sid)
              references students);
```

For Referencing Table

- Add a row that doesn't have the referenced value
  - Reject
  - Could have used NULL/default “always present” value
  - [In principle, add the value to the referenced table….]
- Delete a row => do nothing

For Referenced Table

- Delete a value in referenced table
  - Disallow if there is a reference to it
  - Delete all referencing rows
  - Set the references to default/NULL
- Insert a value => do nothing
- Primary Key update (same as deleting/inserting new value)

For Referenced Table

- Delete a value in referenced table
  - Disallow if there is a reference to it
  - Delete all referencing rows
  - Set the references to default/NULL
- Insert a value => do nothing
- Primary Key update (same as deleting/inserting value)

```
create table enrollment(
 dept varchar(10),
 cid smallint,
 sid int,
 sem smallint,
 yr smallint,
 primary key (dept, cid, sid),
foreign key (dept, cid) references courses
                         on delete cascade
                         on update no action,
foreign key (sid) references students
                         on delete set null);
```

Missing Values

- Missing Values indicated by NULL
- Primary Keys are conceptually not NULL
- NOT NULL constraint can provide more meaning

Constraints

- key and and foreign key constraints are most basic
- attribute domain constraints are common (restricting range of values allowed)
- general constraints over a table can be defined but are less common (and expensive to maintain)
- Enforced as data in inserted/deleted/updated
- Foreign key constraints can be enforced in various ways

Domain Constraints

- Data Type already is the first level of domain constraint
  - Int vs. SmallInt; varchar(10) vs. varchar(20)
  - optimize storage
- One can add additional constraints, e.g. range of allowed values

Domain Constraints In Practice

```
db1=# create table students(sid int check (sid between 1779 and 2100),
db1(#                       last_name varchar(20),
db1(#                       first_name varchar(20),
db1(#                       status smallint not null check (status between 1 and 5),
db1(#                       primary key (sid),
db1(#                       unique (last_name, first_name));
CREATE TABLE
db1=# insert into students values(1776, 'Franklin', 'Benjamin', 4);
ERROR:  new row for relation "students" violates check constraint
"students_sid_check"
DETAIL:  Failing row contains (1776, Franklin, Benjamin, 4).
db1=# insert into students values(1779, 'Washington', 'George', NULL);
ERROR:  null value in column "status" violates not-null constraint
DETAIL:  Failing row contains (1779, Washington, George, null).
```