Nathan Gong ng2695

W1) The 3 basic concepts in the ER data model are entity sets, relationship sets, and attributes.

W2) 1. ER diagrams are conceptually simple and can be easily drawn if the entities, relationships, and attributes are known.
2. ER diagrams are an effective communication tool for database designers with other technical and non-technical stakeholders.
3. ER diagrams can be converted into relational data models using tables, as well as many other data models including hierarchical and network models.

W3) A domain is a set of allowed values, and an SQL data type defines the exact range and type of values that an attribute can have. An atomic domain means that the values of an attribute are indivisible. An example of an atomic attribute is first_name, and an example of a non-atomic attribute is full_name.

W4) Physical data independence refers to how data is physically stored separately from the logical schema or applications that use the data. This means that the data and the schema/application can be used and modified independently of each other.

W5) 1. Designing and implementing the database schema.
2. Controlling user access and security for the data.
3. Ensuring that regular backups of the data are taken.

W6) A natural key is formed from values inherently existing in the real world, and a surrogate key is an artificial system-generated identifier.

W7) This statement may produce incorrect results because as rows are deleted from the table, the value of avg(salary) will change.

W8) It is not always possible.

1. The view may attempt to insert/update values but might only contain a subset of the columns of its reference table, and the table might prevent the insertion of NULL values or it may lead to nonsensical results.
2. The underlying table structure may have changed, causing the view to become stale if it was not maintained.

W9) 3 concepts from SQL DDL that implement integrity independence are primary key, foreign key, and check constraints. One problem caused when integrity rules are only implemented by applications is inconsistent/invalid data being entered into the database.

W10) A composite primary key is a unique identifier consisting of multiple attributes, and may be used when none of its attributes would be a unique key alone.

W11) This is not always a valid approach because the data currently in the table might not be representative of the full set of possible values for each column, and may lead one to believe that a column is a unique key when it is not.

R1)

$\pi_{a,b}(R)$

a	b
1	a
3	c
4	d
5	e
6	

T U S

b	d
a	100
d	200
f	400
g	120
b	300
c	400
e	150

$(\pi_{a,b}(R)) \bowtie (T \cup S)$

a	b	d
1	a	100
3	c	400
4	d	200
5	d	200
6	e	150

R 2)

$\pi_{a,b}(R) \cup \pi_{a,c}(R)$

a	b
1	a
1	d
3	c
4	d
4	f
5	d
5	b
6	e
6	f

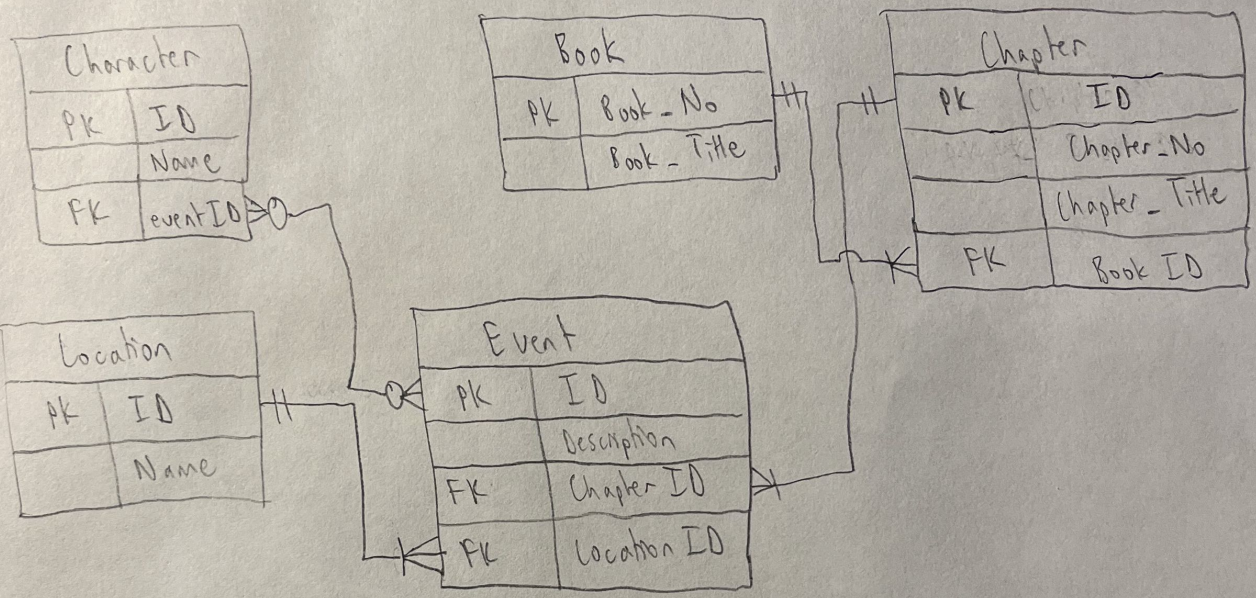
$\sigma_{a>3}(\pi_{a,b}(R) \cup \pi_{a,c}(R)) \bowtie S$

a	b	d
4	d	200
4	f	400
5	d	200
5	b	null
6	e	null
6	f	400

R 3) $\pi_{R.a, R.b, R.c}(\sigma_{T.d=null}(R \bowtie_{R.b=T.b} T))$

R 4) $\pi_S(S \bowtie_T null) \cup \pi_T(T \bowtie_S null) \setminus \pi_T(T \bowtie_S null)$

ER model:



Realizing ER model:

```
CREATE TABLE name_basics(
    nconst  INT NOT NULL,
    first_name VARCHAR(64) NOT NULL,
    last_name VARCHAR(64) NOT NULL,
    nickname VARCHAR(64) NULL,
    PRIMARY KEY (nconst));
```

```
CREATE TABLE title_basics(
    tconst INT NOT NULL,
    title VARCHAR(64) NOT NULL,
    description VARCHAR(64) NOT NULL,
    PRIMARY KEY (tconst));
```

```
CREATE TABLE name_title(
    nconst INT NOT NULL,
    tconst INT NOT NULL,
    is_known_for BOOLEAN NOT NULL,
    role VARCHAR(64) NOT NULL,
    CHECK (role in ('Actor', 'Director', 'Writer',
                    'Producer')),
    PRIMARY KEY (nconst, tconst, role),
    FOREIGN KEY (nconst) REFERENCES name_basics
    (nconst),
    FOREIGN KEY (tconst) REFERENCES title_basics (tconst));
```

SQL 1)

```
SELECT student.ID as student_ID,
       t.no_of_courses,
       student.tot_cred as compute credits,
       student.tot_cred as recorded_credits from student
JOIN (SELECT ID,
             COUNT(course_id) as no_of_courses
             from takes GROUP BY (ID)) as t
  on student.ID = takes.ID;
```

SQL 2)

```
INSERT INTO db_book.department (dept_name, building, budget)
VALUES ('Chem. Eng.', 'Taylor', 150000);
```

SQL 3)

```
UPDATE db_book.department
  SET building = 'Northwest'
WHERE dept_name != 'Comp. Sci.';
```