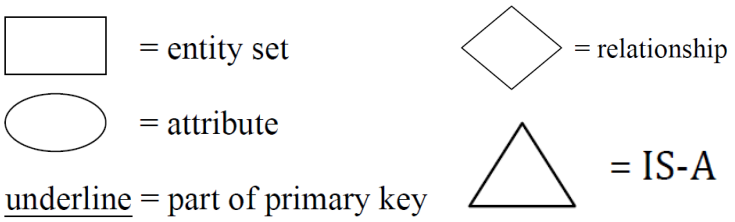
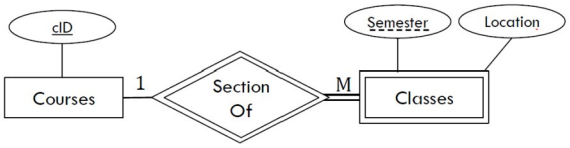
Attribute – a name and a type (column)
Schema – Table name + a set of attributes
Instance – the values in a table (A set of tuples, every row is unique)
Tuple – one row in an instance
Relation – schema plus an instance

Superkey – A set of attributes is a superkey if no two rows are allowed to have the same values in those columns (no duplicates)
Key – a set of fields is a key it is a superkey, and no proper subset of its fields is a superkey
One key is specified as the **primary key**, and others are **candidate keys**
Foreign Key - Attribute whose values are a key in another table



Double line indicates all entities in the set must participate



Weak entity + supporting relationship set

ER → Schema:

Many to Many - new schema for the relationship, primary keys of relating entities as foreign keys, key is a combination of the foreign keys
One to Many - merge relationship into the M side entity
One to One – treat as one to many, merge relationship into one of the tables (prefer side with required participation), mark foreign key as unique
Supporting Relationship – is 1 to M, supporting key is combined with partial key
Double Participation – enforced in software, not in schema
Is-A – 1. One schema per entity, pull down just primary key, or
2. No schema for base type, pull down all attributes
Use method 1 if an entity can be two different derived types or the base type has relationships, use method 2 if the base type is 'abstract'

Schema → SQL

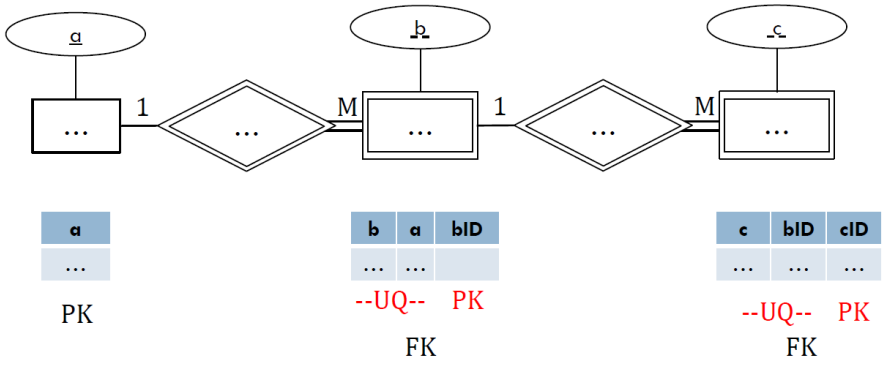
create table <name> (
    <column1Name> <type> <properties>,
    <column2Name> <type> <properties>,
    <table properties>
);

Types – <tiny, small, medium, big>int <unsigned>, float, double, decimal, date, datetime, time, blobs
    char(n) – exactly n characters
    varchar(n) – up to n characters

Column Properties – not null, default 'hello', auto_increment

Table Properties – primary key (col1, col2…)
    unique (col1, col2…)
    foreign key (col1, col2…) references Table(col1, col2…)
        on delete <action> on update <action>
    index (col1, col2…)

FK Actions:
RESTRICT(default): disallow the change
CASCADE: also delete/update in child table
SET NULL: nullify key in child table
SET DEFAULT: set to column's default value



•`bID` now represents complex key in smaller footprint

SQL

Select:

SELECT [DISTINCT] <target-list> FROM <relation-list> [WHERE <qualification>]
[ORDER BY <column>] [DESC] [LIMIT <number>];

- target-list and relation-list are comma separated column and table names respectively
    - target-list can be *
- qualification is a boolean expression, can use AND, OR, NOT
- WHERE Patrons.CardNum = Phones.CardNum - where on joined table
- **IN**: SELECT x FROM … WHERE y IN (SELECT …);
    - nested query must have same column type(s) as x
- **NOT IN**: SELECT Addr FROM CorporateLocs WHERE Addr NOT IN (SELECT Addr FROM

Join:

<table> JOIN <table> ON/WHERE <qualification>;

- join returns a temporary table
- qualification same as select
- joins can be chained, (Patrons JOIN CheckedOut) JOIN Inventory

Natural Join – used in place of JOIN, joins on the columns the two tables have in common
Left Join – keeps all rows from left table, uses NULL if right table doesn't have matching row
Right Join – same as left, but right. BOTH OUTER JOINS REQUIRE ON <condition>
Natural Left/Right Join – only one copy of 'natural' columns

Insert:

INSERT INTO <table>[(col1, col2 ...)] VALUES(val1, val2 …);

- inserts into ever column in order if no columns specified
- specify columns if there are null/default/auto_increment columns

Delete:

DELETE FROM <table> WHERE <qualification>

- qualification same as before
- leave out WHERE to delete all rows...

Update:

UPDATE <table> SET col1 = val1, col2 = val2 WHERE < qualification>

Alter:

ALTER TABLE <table> ADD <column>

- <column> is in the same format as when initially creating the table (name, type, properties)

ALTER TABLE <table> MODIFY <column> <new type>
ADD FOREIGN KEY (<column>) REFERENCES <table>(<column>)

Union:

<select statement> UNION <select statement>

- just like union in relational algebra
- use UNION ALL to keep duplicates

Intersect:

<select statement> INTERSECT <select statement>

- not supported by MySQL
- same as natural join if the schemas are the same

Relational Algebra

π – Projection (select) – get certain columns
σ – Selection (where) – get certain rows
$\times$ – Cross Product (join) – big combine, column names must be disambiguated
$\cup$ – Set Union - 'add', relations must be union compatible
$\cap$ – Set intersect – $A \cap B = A - (A - B)$
– - Set difference – relations must be union compatible
$\bowtie$ - Natural Join – can be given condition too, shortcut for $\sigma_{condition}(R1 \times R2)$
ρ – rename a relation, ρ(new-name, expression)

- can also rename columns ρ(new-name$_{newCol/oldCol}$, expression)

Output of RA query must be a relation (no duplicates)

Division (/):

A

cNo	pNo
c1	p1
c1	p2
c1	p3
c1	p4
c2	p1
c2	p2
c3	p2
c4	p2
c4	p4

B

pNo
p2
p4

A / B

cNo
c1
c4

C

pNo
p2

A / C

cNo
c1
c2
c3
c4

D

pNo
p1
p2
p4

A / D

cNo
c1