Final Review

CSE 180

Queries

SELECT [DISTINCT] c1, c2, ..., cm AGGOP(...)

FROM R1, R2, ..., Rn

[WHERE condition]

[GROUP BY <list grouping attributes>]

[HAVING condition]

[ORDER BY < list of attributes [ASC | DESC]>]

Insert, Delete and Update

```
INSERT INTO R1
    VALUES (c1, …, cn)

DELETE FROM R
    WHERE <condition>

UPDATE R
    SET <attribute> = <new value>
    WHERE <condition>
```

Views and Indexes

CREATE VIEW AS <name>
SELECT ... FROM... [WHERE...]

Logical Data Independence: users can make database without knowing whether they are tables or views -- the queries should work either way

CREATE INDEX <name> ON R(A1, ..., An)

Physical Data Independence: users can make database request on tables without knowing how the data in the table is stored

Constraints

```
ALTER TABLE R
    ADD CONSTRAINT <constraint name>
    FOREIGN KEY (<attributes>) REFERENCES R2(<attributes>)
    [ON UPDATE/RESTRICT/CASCADE]
```

```
ALTER TABLE R
    ADD CONSTRAINT <constraint name>
    CHECK (<constraint>)
```

Triggers

```
CREATE TRIGGER BeerTrig
    AFTER INSERT ON Sells
    REFERENCING NEW ROW AS NewTuple
    FOR EACH ROW
    WHEN (NewTuple.beer NOT IN
        (SELECT name FROM Beers))
    INSERT INTO Beers(name)
        VALUES(NewTuple.beer);
```

The Event

The Condition

The Action

Stored procedure/function

```
CREATE PROCEDURE/FUNCTION <name> <arguments>
        function_name (<arguments> <type>)
RETURNS <type> AS $$
        DECLARE
                variables variable_type;
                … more variables;
        DECLARE CURSOR FOR
                SELECT <attributes>… FROM <table names> …[WHERE <conditions];
        BEGIN
                <variables/ any conditions you want to check>
                OPEN <cursor name>
                LOOP
                        FETCH <cursor name> INTO <attributes in cursor>
                        <anything you want to do>
                END LOOP ;  // close the loop and cursor
                CLOSE <cursor name>;
        RETURN <value>
        END
$$ LANGUAGE plpgsql;
```

ACID

Atomicity: all or nothing; all changes in a transaction are performed or none

Consistency: domain of attribute must have certain restrictions and constraints

Isolation: whether or not multiple transactions trip over each other

Durability: after a transaction completes, changes to data persist

Other misc. facts

=ANY is the same as IN is the same as EXISTS

<>ALL is the same as NOT IN is the same as NOT EXISTS

If no snowboarders (same for <, =, etc)

- >ALL returns all,
- >MAX, >SOME, >MIN returns NO customers

COUNT on an empty set is 0, but SUM, AVG, MIN, MAX on an empty set is NULL

Transactions

Isolation Level	Clean Reads	Repeatable Reads	Simultaneous Existence
READ UNCOMMITTED	N	N	N
READ COMMITTED	Y	N	N
REPEATABLE READ	Y	Y	N
SERIALIZABLE	Y	Y	Y

BEGIN TRANSACTION ISOLATION LEVEL <level>;

…
COMMIT TRANSACTION;

Union, Intersection and Except

	DISTINCT	ALL
R UNION S	MIN(m+n, 1)	m+n
R INTERSECT S	MIN(MIN(m,n,), 1)	MIN(m,n)
R EXCEPT S	IF m>0 AND n=0 THEN 1 ELSE 0	MAX(m-n, 0)

Relational Algebra

- Selection (σ)
- Projection (π)
- Set-theoretic operations:
  - Union (∪)
  - Set-difference (-)
  - Cross-product (x)
  - Intersection (∩)
- Renaming (ρ) and Assignment (¬)
- Natural Join ( ), Theta-Join ( )
- Division (/ or ÷)

Relational Algebra Properties

	Communicative	Associative	Distributive
Union	✔	✔	
Difference			
Product	✔	✔	✔
Intersect	✔	✔	

Armstrong's Axioms

Let X, Y, and Z denote sets of attributes over a relation schema R.

- Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$.
  - FDs in this category are called trivial FDs.
- Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any set Z of attributes.
- Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

Derived from Armstrong's axioms

- Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
- Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- Pseudo-Transitivity: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$.

Completeness and Soundness

Completeness: If a set $\mathcal{F}$ of FDs implies F, then F can be derived from $\mathcal{F}$ using Armstrong's axioms.

$$\text{If } \mathcal{F} \vDash F, \text{ then } \mathcal{F} \vdash F.$$

Soundness: If F can be derived from a set of FDs $\mathcal{F}$ using Armstrong's axioms, then $\mathcal{F}$ implies F.

$$\text{If } \mathcal{F} \vdash F, \text{ then } \mathcal{F} \vDash F.$$

With Completeness and Soundness, we know that

$$\mathcal{F} \vdash F \text{ if and only if } \mathcal{F} \vDash F$$

BCNF and 3NF

R is in _NF if for every FD X → A in F, at least one of following is true:

- X → A is a trivial FD (i.e., A ∈ X), or
- X is a superkey for R, or
- A is a Prime Attribute. That is, A is part of some key of R

1NF

BCNF

3NF

Superkeys, Keys and Prime Attributes

For X→A...

- If X^+ = attr(R), then X is a **superkey**
- If there is not proper subset of X that is a super key, then X is a **key**
    - I.e. if A and AD are superkeys, only A is a key
- A is a **prime attribute** if it is part of some key

Online Analytical Processing Operations

$ of Anheuser-Busch by drinker/bar

	Jim	Bob	Mary
Joe's Bar	45	33	30
Nut-House	50	36	42
Blue Chalk	38	31	40

Roll-up by Bar

$ of A-B / drinker

Jim	Bob	Mary
133	100	112

Drill-down by Beer

$ of A-B Beers / drinker

	Jim	Bob	Mary
Bud	40	29	40
M'lob	45	31	37
Bud Light	48	40	35

- Pivoting: Changing the dimensions used in a cross-tab
- Slicing: Creating a cross-tab for fixed values only
- Rollup: Moving from finer-granularity data to a coarser granularity
- Drill down: Moving from coarser granularity data to finer granularity data

Setting NULL values to 0

How do you change NULL value to 0?

- COALESCE(x, 0) has value x if x isn't NULL, and value 0 if x is NULL.
- Using LEFT OUTER JOIN
- Using UNION