Slides to be used with Ullman/Widom *A First Course in Database Systems*, Prentice-Hall, 1997

SQL2 Embedded SQL

Add to a conventional programming language (C in our examples) certain statements that represent SQL operations.

- Each embedded SQL statement introduced with `EXEC SQL`.
- Preprocessor converts C + SQL to pure C.
  - SQL statements become procedure calls.

Shared Variables

A special place for C declarations of variables that are accessible to both SQL and C.

- Bracketed by

```
EXEC SQL BEGIN/END DECLARE SECTION;
```

- In Oracle Pro/C (not C++) the "brackets" are optional.
- In C, variables used normally; in SQL, they must be preceded by a colon.

Example

Find the price for a given beer at a given bar.

```
Sells(bar, beer, price)

EXEC SQL BEGIN DECLARE SECTION;
    char theBar[21], theBeer[21];
    float thePrice;
EXEC SQL END DECLARE SECTION;
            . . .
/* assign to theBar and theBeer */
            . . .
EXEC SQL SELECT price
    INTO :thePrice
    FROM Sells
    WHERE beer = :theBeer AND
        bar = :theBar;
            . . .
```

Cursors

Similar to PL/SQL cursors, with some syntactic differences.

Example

Print Joe's menu.

```
Sells(bar, beer, price)

EXEC SQL BEGIN DECLARE SECTION;
    char theBeer[21];
    float thePrice;
EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE c CURSOR FOR
    SELECT beer, price
    FROM Sells
    WHERE bar = 'Joe''s Bar';
EXEC SQL OPEN CURSOR c;
while(1) {
    EXEC SQL FETCH c
        INTO :theBeer, :thePrice;
    if(NOT FOUND) break;
/* format and print beer and price */
}
EXEC SQL CLOSE CURSOR c;
```

Oracle Vs. SQL2 Features

- SQL2 expects `FROM` in fetch-statement.
- SQL2 defines an array of characters `SQLSTATE` that is set every time the system is called.
  - Errors are signaled there.
  - A failure for a cursor to find any more tuples is signaled there.
  - However, Oracle provides us with a header file `sqlca.h` that declares a *communication area* and defines macros to access it.
  - In particular, `NOT FOUND` is a macro that says "the no-tuple-found signal was set."

Dynamic SQL

Motivation:

- Embedded SQL is fine for fixed applications, e.g., a program that is used by a sales clerk to book an airline seat.

- It fails if you try to write a program like `sqlplus`, because you have compiled the code for `sqlplus` before you see the SQL statements typed in response to the `SQL>` prompt.

- Two special statements of embedded SQL:

  - ❖ `PREPARE` turns a character string into an SQL query.

  - ❖ `EXECUTE` executes that query.

Example: Sqlplus Sketch

```
EXEC SQL BEGIN DECLARE SECTION;
    char query[MAX_QUERY_LENGTH];
EXEC SQL END DECLARE SECTION;

/* issue SQL> prompt */

/* read user's text into array query */

EXEC SQL PREPARE q FROM :query;
EXEC SQL EXECUTE q;
/* go back to reissue prompt */
```

- Once prepared, a query can be executed many times.
- Alternatively, `PREPARE` and `EXECUTE` can be combined into:

```
EXEC SQL EXECUTE IMMEDIATE :query;
```

Why Decomposition "Works"?

What does it mean to "work"? Why can't we just tear sets of attributes apart as we like?

- Answer: the decomposed relations need to represent the same information as the original.
  - We must be able to reconstruct the original from the decomposed relations.

Projection and Join

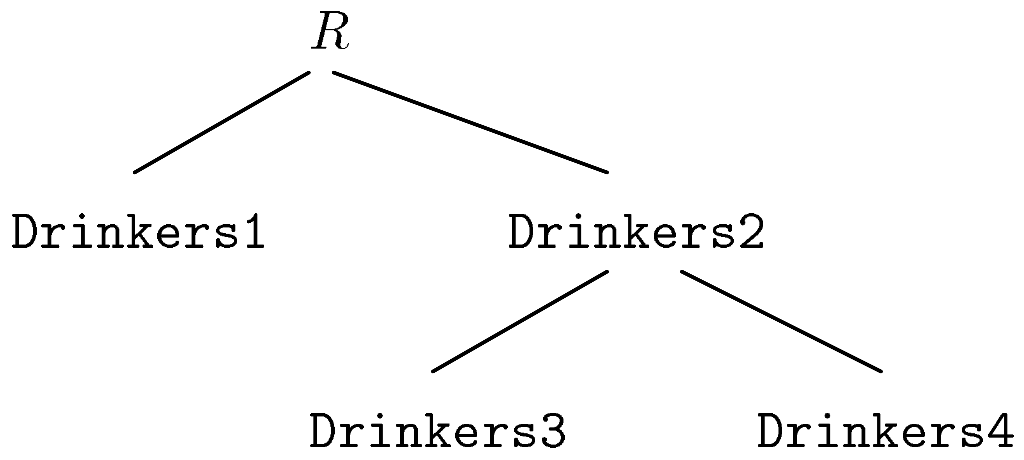
- The operations that relate original and decomposed relations.
- Suppose R is decomposed into S and T. We project R onto S and onto T.

Example

$R =$

name	addr	beersLiked	manf	favoriteBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

- Recall we decomposed this relation as:



- Project onto `Drinkers1(name, addr, favoriteBeer)`:

name	addr	favoriteBeer
Janeway	Voyager	WickedAle
Spock	Enterprise	Bud

- Project onto `Drinkers3(beersLiked, manf)`:

beersLiked	manf
Bud	A.B.
WickedAle	Pete's

- Project onto `Drinkers4(name, beersLiked)`:

name	beersLiked
Janeway	Bud
Janeway	WickedAle
Spock	Bud

Reconstruction of Original

Can we figure out the original relation from the decomposed relations?

- Sometimes, if we natural join the relations.

Example

`Drinkers3` $\bowtie$ `Drinkers4` =

name	beersLiked	manf
Janeway	Bud	A.B.
Janeway	WickedAle	Pete's
Spock	Bud	A.B.

- Join of above with `Drinkers1` = original R.

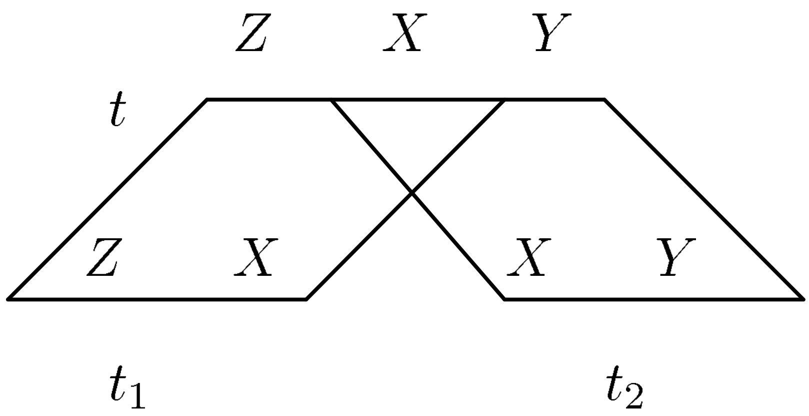
Theorem

Suppose we decompose a relation with schema XYZ into XY and XZ and project the relation for XYZ onto XY and XZ. Then $XY \bowtie XZ$ is *guaranteed* to reconstruct XYZ if and only if either $X \to Y$ or $X \to Z$ holds.

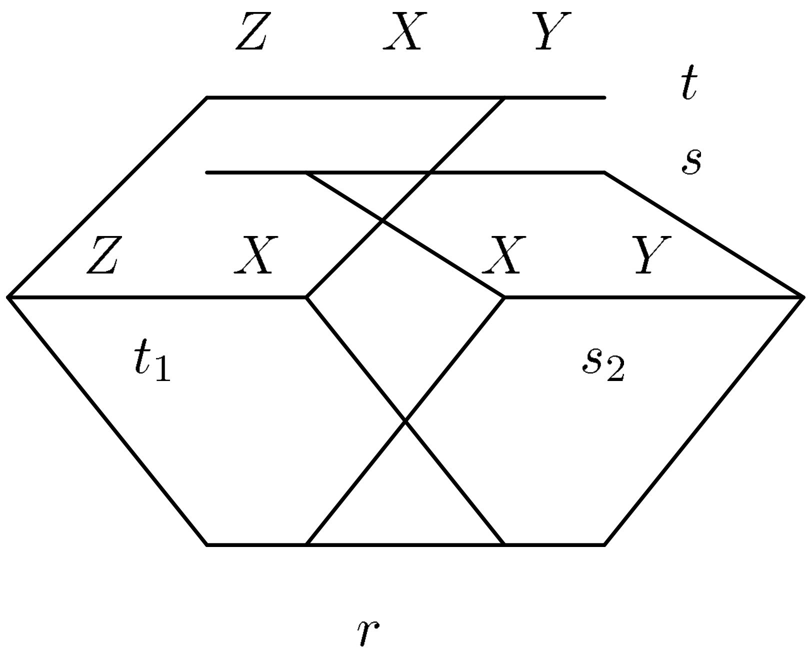
- Notice that whenever we decompose because of a BCNF violation, one of these FD's must hold.

Proof (if)

1. Anything you project comes back in the join.
   - Doesn't depend on FD's.



2. Anything that comes back in the join was in the original XYZ.



- Notice that t_1 and s_2 agree on X; therefore so do t and s.
- If $X \rightarrow Y$, then $r = t$.
- If $X \rightarrow Z$, then $r = s$.
- Either way, r is in original XYZ.

Proof (only-if)

If neither $X \rightarrow Y$ nor $X \rightarrow Z$ holds, then we can find an example XYZ relation where the project-join returns too much.

Z	X	Y
$z1$	x	$y1$
$z2$	x	$y2$

Z	X
$z1$	x
$z2$	x

X	Y
x	$y1$
x	$y2$

Z	X	Y
$z1$	x	$y1$
$z1$	x	$y2$
$z2$	x	$y1$
$z2$	x	$y2$

Application to BCNF Decomposition

- When we decompose R into S and T, it is because there is a FD of the form $(S \cap T) \rightarrow (T - S)$.
- Thus, we can always reconstruct R from S and T.