```
DELIMITER ;;
CREATE PROCEDURE add_working_days(IN start VARCHAR(12), IN end VARCHAR(12))
 BEGIN
   --переменные из таблицы поставщиков
   DECLARE cursor_DATE_BEGIN DATE;
   DECLARE cursor_DATE_END DATE;
   DECLARE cursor_NAME VARCHAR(20);
   DECLARE cursor_SCHEDULE VARCHAR(20);

   --переменная для хранения первого/последнего для периода (откуда уже добавляем)
   DECLARE start_day DATE;
   DECLARE end_day DATE;

   --начальный период может быть больше DATE_BEGIN, переменная хранит эту разницу
   DECLARE first_day INT;
   DECLARE index_schedule INT;

   --период рабочего дня для записи
   DECLARE work_start DATE;
   DECLARE work_end DATE;

   --сколько дней попали в период
   DECLARE interval_days INT;
   -- хранят текущую и следующую букву в расписании
   DECLARE day_val VARCHAR(5);
   DECLARE day_val_next VARCHAR(5);
   -- индексы i - на каком дне от начала действия расписания, n - всего дней (как и
интервал)
   DECLARE i INT DEFAULT 0;
   DECLARE n INT DEFAULT 0;

   DECLARE done INT DEFAULT FALSE;
   DECLARE cursor_i CURSOR FOR SELECT NAME,SCHEDULE,DATE_BEGIN,DATE_END
FROM T_CONTRACTOR_SHERULER;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cursor_i;
   -- конвертируем дату
   SET start = (SELECT STR_TO_DATE(start, '%d.%m.%Y'));
   SET end = (SELECT STR_TO_DATE(end, '%d.%m.%Y'));

   readloop: LOOP
     --получаем ряд из датасета
     FETCH cursor_i INTO cursor_NAME, cursor_SCHEDULE, cursor_DATE_BEGIN,
cursor_DATE_END;

     --берем максимум из начал периодов
     IF (cursor_DATE_BEGIN > start)
     THEN
     SET start_day = cursor_DATE_BEGIN;
     ELSE
     SET start_day = start;
     END IF;
```

```
    --берем минимум из окончаний периодов
    IF (cursor_DATE_END < end)
    THEN
    SET end_day = cursor_DATE_END;

    ELSE
    SET end_day = end;
    END IF;
    -- если получили существующий период, значит пересечение есть
    IF (start_day <= end_day)
    THEN

    SET interval_days = (SELECT DATEDIFF(end_day, start_day));
    SET first_day = (SELECT DATEDIFF(start_day, cursor_DATE_BEGIN));

    SET index_schedule  = first_day,
     n = interval_days,
     i = 0;
    -- пока в периоде есть незаполненные дни
    WHILE (i <= n) DO

    --берем букву из цикличного массива основываясь на отступе периода в таблице и периода в аргументе
    SET day_val = (SELECT substring(cursor_SCHEDULE, (index_schedule %CHAR_LENGTH(cursor_SCHEDULE))+1, 1));

    -- если ночь
    IF (day_val = "н") THEN

    SET work_start = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL i DAY)), '20:00:00'));
    SET work_end = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL (i+1) DAY)), '08:00:00'));
    INSERT INTO table_b(NAME, DATE_BEGIN, DATE_END) VALUES(cursor_NAME, work_start, work_end);
    --двигаем день вперед
    SET i = i+1;
    END IF;

    IF (day_val ='д')
    THEN
    SET work_start = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL i DAY)), '08:00:00'));
    SET work_end = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL i DAY)), '20:00:00'));
    INSERT INTO table_b(NAME, DATE_BEGIN, DATE_END) VALUES(cursor_NAME, work_start, work_end);
    --смотрим на следующую букву в расписании
    SET day_val_next = (SELECT substring(cursor_SCHEDULE, (index_schedule %CHAR_LENGTH(cursor_SCHEDULE))+2, 1));
    -- если это не ночь, то не двигаем сутки на одни вперед
```

```
    IF (day_val_next <> 'н') THEN SET i = i + 1; END IF;
    END IF;

    IF (day_val ='с')
    THEN
    SET work_start = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL i DAY)),
'08:00:00'));
    SET work_end = (SELECT ADDTIME((SELECT DATE_ADD(start_day, INTERVAL i+1 DAY)),
'08:00:00'));
    INSERT INTO T_CONTRACTOR_WORK_DAY(NAME, DATE_BEGIN, DATE_END)
VALUES(cursor_NAME, work_start, work_end);
    SET i = i+1;
    END IF;
    --если выходной, то просто добавляем сутки
    IF (day_val ='в')
    THEN
    SET i = i+1;
    END IF;
    --двигаемся вперед по цикличному списку
    SET index_schedule = index_schedule + 1;
    END WHILE;

    END IF;

    IF done THEN
    LEAVE readloop;
    END IF;
  END LOOP;
  CLOSE cursor_i;
 END;
;;
DELIMITER ;
```