accept\_open\_block\_invitation

DECLARE

    v\_care\_request\_id UUID;

    v\_care\_request RECORD;

    v\_block\_time\_id UUID;

    v\_invited\_parent\_id UUID;

    v\_reciprocal\_date DATE;

    v\_reciprocal\_start\_time TIME;

    v\_reciprocal\_end\_time TIME;

    v\_existing\_block\_date DATE;

    v\_existing\_block\_start\_time TIME;

    v\_existing\_block\_end\_time TIME;

         v\_declined\_count INTEGER;

     v\_new\_scheduled\_care\_id UUID;   -- Store the new scheduled\_care ID

          -- P, Q, N, O tracking variables for child assignment

      v\_provider\_child\_id UUID;      -- P: Retrieved from event\_title

      v\_other\_children UUID[];       -- Q: Retrieved from event\_description

      v\_opened\_block\_id UUID;        -- N: Retrieved from event\_description

      v\_original\_requester\_block\_id UUID; -- O: Retrieved from event\_description

      -- Helper variables for parsing event\_description

      v\_other\_children\_text TEXT;

      v\_opened\_block\_id\_text TEXT;

      v\_original\_requester\_block\_id\_text TEXT;

BEGIN

    -- Get the care request ID from the care response

    SELECT request\_id INTO v\_care\_request\_id

    FROM care\_responses

    WHERE id = p\_care\_response\_id

    AND status = 'pending';

    IF NOT FOUND THEN

        RAISE EXCEPTION 'Care response not found or not in pending status';

    END IF;

    -- Get the complete care request details

    SELECT \* INTO v\_care\_request

    FROM care\_requests

    WHERE id = v\_care\_request\_id;

    IF NOT FOUND THEN

        RAISE EXCEPTION 'Care request not found';

    END IF;

    -- Verify this is an open block request

    IF v\_care\_request.request\_type != 'open\_block' THEN

        RAISE EXCEPTION 'Care request is not an open block request';

    END IF;

    -- Get the block\_time\_id and invited\_parent\_id from the care response being accepted

    -- These are needed for declining other responses

    SELECT block\_time\_id, invited\_parent\_id INTO v\_block\_time\_id, v\_invited\_parent\_id

    FROM care\_responses

    WHERE id = p\_care\_response\_id;

         -- FIXED: For open block invitations, get opened block times from care\_requests table

     -- Now that create\_open\_block\_invitation properly stores the opened block times in requested\_date fields

     SELECT requested\_date, start\_time, end\_time INTO v\_existing\_block\_date, v\_existing\_block\_start\_time, v\_existing\_block\_end\_time

     FROM care\_requests

     WHERE id = v\_care\_request\_id;

     -- Validate that we have the required opened block information

     IF v\_existing\_block\_date IS NULL OR v\_existing\_block\_start\_time IS NULL OR v\_existing\_block\_end\_time IS NULL THEN

         RAISE EXCEPTION 'Missing opened block date/time information in care request';

     END IF;

     -- Get the reciprocal times being offered from care\_requests table (now stored in reciprocal\_date fields)

     -- This is needed for the care\_requests update and for creating the first two blocks

    SELECT reciprocal\_date, reciprocal\_start\_time, reciprocal\_end\_time INTO v\_reciprocal\_date, v\_reciprocal\_start\_time, v\_reciprocal\_end\_time

    FROM care\_requests

    WHERE id = v\_care\_request\_id;

    IF v\_reciprocal\_date IS NULL OR v\_reciprocal\_start\_time IS NULL OR v\_reciprocal\_end\_time IS NULL THEN

         RAISE EXCEPTION 'Missing reciprocal date/time information in care request';

    END IF;

     RAISE NOTICE 'Using opened block times (from scheduled\_care): date=%, start=%, end=%',

         v\_existing\_block\_date, v\_existing\_block\_start\_time, v\_existing\_block\_end\_time;

     RAISE NOTICE 'Using reciprocal times (from UI form): date=%, start=%, end=%',

         v\_reciprocal\_date, v\_reciprocal\_start\_time, v\_reciprocal\_end\_time;

    -- Create scheduled\_care records based on the care\_requests data

    -- RULE: All time information comes from care\_requests table

     -- RULE: Parent C provides care for the RECIPROCAL time (times Parent B offered)

     -- RULE: Parent C receives care for the ORIGINAL time (times from the opened block)

    -- KEY: Populate related\_request\_id to link all blocks together

     -- DEBUG: Log what we're about to create

     RAISE NOTICE '=== CREATING SCHEDULED\_CARE BLOCKS ===';

     RAISE NOTICE 'Parent C (accepting): %', p\_accepting\_parent\_id;

     RAISE NOTICE 'Parent C child: %', p\_accepted\_child\_id;

     RAISE NOTICE 'Original requester: %', v\_care\_request.requester\_id;

     RAISE NOTICE 'Original requester child: %', v\_care\_request.child\_id;

     -- 1. Parent C (accepting parent) providing care for the RECIPROCAL time (times Parent B offered)

     -- This is where Parent C provides care in exchange for receiving care during the opened block

    INSERT INTO scheduled\_care (

        group\_id, care\_date, start\_time, end\_time, care\_type, status, notes,

        parent\_id, child\_id, related\_request\_id

    ) VALUES (

        v\_care\_request.group\_id,

         v\_reciprocal\_date,                           -- Use reciprocal date/time (times Parent B offered)

         v\_reciprocal\_start\_time,                     -- Use reciprocal start time (times Parent B offered)

         v\_reciprocal\_end\_time,                       -- Use reciprocal end time (times Parent B offered)

        'provided',

        'confirmed',

         v\_care\_request.notes || ' - Open block accepted - Parent C providing care',

         p\_accepting\_parent\_id,                       -- Parent C (the accepting parent) provides care

         p\_accepted\_child\_id,                         -- Parent C's child

        v\_care\_request\_id                            -- Link to the original care request!

           ) RETURNING id INTO v\_new\_scheduled\_care\_id;

     RAISE NOTICE 'Created Block 1: Parent C providing care on % %-% (reciprocal time)',

         v\_reciprocal\_date, v\_reciprocal\_start\_time, v\_reciprocal\_end\_time;

     -- 2. Original requester receiving care for the RECIPROCAL time (times Parent B offered)

     -- This is where the original requester receives care from Parent C

    INSERT INTO scheduled\_care (

        group\_id, care\_date, start\_time, end\_time, care\_type, status, notes,

        parent\_id, child\_id, related\_request\_id

    ) VALUES (

        v\_care\_request.group\_id,

         v\_reciprocal\_date,                           -- Use reciprocal date/time (times Parent B offered)

         v\_reciprocal\_start\_time,                     -- Use reciprocal start time (times Parent B offered)

         v\_reciprocal\_end\_time,                       -- Use reciprocal end time (times Parent B offered)

        'needed',

        'confirmed',

         v\_care\_request.notes || ' - Open block accepted - requester receiving care from Parent C',

         v\_care\_request.requester\_id,                 -- The original requester receives care

        v\_care\_request.child\_id,                     -- Their child

        v\_care\_request\_id                            -- Link to the original care request!

    );

     RAISE NOTICE 'Created Block 2: Original requester receiving care on % %-% (reciprocal time)',

         v\_reciprocal\_date, v\_reciprocal\_start\_time, v\_reciprocal\_end\_time;

    -- 3. Parent C (accepting parent) receiving care for the ORIGINAL OPENED BLOCK time

    -- This ensures Parent C sees the opened block where they're receiving care

    INSERT INTO scheduled\_care (

        group\_id, care\_date, start\_time, end\_time, care\_type, status, notes,

        parent\_id, child\_id, related\_request\_id

    ) VALUES (

        v\_care\_request.group\_id,

         v\_existing\_block\_date,                       -- Use ORIGINAL opened block date/time (from requested\_date fields)

         v\_existing\_block\_start\_time,                 -- Use ORIGINAL opened block start time (from start\_time field)

         v\_existing\_block\_end\_time,                   -- Use ORIGINAL opened block end time (from end\_time field)

        'needed',

        'confirmed',

         v\_care\_request.notes || ' - Open block accepted - Parent C receiving care',

         p\_accepting\_parent\_id,                       -- Parent C (the accepting parent) receives care

        p\_accepted\_child\_id,                         -- Parent C's child

        v\_care\_request\_id                            -- Link to the original care request!

    );

     RAISE NOTICE 'Created Block 3: Parent C receiving care on % %-% (original opened block time)',

         v\_existing\_block\_date, v\_existing\_block\_start\_time, v\_existing\_block\_end\_time;

         -- NOTE: care\_requests and care\_responses updates moved to after child assignment to ensure they complete

         -- STEP 3: Use P, Q, N, O information for precise child assignment

     -- This ensures each parent sees the correct children in their blocks

     -- DEBUG: Log the raw P, Q, N, O data from care\_requests

     RAISE NOTICE '=== EXTRACTING P, Q, N, O VARIABLES ===';

     RAISE NOTICE 'Raw event\_title (P): %', v\_care\_request.event\_title;

     RAISE NOTICE 'Raw event\_description (Q, N, O): %', v\_care\_request.event\_description;

     -- Extract P, Q, N, O from the stored information

     v\_provider\_child\_id := v\_care\_request.event\_title::UUID;  -- P stored as text

     -- Parse event\_description to get Q, N, O

     -- The event\_description contains PostgreSQL array literal format, not JSON

     -- Format: {"other\_children" : ["3fbea496-5314-4d27-b6f3-addb40c7c4e8"], "opened\_block\_id" : "d496c9a0-83a6-4141-881c-6d3de31aae99", "original\_requester\_block\_id" : "b9faf939-abc3-4390-a7ae-1973cc672dd0"}

     -- Extract the text values from the event\_description using regex

     v\_other\_children\_text := regexp\_replace(

         regexp\_replace(

             v\_care\_request.event\_description,

             '.\*"other\_children"\s\*:\s\*\[([^\]]\*)\].\*',

             '\1'

         ),

         '"', '', 'g'

     );

     v\_opened\_block\_id\_text := regexp\_replace(

         v\_care\_request.event\_description,

         '.\*"opened\_block\_id"\s\*:\s\*"([^"]\*)".\*',

         '\1'

     );

     v\_original\_requester\_block\_id\_text := regexp\_replace(

         v\_care\_request.event\_description,

         '.\*"original\_requester\_block\_id"\s\*:\s\*"([^"]\*)".\*',

         '\1'

     );

     -- Convert to UUID arrays/values

     IF v\_other\_children\_text != '' AND v\_other\_children\_text != v\_care\_request.event\_description THEN

         v\_other\_children := string\_to\_array(v\_other\_children\_text, ',')::UUID[];

     ELSE

         v\_other\_children := NULL;

     END IF;

     IF v\_opened\_block\_id\_text != '' AND v\_opened\_block\_id\_text != v\_care\_request.event\_description THEN

         v\_opened\_block\_id := v\_opened\_block\_id\_text::UUID;

     ELSE

         v\_opened\_block\_id := NULL;

     END IF;

     IF v\_original\_requester\_block\_id\_text != '' AND v\_original\_requester\_block\_id\_text != v\_care\_request.event\_description THEN

         v\_original\_requester\_block\_id := v\_original\_requester\_block\_id\_text::UUID;

     ELSE

         v\_original\_requester\_block\_id := NULL;

     END IF;

     RAISE NOTICE '=== EXTRACTED P, Q, N, O VALUES ===';

     RAISE NOTICE 'P (Provider child): %', v\_provider\_child\_id;

     RAISE NOTICE 'Q (Other children): %', v\_other\_children;

     RAISE NOTICE 'N (Opened block): %', v\_opened\_block\_id;

     RAISE NOTICE 'O (Original requester block): %', v\_original\_requester\_block\_id;

     -- Validate that we have the required P, Q, N, O information

     IF v\_provider\_child\_id IS NULL THEN

         RAISE NOTICE 'WARNING: P (provider\_child\_id) is NULL - child assignment may fail';

     END IF;

     IF v\_other\_children IS NULL OR array\_length(v\_other\_children, 1) = 0 THEN

         RAISE NOTICE 'WARNING: Q (other\_children) is NULL or empty - child assignment may fail';

     END IF;

     IF v\_opened\_block\_id IS NULL THEN

         RAISE NOTICE 'WARNING: N (opened\_block\_id) is NULL - child assignment may fail';

     END IF;

     IF v\_original\_requester\_block\_id IS NULL THEN

         RAISE NOTICE 'WARNING: O (original\_requester\_block\_id) is NULL - child assignment may fail';

     END IF;

                 -- 1. Add Parent C's child to Parent B's opening block (N)

         RAISE NOTICE '=== STEP 1: Adding Parent C child to Parent B opening block ===';

         IF v\_opened\_block\_id IS NOT NULL THEN

             -- Check if Parent C's child is already in the existing block

             IF NOT EXISTS (

                 SELECT 1 FROM scheduled\_care\_children

                 WHERE scheduled\_care\_id = v\_opened\_block\_id

                 AND child\_id = p\_accepted\_child\_id

             ) THEN

                 -- Add Parent C's child to Parent B's opening block

                 INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

                 VALUES (

                     v\_opened\_block\_id,  -- N: Parent B's opening block

                     p\_accepted\_child\_id, -- Parent C's child

                     p\_accepting\_parent\_id, -- Parent C is providing care

                     'Open block - Parent C child added to Parent B opening block'

                 );

                 RAISE NOTICE 'SUCCESS: Added Parent C child % to Parent B opening block %',

                     p\_accepted\_child\_id, v\_opened\_block\_id;

             ELSE

                 RAISE NOTICE 'SKIP: Parent C child % already exists in Parent B opening block %',

                     p\_accepted\_child\_id, v\_opened\_block\_id;

             END IF;

         ELSE

             RAISE NOTICE 'ERROR: No opened\_block\_id found, skipping Parent B opening block child addition';

         END IF;

         -- 2. Add Parent C's child to Parent A's receiving care block (O)

         RAISE NOTICE '=== STEP 2: Adding Parent C child to Parent A receiving care block ===';

         IF v\_original\_requester\_block\_id IS NOT NULL THEN

             -- Check if Parent C's child is already in the existing block

             IF NOT EXISTS (

                 SELECT 1 FROM scheduled\_care\_children

                 WHERE scheduled\_care\_id = v\_original\_requester\_block\_id

                 AND child\_id = p\_accepted\_child\_id

             ) THEN

                 -- Add Parent C's child to Parent A's receiving care block

                 INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

                 VALUES (

                     v\_original\_requester\_block\_id,  -- O: Parent A's receiving care block

                     p\_accepted\_child\_id,            -- Parent C's child

                     p\_accepting\_parent\_id,          -- Parent C is providing care

                     'Open block - Parent C child added to Parent A receiving care block'

                 );

                 RAISE NOTICE 'SUCCESS: Added Parent C child % to Parent A receiving care block %',

                     p\_accepted\_child\_id, v\_original\_requester\_block\_id;

             ELSE

                 RAISE NOTICE 'SKIP: Parent C child % already exists in Parent A receiving care block %',

                     p\_accepted\_child\_id, v\_original\_requester\_block\_id;

             END IF;

         ELSE

             RAISE NOTICE 'ERROR: No original\_requester\_block\_id found, skipping Parent A receiving care block child addition';

         END IF;

         -- 3. Add Parent A's child (Q) to Parent B's opening block (N) if not already there

         RAISE NOTICE '=== STEP 3: Adding Parent A child (Q) to Parent B opening block ===';

         IF v\_opened\_block\_id IS NOT NULL AND v\_other\_children IS NOT NULL AND array\_length(v\_other\_children, 1) > 0 THEN

             -- Loop through other children (Q) and add them to Parent B's opening block

             FOR i IN 1..array\_length(v\_other\_children, 1)

             LOOP

                 -- Check if this child is already in Parent B's opening block

                 IF NOT EXISTS (

                     SELECT 1 FROM scheduled\_care\_children

                     WHERE scheduled\_care\_id = v\_opened\_block\_id

                     AND child\_id = v\_other\_children[i]

                 ) THEN

                     -- Add Parent A's child to Parent B's opening block

                     INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

                     VALUES (

                         v\_opened\_block\_id,  -- N: Parent B's opening block

                         v\_other\_children[i], -- Q: Parent A's child

                         p\_accepting\_parent\_id, -- Parent C is providing care

                         'Open block - Parent A child (Q) added to Parent B opening block'

                     );

                     RAISE NOTICE 'SUCCESS: Added Parent A child % to Parent B opening block %',

                         v\_other\_children[i], v\_opened\_block\_id;

                 ELSE

                     RAISE NOTICE 'SKIP: Parent A child % already exists in Parent B opening block %',

                         v\_other\_children[i], v\_opened\_block\_id;

                 END IF;

             END LOOP;

         ELSE

             RAISE NOTICE 'WARNING: Cannot add Parent A child to Parent B opening block - missing opened\_block\_id or other\_children';

         END IF;

                   -- 4. Add children to Parent D's (requester's) receiving care block

         RAISE NOTICE '=== STEP 4: Adding children to Parent D receiving care block ===';

         -- Add Parent D's own child to their receiving care block

         INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

         SELECT

             sc.id,

             v\_care\_request.child\_id,  -- Parent D's own child (from the care\_request)

             p\_accepting\_parent\_id, -- Parent C is providing care

             'Open block - Parent D child added to their receiving care block'

         FROM scheduled\_care sc

         WHERE sc.related\_request\_id = v\_care\_request\_id

         AND sc.care\_type = 'needed'  -- Parent D's receiving care block

         AND sc.parent\_id = v\_care\_request.requester\_id  -- Parent D's block (the requester)

         -- Avoid duplicate children

         AND NOT EXISTS (

             SELECT 1 FROM scheduled\_care\_children scc

             WHERE scc.scheduled\_care\_id = sc.id

             AND scc.child\_id = v\_care\_request.child\_id

         );

         GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

         RAISE NOTICE 'Added Parent D child to their receiving care block: % rows affected', v\_declined\_count;

         -- Add Parent C's child to Parent D's receiving care block

         INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

         SELECT

             sc.id,

             p\_accepted\_child\_id,  -- Parent C's child

             p\_accepting\_parent\_id, -- Parent C is providing care

             'Open block - Parent C child added to Parent D receiving care block'

         FROM scheduled\_care sc

         WHERE sc.related\_request\_id = v\_care\_request\_id

         AND sc.care\_type = 'needed'  -- Parent D's receiving care block

         AND sc.parent\_id = v\_care\_request.requester\_id  -- Parent D's block (the requester)

         -- Avoid duplicate children

         AND NOT EXISTS (

             SELECT 1 FROM scheduled\_care\_children scc

             WHERE scc.scheduled\_care\_id = sc.id

             AND scc.child\_id = p\_accepted\_child\_id

         );

         GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

         RAISE NOTICE 'Added Parent C child to Parent D receiving care block: % rows affected', v\_declined\_count;

         -- 5. Add P and Q to Parent C's newly created blocks

          RAISE NOTICE '=== STEP 5: Adding P and Q to Parent C blocks ===';

          -- Parent C's providing care block (reciprocal time) should include Parent C's child and P

          -- First add Parent C's own child to their providing care block

          INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

          SELECT

              sc.id,

              p\_accepted\_child\_id,  -- Parent C's own child

              p\_accepting\_parent\_id, -- Parent C is providing care

              'Open block - Parent C child added to their providing care block'

          FROM scheduled\_care sc

          WHERE sc.related\_request\_id = v\_care\_request\_id

          AND sc.care\_type = 'provided'  -- Parent C's providing care block

          AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

          -- Avoid duplicate children

          AND NOT EXISTS (

              SELECT 1 FROM scheduled\_care\_children scc

              WHERE scc.scheduled\_care\_id = sc.id

              AND scc.child\_id = p\_accepted\_child\_id

          );

          GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

          RAISE NOTICE 'Added Parent C child to their providing care block: % rows affected', v\_declined\_count;

          -- Then add P (provider's child) to Parent C's providing care block

          IF v\_provider\_child\_id IS NOT NULL THEN

              INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

              SELECT

                  sc.id,

                  v\_provider\_child\_id,  -- P: Provider's child (Parent B's child)

                  p\_accepting\_parent\_id, -- Parent C is providing care

                  'Open block - Provider child (P) added to Parent C providing care block'

              FROM scheduled\_care sc

              WHERE sc.related\_request\_id = v\_care\_request\_id

              AND sc.care\_type = 'provided'  -- Parent C's providing care block

              AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

              -- Avoid duplicate children

              AND NOT EXISTS (

                  SELECT 1 FROM scheduled\_care\_children scc

                  WHERE scc.scheduled\_care\_id = sc.id

                  AND scc.child\_id = v\_provider\_child\_id

              );

              GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

              RAISE NOTICE 'Added P (provider child) to Parent C providing care block: % rows affected', v\_declined\_count;

          ELSE

              RAISE NOTICE 'WARNING: Cannot add P to Parent C providing care block - v\_provider\_child\_id is NULL';

          END IF;

          -- Parent C's receiving care block (opened block time) should include P, Q, and Parent C's child

          -- First add Parent C's own child to their receiving care block

          INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

          SELECT

              sc.id,

              p\_accepted\_child\_id,  -- Parent C's own child

              p\_accepting\_parent\_id, -- Parent C is receiving care

              'Open block - Parent C child added to their receiving care block'

          FROM scheduled\_care sc

          WHERE sc.related\_request\_id = v\_care\_request\_id

          AND sc.care\_type = 'needed'  -- Parent C's receiving care block

          AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

          -- Avoid duplicate children

          AND NOT EXISTS (

              SELECT 1 FROM scheduled\_care\_children scc

              WHERE scc.scheduled\_care\_id = sc.id

              AND scc.child\_id = p\_accepted\_child\_id

          );

          GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

          RAISE NOTICE 'Added Parent C child to their receiving care block: % rows affected', v\_declined\_count;

          -- Then add P (provider's child)

          IF v\_provider\_child\_id IS NOT NULL THEN

              INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

              SELECT

                  sc.id,

                  v\_provider\_child\_id,  -- P: Provider's child (Parent B's child)

                  p\_accepting\_parent\_id, -- Parent C is providing care

                  'Open block - Provider child (P) added to Parent C receiving care block'

              FROM scheduled\_care sc

              WHERE sc.related\_request\_id = v\_care\_request\_id

              AND sc.care\_type = 'needed'  -- Parent C's receiving care block

              AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

              -- Avoid duplicate children

              AND NOT EXISTS (

                  SELECT 1 FROM scheduled\_care\_children scc

                  WHERE scc.scheduled\_care\_id = sc.id

                  AND scc.child\_id = v\_provider\_child\_id

              );

              GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

              RAISE NOTICE 'Added P (provider child) to Parent C receiving care block: % rows affected', v\_declined\_count;

          ELSE

              RAISE NOTICE 'WARNING: Cannot add P to Parent C receiving care block - v\_provider\_child\_id is NULL';

          END IF;

          -- Then add Q (other children) if they exist

          IF v\_other\_children IS NOT NULL AND array\_length(v\_other\_children, 1) > 0 THEN

              -- Use a loop to add each child individually to avoid the unnest() in WHERE clause issue

              FOR i IN 1..array\_length(v\_other\_children, 1)

              LOOP

                  -- Check if this child is already in the block

                  IF NOT EXISTS (

                      SELECT 1 FROM scheduled\_care\_children scc

                      INNER JOIN scheduled\_care sc ON scc.scheduled\_care\_id = sc.id

                      WHERE sc.related\_request\_id = v\_care\_request\_id

                      AND sc.care\_type = 'needed'  -- Parent C's receiving care block

                      AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

                      AND scc.child\_id = v\_other\_children[i]

                  ) THEN

                      -- Add this child to Parent C's receiving care block

                      INSERT INTO scheduled\_care\_children (scheduled\_care\_id, child\_id, providing\_parent\_id, notes)

                      SELECT

                          sc.id,

                          v\_other\_children[i],  -- Q: Current child from the array

                          p\_accepting\_parent\_id, -- Parent C is providing care

                          'Open block - Other children (Q) added to Parent C receiving care block'

                      FROM scheduled\_care sc

                      WHERE sc.related\_request\_id = v\_care\_request\_id

                      AND sc.care\_type = 'needed'  -- Parent C's receiving care block

                      AND sc.parent\_id = p\_accepting\_parent\_id  -- Parent C's block

                      LIMIT 1;

                      RAISE NOTICE 'Added child % to Parent C receiving care block', v\_other\_children[i];

                  ELSE

                      RAISE NOTICE 'Child % already exists in Parent C receiving care block, skipping', v\_other\_children[i];

                  END IF;

              END LOOP;

              RAISE NOTICE 'Finished adding Q (other children) to Parent C receiving care block';

          ELSE

              RAISE NOTICE 'WARNING: Cannot add Q to Parent C receiving care block - v\_other\_children is NULL or empty';

          END IF;

         RAISE NOTICE 'Added children to appropriate blocks with related\_request\_id %', v\_care\_request\_id;

     -- Debug: Log what was created

     RAISE NOTICE 'Created 3 scheduled\_care records for open block %: % providing care, % receiving care, % receiving care (opened time)',

         v\_care\_request\_id, p\_accepting\_parent\_id, v\_care\_request.requester\_id, p\_accepting\_parent\_id;

     -- CRITICAL: Update care\_requests status to accepted

     RAISE NOTICE '=== UPDATING CARE\_REQUESTS STATUS ===';

    UPDATE care\_requests

    SET

        status = 'accepted',

        responder\_id = p\_accepting\_parent\_id,

        response\_notes = v\_care\_request.notes || ' - Open block accepted',

         -- Store the accepting party details

        reciprocal\_parent\_id = p\_accepting\_parent\_id,

        reciprocal\_child\_id = p\_accepted\_child\_id,

        reciprocal\_status = 'accepted'

         -- Note: reciprocal\_date, reciprocal\_start\_time, reciprocal\_end\_time are already set correctly

         -- from the create\_open\_block\_invitation function

    WHERE id = v\_care\_request\_id;

     GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

     RAISE NOTICE 'Updated care\_requests status: % rows affected', v\_declined\_count;

    -- Update the care response status to accepted

     RAISE NOTICE '=== UPDATING CARE\_RESPONSES STATUS ===';

    UPDATE care\_responses

    SET status = 'accepted'

    WHERE id = p\_care\_response\_id;

     GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

     RAISE NOTICE 'Updated care\_responses status: % rows affected', v\_declined\_count;

    -- Decline ONLY the specific time slot and parent that was accepted

    -- This prevents over-booking while keeping other time slots and parents available

     RAISE NOTICE '=== DECLINING OTHER RESPONSES ===';

     -- 1. Decline all responses for the same block\_time\_id (same time slot) EXCEPT the accepted one

    IF v\_block\_time\_id IS NOT NULL THEN

        UPDATE care\_responses

        SET status = 'declined'

        WHERE block\_time\_id = v\_block\_time\_id

         AND status = 'pending'

         AND id != p\_care\_response\_id;

         GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

         RAISE NOTICE 'Declined % responses for block\_time\_id %', v\_declined\_count, v\_block\_time\_id;

    END IF;

     -- 2. Decline all responses for the same invited\_parent\_id (same parent) EXCEPT the accepted one

    IF v\_invited\_parent\_id IS NOT NULL THEN

        UPDATE care\_responses

        SET status = 'declined'

        WHERE invited\_parent\_id = v\_invited\_parent\_id

         AND status = 'pending'

         AND id != p\_care\_response\_id;

         GET DIAGNOSTICS v\_declined\_count = ROW\_COUNT;

         RAISE NOTICE 'Declined % responses for invited\_parent\_id %', v\_declined\_count, v\_invited\_parent\_id;

    END IF;

     RAISE NOTICE '=== FUNCTION COMPLETED SUCCESSFULLY ===';

     RETURN TRUE;

EXCEPTION

    WHEN OTHERS THEN

        RAISE NOTICE 'Error in accept\_open\_block\_invitation: %', SQLERRM;

        RAISE EXCEPTION 'Failed to accept open block invitation: %', SQLERRM;

END;